A MATLAB method of lines template for transport equations

H.S. Lee a, C.J. Matthews a,*, R.D. Braddock a, G.C. Sander b, F. Gandola a

a Faculty of Environmental Sciences, Griffith University, Nathan, QLD 4111, Australia
b Department of Civil and Building Engineering, Loughborough University, Leicestershire LE11 3TU, UK

Received 7 March 2003; received in revised form 15 August 2003; accepted 22 August 2003

Abstract

Many environmental problems involve diffusion and convection processes, which can be described by partial differential equations (PDEs). This paper will describe the development of a MATLAB template that generates a numerical solution to PDEs using the method of lines. The template will be applied to various unsaturated flow problems within soil physics to demonstrate the versatility of the method. In particular, the template will generate solutions for three cases (1) one-dimensional Richards’ equation for vertical infiltration; (2) coupled one-dimensional Richards’ equation and solute transport equation for horizontal water and contaminant flow; and (3) two-dimensional Richard’s equation for unsaturated flow over a complex geometry. Where possible, the results from the template will be compared against analytical solutions to determine the accuracy of the numerical solution. In addition, the paper will provide a discussion on possible extensions to the template and future directions.

Keywords: Method of lines; Template; Contaminant transport; Infiltration; Soil physics; Numerical solutions

Software availability

The MATLAB code for the method of lines (MoL) template, including the examples in this paper, are available at http://www.gu.edu.au/school/eve/cmatthews.html. The code consists of small text files (m-files) and are freely available. The template was designed on a PC version of MATLAB release 12.

1. Introduction

Many environmental problems involve diffusion and convection processes. These problems include the dispersal of pollutants into the air and water, and also the spread of species of plants, infections and animals. Each area has its own physics, chemistry, biology and ecology. The modelling in these areas often leads to a partial differential equation (PDE), or transport equation, which is parabolic in nature. Such equations may be linear, in which case analytic solutions are often achievable. However, many of the transport equations are non-linear and numerical simulations are used to understand the particular problem more fully.

Finite-difference and finite-element methods can be applied to these non-linear transport models. Often their application leads to the need to solve large sets of non-linear algebraic equations, and considerable mathematical skill may be needed in implementing the technique (Ames, 1992). The method of lines (MoL) involves discretising the spatial domain and thus replacing the PDE with a vector system of ordinary differential equations (ODEs), for which efficient and effective integrating packages have been developed (Schiesser, 1991; Shampine and Reichelt, 1994). The MATLAB package has strong vector and matrix handling capabilities, a good set of ODE solvers, and an extensive functionality which can be used to implement the MoL (Shampine and Reichelt, 1994).

Many of these transport equations are similar in structure, even where they have one, two, or even three space dimensions, and even where the functional representations of the underlying diffusive and convective processes are quite complex. A template has been developed in MATLAB to handle transport equations, using the
MoL and the extensive functionality of the language (Lee et al., 1998). The template has many built-in housekeeping features related to lattice generation and the creation of the system of ODEs. Further, the coding of the essential ODEs is simplified, as the coding required closely follows the mathematical description.

This paper aims to show how this template can be used to handle transport equations which incorporate complex physics and chemistry. The template is described and illustrated by application to problems arising in soil physics, where the mathematical descriptions of the diffusive and convective processes are complex. The template is applied to problems in one and two dimensions, to coupled systems of parabolic PDEs, and to a problem with a complex geometric domain.

2. Richards’ equation

The two-dimensional hydraulic pressure \((h)\) form of Richards’ equation is (Hillel, 1980)

\[
C_w \frac{\partial h}{\partial t} = \frac{\partial}{\partial x} \left( K \frac{\partial h}{\partial x} \right) + \frac{\partial}{\partial z} \left( K \frac{\partial h}{\partial z} \right) - \frac{\partial K}{\partial z},
\]

where \(h = h(x,z,t)\) is the hydraulic pressure, \(K\) is the hydraulic conductivity, \(C_w (= \partial \theta/\partial h)\) is the water capacity, \(\theta\) is the water content, and the spatial axes are \(x\) (horizontal) and \(z\) (vertical downwards). Eq. (1) describes unsaturated flow through a homogenous soil where the hydraulic functions, \(K(h)\) and \(C_w(h)\), are highly non-linear functions of \(h\). The hydraulic functions are usually developed by function fitting to experimental data (Viaene et al., 1994). The term \(- \partial K/\partial z\) in (1) is a convective term representing the effects of gravity, while the other terms represent non-linear diffusion. The detail of these terms, whilst important in soil physics, is not as important to this paper. In fact, the mathematical functions that represent these terms can be replaced with functions to suit other areas of modelling.

For one-dimensional flow in the vertical direction, Eq. (1) reduces to

\[
C_w \frac{\partial h}{\partial t} = \frac{\partial}{\partial z} \left( K \frac{\partial h}{\partial z} \right) - \frac{\partial K}{\partial z}.
\]

Sander et al. (1988) provide an analytic solution to the water redistribution problem in the vertical direction for a specific form of the hydraulic functions and boundary conditions. This analytic solution can be used to test the accuracy of a numerical solution obtained from the MoL to the same problem.

