Three-Dimensional Field-Scale Coupled Thermo-Hydro-Mechanical Modeling: Parallel Computing Implementation

Philip James Vardon[1]; Peter John Cleall[2]; Hywel Rhys Thomas[3]; Roger Norman Philp[4]; and Ioana Banicescu[5]

Abstract: An approach for the simulation of three-dimensional field-scale coupled thermo-hydro-mechanical problems is presented, including the implementation of parallel computation algorithms. The approach is designed to allow three-dimensional large-scale coupled simulations to be undertaken in reduced time. Owing to progress in computer technology, existing parallel implementations have been found to be ineffective, with the time taken for communication dominating any reduction in time gained by splitting computation across processors. After analysis of the behavior of the solver and the architecture of multicore, nodal, parallel computers, modification of the parallel algorithm using a novel hybrid message passing interface/open multiprocessing (MPI/OpenMP) method was implemented and found to yield significant improvements by reducing the amount of communication required. This finding reflects recent enhancements of current high-performance computing architectures. An increase in performance of 500% over existing parallel implementations on current processors was achieved for the solver. An example problem involving the Prototype Repository experiment undertaken by the Swedish Nuclear Fuel and Waste Management Co. [Svensk Kärnbränslehantering AB (SKB)] in Äspö, Sweden, has been presented to demonstrate situations in which parallel computation is invaluable because of the complex, highly coupled nature of the problem. **DOI: 10.1061/(ASCE)GM.1943-5622.0000019.**

CE Database subject headings: Three-dimensional models; Computer analysis; Parallel processing; Transport phenomena; Geotechnical models; Soil mechanics; Unsaturated soils; Computation.

Author keywords: Three-dimensional models; Computer analysis; Parallel processing; Transport phenomena; Geotechnical models; Soil mechanics; Unsaturated soils; High-performance computing; Multithreaded; Message passing; Prototype Repository project.

Introduction

Many physical systems are influenced by bidirectional coupling between a number of phenomena. For example, the thermo-hydro-mechanical (THM) behavior of porous materials is important in relation to assessment of proposed deep geological nuclear waste disposal repositories, oil extraction methods, geothermal energy systems, and carbon sequestration technologies. Such coupled problems can often be reduced geometrically to two or even one dimension but in many cases are inherently three-dimensional owing to the shape of the domain, the processes that are occurring, or the materials that are present within or that contain the domain. When undertaking numerical simulations of such problems, a large domain is often required, and where a high resolution of results is demanded, computational effort can become significant. Moreover, physical process timescales within the domain can be long, e.g., 1,000 years, and further processes such as those involving chemistry and biology may be of interest and further increase computational demand.

In such computationally expensive applications in terms of the time taken to run the application, it is worth the investment in development time to seek an improved solution over serial execution. Inevitably, computational techniques reflect advances in technological hardware, i.e., the processors and memory hierarchy (e.g., cache, RAM, disk). Consequently, computer codes which were previously considered optimal may have to be revisited with respect to the "moving" technological "goalposts." For this work, one technique of reducing the time taken for the analysis is examined, but it is acknowledged that others may be used in conjunction with this. Some discussion of these techniques is presented as background information.

High-performance computing (HPC) techniques, in particular, parallel and distributed computing, can be used to vastly reduce computational time (e.g., Thomas et al. 2003b; Smith and Griffiths 2004), although the possible reduction is application- and implementation-specific. Distributed computing is mainly used for easily parallelizable applications such as Monte Carlo analyses, where communication between processes is limited; in fact, in many cases, communication is not required at all until the calculations

[1]Research Fellow, Geoenvironmental Research Centre, Cardiff School of Engineering, Cardiff Univ., Cardiff, UK (corresponding author). E-mail: VardonPJ@Cardiff.ac.uk

[2]Lecturer, Geoenvironmental Research Centre, Cardiff School of Engineering, Cardiff Univ., Cardiff, UK. E-mail: Cleall@Cardiff.ac.uk

[3]Professor and Director, Geoenvironmental Research Centre, Cardiff School of Engineering, Cardiff Univ., Cardiff, UK. E-mail: ThomasHR@Cardiff.ac.uk

[4]Lecturer, Computer Science, Cardiff Univ., Cardiff, UK. E-mail: Roger.Philp@astro.cf.ac.uk

[5]Professor, Dept. of Computer Science and Centre for Computational Sciences, Mississippi State Univ., Starkville, MS 39759. E-mail: ioana@CSE.MsState.Edu

Note. This manuscript was submitted on July 3, 2009; approved on February 24, 2010; published online on September 18, 2010. Discussion period open until September 1, 2011; separate discussions must be submitted for individual papers. This paper is part of the ***International Journal of Geomechanics***, Vol. 11, No. 2, April 1, 2011. ©ASCE, ISSN 1532-3641/2011/2-90–98/$25.00.

Int. J. Geomech. 2011.11:90-98.

are complete. In this study, a tightly coupled computationally expensive calculation is required, and therefore parallel computing is used. Here, multiple processors are used to cooperatively distribute the work, thereby decreasing the overall analysis time in comparison with that of a single processor. In general, to use parallel computation, the programmer must implement a parallel algorithm and find which parts of the calculation can be distributed effectively.

In this paper, a fully coupled finite-element THM model is presented that utilizes parallel computation algorithms to undertake numerical simulations. To investigate the impact of computational architecture, a comparison between a MIPS (600 MHz) processor-based parallel computer with a Myrinet interconnect and an Intel (3 GHz) processor-based parallel computer with an InfiniBand interconnect is made. A field-scale problem, the Prototype Repository Experiment, undertaken by the Swedish Nuclear Fuel and Waste Management Co. [Svensk Kärnbränslehantering AB (SKB)] in Äspö, Sweden (Svemar and Pusch 2000; Johannesson et al. 2007), has been modeled and the results briefly presented to illustrate the points made throughout the paper. The theoretical and numerical formulation of the THM model is first presented with high-performance computing, including a brief analysis of computational trends and an overview of the computational implementation of the theoretical formulation. Benchmark simulations are then presented along with some results to highlight the importance of large-scale modeling. An analysis of computational behavior in parallel is then presented, with a new parallel algorithm proposed in the subsequent section, followed by the experimental results. Finally, conclusions are presented.

