Parallel Processing of a Groundwater Contaminant Code

R. C. Arnett
L. E. Greenwade
Idaho National Engineering and Environmental Laboratory
P. O. Box 1625
Idaho Falls, Idaho USA 83415

Introduction and Problem Description

The U. S. Department of Energy's Idaho National Engineering and Environmental Laboratory (INEEL) is conducting a field test of experimental enhanced bioremediation of trichoroethylene (TCE) contaminated groundwater. TCE is a chlorinated organic substance that was used as a solvent in the early years of the INEEL and disposed in some cases to the aquifer. There is an effort underway to enhance the natural bioremediation of TCE by adding a non-toxic substance that serves as a feed material for the bacteria that can biologically degrade the TCE.

Figure 1 presents a planar view of the contaminated site including selected wells. Phase 1 of the field test lasted 40 weeks and the injection frequency of the organic feed material (electron donor) for the enhanced bioremediation of TCE was 1-2 times a week for the entire duration

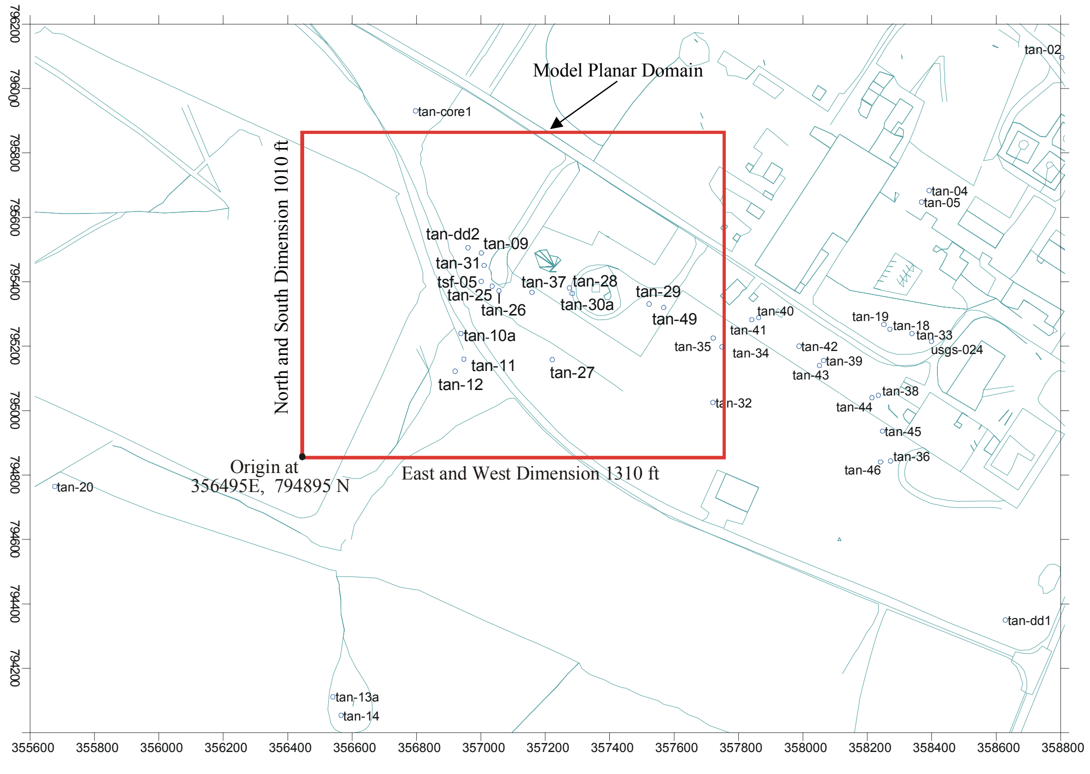

Figure 1. Map of Contaminated Area

Work supported by the U.S. Department of Energy, Assistant Secretary for Environmental Management, under DOE Idaho Operations Office Contract DE-AC07-99ID13727.