The \(\theta\)-based form of Richards’ equation for horizontal flow in a homogenous soil is

\[
\frac{\partial \theta}{\partial t} = \frac{\partial}{\partial x} \left( D \frac{\partial \theta}{\partial x} \right),
\]

where \(\theta\) is the water content, and \(D(\theta) = K/C_w\) is the hydraulic diffusivity. The convection–dispersion equation in the horizontal direction for a solute in an incompressible homogeneous medium (in the absence of sources and sinks) is

\[
\frac{\partial(qc)}{\partial t} = \frac{\partial}{\partial x} \left( D \frac{\partial qc}{\partial x} \right) - \frac{\partial(qc)}{\partial x},
\]

where \(c\) is the solute concentration, \(q = -D(\partial \theta/\partial x)\) is the Darcy flux of water, and \(D_s\) is the coefficient of dispersion (Gandola et al., 2001). Note that

\[
D_s = D_m + \alpha |q|/\theta,
\]

where \(D_m\) is the molecular diffusivity of solute, and \(\alpha\) is the longitudinal dispersivity. Generally the molecular diffusivity is small and is neglected. Applying the product rule to the terms \(\partial(qc)/\partial t\) and \(\partial(qc)/\partial x\) in Eq. (4), and the conservation of mass identity

\[
\frac{\partial \theta}{\partial t} = -\frac{\partial q}{\partial x},
\]

Eq. (4) reduces to

\[
\frac{\partial c}{\partial t} = \frac{\partial}{\partial x} \left( D_s \frac{\partial c}{\partial x} \right) - q \frac{\partial c}{\partial x}.
\]

Gandola et al. (2001) have provided an exact solution for the coupled transport of water and solute (i.e. Eqs. (3) and (7), respectively) for particular forms of \(D(\theta)\) and \(D_s\), and for particular boundary conditions.

3. Representation of derivatives

Consider the lattice

\[
z_i = z^{*} + i\Delta z, \quad i = 0, 1, 2 \ldots n,
\]

where \(\Delta z\) is fixed, \(z^{*}\) is some (arbitrary) starting point, \(i\) is a counter, and the lattice has \(n + 1\) nodes. Also consider a differentiable function \(f(z)\), and note that finite-difference approximations to any finite order of approximation can be obtained for \(f'(z)\) (Wilson and Turcotte, 1994). The errors in these approximations can also be estimated, and the error becomes smaller as the order of the approximation increases. In applying the method of lines, the fourth-order formulae to approximate the first derivative \(f'(z)\) can be used, and these formulae are represented in the matrix (Schiesser, 1991)

\[
B = \frac{1}{24\Delta z} \begin{bmatrix}
-50 & 96 & -72 & 32 & -6 \\
-6 & -20 & 36 & -12 & 2 \\
2 & -16 & 0 & 16 & -2 \\
-2 & 12 & -36 & 20 & 6 \\
6 & -32 & 72 & -96 & 50
\end{bmatrix}.
\]
The middle row of $B$ represents the fourth-order central, or symmetric, finite-difference representation of $f'(z)$, while the first and second rows of $B$ represent backward-difference representations. The fourth and fifth rows represent the corresponding forward-difference representations.

The rows of $B$ are used to obtain derivative representations of $f(z)$ on the lattice. The fourth-order central-difference representation of $df(z_i)/dz$ is given by

$$
\frac{df(z_i)}{dz} = \frac{1}{24\Delta z^2} [12, -16, 0, 16, -2] \otimes f_i,
$$

where $\otimes$ is the usual mathematical dot or scalar product, and uses the third row of $B$. The first upwind finite-difference representation involves the second row of $B$, and

$$
\frac{df(z_i)}{dz} = \frac{1}{24\Delta z^2} [-6, -20, 36, -12, 2] \otimes f_i.
$$

Notice the displacement of the counter $i$ on the vector of $f(z)$ values being used. Finite-difference representations of $f'$ to any order can be written in similar form (Wilson and Turcotte, 1994).

### 3.1. Application to one-dimensional Richards’ equation

Consider Eq. (2) on the lattice and define $h$ as the vector of values of $h(z,t)$ at the nodal points. Then the derivatives $\partial h/\partial z$ at the nodes can be calculated using the derivative representations in Eq. (8). The diffusive terms are discretised using central-difference representations or the middle row in the matrix $B$ (van de Wouwer et al., 2001). The forward or backward representations are used near the ends of the lattice so as to avoid the use of fictitious points (Schiesser, 1991). Thus

$$
h_i^d = A_i^d h,
$$

where $h_i^d$ is the vector of first derivatives $\partial h/\partial z$ on the lattice for the diffusion terms. The $(n + 1, n + 1)$ matrix $A_i^d$ is the first-order derivative matrix and it is sparse. The $i$th row corresponds to the $i$th node, and the non-zero elements correspond to a row of $B$, depending on the form of the representation used for the derivative at that node. For the one-dimensional Richards’ equation, the first two rows of $A_i^d$ are constructed from the first two rows of $B$, completed with zeros to provide forward-difference representations for $i = 0, 1$. Similarly, the last two rows of $A_i^d$ provide backward-difference representations for $i = n - 1, n$. The other rows of $A_i^d$ are constructed using the central-difference representation in $B$, again completed with zeros.