Theoretical and Numerical Formulation

The model presented in this paper is a fully coupled THM model, COMPASS, which has been described in detail elsewhere (Thomas and He 1997). A standard finite-element method is employed to achieve spatial discretization, and the resulting system of coupled equations can be expressed in matrix form as

$$\begin{bmatrix} \mathbf{K_{ll}} & \mathbf{K_{lT}} & \mathbf{K_{la}} & - \\ \mathbf{K_{Tl}} & \mathbf{K_{TT}} & \mathbf{K_{Ta}} & - \\ \mathbf{K_{al}} & - & \mathbf{K_{la}} & - \\ - & - & - & - \end{bmatrix} \begin{bmatrix} \mathbf{u_{ls}} \\ \mathbf{T_s} \\ \mathbf{u_{as}} \\ \mathbf{u_s} \end{bmatrix} + \begin{bmatrix} \mathbf{C_{ll}} & \mathbf{C_{lT}} & \mathbf{C_{la}} & \mathbf{C_{lu}} \\ \mathbf{C_{Tl}} & \mathbf{C_{TT}} & \mathbf{C_{Ta}} & \mathbf{C_{Tu}} \\ \mathbf{C_{al}} & \mathbf{C_{aT}} & \mathbf{C_{aa}} & \mathbf{C_{au}} \\ \mathbf{C_{ul}} & \mathbf{C_{uT}} & \mathbf{C_{ua}} & \mathbf{C_{uu}} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{u}}_{\mathbf{ls}} \\ \dot{\mathbf{T}}_{\mathbf{s}} \\ \dot{\mathbf{u}}_{\mathbf{as}} \\ \dot{\mathbf{u}}_{\mathbf{s}} \end{bmatrix} + \begin{bmatrix} \mathbf{f_l} \\ \mathbf{f_T} \\ \mathbf{f_a} \\ \mathbf{f_u} \end{bmatrix} = \{0\} \quad (1)$$

where $\mathbf{K}$, $\mathbf{C}$, and $\mathbf{f}$ = matrices of coefficients of the equation; $\mathbf{u}_{ls}$, $\mathbf{T}_s$, $\mathbf{u}_{as}$, and $\mathbf{u}_s$ = numerically approximated vectors of the pore-water pressure, temperature, pore-air pressure, and displacements; and the dot modifier = time differential. Subscripts of the coefficient matrices **l**, **T**, **a**, and **u** refer to the moisture, temperature, air, and displacement, respectively, with the first subscript referring to the related governing equation and the second, where it occurs, to the contributing variable.

The formulation is based upon the principles of conservation of mass for the hydraulic and air phases, conservation of energy for the temperature field, and stress equilibrium for the deformation field. The hydraulic formulation assumes moisture is made up of both liquid and vapor phases and that the amount of moisture in the vapor phase is governed by the temperature and suction at any location. The mechanisms of flow considered are pressure-driven flow, i.e., Darcian flow, and diffusive flow. The air transfer formulation includes free air and dissolved air and is based upon pressure-driven flow. For heat transfer, local thermal equilibrium is assumed and conduction, convection, and latent heat of vaporization are considered as mechanisms of heat transfer. An elastoplastic constitutive model is used to define a stress-strain constitutive relationship and includes development of strain due to changes in temperature, net mean stress, and suction.

A finite-difference scheme is applied to achieve temporal discretization. In particular, a fully implicit midinterval forward-difference time-stepping algorithm is used, so that the solution of the vector of variables, $\boldsymbol{\phi}$ (where $\boldsymbol{\phi} = [\mathbf{u}_{ls}, \mathbf{T}_s, \mathbf{u}_{as}, \mathbf{u}_s]^T$), can be found. This can be expressed as

$$\mathbf{A}^{n+1/2}\boldsymbol{\phi}^{n+1} + \mathbf{B}^{n+1/2}\left[\frac{\boldsymbol{\phi}^{n+1} - \boldsymbol{\phi}^{n}}{\Delta t}\right] + \mathbf{C}^{n+1/2} = \{0\} \quad (2)$$

where Δt = time-step. Rearranging this yields

$$\left[\mathbf{A}^{n+1/2} + \frac{\mathbf{B}^{n+1/2}}{\Delta t}\right]\boldsymbol{\phi}^{n+1} = \left[\frac{\mathbf{B}^{n+1/2}\boldsymbol{\phi}^{n}}{\Delta t} - \mathbf{C}^{n+1/2}\right] \quad (3)$$

which can be expressed in general matrix form as

$$[a]\boldsymbol{\phi}^{n+1} = [b] \quad (4)$$

An iterative *predictor–corrector* algorithm to allow for nonlinearity in the system is implemented. Fig. 1 shows the overall algorithm both in serial and parallel, including the nonlinear and time-stepping loops. The diagram also indicates the key loops within sections of the code. The solution to this form of equations has been achieved utilizing a Krylov subspace method; in particular, a biconjugate gradient (BiCG) solver (Barrett et al. 1995) has been implemented here, owing to the nonsymmetric and sparse nature of the matrices ($[a]$ and $[b]$). In particular, the iterative solver allows the full maintenance of matrix sparsity in memory during solution, i.e., it does not store zero values and therefore enables larger scale analyses to be undertaken, in comparison to many direct schemes, e.g., Gaussian elimination, where full sparsity cannot be maintained. The pseudocode for the BiCG iterative solver is shown in Fig. 2 (after Barrett et al. 1995), where A = matrix; M = preconditioner, i.e., a method of solving a similar system of equations of which there are a variety of types; i = integer counter; α, β and ρ = real numbers; and the remainder are vector quantities. Matrix A, in the nomenclature after Barrett et al. (1995) is equivalent to $[a]$ in Eq. (4), b is equivalent to $[b]$, and x is equivalent to $\boldsymbol{\phi}$. The solver can be summarized as a series of nine vector-vector calculations and 2 matrix-vector calculations and a single real number division, assuming that the preconditioner, M, is a vector quantity, e.g., the Jacobi preconditioner.

High-Performance Computing

The computational effort required to undertake simulations such as the THM behavior of a system is dependent upon the size of the domain, the level of discretization, and the resolution of transient results required (Cleall et al. 2006b). If this effort is very time-consuming, i.e., the time taken for solution is measured in days, then consideration of development and utilization of HPC is appropriate. In this paper, HPC is defined as the implementation of a parallel processing algorithm.

The historical definitions for types of parallel computers (Flynn 1972) are outdated (van de Steen and Dongarra 1996). To reflect this fact, architectures are referred to here as shared-memory, where multiple processors have access to a single memory block, or

Int. J. Geomech. 2011.11:90-98.