Groundwater Modeling

A groundwater model was prepared to simulate the process, calculate the mass of TCE degraded, and provide assistance to the design of a proposed full-scale bioremediation. The model grid was 78 x 108 x 5 cells and included some grid refinement around selected wells. Figure 2 is a planar view of the model grid. There were 5 high flow or aquifer type zones and the wells were completed in different zones. This required a minimum of 5 model layers with vertical leakage between them. Figure 3 is a cross-sectional view of the grid with selected wells and their completion intervals.

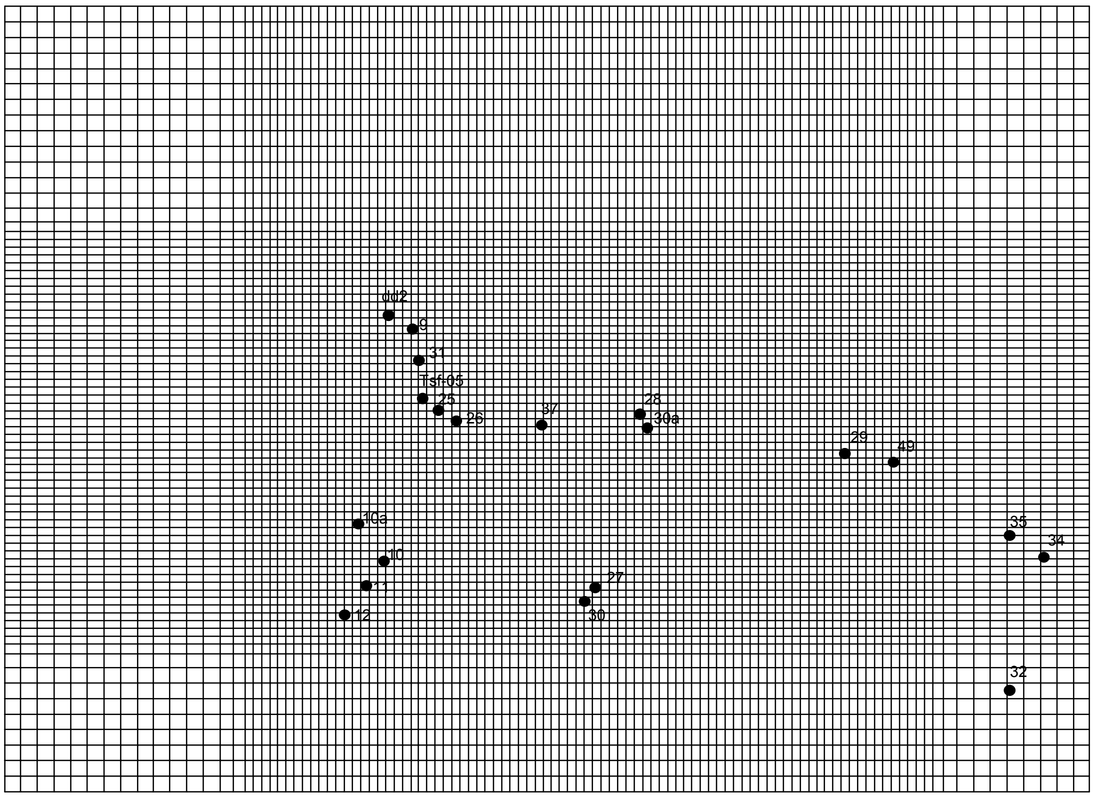

Figure 2. Planar view of model grid with well locations

Two groundwater simulation codes were used. Groundwater flow was simulated with MODFLOW (McDonald and Harbaugh, 1988), a widely used modular, three-dimensional, finite-difference computer model for simulating groundwater flow. It was developed and maintained by the U.S. Geological Survey (USGS). The MODFLOW code uses a block-centered finite-difference numerical approach to solve steady-state or transient problems. Layers can be simulated as confined, unconfined, or a combination of the two. Flows resulting from external stresses such as wells, rivers, drains, areal recharge, and evapotranspiration can also be simulated. The finite-difference equations are solved using a variety of optional solvers including preconditioned conjugate gradient, strongly implicit procedure (SIP), and a slice-successive overrelaxation (SSOR) procedure. MODFLOW was used to simulate the groundwater flow system and the flow field is one of the inputs needed by transport code described below.

