United States
Environmental Protection
Agency

National Risk Management
Research Laboratory
Ada, OK 74820

Research and Development | EPA/600/SR-97/007 | February 1997

Project Summary

Ground-Water Model Testing: Systematic Evaluation and Testing of Code Functionality and Performance

Paul K. M. van der Heijde and D. A. Kanzer

Effective use of ground-water simulation codes as management decision tools requires the establishment of their functionality, performance characteristics, and applicability to the problems at hand. This is accomplished through systematic code-testing protocol and code selection strategy. The protocol contains two main elements: functionality analysis and performance evaluation. Functionality analysis is the description and measurement of the capabilities of a simulation code; performance evaluation concerns the appraisal of a code's operational characteristics (e.g., computational accuracy and efficiency, sensitivity for problem design and parameter selection, and reproducibility).

Testing of qround-water simulation codes may take the form of (1) benchmarking with known, independently derived analytical solutions; (2) intracomparison using different code functions inciting the same system responses; (3) intercomparison with comparable simulation codes; or (4) comparison with field or laboratory experiments. The results of the various tests are analyzed using standardized statistical and graphical techniques to identify performance strengths and weaknesses of code and testing procedures. The protocol is demonstrated and evaluated using a three-dimensional finite difference flow and solute transport simulation code, FTWORK.

Introduction

Ground-water modeling has become an important methodology in support of the planning and decision-making processes involved in ground-water resources development, ground-water protection, and aquifer restoration. In ground-water modeling, it is crucial that the code's credibility is established and its suitability is determined. This is conducted through systematic evaluation of code's correctness, performance, sensitivity to input uncertainty, and applicability to typical field problems. Such a systematic approach is referred to as "*code-testing and evaluation protocol*." Without subjecting a ground-water simulation code to such systematic testing and evaluation, results obtained with the code may suffer from low levels of confidence. Acceptance of a modeling code depends not only on a series of successful tests, but also a history of successful applications to a variety of site conditions and management problems.

Reviewing the existing literature indicates that previous code-testing studies appear to be (1) lacking in systematically addressing code features and providing insight in the completeness and effectiveness of the performed testing, and (2) inconsistent and incomplete for documentation describing the code's functions and features. A new code-testing protocol known as *functionality analysis, performance evaluation and applicability assessment protocol* (van der

Heijde et al., 1993) is presented to address these deficiencies.

The report begins with a review of existing code-testing literature. The formulation of a comprehensive code-testing protocol is developed. Testing strategies are presented using various graphical and statistical tools. The protocol is then demonstrated using a numerical code, FTWORK (Faust et al., 1990), which is designed to simulate three-dimensional flow and solute transport in the saturated zone of the subsurface.

Code Testing

A systematic approach to code testing combines elements of error-detection, evaluation of the operational characteristics of the code, and assessment of its suitability to solve certain types of management problems, with well-designed test problems, relevant test data sets, and informative performance measures.

The code-testing protocol described in the report is applied in a step-wise fashion (Table 1). First, the code is analyzed with respect to its simulation functions and operational characteristics. Potential code performance issues are identified, based on analysis of simulated processes, mathematical solution methods, computer limitations and execution environment. This is followed by the formulation of a test strategy, consisting of design or selection of relevant test problems. The set of test problems is chosen such that all code functions and features of concern are addressed. Results of the testing are documented in tables and matrices providing an overview of the completeness of the testing in various types of informative graphs, and with a set of statistical measures. The actual testing may take the form of

1) benchmarking using known, independent derived analytical solutions,
2) intracomparison using different code functions inciting the same system responses;
3) intercomparison with comparable simulation codes; or
4) comparison with field or laboratory experiments.

It is important that each test be documented with respect to test objectives, model setup for both the tested code and the benchmark, if applicable (structure, discretization, parameters), and results for each test (for both the tested code and the benchmark).

Functionality of a ground-water modeling code is defined as the set of functions and features which the code offers the user in terms of model framework geometry, simulated processes, boundary conditions, and analytical and operational capacities. The code's functionality must be defined in sufficient detail for potential users to assess the code's utility, as well as to enable the code developer to design a meaningful code-testing strategy. *Functionality analysis* involves the identification and description of the code's functions, and the subsequent evaluation of each code function or group of functions for conceptual correctness and error-free operation. The information generated by functionality analysis is organized into a summary structure, or matrix that brings together the description of code functionality, code-evaluation status, and appropriate test problems. This *functionality matrix* is formulated by combining a complete description of the code functions and features with the objectives of the test cases. The functionality matrix illustrates the extent of the functionality analysis.