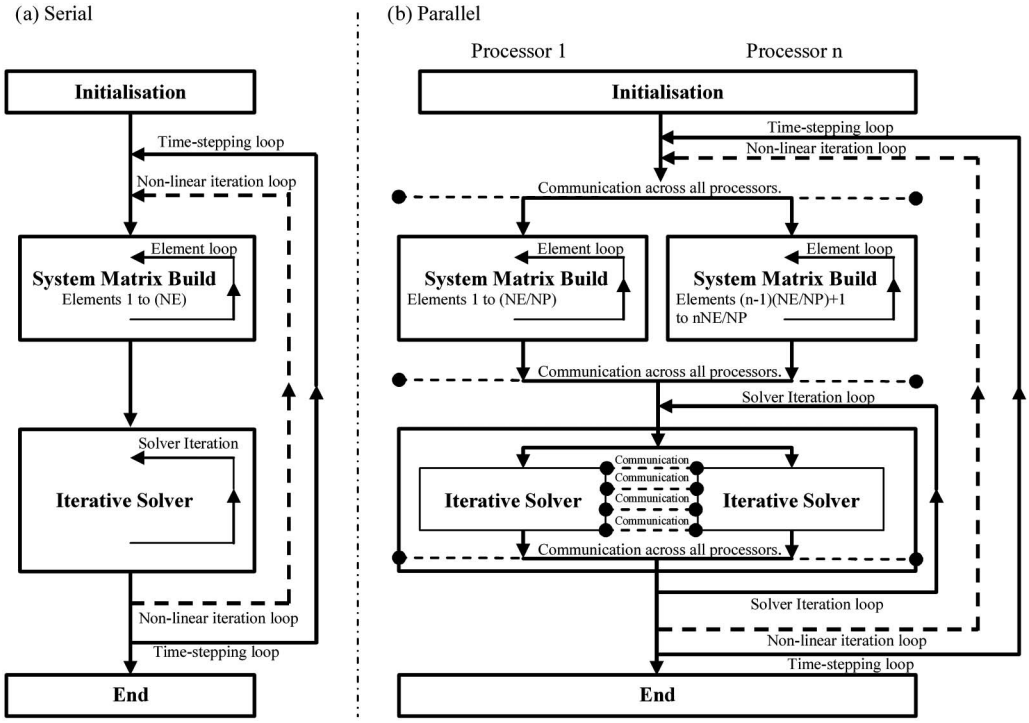

Fig. 1. Schematic of main areas of computation; NE is the number of finite elements and NP is the number of processing cores

Compute $r_0 = b - Ax_0$ *for an initial guess* x_0
Choose $\tilde{r}_0$ *(for example* $\tilde{r}_0 = r_0$*)*
for $i = 1, 2, \ldots$
  solve $Mz_{i-1} = r_{i-1}$
  solve $M^T\tilde{z}_{i-1} = \tilde{r}_{i-1}$
  $\rho_{i-1} = z_{i-1}^T\tilde{r}_{i-1}$
  if $\rho_{i-1} = 0$ *then method fails*
  if $i = 1$
    $p_i = z_{i-1}$
    $\tilde{p}_i = \tilde{z}_{i-1}$
  else
    $\beta_{i-1} = \rho_{i-1}/\rho_{i-2}$
    $p_i = z_{i-1} + \beta_{i-1}p_{i-1}$
    $\tilde{p}_i = \tilde{z}_{i-1} + \beta_{i-1}\tilde{p}_{i-1}$
  end if
  $q_i = Ap_i$
  $\tilde{q}_i = A\tilde{p}_i$
  $\alpha_i = \rho_{i-1}/\tilde{p}_i^T q_i$
  $x_i = x_{i-1} + \alpha_i p_i$
  $r_i = r_{i-1} + \alpha_i q_i$
  $\tilde{r}_i = \tilde{r}_{i-1} + \alpha_i \tilde{q}_i$
end

Fig. 2. Pseudocode of BiCG iterative solver (after Barrett et al. 1995)

distributed-memory, where each processor has its own block of memory. A third architecture is now common due to multicore processors, where multiple processing cores on a single processor have access to a single block of memory but not to the memory of other processors; this configuration is sometimes termed a hybrid architecture.

Shared-memory and distributed-memory parallel programming paradigms are the two main forms of parallel processing programming (Roosta 2000), with two main implementations of each paradigm being open multiprocessing (OpenMP) and message passing interface (MPI), respectively. The main difference is that OpenMP can only be used on shared-memory platforms, whereas MPI can be used across all architectures.

The progression of technology is an important consideration, in particular, the relative progression of different components of a system (i.e., processor, memory, interconnect), and as such are transiently evolving resources. Therefore, the most efficient parallel computational method for solving any particular problem may not be unique. For example, from 1986 to 2002, processors became, on average, 52% faster than the previous year in terms of mega-floating point operations per second or MFLOPS (Hennessy and Patterson 2007), and in comparison, the bandwidth of interconnects only increased by 26% per year on average over the 14 years ending in 2007. Intel predicted that the number of transistors per processor chip will increase in line with Moore's law, i.e., exponentially (Intel 2006), and to allow for this increase in number of transistors per chip within the bounds of other physical constraints, there has been a shift to multicore processors (Asanovic et al. 2006; Intel 2006). In addition, problems that were previously "unsolvable" because of their computational demands have now become solvable.

Computational Implementation

The main computational stages of the analysis are shown in Fig. 1(a) for a serial analysis. The *initialization* and *end* of the analysis are undertaken once, and the time taken is insignificant when considering the overall time of the computation. The *system-matrix build* and the *iterative solver* stages are undertaken in each nonlinear iteration of each time-step and form the majority of the analysis time. The system-matrix build stage has a fixed amount of, essentially, nonsequential calculation to complete, and all information required to complete this step is known at the start of an iteration, making this stage highly parallelizable. The iterative solver stage relies upon estimation of solutions

Int. J. Geomech. 2011.11:90-98.

(Barrett et al. 1995) and a convergent solution being found. The computational effort needed for each pass through the iterative solver is variable and is dependent upon the eigenvalue spectrum and the ability to provide a reasonable initial estimation of the solution. In a boundary/initial-value system such as that considered here, the time taken in this section can vary throughout the analysis. Also, the result from each previous iteration is required so that a good estimate can be made for the next, the result being that any parallelism must be undertaken within each solver iteration. Therefore, the amount of work that can be undertaken within this stage before communication is required is greatly reduced.

Fig. 1(b) shows schematically the implementation of a parallel algorithm alongside the original serial implementation. The salient features are that the initialization and end stages are serial, the system-matrix build stage is parallel with communication only at the end, and the iterative solver stage has many communications within a single iteration. Specifically, the iterative solver section solves the linear algebra problem posed in Eq. (4) by use of a preconditioned BiCG solver; initially, the implementation previously presented by Owen (2000) is used. This has serial speed advantages and maintains sparsity in the system-matrix, vastly reducing storage required over direct solvers (Barrett et al. 1995). However, this method requires full knowledge of vectors throughout the solver; hence, it is more complicated to parallelize (Duff and van der Vorst 1999).

To measure computational efficiency, two metrics are used: speed-up, S, and efficiency, E, (for number of processors, subscript n). These are defined as

$$S_n = \frac{T_1}{T_n} \tag{5}$$

and

$$E_n = \frac{S_n}{n} \tag{6}$$

where T_1 = time of a serial computation; and T_n = time taken for a parallel computation with n processors. It should be noted that the serial computation used for comparison must be the best serial computation available, not merely the parallel algorithm executed on a single processing core.