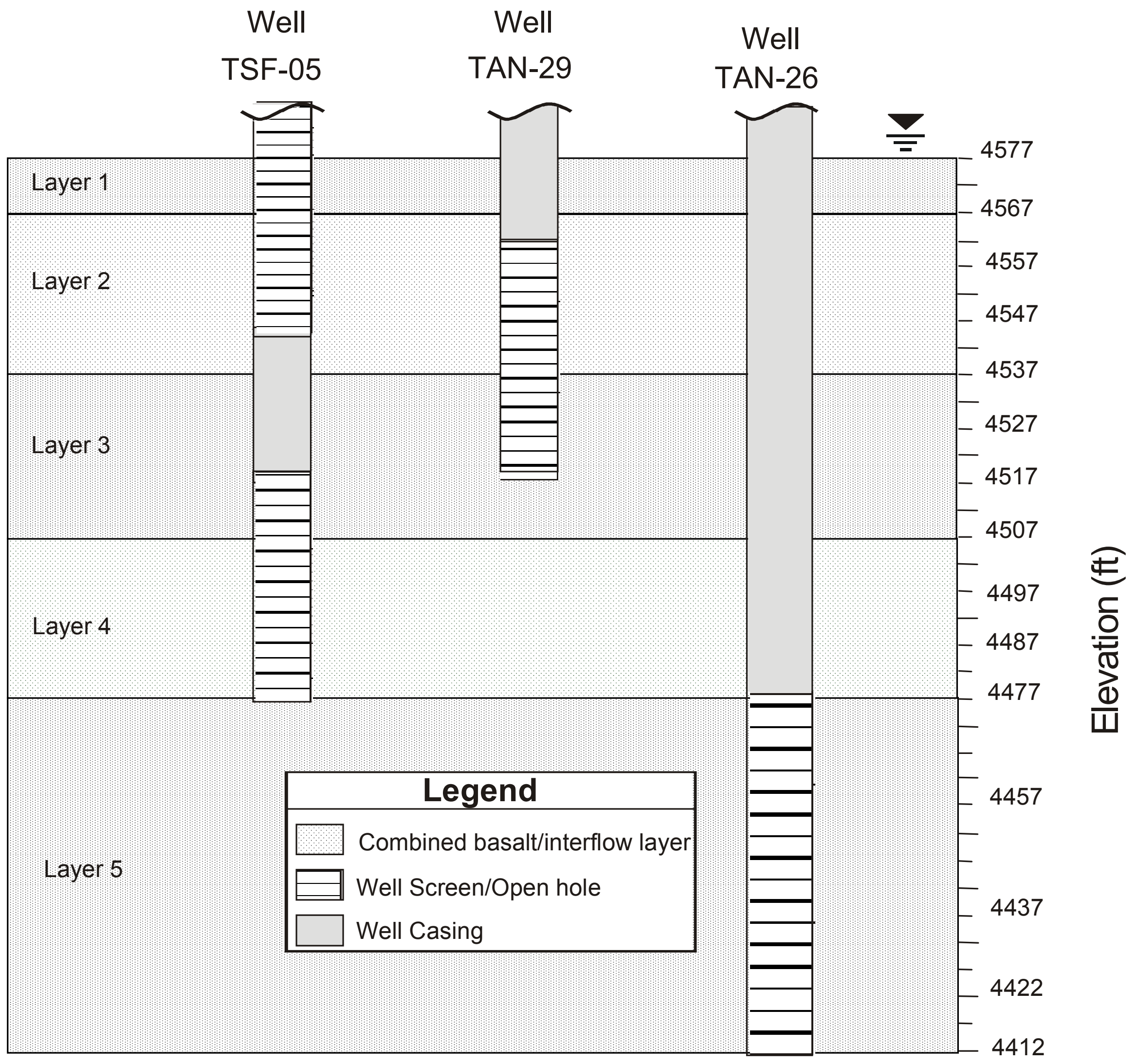

Figure 3. Selected Well Completions

Contaminant transport within the flow field was simulated with the MT3DMS code. MT3DMS is a modular three-dimensional multispecies transport model for simulating advection, dispersion, and sorption or simple first-order reactions (Zheng and Wang, 1999). MT3DMS can be used to simulate changes in concentrations of miscible contaminants in groundwater considering advection, dispersion, diffusion and some basic chemical reactions, with various types of boundary conditions and external sources or sinks. The basic chemical reactions included in the model are equilibrium-controlled or rate-limited linear or non-linear sorption, and first-order irreversible or reversible kinetic reactions. MT3DMS is designed for use with any block-centered finite-difference flow model, such as MODFLOW, assuming constant fluid density and full saturation.

MT3DMS is implemented with an optional, dual-domain formulation for modeling mass transport. With this formulation, the porous medium is regarded as consisting of two distinct domains, a mobile domain where transport is predominately by advection and an immobile domain where transport is predominately by molecular diffusion. Instead of a single "effective" porosity for each model cell, two porosities, one for the mobile domain and the other for the immobile domain, are used to characterize the porous medium. The exchange between the mobile and immobile domains is specified by a mass transfer coefficient. The dual-domain advective-diffusive model may be more appropriate for modeling transport in fractured media or extremely heterogeneous porous media than the single porosity advective-dispersive model, provided that the porosities and mass transfer coefficients can be properly characterized. This dual-domain capability was used to simulate transfer between an immobile phase (liquid) and dissolved phase TCE.

There are approximately 30,000 active nodes in the model. Time step size was controlled by injection frequency and a rather high flow velocity. There were 168 stress periods (periods during which the external stress [pumping] changed or transitioned between changes) in the flow model and multiple flow model time steps for each stress period. The transport code read the head and flux file created by the flow model and computed the contaminant transport including dispersion and decay. There were many transport steps for each flow time step. An optional implicit solver in MT3DMS allows a Courant number (number of grid cells a particle of water can travel in a given transport step) greater than one, but mass balance analysis indicated the maximum value of the Courant number should not be greater than two. This limited the maximum transport step size to 4×10^{-3} days; transport step sizes during and immediately after injection were much smaller.