Let $K_h$ be the vector of $dK/dh$ values evaluated at each node on the lattice. Then

$$
K_i = K_h(A_i^d h),
$$

where $K_i$ is the vector of convective first derivatives $\partial K/\partial z$ on the lattice, and $u \cdot v$ is the MATLAB operator for element-by-element multiplication, where $u$ and $v$ are appropriate vectors. Note that $dK/dh$ is derived directly from $K(h)$, and $A_i^d$ constitutes a numerical approximation to $\partial h/\partial z$ thereby giving an approximation to $\partial K/\partial z$ for each node on the lattice. In addition, the $(n + 1, n + 1)$ matrix $A_i^d$ uses backward or upwinding derivatives, from row 4 of $B$, for its general row construction, that is, from row 4 to row $n$ in $A_i^d$. At the start of the lattice, the first three rows of $A_i^d$ are constructed from the first three rows of $B$, completed with zeros, and the last row of $A_i^d$ uses the last row of $B$.

Then Eq. (2) can be discretised to the form

$$
C_u(h) \frac{dh}{dt} = A_i^d(K \cdot A_i^d h) - K_h(A_i^d h),
$$

where $C_u(h)$ is the vector of values of $C_u(h)$ at the lattice point. Eq. (11) is now a system of ordinary differential equations (ODEs), with each ODE corresponding to a node on the lattice. An ODE integrator is then used to integrate Eq. (11) forward in time. Note that Eq. (11) may be stiff requiring a stiff ODE solver which is provided in the MATLAB suite (Shampine and Reichelt, 1994). An initial condition appropriate to the physical problem being considered is used to start the time integrator.

Depending on the type of boundary conditions involved, different procedures at different parts of the stage-wise differentiation have to be used to incorporate the boundary conditions directly into the set of ODEs. For Dirichlet boundary conditions, the values are assigned to the respective boundary points before computation of the derivatives, so that the correct dependent variables, $h$, are retained. Also zeroing the time derivatives at the boundaries is recommended to restrain the integrator from moving the constant value away from its true value (Schiesser, 1991). Where Neumann boundary conditions are involved, the boundary conditions are imposed on the boundary points in the first-order derivative vector, $h^d$, so that the correct first-order derivatives at the boundaries move into the second-order derivative calculation. If mixed-type conditions are involved, these are treated in the same manner as for the Neumann-type...
problem. This implies that the method is able to handle all types of time-dependent boundary conditions.

4. The template

The essential processes for solving transport equations such as Eqs. (1), (2), (3) and (7) using the MoL, may be automated using a template and the functionality and matrix-based capabilities of MATLAB. The conceptual steps are shown on the left-hand side of Fig. 1, while the right-hand side shows the corresponding subroutine structure and information pathways for the computer-based implementation. The conceptual steps of initialising parameters, setting up the initial profile and the output steps are common features of any numerical solution. The other conceptual elements are essential to the implementation of the MoL technique. However, all depend heavily on the dimensionality of the problem, i.e. one-space dimension as in Eq. (2), or two-space dimensions as in Eq. (1).

Consider the one-dimensional Richards’ equation as discussed in Section 3.1. The initialisation step defines the essential parameters for the model. For example, the ‘Initialise’ routine will define soil parameters for the hydraulic functions \( K(h) \) and \( \theta(h) \) and other physical parameters, such as a constant rainfall rate. Also, the routine will initialise geometric parameters, such as the total length in the \( z \)-direction and the total number of nodes over this length. The ‘Domain’ routine generates the spatial lattice, the indexing of these nodes and the housekeeping vectors, which identify the nodes associated with the boundaries of the physical domain. This subroutine is particularly simple for the one-dimensional case, but becomes more complex for two- or three-dimensional cases, particularly where the boundaries are not all parallel to the co-ordinate axes. The ‘Initial’ routine sets up the initial values of \( h \) on the lattice. Note that the matrix-based structure of MATLAB enables mathematical formulae to be expressed directly into the code.

The ‘FODM’ (first-order differentiation matrix) routine accepts information on the structure and indexing of the grid, including the location of boundaries, to generate the matrices \( A_d^z \) and \( A_c^z \). The differential matrices, \( A_d^z \) and \( A_c^z \), are then passed to the ‘RHS’ routine, which
generates the right-hand side of Eq. (11). Values of \( h \) are passed from the ‘ODE45’ routine to the ‘RHS’ routine to calculate the right-hand side of Eq. (11) to give an approximation for \( \frac{\partial h}{\partial t} \). The \( \frac{\partial h}{\partial t} \) vector is then returned to the ODE solver calculating \( h \) at the next time step. In general, ‘ODE45’ from the MATLAB library is used as the ODE solver in the template, which is based on a fourth- and fifth-order pair of Runge Kutta formulae. However, changing the ODE solver within the template is a minor change since all ODE solvers have been standardised within MATLAB.

The last step, ‘Output’ routine, will provide a graphical representation of the numerical output, which will usually involve a plot of \( h \) over the spatial domain. When an analytical solution is available, this paper will compare the numerical and analytical solutions by calculating the relative error

\[
\varepsilon_i = \frac{H_i - h_i}{H_i},
\]

at each node. Here, \( H_i \) is the analytic solution at node \( i \), and \( h_i \) is the numerical solution at node \( i \). Note that other error measures may be used. The functionality of the graphics package in MATLAB is large and well suited to obtaining information on the output. In particular, a host of graphing functions is available for results from two- and three-dimensional models, including contour, quiver and surface plots.

