Presented at the 3rd Asian Conf. on Unsaturated Soils, Nanjing, China, 2007

PARALLEL COMPUTING OF UNSATURATED SOILS USING ELEMENT-BY-ELEMENT AND DOMAIN DEOMPOSITON METHODS

Youliang Zhang, Domenico Gallipoli & Charles Augarde
School of Engineering, Durham University, U.K.

Abstract: The finite element (FE) simulation of large scale boundary value problems in unsaturated soils is particularly time-consuming owing to the complex properties of the unsaturated porous media. To speed up analyses, a parallel FE code has been developed in this work by using C++ and MPI running on a Linux parallel computer cluster. The "Divide and Conquer" strategy has been used to partition the task among the processors with minimum data exchanges and load balancing. The linear system of equations was solved using the iterative solver BiCGSTAB incorporated within an element-by-element method. One serial and one parallel calculation are presented to validate the code and to test the parallel performance the algorithm.

1. INTRODUCTION

Understanding the behaviour of unsaturated soils is important for many geotechnical applications in natural soil deposits (where the water table is at some depth below ground surface) as well as in compacted soils used as fill material. The accurate analysis of unsaturated soils by the finite element methods requires consideration of the physical coupling among three phases, i.e. the gas, liquid and solid phases. However, the highly nonlinear nature of the governing differential equations and the presence of air pressure as an additional nodal unknown, make computations expensive. Besides, most geotechnical problems cannot be accurately studied in two dimensions and they need instead three-dimensional models. Such features increase the amount of computation beyond the capabilities of any single personal computer. Alternative approaches are therefore required such as parallel computing by using a cluster of computers working simultaneously.

The choice of a parallel strategy is crucial in order to achieve high performance in terms of efficiency and extensibility. A parallel frontal technique was used (Thomas et al. 1998) to reduce computational time in multi-physics problems. However, direct methods lead to fill-in, which makes expensive to solve and difficult to parallelize large sparse linear systems and they are therefore being progressively replaced by iterative methods. Among iterative solvers, the element-by-element (EBE) method is one of the most attractive due to its economy in storage and the ease to implement. The EBE was introduced by Hughes et al (1983) and Ortiz et al (1983). This paper describes a parallel finite element code for coupled multi-physics problems in unsaturated soils developed by using Domain Decomposition Methods (DDM) and an EBE iterative solver.

2. UNSTURATED SOILS SIMULATIONS

2.1 Governing eqations for unsaturated soils

Unsaturated soil is a three-phase porous medium consisting of solid grains, pore liquid and pore gas. In this work it is assumed that the gas phase only contains air whereas the liquid phase contains liquid water and dissolved. The coupled study of water flow, air flow and mechanical equilibrium requires the statement of governing differential equations as well as constitutive relationships between physical variables and primary unknowns. The mass balance of water flow leads to the following equation:

$$\frac{\partial n S_r \rho_w}{\partial t} + div\left(\rho_w \mathbf{v}_w\right) = 0 \qquad (1)$$

The mass balance of air flow within the gas phase as well as in the liquid phase (i.e. in the form of dissolved air) leads to the following equation:

$$\frac{\partial}{\partial t}\left[n\rho_a\left(1 - S_r + HS_r\right)\right] + \nabla \cdot \left[\rho_a\left(\mathbf{v}_a + H\mathbf{v}_w\right)\right] = 0 \qquad (2)$$

The mechanical equilibrium leads to the following equation expressed in terms of net stress $\sigma^*_{ij} = \sigma_{ij} - \delta_{ij} u_a$ and pore gas pressure u_a:

$$\sigma^*_{ij,j} + u_{a,i} + b_i = 0 \qquad (3)$$

In the above equations, n is porosity, S_r is the liquid degree of saturation, ρ_w and ρ_a are the water and air density respectively, $\mathbf{v}_w$ and $\mathbf{v}_a$ are the liquid and gas flux vectors respectively, H is Henry's constant of solubility of dry air in water,

σ_{ij} is the total stress, δ_{ij} is the Kronecker delta and b_i is the body force.

The above governing equations can be re-formulated in terms of the primary unknowns of displacement, pore gas pressure and pore liquid pressure by using a set of constitutive equations introduced in the following section. Besides, initial conditions of displacement, pore gas pressure and pore liquid pressure need to be specified at time t=0 over the full soil domain. The boundary conditions can be specified either as imposed values on nodes or as fluxes on the boundary.