Performance evaluation is aimed at characterizing the operational characteristics of the code in terms of:

1) *computational accuracy* (e.g., in comparison with a benchmark);
2) *reliability* (e.g., reproducibility of results, convergence and stability of solution algorithms, and absence of terminal failures);
3) *sensitivity* for grid orientation and resolution, time discretization, and model parameters;
4) *efficiency* of coded algorithms (in terms of numerical accuracy versus code execution time, and memory and mass storage requirements); and
5) *resources required* for model setup and analysis (e.g., input preparation time, effort needed for graphical representation of simulation output).

Results of the performance evaluation are reported both quantitatively and qualitatively in checklists and in tabular form. Reporting on performance evaluation should provide potential users information on the performance as a function of problem complexity and setup, selection of simulation control parameters, and spatial and temporal discretization. The functionality matrix and performance tables, together with the supporting test results and comments, should provide the information needed to select a code for a site-specific application and to evaluate the appropriateness of a code used at a particular site.

Testing Strategy

Comprehensive testing of a code's functionality and performance is accomplished through a variety of test methods. Determining the importance of the tested functions and the ratio of tested versus non-tested functions provides an indication of the completeness of the testing. Based on the analysis of functionality and performance issues, a code-testing strategy is developed. Such a code-testing strategy should consist of:

1) formulation of test objectives (as related to code functionality and performance issues), and of test priorities (Table 2);
2) selection and/or design of test problems and determination of type and extent of testing for selected code functions;
3) determination of level of effort to be spent on sensitivity analysis for each test problem;
4) selection of the qualitative and quantitative measures to be used in the evaluation of the code's performance; and
5) determination of the level of detail to be included in the test report and the format of reporting.

The test procedure includes three levels of testing (van der Heijde and Elnawawy, 1992). At Level I, a code is tested for correctness of coded algorithms, code logic and programming errors by: (1) conducting step-by-step numerical walk-throughs of the complete code or through selected parts of the code; (2) performing simple, conceptual or intuitive tests aimed at specific code functions; and (3) comparing with independent, accurate benchmarks (e.g., analytical solutions). If the benchmark computations themselves have been made using a computer code, this computer code should be, in turn, subjected to rigorous testing by comparing computed results with independently derived and published data.

At Level II, a code is tested to: (1) evaluate functions not addressed at Level I; and (2) evaluate potentially problematic combinations of functions. At this level, code-testing is performed by intracomparison (i.e., comparison between runs with the same code using different functions to represent a particular feature), and intercomparison (i.e., comparison between different codes simulating the same problem). Typically, synthetic data sets are used representing hypothetical, often simplified ground-water systems.

At Level III, a code (and its underlying theoretical framework) is tested to determine how well a model's theoretical foundation and computer implementation describes actual system behavior, and to demonstrate a code's applicability to representative field problems. At this level, testing is performed by simulating a field or laboratory experiment and comparing the calculated and independently observed cause-and-effect responses. Because measured values of model input, system parameters and system responses are samples of the real system, they inherently incorporate measurement errors, are subject to uncertainty, and may suffer from interpretive bias. Therefore, this type of testing always retains an element of incompleteness and subjectivity.

The test strategy requires that Level I testing is conducted (often during code development), and, if successfully completed, is followed by Level II testing. The code may gain further credibility and user confidence by subjecting it to Level III testing (i.e., field or laboratory testing). Ideally, code testing should be performed for the full range of parameters and stresses the code is designed to simulate, in practice this is often not feasible due to budget and time constraints. Therefore, prospective code users need to assess whether the documented tests adequately address the conditions expected in the target application(s). If previous testing has not been sufficient in this respect, additional testing may be necessary.

Evaluation Measures

Evaluation of code-testing results should be based on: (1) visual inspection of the graphical representation of variables computed with the numerical model and its benchmark; and (2) quantitative measures of the goodness-of-fit. Such quantitative measures or *evaluation* or *performance criteria*, characterize the differences between the results derived with the simulation code and the benchmark, or between the results obtained with two comparable simulation codes.

Graphical measures are especially significant to obtain a first, qualitative impression of test results, and to evaluate test results that do not lend themselves to statistical analysis. For example, graphical representation of solution convergence characteristics may indicate numerical oscillations and instabilities in the iteration process. Practical considerations may prevent the use of all data-pairs in the generation of graphical measures. Thus, a subset of data-pairs may be selected for use with graphical measures. There are five types of graphical evaluation techniques particularly suited: (1) X-Y plots or line graphs of spatial or temporal behavior of variables; (2) one-dimensional column plots or histograms (for test deviations); (3) combined plots of line graphs and column plots of deviations; (4) contour and surface plots; and (5) three-dimensional column plots or histograms. The conclusions from visual inspection of graphic representations of testing results may be described qualitatively (and subjectively) by such attributes as "poor," "reasonable," "good," and "very good."