Benchmark Simulations

The THM behavior of underground high-level nuclear waste (HLW) repositories is of considerable importance when considering the performance assessment of such a system. Several large-scale in situ tests have been undertaken and analyzed (e.g., Thomas et al. 2003a, 2009). However, these have, in general, been restricted to two-dimensional analyses.

In some cases, repositories are proposed to be sited in crystalline rock which is often heavily fractured, and such fractures can cause preferential flowpaths (Neretnieks 1993) which exacerbate the inherent three-dimensionality and complexity of the system. Therefore, it is necessary to model the system in three dimensions and include within the modeled domain the entire repository and the surrounding rock mass. The results are required to be detailed in and around the repository structure; hence, a highly detailed model resolution is needed. In turn, this results in a high level of computational resource in preprocessing, postprocessing, and in particular, within the processing aspect of modeling. The time spent in the processing stage for applications such as these may be, for example, of the magnitude of weeks or longer. Other challenges are met at the visualization stage of both pre- and postprocessing (Cleall et al. 2006a, b).

The Prototype Repository Project (Johannesson et al. 2007) being undertaken by SKB in Äspö, Sweden, aims to investigate, on a full scale, the performance of engineered barriers and near-field crystalline rock in a simulated nuclear waste repository. The experiment seeks to create realistic conditions, as far as is practicable. Sited approximately 450 m below ground surface (Svemar and Pusch 2000), the experiment involves the construction of a repository tunnel, including six deposition holes, and the emplacement of six heater canisters, bentonite buffer blocks, bentonite pellets, and backfill material. Many aspects of this experiment, including full details of the experimental configuration and testing plan, have been previously reported in detail. Dahlström (1998) reported the planning stage; Svemar and Pusch (2000) reported from the early stages; SKB reported various data (for example, Goudarzi and Johannesson 2006); and initial modeling work was reported by Cleall et al. (2006a). This experiment is yielding a unique set of measured results, with a spatial and time scale previously not available.

There are three main stages to the repository project: (1) the preplacement phase, where the tunnel and deposition holes have been created; (2) the placement phase, when heaters, buffer material, and backfill are installed; and (3) the postplacement phase, when all material has been put in place.

Numerical Analysis

The rock, buffer, and backfill materials within a waste repository can be considered to be porous media. In the problem considered here, both large-scale and localized behavior are of importance within a large domain. Therefore, a continuum-model approach has been selected but modified to include hydraulic features at a high resolution. There are two areas of consideration within the rock domain: first, the far field, in which an effective continuum approach will be used; and second, where identified hydraulic features which may be zones of increased or decreased hydraulic conductivity will be included explicitly within the host rock. Tetrahedral elements are used owing to the ease with which different mesh resolutions can be considered across a domain; this helps to reduce computation while allowing accuracy to be maintained. A domain visualization is shown in Fig. 3. This geometrical domain (160 m wide, 100 m high, and 100 m deep) has been discretized by over 480,000 tetrahedral elements.

Two pertinent stages of the experiment are considered in this paper: (1) the preplacement phase, where an isothermal hydraulic analysis is performed and the results discussed in terms of their three-dimensional nature; and (2) a representative THM analysis of the postplacement phase. For this second set of analyses, the focus is on the computational performance aspect. In each case, a full set of material parameters is required and has been defined previously in Cleall et al. (2006a).

For analysis of the preplacement phase of the test, an uncoupled hydraulic analysis is performed. For this analysis, a hydrostatic boundary condition is applied on the outer rock boundary, numerical tests having been undertaken to ensure that this boundary is a sufficient distance from the repository to have a negligible impact on the results. Also, following the approach adopted by Thomas et al. (2003a, 2009), a zero pore-water pressure fixed boundary condition is set on the repository tunnel and deposition-hole walls, with the initial conditions in the rock mass being hydrostatic.

For the postplacement phase analysis, a coupled THM analysis is undertaken. Again, a hydrostatic boundary condition is applied on the outer boundary of the domain along with a fixed temperature (at the ambient value) and zero displacement normal to the boundary surface. The presence of the heating canisters is represented by

Int. J. Geomech. 2011.11:90-98.

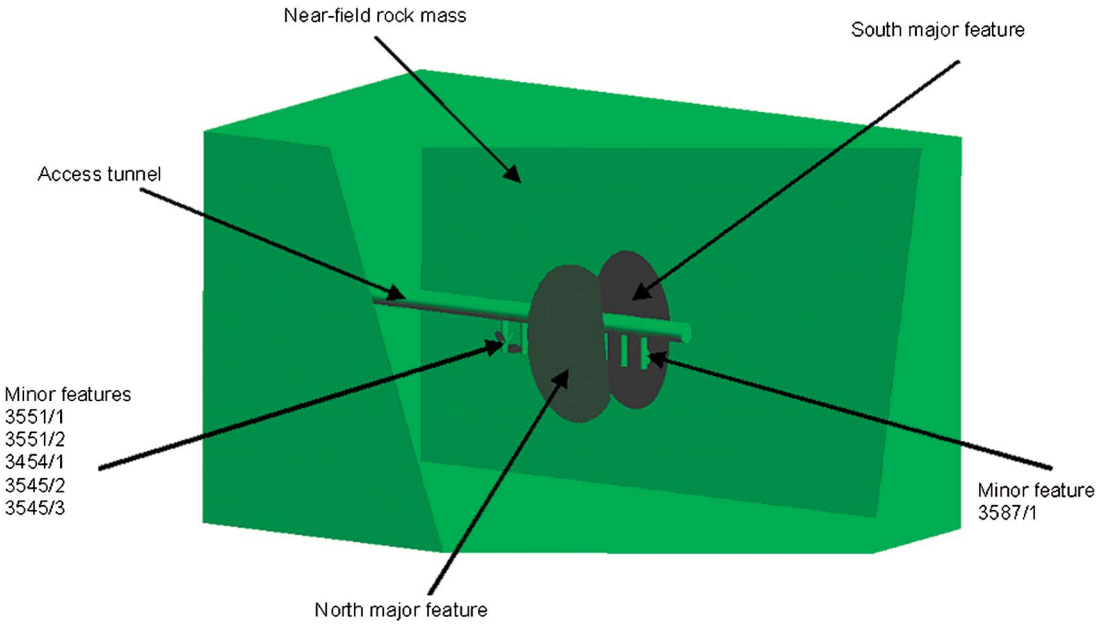

Fig. 3. Cut-away visualisation of model domain

application of a transiently variable thermal flux. The final results of the preplacement phase form the initial hydraulic conditions with the initial temperature set at the ambient value.