2.2 Constitutive equations

It is assumed that the air behaves as an ideal gas obeying the following law:

$$\rho_a = \frac{M}{TR}\left(u_a + p_{atm}\right) \tag{4}$$

where T is the absolute temperature, R is the gas constant, M is the molecular weight of air and p_{atm} is the atmospheric pressure (note that pore gas pressure, pore liquid pressure and stresses are all relative to the atmosphere in this formulation).

The liquid and gas fluxes, $\mathbf{v}_w$ and $\mathbf{v}_a$ are given by the generalized multiphase Darcy's law as:

$$\mathbf{v}_w = -\frac{\mathbf{k}k_{rw}}{\mu_w}\left(\nabla u_w + \boldsymbol{\gamma}_w\right) \tag{5}$$

$$\mathbf{v}_a = -\frac{\mathbf{k}k_{ra}}{\mu_a}\left(\nabla u_a + \boldsymbol{\gamma}_a\right) \tag{6}$$

where $\mathbf{k}$ is the intrinsic permeability tensor, $\boldsymbol{\gamma}_w$ and $\boldsymbol{\gamma}_a$ are the specific weights, k_{rw} and k_{ra} are the relative permeabilities and μ_a and μ_w are the viscosities (the subscripts $_a$ and $_w$ refer to the gaseous and liquid phases respectively). The values of relative permeabilities range from 0 to 1, and are assumed to be nonlinear functions of the degree of saturation.

The degree of saturation is a unique function of net stress σ^*_{ij} and suction $s = u_a - u_w$ through a water retention relationship such as:

$$S_r = f\left(\sigma^*_{ij}, s\right) \tag{7}$$

The following elastic relationship is adopted between the "average soil skeleton stress", whose definition is $\bar{\sigma}_{ij} = \sigma_{ij} - \delta_{ij}(u_a - S_r s)$, and the strain ε_{kl}:

$$\mathrm{d}\bar{\sigma}_{ij} = \mathbf{D}_{ijkl}\mathrm{d}\varepsilon_{kl} \tag{8}$$

where $\mathbf{D}_{ijkl}$ is the elastic stiffness tensor.

The above equation can also be rewritten in terms of net stress and suction as:

$$\mathrm{d}\sigma^*_{ij} = \mathbf{D}_{ijkl}\left(\mathrm{d}\varepsilon_{kl} - h_{kl}\mathrm{d}s\right) \tag{9}$$

where:

$$h_{kl} = \mathbf{D}^{-1}klij\ \delta_{ij}\left(s\frac{\mathrm{d}S_r}{\mathrm{d}s} + S_r\right) \tag{10}$$

A linear elastic relationship is assumed between average soil skeleton stress and strain so the stiffness tensor $\mathbf{D}$ is independent of stress state.

2.3 Time and spatial discretisation

Spatial discretization by the Galerkin finite element method of Eqs. 1, 2 and 3 and the relevant constitutive relationships yield the algebraic system given in Eq. 11, whose matrix and vector coefficients are discussed in detail in (Gens et al. 1995). The nodal unknowns of pore liquid pressure $\mathbf{u_w}$, pore gas pressure $\mathbf{u_a}$ and displacement $\mathbf{u}$ appear in all equations making the problem fully coupled. This non linear system must be simultaneously solved for the three primary unknowns. The dependency of the matrix coefficients in Eq. 11 on the primary unknowns through constitutive equations requires an iterative procedure for solution in each time step.

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & \mathbf{K}_{gg} & \mathbf{K}_{gw} \\ 0 & 0 & \mathbf{K}_{ww} \end{bmatrix} \begin{bmatrix} \bar{\mathbf{u}} \\ \bar{\mathbf{u}}_a \\ \bar{\mathbf{u}}_\mathbf{w} \end{bmatrix} + \begin{bmatrix} \mathbf{C}_{uu} & \mathbf{C}_{ug} & \mathbf{C}_{uw} \\ \mathbf{C}_{gu} & \mathbf{C}_{gg} & \mathbf{C}_{gw} \\ \mathbf{C}_{wu} & \mathbf{C}_{wg} & \mathbf{C}_{ww} \end{bmatrix} \begin{bmatrix} \frac{\partial \bar{\mathbf{u}}}{\partial \mathbf{t}} \\ \frac{\partial \bar{\mathbf{u}}_a}{\partial t} \\ \frac{\partial \bar{\mathbf{u}}_w}{\partial t} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathbf{f}_u}{\partial t} \\ \mathbf{f}_a \\ \mathbf{f}_w \end{bmatrix} \tag{11}$$

For simplicity, Eq. 11 can also be written in a concise matrix form as:

$$\mathbf{K}(\mathbf{X})\mathbf{X} + \mathbf{C}(\mathbf{X})\frac{\partial \mathbf{X}}{\partial t} = \mathbf{F}(\mathbf{X}) \tag{12}$$

where $\mathbf{X}$ indicates the vector of all nodal unkowns.