All this caused a considerable computational burden. Simulation times were very long for a groundwater problem.

Original Performance

The computational time for the flow simulation was moderate, on the order of ½ hour. The computational time for the contaminant transport was considerable. The total wall clock time of the MT3DMS transport code for a complete simulation was about 5 1/2 days for a single species and more than 10 days for two species.

A process known as model calibration or history matching was used to estimate unknown or uncertain model parameter values. In model calibration, the results of model run are compared to field measured concentrations and then some of the model parameter values are changed to improve the match. An acceptable match implies a good set of parameter values. The implications of the long run times for the project schedule and budget were severe since the results of a previous run were needed to determine changes in parameter values for the next run. In the early stages of model calibration, it was not necessary to complete a simulation in order to obtain sufficient results to make parameter changes for the next run. Hence, a run could sometimes be terminated prior to completion. Still, 1-2 wall clock days were needed to get the minimum information to make a reasonable estimate of needed changes. Because many calibration runs were needed, the projected calibration time was estimated to cover 8-10 months or more, a period that was not compatible with the project schedule or the budget.

Performance Analysis

The code's performance was analyzed to determine ways of shortening the run time. The code provides several optional solvers. The subsurface was a fractured media and resulted in relatively high velocities, which limited the options. More than an incremental speed-up in the code was needed.

The code was originally run on a high-end NT workstation, but was transferred to a Silicon Graphics (SGI) Origin 2000 shared memory computer with 24 processors. Because of the need to considerably speed up the code and the availability of the shared memory, multi-processing computer, a quick investigation of the feasibility of parallel processing was done. This started with profiling the code to determine the portions performing the bulk of the computing and if those portions were small enough to warrant modifying the code to run in parallel on multiple processors. It was realized that Amdahl's law applied and that about 80-90% or more of the computations should be limited to a small part of the code to make parallel processing feasible.