Preplacement Phase Results

The results presented in this section consider the preplacement stage of the repository. These results are important because they form the initial conditions of the analyses of the latter stages, as mentioned in the previous section. In Fig. 4, a contour plot of the pore-water pressure is presented. The impact of the shape of the repository, along with the effects of the hydraulic fractures, is apparent; specifically the edge of the south major fracture is highlighted, and its influence can be seen in the rock mass adjacent to it. The effects of this hydraulic feature extend beyond the plane shown, and the impact of the three-dimensional nature of the problem can be seen, thus substantiating the need for this work.

The efficiency of the computation has been measured for the hydraulic analysis and is similar to the THM example presented subsequently. Whereas the problem considered in this section has a more complex mesh, having over 480,000 elements compared to only approximately 100,000 used in the THM benchmark simulations, it only considers one degree of freedom per node compared to five for the THM example, which, in fact, results in a comparable number of unknowns within the linear system to be solved.

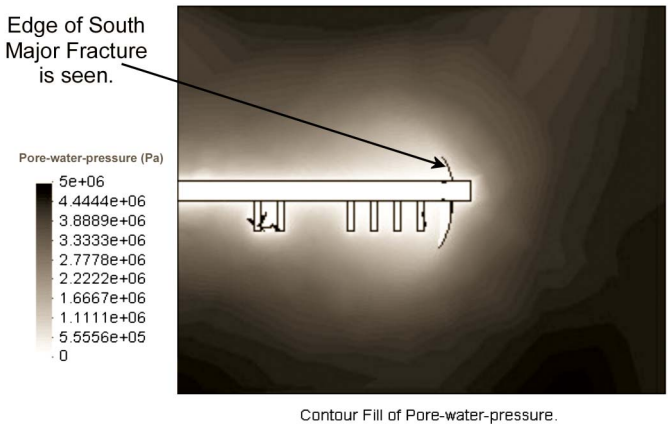

Fig. 4. Contour plot of pore-water pressure (Pa) along the centerline of the repository

Initial Computational Performance for THM Analysis

A series of coupled THM simulations based upon the postplacement phase of the experiment have been performed to gain insight into the computational performance of the adopted approach. The simulations have been run on a 600 MHz MIPS R14000 processor-based parallel computer, using an Ethernet interconnect, and on an Intel Woodcrest Dual Core Xeon 5160 3 GHz processor-based parallel computer, using an InfiniBand interconnect. The code has been optimized in each case for the hardware used.

The domain has the same dimensions and material parameters as considered in the preplacement simulation but has been simplified with removal of the fractures shown in Fig. 3, resulting in a problem that allows a large number of analyses to be undertaken in a relatively short time span to investigate the computational performance. The domain, therefore, includes the rock mass, the access tunnel and the deposition holes. However, some tests have been undertaken with the fractures included, and it has been found that this simplification does not affect the overall trends of the performance in terms of increase of speed and efficiency.

The simplified domain has been discretized with a number of meshes so that the impact of problem size on computational performance can be assessed. Details of these meshes are contained in Table 1. In this initial THM analysis, the mesh with over 100,000 tetrahedral elements is used and, for these tests, is run for 10 time steps only, with over 80,000 unknowns being considered.

The impact of implementing the HPC algorithm on the performance of the code can be most clearly seen if timings are examined for two distinct stages of the computation—the system-matrix build and the iterative solver, as introduced previously. Performance is shown in Fig. 5 against the number of processing cores for each of these stages, along with the total analysis time. This

Int. J. Geomech. 2011.11:90-98.

Table 1. Characterization of Analyses with Different Number of Finite Elements

Number of finite elements	Number of degrees of freedom	Square of number of degrees of freedom (a)	Size of system-matrix (b)	Sparsity of system-matrix (a/b)
100,000	99,534	9.91×10^9	8.63×10^6	1,148.3
350,000	371,172	1.38×10^{11}	3.23×10^7	4,272.4
500,000	527,388	2.78×10^{11}	4.56×10^7	6,096.5

removes any impact of the fixed computational overheads related to initialization which would become relatively more insignificant as the overall time scale considered by the analyses increases. It is worth restating that the amount of calculation required for any particular matrix build, and therefore time, will be constant throughout any analysis, but the amount of calculation required by the solver changes due to its iterative nature.

It can be seen in Fig. 5 that on both sets of parallel computers the system-matrix build-time shows good improvement with an increasing number of processors. For processor numbers below eight, good increases in speed are found. A speed-up of 4.4 times (4.4×; $E_8 = 55\%$ efficiency) for eight processors is found for the R14000 and 4.8× ($E_8 = 60\%$) for the Woodcrest processors. The efficiency reduces as the number of processing cores increases because the number of communications required increases in proportion to the square of the number of processors. In real time, the Woodcrest processors are 18.8 times faster for the matrix build when running with eight processors. It is worth highlighting the need for HPC here—even with the Woodcrest processors a serial calculation of the simplified benchmark takes approximately 70 s for 10 time steps. With analyses routinely taking more than 5,000 time steps and many times more unknowns, the timescales can easily become unmanageable.

The solver stage for the R14000 processors shows good speeds—a 2.9× speed-up ($E_8 = 37\%$)—despite the large amount of communication required between processors. However, the Woodcrest processors show a decrease in speed when going from one to two or four processors, owing to the introduction of the HPC algorithm, and then show an increase in speed for eight processors (1.2×; $E_8 = 15\%$). For a larger model, the amount of work would increase whereas the amount of communication would remain approximately constant, hence, greater efficiency would be expected. In real time, however, the performance of the Woodcrest processors (when using eight processors) is still 8.2 times faster than the R14000 processors for the solver, but the potential for gains through parallel computation is not realized.

New Parallel Solver Algorithm

It is clear from the previous section that the system-matrix build section of the code is able to be parallelized and to produce increased performance with reasonable efficiency. However, the iterative solver exhibits inefficient behavior on a modern parallel computer largely as a result of communication effects. Further analysis of the solver algorithm is required to identify the most effective HPC implementation for this stage of the analysis. It is of the utmost importance that many of the quantities calculated are reused within the same iteration. Therefore, communication is needed throughout the iteration to parallelize these operations.

In the parallel implementation tested in the previous section (Owen 2000), all calculations were parallelized and the results reassembled by message passing before the next calculation was able to be undertaken. It is clear that currently this approach is not able to produce an efficient parallel solution. In trying to consider options for a new parallel algorithm, future computational developments were considered. It was noted previously that HPC systems currently available are utilizing commodity processors and are of a hybrid form, i.e., made up of a number of nodes, each with a small number of processing cores accessing shared-memory.