5. Example with one-dimensional transport equation

The analytical solution of Sander et al. (1988) applies to constant flux infiltration into a dry soil. The form of the diffusivity and hydraulic functions within the solution are applicable to realistic soils and, therefore, is an excellent test case for any numerical scheme (Watson et al., 1995). The test case considered here is Case 1 from Watson et al. (1995), where the initial condition is \( h(z,t=0) = -\infty \), or \( \Theta = 0 \), where \( \Theta \) is the reduced water content given by

\[
\Theta = \frac{\theta - \theta_i}{\theta_s - \theta_i},
\]

where \( \theta_i \) is the initial water content and \( \theta_s \) is the saturated water content. The initial condition on \( h \) is difficult to handle numerically, and Eq. (2) is written in terms of \( \Theta \), where

\[
\frac{\partial \Theta}{\partial t} = \frac{\partial}{\partial z} \left[ D(\Theta) \frac{\partial \Theta}{\partial z} \right] - \frac{\partial K(\Theta)}{\partial \Theta} \frac{\partial \Theta}{\partial z}.
\]

The analytical solution of Sander et al. (1988) used the hydraulic function

\[
K(\Theta) = \frac{K_{sat}(1-\nu)\Theta}{(1-\nu\Theta)},
\]

where \( K_{sat} \) and \( \nu \) are parameters. The analytic solution was obtained for the initial condition

\[
\Theta(z,0) = 0,
\]

and the boundary conditions

\[
t > 0, \quad z = 0, \quad -D(\Theta) \frac{\partial \Theta}{\partial z} + K(\Theta) = Q,
\]

\[
t > 0, \quad z \to +\infty, \quad \Theta = 0,
\]

where \( Q \) is a constant surface flux due to rainfall,

\[
D(\Theta) = \frac{D_o}{(1-\nu\Theta)^2},
\]

and \( D_o \) is a constant.

The analytic solution describes a wetting front moving into a previously dry soil. The boundary condition equation implies an infinite domain, and this cannot be handled numerically. This boundary condition was replaced by

\[
t > 0, \quad z = L, \quad \Theta = 0,
\]

with \( L = 100 \) cm, and the solutions to this case will approximate the Sander et al. (1988) solution up until the wetting front starts to interact with the bottom boundary.

The problem represented by Eqs. (14), (16), (17) and (20) was solved using the MoL template. The infiltration problem was given a constant flux of \( Q = 0.8 \times K_{sat} \) with the soil parameters \( \theta_i = 0.06 \) cm\(^3\)/cm\(^3\), \( \theta_s = 0.35 \) cm\(^3\)/cm\(^3\), \( \nu = 0.85 \), \( D_s = 2.75862 \) cm\(^2\) min\(^{-1}\) and \( K_{sat} = 0.1/\theta_s \theta_i \) cm\(^2\) min\(^{-1}\) (Watson et al., 1995). Within the template, these parameters are initialised in ‘Initialise’ along with the geometric parameters \( L = 100 \) cm and \( N = 101 \), where \( N \) is the total number of nodes within the system. The ‘Domain’ function accepts the geometric parameters and returns a matrix \( G \), that represents the grid system for the given problem, and the step size \( \Delta z \). For this case, \( G \) will be a column vector of length \( N \) with the vector entries containing the node numbers. From \( G \), the initial condition is set and stored in \( h_0 \), and the nodes that represent the top and bottom boundary are specified and stored in vectors \( b1 \) and \( b2 \), respectively. The ‘FODM’ routine accepts \( G, b1, b2 \) and \( \Delta z \) to create the differential matrices \( A_h \) and \( A_e \). These matrices are used by the ‘RHS’ routine to calculate a discretised form of Eq. (14), in a similar fashion to Eq. (11), to give an approximation to \( \partial \Theta/\partial t \) at a specific time. The ‘ODE45’ routine accepts \( h_0 \) at the start of the simulation and uses the ‘RHS’ routine to approximate \( \partial \Theta/\partial t \) at the required time interval. The ODE integrator uses the approximation of \( \partial \Theta/\partial t \) to approximate the solution at the next time step, which in turn becomes input data for the ‘RHS’ routine.
6. Coupled transport equations

Now consider the joint redistribution of water and solute in the horizontal direction, described by Eqs. (3) and (7) for 0 ≤ x ≤ L. For simplicity, the initial and boundary conditions are taken in the form

\[ t = 0, \quad \theta = \theta_1, \quad c = c_1, \quad x \in [0, L], \]
\[ t > 0, \quad \theta = \theta_s, \quad c = c_s, \quad x = 0, \]  \quad (21)
\[ t > 0, \quad \theta = \theta_s, \quad c = c_s, \quad x = L. \]

where \( c_i \) is the initial solute concentration and \( c_s \) is a surface solute concentration with \( c_i < c_s \). Eqs. (3) and (7) are coupled transport equations where the water content \( \theta(x, t) \) drives the dispersion of the solute, but the solute concentration \( c(x, t) \) does not affect the flow of water.