The temporal discretization of Eq. 12 is accomplished through an implicit time difference scheme:

$$\left[\mathbf{C} + \Delta t\mathbf{K}\right]\mathbf{X}_{n+1} = \mathbf{C}\mathbf{X}_n + \Delta t\mathbf{F}_{n+1} \qquad (13)$$

where matrices $\mathbf{K}$, $\mathbf{C}$ and vector $\mathbf{F}$ are a function of the primary unknown vector $\mathbf{X} = \mathbf{X}_{n+1}$. The solution vector $\mathbf{X}_{n+1}$ is obtained through the iterative Picard algorithm also known as the successive substitution algorithm. The Picard algorithm is a relative simple iterative method to solve non-linear algebraic systems and can be illustrated by the following list of steps:

(1) Specify an initial solution $\mathbf{X}_0$
(2) For i=1,2,… do
  (2.1) compute the coefficient matrix $\mathbf{K}$, $\mathbf{C}$ and vector $\mathbf{F}$
  (2.2) solve the linear system Eq. (13)
  (2.3) check for convergence. If not converged, goto (2)

In the above algorithm, step (2.2) is the most computational expensive process that is performed in each iteration to solve the non-symmetric algebraic system of Eq. 13. For this purpose, the iterative solver EBE-BiCGSTAB is employed, which has proven to deal effectively with non-symmetric algebraic systems.

3. PARALLEL IMPLEMENTATION OF ELEMENT-BY-ELEMENT SOLVER

The parallel solution of the global system of equations comprises three components: the partition of the task, the solving of a linear system by EBE-BiCGSTAB, and the communication among the processors. The communications are achieved by the Message Passing Interface library (MPI). The other two components are discussed in the following sections.

3.1 Task partition by domain decomposition method

Here the Domain Decomposition Method (DDM) is utilized as a "divide and conquer" strategy to partition the computational task into some smaller subtasks, and at the same time to minimize the communications and to balance the loads among the processors. In the DDM the physical domain of the finite element model is divided into a number of non-overlapping sub-domains mapped onto different processors of a parallel computer. Generally, the number of subdomains equals the number of processors in the parallel computer. Computational tasks such as the calculation of elemental stiffness matrices and external force vectors in each sub-domain are accomplished on separate processors. There are three basic steps for the DDM algorithm. Firstly, the global finite element mesh is partitioned into a number of sub-domains which are then distributed to different processors of parallel computers. The graph partition algorithm used in this work is "Metis" proposed by Karypis et al. (1998). Secondly, the distributed and unassembled global system of equation is solved by means of the iterative solver BiCGSTAB in conjunction with element-by-element methods. Finally, calculated results from each processor are collected by the master processor where they are processed for visualization.

3.2 Element-by-element method

In the element-by-element method, the global system (Eq. 13) is never assembled, a major advantage over other iterative solvers in storage terms. The most important part of the EBE method is the implementation of the matrix-vector multiplication. This can be described as follows.

Let $\mathbf{A_e}$ be the elemental matrix, $\mathbf{x_e}$ the elemental vector, $\mathbf{A}$ be the global matrix, $\mathbf{x}$ the global vector. Then the global matrix-vector product $\mathbf{Ax}$ can be formulated at the elemental level by the following equation.

$$\mathbf{A}\mathbf{x} = \left(\sum_{e=1}^{n} \mathbf{B}_e^T \mathbf{A}_e \mathbf{B}_e\right)\mathbf{x} = \sum_{e=1}^{n}\left[\mathbf{B}_e^T\left(\mathbf{A}_e\mathbf{x}_e\right)\right] \qquad (14)$$

where $\mathbf{B}_e$ is a Boolean connectivity matrix which maps the entries of the elemental matrix into the global matrix. By this way the global matrix-vector product $\mathbf{Ax}$ can be represented by operations at the element level $\mathbf{A}_e\mathbf{x}_e$.