Options to Restrict Communication Effects

The level of parallelism considered is important; in general, the coarser the parallelism, the more efficient it is likely to be. For example, in this case, the parallelism of the system-matrix build is undertaken at the highest loop level and can be seen to yield good efficiency. However, this is impractical for an iterative solver such as is considered in this paper. However, the parallelism could be restricted to the larger calculations, reducing the comparative communication times, although if one can only parallelize a small number of calculations, the performance gains are reduced. The options considered to reduce the communication effects were:

1. *Shared-memory only*—by eliminating message passing and sharing a single memory;

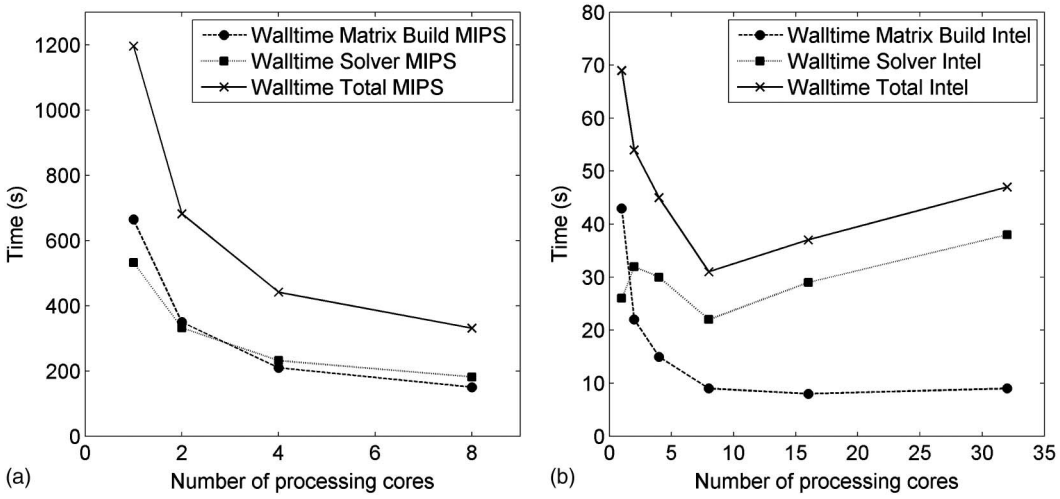

Fig. 5. Initial performance of parallel algorithm for (a) 600 MHz MIPS-based parallel computer, and (b) 3 GHz Intel-based parallel computer

Int. J. Geomech. 2011.11:90-98.

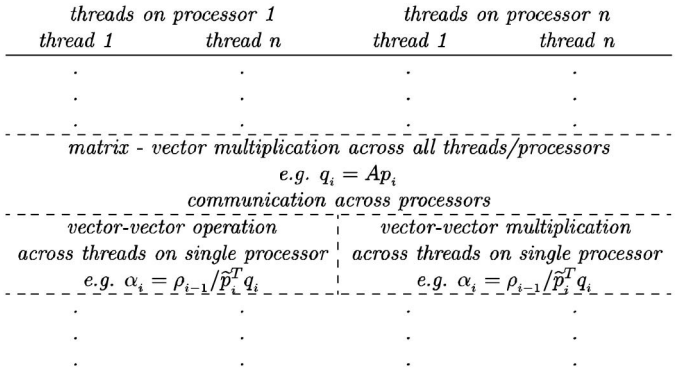

Fig. 6. Outline of multilevel parallelism scheme for BiCG solver

2. *Overlapping communication with computation*—White and Bova (1999) state that in most standard MPI implementations, communication and computation do not overlap. It is suggested that a multithreaded model, reserving a thread for communication, could achieve this (Mao et al. 2006); and
3. *Multilevel parallelism*—the largest calculations, the matrix-vector calculations, could be carried out over a number of nodes, whereas the smaller calculations, the vector-vector calculations, would be carried out on-node, using shared-memory.

Option (3) would converge to option (1) when executed on a single node; therefore, option (1) has been disregarded in further discussion.

Limited overlaps are available for option (2), especially for the longer calculations. With that fact in mind and supported by a simple Hockney performance model analysis (Hockney 1994), it was decided that this approach would only yield limited, if any, performance gain. Lending support to the Hockney approach, results were achieved that were similar to those found in the section "Initial Computational Performance for THM Analysis." Therefore, it was decided that option (3), illustrated by pseudocode in Fig. 6, was the most sensible to pursue. Fig. 6 shows how the two major calculation types in the BiCG solver are proposed to be undertaken. The computational resource is shown at the top of the figure with *Threads on processor 1* shown and subdivided from 1 through *n* threads. An excerpt from the pseudocode is shown where a matrix-vector calculation is carried out over all threads, including all available processors and requiring communication to complete this calculation. A vector-vector operation is then shown being undertaken simultaneously on the threads on each processor with no communication required. Moreover, option (3) is the most suited to portability, being able to be executed on fully shared-memory, fully distributed-memory, and hybrid systems. As such, this method has been implemented, with vector-vector calculations being undertaken on-node and matrix-vector multiplication both on and across nodes, using a combination of multithreading and message passing.

Performance of the New Parallel Algorithm

The performance of the proposed solution has been analyzed on two currently used HPC systems, both hybrid parallel computers. Raptor, situated at Mississippi State University, is based upon Dual Core 2.6 GHz Opteron 2218 processors, with two chips per node, 8 GB of shared-memory per node, and a Gigabit Ethernet interconnect; it has 2,048 processing cores in total. Merlin, situated at Cardiff University in the United Kingdom, has 2,048 Intel Xeon (Harpertown/Seaburg) 3.0 GHz cores, on quad-core chips, with two chips per node, 16 GB of shared-memory per node, and an InfiniBand Connect-X 20 Gbps interconnect. The characteristics of the two computer systems are summarized in Table 2. The number of nodes used for the performance tests was limited to a maximum of eight in each case, owing to (a) high demand, (b) likely cost benefit, and (c) the tightly coupled nature of the formulation. The range of nodes corresponds to 4–32 processing cores on Raptor and 8–32 cores on Merlin.

The performance of the multilevel parallel implementation of the iterative solver is shown in Fig. 7 for two different size problems and on two different parallel computers. Figs. 7(a) and 7(b) show the performance for the 100,000- and the 350,000-element problems being executed on Raptor, and Figs. 7(c) and 7(d) show the performance of the same problems executed on Merlin.

For this application and parallel implementation, it can be seen that, with a larger problem size, the solver is more efficient on Merlin. Fig. 7(c) shows that, for the 100,000 finite problem, the maximum speed increase is 4.5×, whereas, as shown in Fig. 7(d), for a 350,000 finite simulation, the speed-up is 5.8×. It is hypothesized that this speed difference is largely caused by the decrease in communication time proportional to the calculation time for the matrix-vector multiplications. However, this behavior is not exhibited on the Raptor system, where speed-ups remain largely constant.