The template in Fig. 1 is readily applied to this problem (Eqs. (21) and (27)) with some changes to the computer routines to handle the coupled system of ODEs. The lattice is defined as

\[ x_i = i \Delta x, \quad i = 0, 1, 2 \ldots n, \]

with a commensurate change from \( \Delta z \) to \( \Delta x \) in \( B \) (see Eq. (8)), and in the ‘Domain’ routine. The major change is the setting up of two vectors \( \theta = (\theta_i) \) and \( c = (c_i) \), which are the values of \( \theta(x, t) \) and \( c(x, t) \) at the node points on the lattice. The initial condition for \( \theta \) and \( c \) are both assigned in the ‘Initial’ routine and are combined to give a single column vector \( [\theta \ c]^T \). As before, the ‘FODM’ routine generates two first-order differentiation matrices \( A_{\theta}^i \) and \( A_{c}^i \), where \( A_{\theta}^i \) is used in discretising the diffusion terms in both Eqs. (3) and (7). The convective differentiation matrix \( A_{c}^i \) will be used in representing the term \( \partial c / \partial x \) in Eq. (7). Again, the only real change is the change of co-ordinate axis.

Now let

\[ \Theta^i = A_{\theta}^i \theta, \]  \quad (22)

and

\[ c^i = A_{c}^i c, \]  \quad (23)

where \( \Theta^i \) and \( c^i \) are the vectors of the first derivatives \( \partial \theta / \partial x \) and \( \partial c / \partial x \) in the diffusion terms evaluated on the lattice. Further let

\[ c^i = A_{c}^i c, \]  \quad (24)

be the upwinding or convective derivative. Then, applying the MoL to Eqs. (3) and (7) yields

\[ \frac{d \Theta}{dt} = A_{\theta}^i (D A_{\theta}^i \Theta) \]  \quad (25)

and

\[ \frac{dc}{dt} = (A_{c}^i (\theta \cdot D A_{c}^i c)) / \theta - q A_{c}^i c, \]  \quad (26)
where (●) signifies the element-by-element product of MATLAB, D = D(θ), K = K(θ) and q = -D(Aθ) + K.

On combining
\[
\frac{d}{dt} [\begin{bmatrix} \Theta \\ \mathbf{c} \end{bmatrix} = [A_{\theta}(D_{\theta}\mathbf{c})] + \dot{\theta} \mathbf{A}_{\theta}^T \mathbf{c} - \mathbf{q} \mathbf{A}_{\theta}^T \mathbf{c} ]
\]
which is a coupled system of ODEs. The ‘RHS’ routine accepts estimates of θ and c, as a single column vector, and returns the vector form of the right-hand side of Eq. (27). Note that the ‘RHS’ routine must split the column vector [θ, c] into its separate components θ and c to take advantage of the domain indexing from the ‘Domain’ routine and to evaluate Eqs. (22)–(26). Once Eqs. (25) and (26) have been evaluated, the vectors are combined into a single vector (Eq. (27)), which is passed to ‘ODE45’ routine. Finally, the ‘Output’ routine needs to be altered to show the appropriate graphs of both θ(x,t) and c(x,t), which again involves the splitting of the resultant vector.

A simple exact solution to Eqs. (3) and (7) has been obtained by Gandola et al. (2001) for specific functional forms for D(θ) and D_\theta and are given by
\[
D(\Theta) = \frac{N}{2(\Theta)^N}(1 - \frac{\Theta^N}{N + 1})
\]
and
\[
D_\theta(\Theta) = \frac{\theta_i - \theta_f}{\Theta_{\theta} - \Theta_{\theta_0}}D(\Theta) - \frac{N^2}{2(N + 1)(N + 2)}\Theta^{2N+1}
\]
where Θ is given by eq. (13), D_\theta(\Theta) = \theta D_\theta(\Theta) and N is a parameter that controls the steepness of the water and concentration profiles. A restriction on this solution is that \(\theta_i > \theta_f/(N + 3)\) in order for \(D_\theta^2 \geq 0\).

Given eqs. (21), (28) and (29), the exact solution to the reduced form of Eqs. (3) and (7) is given by
\[
\Theta = \frac{\Theta - \Theta_f}{\Theta_i - \Theta_f} = (1 - \eta)^{1/N},
\]
\[
c = \frac{c - c_i}{c_s - c_i} = (1 - \eta)^{1/N},
\]
where η is the Boltzmann variable given by η = x/\(\sqrt{t}\) (Smiles et al., 1978). This solution applies only for a semi-infinite domain, thus comparisons with the numerical simulations are only possible while the fronts do not interact with the boundary at \(L = 5\).

To generate a numerical solution, the following parameters were given the values of \(\theta_i = 0.1 \text{ cm}^3/\text{cm}^3\), \(\theta_f = 0.4 \text{ cm}^3/\text{cm}^3\), \(\Theta_i = 0.05 \text{ mol/cm}^3\), \(\Theta_f = 0.3 \text{ mol/cm}^3\), \(L = 5 \text{ cm}\), \(\Delta x = 0.025 \text{ cm}\) and \(N = 3\). Numerical solutions from the template where obtained at \(t = 2 \text{ min}\) and plots of both \(\theta(x)\) and \(c(x)\) are shown in Fig. 4(a) and (b) respectively. Fig. 5(a) and (b) also shows the corresponding relative error plots highlighting the difference between the numerical solution and the exact solution given by Eqs. (30) and (31). From Fig. 5, it is evident that a large jump in relative error occurs at \(x = 1.42\), which is the location of the water front. The jump in error is caused by a discontinuity in the \(\theta\) and \(c\) profiles, where the slope changes instantaneously from a large value to zero (Fig. 4). The accuracy of the solution around the discontinuity is generally maintained at 3 to 4 significant figures; however, there is a larger zone of error in the concentration profile (Fig. 5(b)) at 2 significant figure accuracy. In addition, the difference between the total mass in the system between the numerical and exact solutions at \(t = 2 \text{ min}\) has an accuracy of four significant figures for both profiles (Gandola et al., 2001).