3.3 The EBE- BiCGSTAB algorithm

The BiCGSTAB algorithm is a Krylov subspace iterative method for solving a large non-symmetric linear system of equations. A typical algorithm is presented as follows, (Barrett et al., 1994).

Given $\mathbf{A}$, $\mathbf{b}$ and $\mathbf{x}_0$, solve $\mathbf{x}$
i=0
$\mathbf{r_1} = \mathbf{b} - \mathbf{A} * \mathbf{x}$ (using EBE methods)
$\mathbf{r_0} = \mathbf{r_1}$

```
Do
    i=i+1
    ρ1 = r0^T r1
    if (i == 1)        p = r1;
    else
      β = (ρ1/ ρ2) * (α/ω)
      p = r + (p - v* ω) *β
   endif
   M z= p
   v = A z              (using EBE methods)
   α =ρ1 / (r0 v)
   s = r1 -  v *α
   check convergence, if converged
     x = x + z*α
   endif
   M z1= s
    t = A* z1           (using EBE methods)
    ω = (t * s) / (t*t)
   x =  x + z *α+  z1*ω
  r = s - t *ω
   ρ2 =ρ1
    check convergence, continue if necessary
End
```

It can be seen that the most time-consuming parts of the algorithm are inner products, vector updates, matrix-vector products (commented int the above algorithm), and preconditioner solves. To compute an inner product of two vectors in parallel, each processor first computes the inner product of its own part, and then values on the boundaries between subdomains have to be exchanged with neighbouring processors to calculate the global inner product. This step therefore requires communication.

For vector updates, each processor updates its own segment. Preconditioning is often the most problematic part of parallelizing an iterative method. In this study, the diagonal proconditioner is chosen in view of its ease to implement and parallelize. In this case the preconditioning matrix **M** is **M** = diag(**A**).

4. NUMERICAL EXAMPLES

4.1 The parallel computer and performance analysis methods

The parallel computer used is a Linux cluster called Hamilton at Durham University which consists of 96 dual-processor dual-core Opteron 2.2 GHz nodes with 8 GBytes of memory and a Myrinet fast interconnect for running MPI code, and 135 dual-processor Opteron 2 GHz nodes with 4 GBytes of memory and a Gigabit interconnect. The operating system is SuSE Linux 10.0 (64-bit). The system has 3.5 Terabytes of disk storage.

To analysis the performance of the algorithm, we need to define speedup and efficiency. Let t_1 be the time to execute a given problem with one processor, and t_p the time needed to execute the same problem with p processors. Then the speedup is the ratio of the single processor CPU time over CPU time of p processors.

$$S_p = t_1 / t_p \tag{15}$$

The efficiency is defined as the speedup ratio over the number of processors used (p),

$$E_p = S_p / p = t_1 / (p t_p) \tag{16}$$

4.2 Serial computation

The finite element code described in Section 2 has been validated by simulating the drainage experiment described by Liakopoulos (1965), which is often used as a reference case for assessing the performance of finite element codes in unsaturated soils (see, for example, Lewis & Schrefler 1998). The validation of the finite element code has been carried out by means of a serial computation performed on a single processor.

Liakopulos experiment is a simple test involving drainage of a 1m high column of Del Monte sand initially saturated and subjected to an uniform flow field with a supply of water to the top surface and free drainage from the bottom surface. At t=0, the water supply to the top surface is stopped while maintaining free drainage from the bottom surface. Mechanical equilibrium under self weight is also assumed at t=0 together with a null value of suction throughout the sample. The boundary conditions imposed during drainage are as follows: on the bottom surface horizontal and vertical displacements are restrained while atmospheric air and water pressures are imposed; on the top surface null load and null water flow are imposed while air pressure is maintained at atmospheric value; on the lateral surface water and air flow are null and horizontal displacements are restrained. The constitutive relationships and model parameters adopted in this work are the same as those presented by Lewis & Schrefler (1998).

Table 1. Material parameters for the drainage test.