An SGI integrated performance tool package called "speedshop" was used to run performance experiments on the executable and produce results that allowed the user to analyze code performance. One of the speedshop options, called "usertime," returns CPU time or the time the program is actually running plus the time the operating system is performing services for the program. The display generated by the "prof" utility from the speedshop output breaks the program time down into the time used by each function in the program. Speedshop uses statistical callstack profiling, based on CPU time, with a default sample interval of 30 milliseconds.

The prof results showed that 94.6% of the CPU time was spent in an advective forward particle tracking routine and a Runga Kutta subroutine called from a loop within the tracking routine. Since 80-90 % of the CPU time was spent in a limited part of the code, the particle tracking routine and its Runga Kutta subroutine were examined to determine the feasibility of modifying them to run in parallel and obtain a large code speed-up.

Parallel Processing Performance

A PARALLEL DO and other OPENMP directives were added to the major particle loop of the particle tracking routine. The Runga Kutta subroutine was called within this loop so it was included in the parallelization without specific modifications to the subroutine. The results from the parallel code compared quite closely to those of the original code.

The full simulation wall clock time was reduced from 5 ½ days to about 26 hours with 10 processors. This allowed a reasonable schedule for calibrating the model without oversimplification.

Planned Future Work and Improvements

Parallel processing a groundwater transport code has shown very substantial benefits and indicates the value of further investment in this area. At the same time, the code parallelization for this project was unplanned, need driven, and performed with a narrow focus. There was an immediate problem that had to be solved under crisis conditions. Only that part of the code used for this project was evaluated and parallelized. Other optional solvers that might be more appropriate for other problems were not examined or parallelized. The original code was not tuned to streamline it and eliminate computational inefficiencies prior to the parallelization.
Our experience with this effort indicated that using two processors reduced run time to about 70% of the original code for an efficiency factor of about 70%. Using ten processors resulted in an efficiency rating of about 45%. It would be very desirable and should be possible to tune the code in order to raise the efficiency rating quite substantially.

In addition, the number of processors available on the shared memory computer limited the total speed up. There were times when insufficient processors were available for this simulation because of the needs of other users. Requesting 40% of a major resource (10 of the total 24 processors) became an issue to the user community and at times degraded the total system performance significantly.

There are other available computing resources that could provide relief. One attractive alternative to a shared memory computer are clusters of individual processors, each with it's own memory. One such available cluster has 40 individual CPU's all connected together on a rack. Other remote, clustered CPU's could be connected via a network. This approach offers flexibility in the number and scheduling of CPU's. However, use of clustered CPU's would require implementation of the Message Passing Interface (MPI) protocol whereby information and data are passed via software calls between the various CPU's working in parallel on the simulation. To taking full advantage of a variety of available hardware, it would be necessary to seriously evaluate MPI as an alternative method of implementing parallel processing.

We have performed the preliminary modeling using the multi-species MT3DMS groundwater transport code. This code does not consider reaction among species such as competing electron donors (oxygen, nitrate, sulfate, and TCE) or production of different degradation species (DCE, vinyl chloride, ethene, etc.) A code called RT3D, built on MT3DMS, has the capability to simulate such reactions in addition to groundwater transport of each species. There is a desire to consider some of the most important reactions and extensive field data of various competing electron donors and degradation products are available. The added computational burden of additional reacting species to the TAN ISB model is substantial and parallel processing is the only feasible solution. Plans are underway to fully update, tune, and carefully and systematically parallelize the RT3D code.

References

McDonald, M. G. and A. W. Harbaugh, 1988, *A Modular Three-Dimensional Finite-Difference Groundwater Flow Model*, In: Techniques of Water-Resources Investigations of the United States Geological Survey, Book 6, Modeling Techniques, U. S. Geological Survey, Federal Center, Box 25425, Denver, CO 80225.

Zheng, C and P. P. Wang, 1999, *MT3DMS A Modular Three-Dimensional Multispecies Transport Model for Simulation of Advection, Dispersion and Chemical Reactions of Contaminants in Groundwater Systems*, SERDP-99, revised November 1999, prepared for the U. S. Army Corps of Engineers by the Department of Geological Sciences, University of Alabama, 202 Bevill Research Building, Tuscaloosa, Alabama 35487, USA