In addition, if the simulation is started at some arbitrary time \(t^*\), where \(t^* > 0\) a slightly higher accuracy can be obtained. This results in a very simple change to
Fig. 5. Plot of relative error in (a) $\theta$ and (b) $c$ against horizontal distance at $t = 2$ min for $\Delta x = 0.025$.

Fig. 6. Plot of relative error in (a) $\theta$ and (b) $c$ against horizontal distance at $t = 2$ min for $\Delta x = 0.025$ starting at $t^* = 0.2$ min.

7. A two-dimensional transport equation

Weeks et al. (2003) considered the infiltration of water through a soil liner covering mine waste. They considered a piecewise linear dump and liner as an approximation to the shapes commonly used in practice (Hoekstra and Berkhout, 1994). In Fig. 7, the cover liner domain is given by A, B, C, D, E, F, G and H, which overlays a mound of waste. A two-dimensional Richards’ equation is applied to the liner region, with an initial condition, at $t = 0$, is given by

$$\forall [x,z] \in \text{domain}, \quad h(x,z) = h_0,$$

and boundary conditions, at $t > 0$, are given by

On AB: $$-K\left(\frac{\partial h}{\partial z}\right) = Q$$

The template, which is implemented in the code for this problem, where the initial condition is changed from Eq. (21) at $t = 0$ to one described by Eqs. (30) and (31) and respectively at $t = t^*$. This change moves away from the discontinuous initial condition in Eq. (21) thereby improving the accuracy of the numerical solution. For example, Fig. 6(a) and (b) shows the relative error plots for both $\theta$ and $c$ over the length of the soil profile at $t = 2$ min, but starting the simulation at $t^* = 0.2$ min. In comparison with Fig. 5, Fig. 6 shows that for both profiles the large jump in relative error decreases, but the magnitude of error around the jump is maintained. This suggests that the majority of the error is caused by the discontinuity at the wetting front. Note that as the simulation advances with time the relative error for both profiles decreases as long as there is no interaction with the boundary at $L = 5$ cm.
On BC and FG: \( K \frac{\partial h}{\partial x} \sin(\gamma) - \left( K \frac{\partial h}{\partial z} - K \right) \cos(\gamma) = Q \cos(\gamma) \)  
\( \frac{\partial h}{\partial x} = 0 \)  
\( \frac{\partial h}{\partial z} = 0 \)  
\( \frac{\partial h}{\partial z} - 1 = 0 \)  
\( \frac{\partial h}{\partial z} = 0 \)

where \( Q \) is the rainfall flux, \( h_i \) is a constant hydraulic pressure and \( \gamma \) is the angle of the sloping face of the dump. The surfaces AH and CD have a no-flow condition imposed representing symmetry lines with other parts of the dump and liner. The lower edge of the liner is modelled as a no-flow barrier as an approximation to a capillary barrier, and a natural drainage condition is applied on DE. The shape of the cover liner is two-dimensional and complex, and contains edges which are not aligned parallel to the co-ordinate axes. The spatial discretisation is selected so that \( \frac{\Delta z}{\Delta x} \tan(\gamma) \), ensuring that nodal points lie along the sloping face, and that the boundary conditions can be applied. In addition, the lengths AH = ED \( \tan(\gamma) \), CD = AB \( \tan(\gamma) \) and \( L_z = L_x \) \( \tan(\gamma) \) are related through the geometry. The model does allow the extension of AB by tagging an additional block of nodes to the top flat region ABGH and consequently extending the length \( L_x \). This feature was included in the model by Weeks et al. (2003) to analyse the flow behaviour in the top flat region including the effect of lateral drainage caused by the sloping region.

The hydraulic functions of Touma et al. (1984) are used, and

\( \theta(h) = \theta_i + (\theta_s - \theta_i) \left( \frac{1}{1 + (-m_i h)^{m_4}} \right)^{m_5} \)  
\( K(\theta) = m_3 \theta^{m_4} \),

where \( m_1 = 0.044, m_2 = 2.22, m_3 = 0.55, m_4 = 18130 \) cm/h, \( m_5 = 6.07, \theta_s = 0.312 \) cm²/cm³, and \( \theta_i = 0.0265 \) cm²/cm³.

The conceptual template in Fig. 1 can be applied to this problem, but the implementation requires some alterations to the computer routines. The ‘Initialise’ routine needs to be extended to accept values associated with the new geometry, that is, the lengths \( L_z, AH, AB, \) the extension to the top flat region (AB ex) and the spatial step size in the z-direction (\( \Delta z \)). These values are passed to the ‘Domain’ routine, which has been extended to generate a two-dimensional lattice over the problem region and two numbering systems for the node indexing. As before, the ‘Domain’ routine will number the nodes in a column-wise manner or in the z-direction. This numbering process starts at A, numbers each node moving downwards in the column (in the z-direction) and moves to the next column once the bottom of the liner is reached. This numbering system starts at A, numbers each node moving downwards in the column (in the z-direction) and finishes at D (see Fig. 7).