There are three general procedures, coupled with standard linear regression statistics and estimation of error statistics, to provide quantitative goodness-of-fit measures (Donigian and Rao, 1986): (1) *paired-data performance* -- the comparison of simulated and observed data in time and space; (2) *time and space integrated, paired-data performance* -- the comparison of spatially and temporally integrated or averaged simulated and observed data; and (3) *frequency domain performance* -- the comparison of simulated and observed frequency distributions. The organization and evaluation of code intercomparison results can be cumbersome due to the potentially large number of data-pairs involved if every computational node is included in the analysis. This can be mitigated by analyzing smaller, representative sub-samples of the full set of model domain data-pairs. The representativeness of the selected data-pairs is often a subjective judgment. For example, in simulating one-dimensional, uniform flow, the data pairs should be located at least on two lines parallel to the flow direction, one in the center of the model domain and one at the edge.

Useful quantitative evaluation measures for code-testing include: (1) *Mean Error* (ME), defined as the mean difference (i.e., deviation) between the model calculated values versus the benchmark values; (2) *Mean Absolute Error* (MAE), defined as the average of the absolute values of the deviations; (3) *Positive Mean Error* (PME) and *Negative Mean Error* (NME), defined as the ME for the positive deviations and negative deviations, respectively; (4) *Mean Error Ratio* (MER), a composite measure indicating systematic overprediction/ underprediction by the code; (5) *Maximum Positive Error* (MPE) and *Maximum Negative Error* (MNE), defined as the maximum positive and negative deviations, respectively, indicating potential inconsistencies or sensitive model behavior; and (6) *Root Mean Squared Error* (RMSE), defined as the square root of the average of the squared differences between the model calculated values and its benchmark equivalents.

Various computed variables may be the focus of graphic or statistical comparison, including hydraulic heads (in space and time), head gradients, global water balance, internal and boundary fluxes, velocities (direction and magnitude), flow path lines, capture zones, travel times, and location of free surfaces and seepage surfaces, concentrations, mass fluxes, and breakthrough curves at observation points and sinks (wells, streams).

Code-testing Protocol Demonstration

The code-testing and evaluation protocol is applied to a block-centered finite-difference simulation code, FTWORK (Faust et al., 1990), which was designed to simulate transient and steady-state three-dimensional saturated ground-water flow and transient transport of a single dissolved component under confined and unconfined conditions. To demonstrate the use of the code-testing protocol, the following steps have been taken, featuring the FTWORK code: (1) identifying and examining code functionality; (2) determining type and objectives of tests performed and documented by the code developers; (3) evaluating the suitability of performed tests for use in protocol demonstration; (4) compiling protocol summary structure (*i.e.,* checklists) using performed tests; (5) designing and conducting new tests, based on deficiencies in performed tests; and (6) summarizing the combined results of tests performed by code developers and tests performed as part of the protocol demonstration.

Most of the tests originally performed by the developers were adapted, augmented, and reanalyzed to ensure consistency with the protocol. Additional tests were designed and executed to evaluate capabilities and characteristics of the FTWORK code, not addressed in the FTWORK documentation.

Discussion and Conclusions

Historically, reporting on simulation code-testing has been limited to the use of author-selected verification problems. Few studies have focused on author-independent evaluation of a code, or at code intercomparison. Main deficiencies in reported code-testing efforts include

incompleteness of the performed testing, absence of discussion regarding tested code functions as compared with available code functions and features, and lack of detail in test problem implementation. This makes it difficult to recreate the data sets for additional analysis. The protocol presented in this report aims to address these issues. In addition, the protocol covers many other test issues, ranging from performance and resource utilization to usefulness as a decision-making support tool.

The code-testing protocol is designed to be applicable to all types of simulation codes dealing with fluid flow transport phenomena in the unsaturated and saturated zones of the subsurface. Selection and implementation of test problems will differ for the different types of codes. However, evaluation techniques are in principle independent of code type. Test results are presented in a form that is unbiased by the requirements posed by specific applications. It aims to provide enough detail to establish confidence in the code's capabilities and to efficiently determine its applicability to specific field problems. Because users of code-testing results may differ in terms of objectives, the protocol leaves it to the users to determine if a tested code is suitable to their needs.

The most critical element of the code-testing protocol is the functionality analysis (including elements of what is often called "code verification"). Many different test configurations can be used and, for some code types, a large number of benchmark solutions may be available. For other code types, intercomparison may be the only available option. Selection of benchmarks and design of test problems should be guided by test objectives and in the context of the completeness of the testing exercise. Protocol tools such as functionality tables and functionality matrices are effective aids in the design of test problems. Well-designed tests not only identify code functionality problems, but should also provide important information on correct implementation of code features. Functionality analysis may be limited because not all code features can be adequately addressed using benchmark solutions. Often, code intracomparison, code intercomparison and conceptual testing are required, resulting in a more subjective assessment of code accuracy and operational constraints.