Parameter	value
Young's modulus(E)	1.3MPa
Poisson's ratio (v)	0.4
Initial porosity (n)	0.2975
Water density (ρ_w)	1000 kg m^{-3}
Intrinsic permeability (k)	4.5×10^{-13} m^2
Water viscosity (μ_a)	1.0×10^{-3} Pa s
Air viscosity (μ_w)	1.8×10^{-5} Pa s

The degree of saturation is described by:

$$S_r = 1 - 1.9722\times10^{-11} s^{2.4279} \quad (17)$$

Water elative permeability also is a function of degree of saturation, valid for $S_r \geq 0.91$, is:

$$k_{rw} = 1 - 2.207\left(1 - S_r\right)^{1.0121} \quad (18)$$

The relative permeability of air, k_{ra}, is a function of the degree of saturation and given by:

$$k_{ra} = \left(1 - S_e\right)^2 \left(1 - S_e^{\frac{5}{3}}\right), \quad S_e = \frac{S_r - 0.2}{1 - 0.2} \quad (19)$$

For the serial simulations, the model is divided into 20 8-node isoperimetric elements. A 10 seconds time-step is applied. A calculation of 720 time steps, i.e. 120 minutes was performed. The variation of suction and vertical displacements at 5mins, 10mins, 20mins, 30mins, 60mins, and 120mins are shown in Fig. 2 and Fig. 3.

4.3 4.3 Parallel computational results

A larger mesh than in the previous case has been used to investigate the performance of the parallel computation. Such mesh consists of 300 elements and is decomposed in 6 subdomains as shown in Fig. 1(b). A calculation of 120 time steps of 1min each, i.e. 120 minutes, has been performed. To assess the parallel performance of the code, the same analysis described in section 4.2 has been carried out by using 1, 2, 4, 6, 8, 15, and 20 processors. The CPU time for each of these computations was recorded and the relationships between number of processors and CPU time, speedup, efficiency are shown in Figs. 4, 5 and 6. Inspection of Fig. 4 indicates that when a single processor is used the CPU time is 4605s whereas, when 20 processors are used, the CPU time drops to 478s. Inspection of Fig. 6 indicates that the maximum efficiency is achieved with 8 processors and this is therefore the optimal number of processors for this particular model. When the number of processors increases above 8 the efficiency deteriorates. This is due to the increase of data exchange between processors and the consequent increase in the amount of time spent in communication among nodes, which offsets the benefit of employing additional processors.

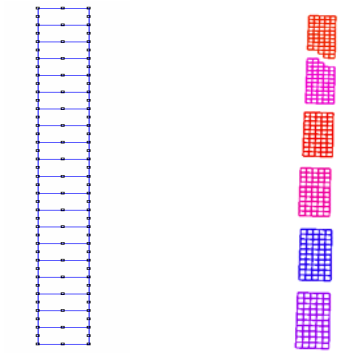

Figure 1. The finite element model and mesh (a) A mesh with 20 8-node isoparametric elements (b) Domain decomposition of a mesh with 300 elements and 6 subdomains.

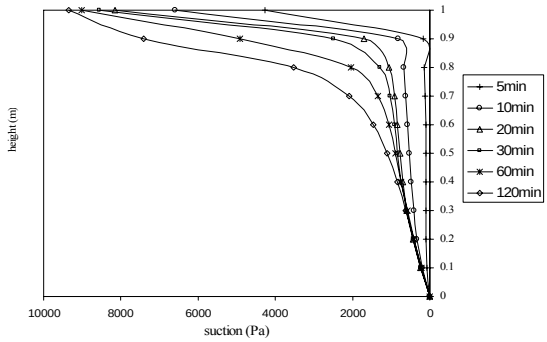

Figure 2. Profile of suction.

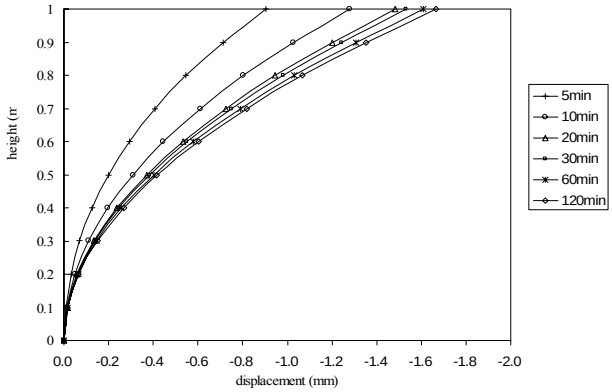

Figure 3. Profile of vertical displacements.