The column-wise numbering system is the main node indexing for the template and is used to construct all housekeeping vectors, like boundary indexing vectors. The boundary indexing vectors contains the indices of the nodes, which lie on each of the piecewise linear boundaries, and are used by the template to apply the boundary conditions within the numerical scheme. In addition, the new ‘Domain’ routine generates a row-wise node index, which is used as input data for the ‘FODM’ routine to generate the differential matrix in the x-direction (A x).
For the two-dimensional problem, the ‘FODM’ routine has been extended to generate a differential matrix for an arbitrary spatial dimension by accepting information on the appropriate node indexing in the form of a column vector \( k \). The node numbers within \( k \) are always stored in a column-wise fashion regardless of the type of node indexing. This allows the FODM routine to take advantage of all housekeeping vectors to determine the position of various differencing equations within the differential matrix. Using this technique, the FODM routine can generate differential matrices for the appropriate set of nodes, that correspond to the sloping terms of \( \partial z \) and \( \partial x \) in the gradient in the \( x \) and \( z \)-directions, \( A_x^t \) and \( A_z^t \), including the upwind matrices \( A_x^c \) and \( A_z^c \). Note that both \( A_x^t \) and \( A_z^t \) are ordered in a row-wise fashion and cannot be directly applied to \( h \).

To overcome this problem, the template creates a reordering vector \( R \), which is an index vector that converts any domain vector from a column-wise numbering system to a row-wise numbering system. To construct \( R \), the column-wise numbering system is kept but the node numbers are stored in \( R \) in a row-wise fashion. Therefore, to apply \( A_x^t \) or \( A_z^t \) to a vector \( v \) within MATLAB, the reordering vector must be used as follows:

\[
v_x^t(R) = A_x^t(v(R)).
\]

In Eq. (40), \( R \) rearranges \( v \) to a row-wise numbering system and is applied to the differential matrix to evaluate \( v_x^t \). The second application of \( R \) ensures that the resultant vector keeps the original column-wise numbering system so that it can be applied to other domain vectors.

Given that \( R \) is used correctly to evaluate approximations to derivatives in the \( x \)-direction, Eq. (1) in MoL is effectively discretised to the form

\[
\frac{\partial h}{\partial t} = A_x^t(K A_z^t h) + A_z^t(K A_x^t h) - K_h(A_z^t h).
\]

The ‘RHS’ routine needs to be modified to accommodate the above changes in evaluating Eq. (41). Also, boundary conditions need to be applied to a set of nodes, which is essentially automatic using MATLAB and the boundary index vectors. The only other major difference is the handling of the sloping boundary condition given by Eq. (34). In applying Eq. (34), it was assumed that \( \partial h/\partial x \) would give a reasonable approximation to the hydraulic gradient in the \( x \)-direction since there is a gravity component in the \( z \)-direction. Eq. (34) was then rearranged in terms of \( \partial h/\partial z \) and the condition was imposed on the appropriate set of nodes, that correspond to the sloping boundary, in \( h_x \), where \( h_x = A_x^t h \). As a corrector step, Eq. (34) is then rearranged for \( \partial h/\partial x \) and the boundary condition is then imposed on the same set of nodes in \( h_x \) (\( = A_x^t h \)), which include the new values of \( \partial h/\partial z \) from \( h_x \).

To demonstrate flow behaviour from the cover liner model, the dimensions of the dump where set at \( L_x = 1 \) m, \( AB = 0.4 \) m and \( AH = 0.25 \) m with no extension of the top flat region, i.e. \( AB_{es} = 0 \). The rainfall flux was set at 0.5 K s\(^{-1}\) corresponding to a moderate rainfall rate. Fig. 8(a) and (b) shows the contour lines of \( h(x,z,t) \) at \( t = 6 \) and 18 min for a small waste dump, respectively. From Fig. 8, a range of expected flow behaviour is evident with water infiltrating parallel to the sloping surface (Philip, 1991) and the top flat section with a smooth transition between the two flow domains around the corner points B and G. The influence of the bottom no flow boundary is also evident with an accumulation of water along the boundaries HG and GF. As time progresses, Fig. 8(b) also shows lateral divergence from the curvature of the contours around the corner points B and G and the influence of the natural drainage condition from the drier zone along boundary ED particular towards E. These types of flow behaviours are evident in Weeks et al. (2003) and a comparison between the two numerical
solutions for a more realistic size dump, \( L_x = 5 \text{ m} \), has shown good agreement (Matthews, 2002). Note that the ‘Output’ routine can also display the time development of the solution for a particular nodal point within the domain or for the whole domain.

8. Discussion and extensions

The MATLAB template has been applied to three problems, which involve complex functional representation of the processes of diffusion and convection. In each case, the template can be applied to generate the numerical solution. Changes to the principal routines are required to handle a pair of coupled transport equations, and also a two-dimensional problem with a complicated geometry. All of these enabling routines are available at http://www.gu.edu.au/school/eve/cmatthews.html.

In each of the three applications, accurate numerical solutions can be obtained, provided that the solution profile does not contain strong slopes. The accuracy obtained does depend on the spatial step size and the error tolerances used in the integrator. MATLAB does contain an extensive suite of non-stiff solvers, and also stiff solvers, and these can be easily utilised where required. The suit also includes differential algebraic equation (DAE) approaches to solving a system of ODEs, which have particular advantages over an ODE integrator approach (Tocci et al., 1997). The uses of both approaches and their relevance to particular forms of the transport equation will be explored in future work.