When executing the analyses on Raptor, it is seen that, on a single node, good performance is achieved—a 3.3× increase in speed on four processing cores is found, as shown on Fig. 7(a). The efficiency of moving from one to two cores on a single node is higher than moving from, for example, four to eight cores. This effect is attributed to the performance of the memory bus. However, when moving to more nodes, the overhead attached to message passing is too great, and this overhead increases with the amount of nodes. In Fig. 7(c), the same analysis is executed on Merlin, and this overhead can also be seen. However, the effect is much reduced owing to the shorter communication times of a faster interconnection network that has both lower latencies and higher bandwidth and allows time reductions by splitting computation across nodes. It can also be seen by the asymptotic shape of the curves on Merlin that without a vastly increased communication speed, mainly in terms of latency, or an algorithm with reduced communication, no further speed-up is possible. The vectors that are calculated within each node also become more significant within the algorithm as the number of nodes increases and act as a base load where no further time reduction is possible.

Table 2. Systems Characteristics

System name	# of nodes	# of chips/node	# of cores/chip	Total # of cores	# of nodes used in experiments	Processor speed (GHz)	RAM/node (GB)	Interconnect
Raptor	512	2	2	2048	1–8	2.6	8	1 GB Ethernet
Merlin	256	2	4	2048	1–8	3.0	16	20 GB InfiniBand

Int. J. Geomech. 2011.11:90-98.

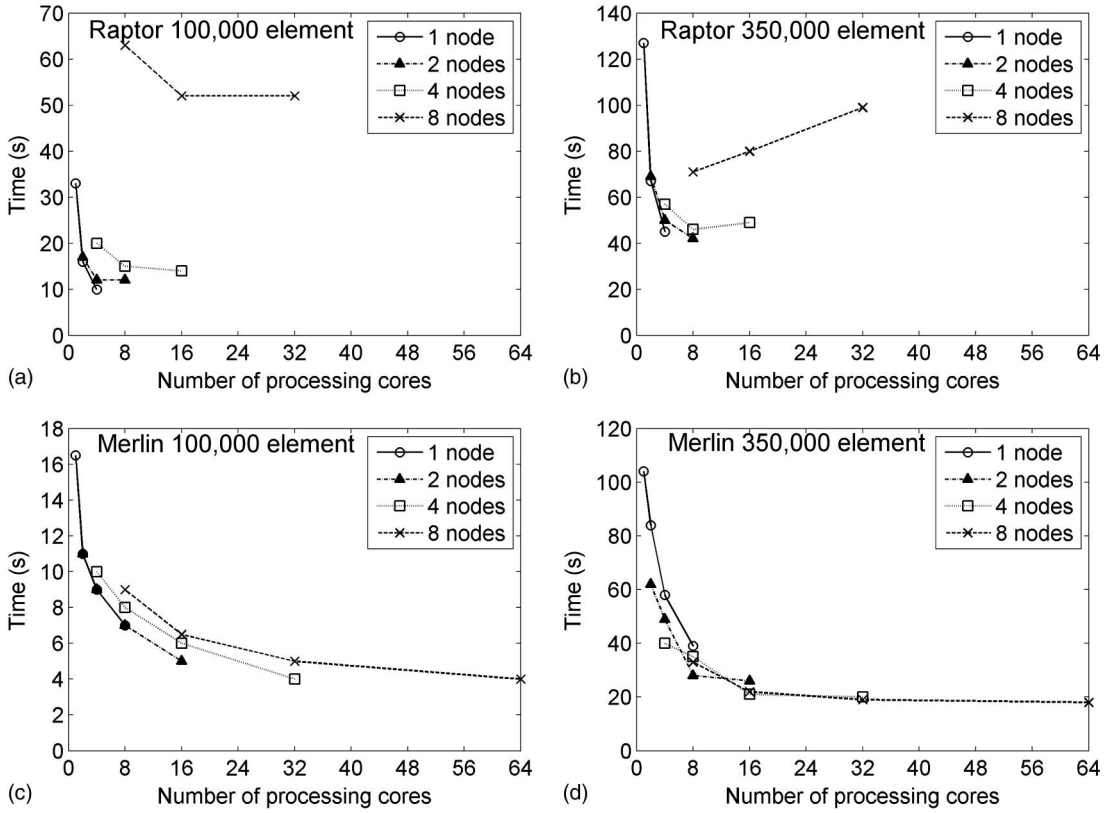

Fig. 7. Wall time of solver for two different size simulations on two HPC systems: (a) 100,000-element simulation on Raptor; (b) 350,000-element simulation on Raptor; (c) 100,000-element simulation on Merlin; and (d) 350,000-element simulation on Merlin

This overhead, found when moving to execute on more nodes, is also reduced for larger numbers of finite elements on Raptor, as illustrated by the proportionally smaller gaps between nodes on Fig. 7(b) compared with Fig. 7(a). Here, the terminology, "gap," is used to signify the ratio between the difference in execution times of two subsequent runs on the same total number of cores but over a different number of nodes. However, the pattern shown in Fig. 7(d) is more complex. By increasing the number of nodes and using the same overall number of processing cores, time is reduced. The exact reasons for this are unknown because it is hard to isolate individual contributions, but it is possible that a number of factors are contributing, ranging from the bandwidth of the processor bus to the memory speed, the memory usage, data memory locality, as well as the latencies and bandwidth of the interconnect. However, there are still speed increases by using all cores on the node compared to using fewer, albeit with a lower efficiency, which in practice is useful.

From this analysis, it can also be seen that the critical component is the interconnect, which is consistent with the coupled nature of the problem and the communication that is required for each solver iteration. The Merlin system [Figs. 7(c) and 7(d)] has a substantially faster interconnect, with up to 20 times more bandwidth and for this application achieves a speed increase of 5×, with 32 processing cores for a 350,000 finite-element simulation, although this is likely to increase if the number of finite elements increases. In comparison, the 100,000 finite-element simulation achieved a 4× speed-up for 32 processing cores. Care must be taken when running analyses that the interconnect is able to provide message passing speeds complementary to processing speeds and problem type to enable speed-up advantages.

Conclusions

The work presented in this paper confirms that HPC, and in particular, parallel processing, is a valuable tool in computational analysis. A method of reducing the time taken for large-scale, fully coupled thermo-hydro-mechanical simulations, which can be significant, has been presented. Results from the preplacement phase of the Prototype Repository project undertaken by SKB in Äspö, Sweden, have been presented and demonstrate the need for field-scale three-dimensional analyses.