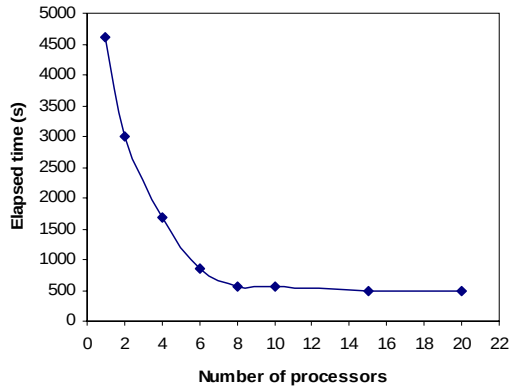

Figure 4. CPU time for different numbers of processors used.

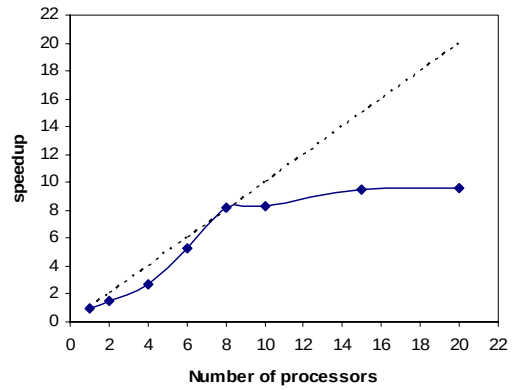

Figure 5. Speedup for different numbers of processors used.

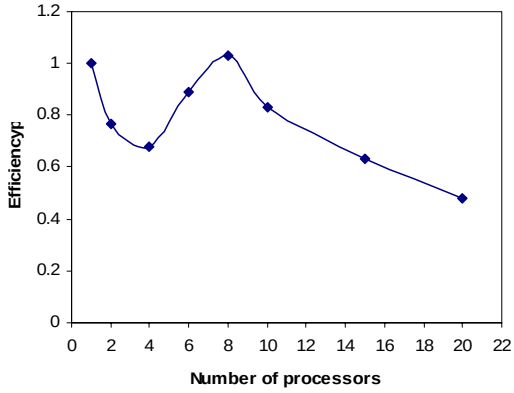

Figure 6. Efficiency for different numbers of processors used.

5. CONCLUSIONS

A parallel finite element code for unsaturated soils using domain decomposition and an element-by-element iterative solver has been presented. By running the code on a Linux cluster of parallel computers, good levels of speedup and efficiency have been achieved. Given the small scale of the boundary value example analysed in this paper, a maximum number of 20 processors has been used. It is envisaged that the code will be applied to larger computations in the next future where a greater number of processors will be employed.

REFERENCES

Barrett, R., Berry, M., Chan, T.F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C. and Van der Vorst, H. 1994. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* 2nd Edition, Philadelphia: SIAM.

Gens, A., Jouanna, P., and Schrefler, B.A. 1995. *Modern issues in non-saturated soils*. CISM course and Lectures No.357, New York: Springer-Verlag.

Hamilton cluster http://hamilton.dur.ac.uk/

Hughes, T.J.R., Levit, I. and Winget, J. 1983. *An element-by-element solution algorithm for problems of solid and structural mechanics.* Computer Methods in Applied Mechanics and Engineering 36: pp. 241-254.

Karypis, G. and Kumar, V. 1998. *Multilevel k-way partitioning scheme for irregular graphs*. Journal of Parallel and Distributed Computing 48(1): pp. 96-129.

Lewis, R.W. and Schrefler, B.A. 1998. *The Finite Element Method in the Static and Dynamic Deformation and Consolidation of Porous Media.* 2nd Edition, Chichester: John Wiley.

Liakopoulos, A.C. 1965. *Transient flow through unsaturated porous media*. Ph.D. Thesis, University of Berkely, California.

Ortiz, M., Pinsky, P.M. and Taylor, R.L. 1983. *Unconditionally Stable Element-By-Element Algorithms for Dynamic Problems*. Computer Methods in Applied Mechanics and Engineering 36 (2): pp. 223-239.

Saad, Y. 1996. *Iterative methods for sparse linear systems.* Boston: PWS..

The Message Passing Interface (MPI) standard. http://www-unix.mcs.anl.gov/mpi/

Thomas, H.R., Yang, H.T., He, Y. and Jefferson, A.D. 1998. *Solving coupled thermo-hydro-mechanical problems in unsaturated soil using a substructuring Frontal technique.* Communications in Numerical Methods in Engineering 14(8): pp. 783-792.