It should be noted that the three applications do not contain source or sink terms such as may arise from chemical reactions, biological interactions between species, and even groundwater uptake by plants. Such effects can be readily added in the ‘RHS’ routine for a particular problem. For example, assume that the solute is salt in Eq. (7), and that salt dissolves from the soil into the water through some source term

\[ S = S(x,t,\theta,c), \]

where \( S \) is assumed to depend on space and time, as well as water content and salt concentration. This is represented by the addition of \( S \) to the right-hand side of Eq. (7). Let

\[ s = (S(x,t,\theta,c)), \]

be the vector of values of \( S \) at the lattice points \( x \). Then Eq. (26) is altered to

\[ \frac{dc}{dt} = (A_d^s(\theta,F,J)A_c^s) - q \cdot A_c^s + s, \]

and the necessary alterations to the ‘RHS’ routine are obvious. Note that this source term does not depend on the gradients of either water content or solute concentration. If this were the case, then the appropriate differentiation matrix would need to be included.

Even though the numerical examples within this paper focused on the solution of Richards’ equation, the template could by applied to other problems that involve convective–diffusion processes. The dispersion of a pollutant in the air usually involves convection by a given wind field, as well as diffusion governed by eddy viscosity terms. The representation of such terms has its own physics, but the governing equations are essentially parabolic. They may contain source or sink terms due to settling of particulates or chemical reactions with other species. Obviously, the template can be extended to more than two species (for example, \( \theta \) and \( c \) in Eq. (25)), provided that the interspecies interaction terms are available. In addition, many biological and evolutionary processes involve the spread of species by dispersion (diffusion and convection), as well as by competition (Murray, 2001). Here the focus is often placed on the ‘action–reaction’ terms representing the competition between the species. The treatment of such problems using the template and the method of lines is technically feasible.

Note that fourth-order finite differences are used in matrix \( B \) in Eq. (8). Other orders of finite-difference representation can be used, and note that the symmetric or centred formulae have better truncation error properties over the non-symmetric, non-centred formulae of the same order. Indeed, appropriate derivative representations to high order are available and in MATLAB code (Wilson and Turcotte, 1994), and can be readily incorporated into an appropriate form of \( B \). From there, different order representations can be used to generate the differentiation matrices \( A_d^s \) and \( A_c^s \) to almost any desired order.

A prime feature of the template is that it provides an easy way to incorporate the finite differencing into the right-hand side of the partial differential equation, and thus to numerically evaluate these terms. In the MoL, this is then used by an integrator to advance the solution in time. The template can also be used in other common time-stepping techniques where time is also discretised, on a lattice \( t_j = j\Delta t, \ j = 0, 1, 2, ... \), and \( \Delta t \) is a prescribed time-step. Then Eq. (3) is discretised to the form

\[ \theta_{j+1} - \theta_j = \Delta t F(j, j+1, ...) \]

where \( \theta_j \) is the vector of \( \theta \)-values at the spatial nodes at time \( j \). \( F \) is a function of the right-hand side discretised using the template at various time lines \( j, j+1 \), as well as possibly at other times. The choice of the function \( F \) depends on the classical method being used. The choice

\[ F = [A_d^s J(J)A_c^s], \]

gives the simple forward-time, centred-difference, explicit scheme, while
\[ F = \frac{1}{2} \left( [A^0_i \cdot D(\theta) \cdot A^0_i(\theta)] + [A^i \cdot D(\theta) \cdot A^0_i(\theta)]_{i+1} \right) \]  

Equation (47) gives the classical Crank–Nicolson scheme (Ames, 1992). Where Eq. (3) is non-linear, i.e. \( D=D(\theta) \), then Eq. (45) for the Crank–Nicolson scheme with \( F \) given in Eq. (47), is a set of non-linear algebraic equations for \( \theta_{i+1} \), and is usually solved using some variation of the Newton scheme. The generation of \( F \) in Eq. (47) is frequently done by hand, and requires technical skill. The template provides the means of easily constructing a computational form for carrying out the required functional evaluations. Where the Jacobian or Hessian matrix needs to be evaluated, appropriate differentiation matrices can be readily constructed and applied to \( F \). Whilst the practical aspects of performing the calculations are handled by the template, the theoretical analysis of convergence and error properties still remains.

9. Conclusions

In this paper, the MATLAB template for MoL was applied to three unsaturated flow problems within soil physics. The numerical solution showed good agreement with analytical solutions for one-dimensional constant flux infiltration and one-dimensional coupled water and solute transport except around steep fronts. For the third application, the template was used to model two-dimensional unsaturated flow through a cover liner for a waste dump. The model showed expected flow behaviour and it was noted that good agreement is obtained when compared with another numerical solution. These three problems demonstrate the versatility of the MoL template in handling one- and two-dimensional transport equations and a coupled set of transport equations. In theory, the template could also handle transport equations with source/sink terms and can be applied to other areas of science that involve convective and diffusion processes. A feature of the MoL template is the separation of the time and spatial discretisation allowing for a variety of time stepping schemes to be employed. In particular, the MATLAB library provides a DAE and ODE approach to time integration that can be readily applied in the MoL template. These approaches could be used to gain better accuracy around steep wetting fronts and will be explored in future work. For those familiar with MATLAB, the MoL template provides a relatively easy approach to the numerical solution of transport equations that will be improved over time with the continual exposure to variety of problems within the scientific fields.

References