The functionality analysis, performance evaluation and applicability assessment protocol, presented in the full report, provides a comprehensive framework for systematic and in-depth evaluation of a variety of ground-water simulation codes. While allowing flexibility in implementation, it secures, if properly applied, addressing all potential coding problems. It should be noted that the protocol does not replace scientific review nor the use of sound programming principles. Most effectively, the code-testing under the protocol should be performed as part of the code development process. Additional testing in accordance with the protocol may be performed under direction of regulatory agencies, or by end-users. If properly documented, code-testing in accordance with the protocol supports effective independent review and assessment for application suitability. As such, the protocol contributes significantly to improve quality assurance in code development and use in ground-water modeling.

REFERENCES

Donigian, Jr., A.S. and Rao, P.S.C. 1986. Example model testing studies. In: *Vadose Zone Modeling of Organic Pollutants* (eds. S.C. Hern and S.M. Melancon), Lewis Publishers, Chelsea, Michigan.

Faust, C.R., P.N. Sims, C.P. Spalding, P.F. Anderson, and D.E. Stephenson. 1990. FTWORK: A three-dimensional groundwater flow and solute transport code. WRSC-RP-89-1085. Westinghouse Savannah River Company, Aiken, South Carolina.

van der Heijde, P.K.M., and O.A. Elnawawy. 1992. Quality Assurance and Quality Control in the Development and Application of Ground-Water Models. EPA 600/R-93/011, Office of Research and Development, U.S. Environmental Protection Agency, Washington, D.C.

van der Heijde, P.K.M., S.S. Paschke, and D.A. Kanzer. 1993. Ground-water flow and solute transport model functionality testing and performance evaluation. In: H.J. Morel-Seytoux (ed.), Proc. Thirteenth AGU Hydrology Days, Fort Collins, Colorado, pp. 378-389.

Table 1. Procedures of code-testing and evaluation protocol.

CODE TESTING AND EVALUATION PROTOCOL	
Step 1	analyze the code documentation with respect to simulation functions, operational features, mathematical framework, and software implementation;
Step 2	identify code performance issues based on understanding of simulated processes, mathematical methods, computer limitations, and software environment;
Step 3	develop testing strategy that addresses relevant code functionality and performance issues including selection and/or design of test problems and determination of appropriate evaluation measures;
Step 4	execute test problems and analyze results using standard graphic and statistical evaluation techniques;
Step 5	collect code performance issues and code test problems in overview tables and display matrices reflecting correctness, accuracy, efficiency, and field applicability;
Step 6	identify performance strengths and weakness of code and testing procedure;
Step 7	document each test setup and results in report form and as electronic files (text, data, results, graphics); and
Step 8	communicate results (e.g., prepare executive summary, overview report, etc.).

Table 2. Major test issues for three-dimensional finite-difference saturated ground-water flow and solute transport codes.

General Features
- mass balances (regular versus irregular grid)
- variable grid (consistency in parameter and stress allocation)

Hydrogeologic Zoning, Parameterization, and Flow Characteristics
- aquifer pinchout, aquitard pinchout
- variable thickness layers
- storativity conversion in space and time (confined-unconfined)
- anisotropy
- unconfined conditions
- dewatering
- sharp contrast in hydraulic conductivity

Boundary Conditions for Flow
- default no-flow assumption
- areal recharge in top active cells
- induced infiltration from streams (leaky boundary) with potential for dewatering below the base of the semi-pervious boundary
- drain boundary
- prescribed fluid flux
- irregular geometry and internal no-flow regions

Transport and Fate Processes
- hydrodynamic dispersion (longitudinal and transverse)
- advection-dominated transport
- retardation (linear and Freundlich)
- decay (zero and first-order)
- spatial variability of dispersivity
- effect of presence or absence cross-term for dispersivity

Boundary Conditions for Solute Transport
- default zero solute-flux assumption
- prescribed solute flux
- prescribed concentration on stream boundaries
- irregular geometry and internal zero-transport zones
- concentration-dependent solute flux into streams

Sources and Sinks
- effects of time-varying discharging and recharging wells on flow
- multi-aquifer screened wells
- solute injection well with prescribed concentration (constant and time-varying flow rate)
- solute extraction well with ambient concentration

Paul K. van der Heijde and David A. Kanzer are with the International Ground Water Modeling Center, Institute for Ground-Water Research and Education, Colorado School of Mines, Golden, CO 80401-1887.

***Joseph R. Williams** is the EPA Project Officer (see below).*

The complete report, entitled "Ground-Water Model Testing: Systematic Evaluation and Testing of Code Functionality and Performance," (Order No. xxx; Cost: $xx.00, subject to change) will be available only from:

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161
Telephone: 703-487-4650

The EPA Project Officer can be contacted at

Subsurface Protection and Remediation Division
National Risk Management Research Laboratory
U.S. Environmental Protection Agency
Ada, OK 74820