Initially, the performance of an existing parallel algorithm was tested on an Intel Woodcrest processor (3 GHz)–based parallel computer and compared to results for a MIPS (600 MHz) processor-based parallel computer. The parallel computer using the Intel Woodcrest processors shows a 10.7× real time performance increase for eight processors on each computer, showing the transient nature of computation technology. The parallel computer using the MIPS (600 MHz) processors shows good speed-up results, 3.6 times over eight processors, whereas the parallel computer using Intel Woodcrest (3 GHz) processors shows 2.2 times speed-up over eight processors. On both computers, the system-matrix build component of the algorithm shows good parallel performance, as did the iterative BiCG solver on the slower MIPS computer. However, solver performance on the Intel Woodcrest–based parallel computer was poor, with only a 1.2× speed increase on eight cores, owing to a comparatively large communication overhead.

A new parallel implementation of the BiCG solver was also presented. In particular, multilevel parallelism was used, reflecting the use of current hardware technology. Both multithreading on a single node and message passing across nodes have been implemented. Speed-ups of up to 6× have been achieved for the solver,

Int. J. Geomech. 2011.11:90-98.

although care must be taken when selecting the hardware architecture and configuration of the HPC system. For this application, the bandwidth and latencies of the interconnect relative to the processing speed must be considered. This speed-up can be used to inform and enable field-scale simulations of large-scale problems such as nuclear waste disposal repositories.

Acknowledgments

Support for the first writer from an Engineering and Physical Sciences Research Council (EPSRC) studentship and funding by the European Commission (EC) via the Prototype Repository Project (FIKW-2000-00055), along with access to SKB's high quality data set related to the Prototype Repository project are gratefully acknowledged. Support and use of the computing facilities both at the High-Performance Computing Collaboratory (HPC^2) at Mississippi State University and at Advanced Research Computing @ Cardiff (ARCCA) at Cardiff University are also appreciatively acknowledged. The writers would also like to thank the National Science Foundation for its partial support of this work through the grants NSF EPS #0556308 and NSF IIP #1034897.

References

Asanovic, A., et al. (2006). *The landscape of parallel computing research: A view from Berkeley*, Univ. of California, Berkeley, CA.

Barrett, R., et al. (1995). *Templates for the solution of linear systems: Building blocks for iterative methods*, Wiley, New York.

Cleall, P. J., Melhuish, T. A., and Thomas, H. R. (2006a). "Modeling the three-dimensional behavior of a prototype nuclear waste repository." *Eng. Geol.*, 85(1–2), 212–220.

Cleall, P. J., Thomas, H. R., Melhuish, T. A., and Owen, D. H. (2006b). "Use of parallel computing and visualization techniques in the simulation of large scale geoenvironmental engineering problems." *Future Gener. Comput. Syst.*, 22(4), 460–467.

Dahlström, L-O. (1998). "Äspö HRL—Test plan for the prototype repository." *Progress Rep. No. HRL-98-24*, Swedish Nuclear Fuel and Waste Management (SKB), Stockholm, Sweden.

Duff, I. S., and van der Vorst, H. A. (1999). "Developments and trends in the parallel solution of linear systems." *Technical Rep. TR/PA/99/10*, Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique (CERFACS) (European Centre for Research and Advanced Training in Scientific Computation), Toulouse, France.

Flynn, M. (1972). "Some computer organizations and their effectiveness." *IEEE Trans. Comput.*, C-21(9), 948–960.

Goudarzi, R., and Johannesson, L.-E. (2006). "Äspö Hard Rock Laboratory, Prototype Repository—Sensor data report (Period 010917-061201)." *16, IPR-07-05*, Swedish Nuclear Fuel and Waste Management (SKB), Stockholm, Sweden.

Hennessy, J., and Patterson, D. (2007). *Computer architecture: A quantitative approach*, 4th Ed., Morgan Kauffman, San Francisco.

Hockney, R. (1994). "The communication challenge for MPP: Intel Paragon and Meiko CS-2." *Parallel Comput.*, 20(3), 389–398.

Intel. (2006). "White paper—Intel architecture and silicon cadence. The catalyst for industry innovation."

Jiayin, M, Bo, S., Yongwei, W., and Guangwen, Y. (2006). "Overlapping communication and computation in MPI by multithreading." *Proc., 2006 Int. Conf. on Parallel & Distributed Processing Techniques and Applications (PDPTA'06)*, CSREA Press, Bogart, GA, 52–57.

Johannesson, L.-E., Börgesson, L., Goudarzi, R., Sandén, T., Gunnarsson, D., and Svemar, C. (2007). "Prototype Repository: A full scale experiment at Äspö HRL." *Phys. Chem. Earth*, 32(1–7), 58–76.

Neretnieks, I. (1993). "Solute transport in fractured rock—applications to radionuclide waste repositories." *Flow and contaminant transport in fractured rock*, J. Bear, C.-F. Tsang, and G. de Marsily, eds., Academic Press, San Diego, 39–127.

Owen, D. H. (2000). "Preconditioned parallel iterative solution methods for coupled finite element analyses." Ph.D. thesis, Univ. of Wales, Cardiff, UK.

Roosta, S. H. (2000). "Parallel processing and parallel algorithms." *Theory and computation*, Springer-Verlag, New York.

Smith, I. M., and Griffiths, D. V. (2004). *Programming the finite element method*, 4th Ed., Wiley, Chichester, UK.

Svemar, C., and Pusch, R. (2000). "Äspö Hard Rock Laboratory." *Prototype Repository, project description*, *FIKW-CT-2000-00055, IPR-00-30*, Swedish Nuclear Fuel and Waste Management (SKB), Stockholm, Sweden.

Thomas, H. R., Cleall, P. J., Chandler, N., Dixon, D., and Mitchell, H. P. (2003a). "Water infiltration into a large scale in situ experiment in an underground research laboratory—Physical measurements and numerical simulation." *Géotechnique*, 53(2), 207–224.

Thomas, H. R., Cleall, P. J., Dixon, D., and Mitchell, H. P. (2009). "The coupled thermal-hydraulic-mechanical behavior of a large scale in situ heating experiment." *Géotechnique*, 59(4), 401–413.

Thomas, H. R., and He, Y. (1997). "A coupled heat-moisture transfer theory for deformable unsaturated soil and its algorithmic implementation." *Int. J. Numer. Methods Eng.*, 40(18), 3421–3441.

Thomas, H. R., Yang, H. T., He, Y., and Cleall, P. J. (2003b). "A multilevel parallelized substructuring frontal solution for coupled thermo/hydro/mechanical problems in unsaturated soil." *Int. J. Numer. Anal. Meth. Geomech.*, 27(11), 951–965.

van de Steen, A. J., and Dongarra, J. J. (1996). "Overview of recent supercomputers." *Technical report, Dept. of Computer Science*, 6th Ed., Univ. of Tennessee, Knoxville, TN.

White, J. B., and Bova, S. W. (1999). "Where's the overlap? An analysis of popular MPI implementations." *Proc., Third MPI Developer's and User's Conf., MPIDC '99*, MPI Software Technology Press, Starkville, MS.

Int. J. Geomech. 2011.11:90-98.