Designation: D6025 – 96 (Reapproved 2008)

Standard Guide for Developing and Evaluating Groundwater Modeling Codes[1]

This standard is issued under the fixed designation D6025; the number immediately following the designation indicates the year of original adoption or, in the case of revision, the year of last revision. A number in parentheses indicates the year of last reapproval. A superscript epsilon (ε) indicates an editorial change since the last revision or reapproval.

1. Scope

1.1 This guide covers a systematic approach to the development, testing, evaluation, and documentation of groundwater modeling codes. The procedures presented constitute the quality assurance framework for a groundwater modeling code. They include code review, testing, and evaluation using quantitative and qualitative measures. This guide applies to both the initial development and the subsequent maintenance and updating of groundwater modeling codes.

1.2 When the development of a groundwater modeling code is initiated, procedures are formulated to ensure that the final product conforms with the design objectives and specifications and that it correctly performs the incorporated functions. These procedures cover the formulation and evaluation of the code's theoretical foundation and code design criteria, the application of coding standards and practices, and the establishment of the code's credentials through review and systematic testing of its functional design and through evaluation of its performance characteristics.

1.3 The code's functionality needs to be defined in sufficient detail for potential users to assess the code's utility as well as to enable the code developers to design a meaningful code 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.

1.4 Groundwater modeling codes are subject to the software life cycle concept that consists of a design phase, a development phase, and an operational phase. During the operational phase the software is maintained, evaluated regularly, and changed as additional requirements are identified. Therefore, quality assurance procedures should not only be established for software design, programming, testing, and use, but also for code maintenance and updating.

1.5 Quality assurance in the development of groundwater modeling codes cannot guarantee acceptable quality of the code or a groundwater modeling study in which the code has been used. However, adequate quality assurance can provide safeguards against the use in a modeling study of faulty codes or incorrect theoretical considerations and assumptions. Furthermore, there is no way to guarantee that modeling-based advice is entirely correct, nor that the groundwater model used in the preparation of the advice (or any scientific model or theory, for that matter) can ever be proven to be entirely correct. Rather, a model can only be invalidated by disagreement of its predictions with independently derived observations of the studied system because of incorrect application of the selected code, the selection of an inappropriate code, the use of an inadequately tested code, or invalidity of or errors in the underlying theoretical framework.

1.6 This guide is one of a series of guides on groundwater modeling codes and their applications, such as Guides D5447, D5490, D5611, D5609, D5610, and D5718. Other standards have been prepared on environmental modeling, such as Practice E978.

1.7 Complete adherence to this guide may not always be feasible. If this guide is not integrally followed, the elements of noncompliance should be clearly identified and the reasons for the partial compliance should be given. For example, partial compliance might result from inadequacy of existing field techniques for measuring relevant model parameters, specifically in complex systems.

1.8 *This guide offers an organized collection of information or a series of options and does not recommend a specific course of action. This document cannot replace education or experience and should be used in conjunction with professional judgment. Not all aspects of this guide may be applicable in all circumstances. This ASTM standard is not intended to represent or replace the standard of care by which the adequacy of a given professional service must be judged, nor should this document be applied without consideration of a project's many unique aspects. The word "Standard" in the title of this*

[1] This guide is under the jurisdiction of ASTM Committee D18 on Soil and Rock and is the direct responsibility of Subcommittee D18.21 on Groundwater and Vadose Zone Investigations.

Current edition approved Sept. 15, 2008. Published November 2008. Originally approved in 1996. Last previous edition approved in 1996 as D6025 – 96 (2002). DOI: 10.1520/D6025-96R08.

document means only that the document has been approved through the ASTM consensus process.

2. Referenced Documents

2.1 *ASTM Standards:*[2]

D653 Terminology Relating to Soil, Rock, and Contained Fluids

D5447 Guide for Application of a Groundwater Flow Model to a Site-Specific Problem

D5490 Guide for Comparing Groundwater Flow Model Simulations to Site-Specific Information

D5609 Guide for Defining Boundary Conditions in Groundwater Flow Modeling

D5610 Guide for Defining Initial Conditions in Groundwater Flow Modeling

D5611 Guide for Conducting a Sensitivity Analysis for a Groundwater Flow Model Application

D5718 Guide for Documenting a Groundwater Flow Model Application

E978 Practice for Evaluating Mathematical Models for the Environmental Fate of Chemicals (Withdrawn 2002)[3]

3. Terminology

3.1 *Definitions:*

3.1.1 *code verification, n— in groundwater modeling*, the process of demonstrating the consistency, completeness, correctness, and accuracy of a groundwater modeling code with respect to its design criteria by evaluating the functionality and operational characteristics of the code and testing embedded algorithms and internal data transfers through execution of problems for which independent benchmarks are available (**1**).[4]

3.1.1.1 *Discussion*—In software engineering, verification is the process of demonstrating consistency, completeness, and correctness of the software (**2**). Practice E978 defines verification as "...the examination of the numerical technique in the computer code to ascertain that it truly represents the conceptual model and that there are no inherent problems with obtaining a solution." In this guide, the term code verification is used. The objective of the code verification process is threefold: (*1*) to check the correctness of the program logic and the computational accuracy of the algorithms used to solve the governing equations; (*2*) to ensure that the computer code is fully operational (no programming errors); and (*3*) to evaluate the performance of the code with respect to all of its designed and inherent functions (**1**).

A code can be considered "verified" when all its functions and operational characteristics have been tested and have met specific performance criteria, established at the beginning of the verification procedure. Considering a code verified does not imply that a groundwater model application constructed with the code is verified.

NOTE 1—In groundwater modeling, the term "validation" is sometimes used to describe the process of determining how well a groundwater modeling code's theoretical foundation and computer implementation describe actual system behavior in terms of the degree of correlation between calculated and independently observed cause-and-effect responses of the reference groundwater system for which the code has been developed (**1,3**). This process is also referred to as field demonstration, field comparison, or extended verification (**4**).

NOTE 2—Validation as described in Note 1 is by nature a subjective and open-ended process. As there is no practical way to determine that a groundwater modeling code correctly represents the reference system, the code can never be considered "validated." Therefore, this guide does not endorse the use of the term validation in the context of groundwater modeling (**1,3,4**).

3.1.2 *computer code (computer program), n*— the assembly of numerical techniques, bookkeeping, and control language that represents a model from acceptance of input data and instructions to delivery of output.

3.1.3 *functionality, n— of a groundwater modeling code*, the set of functions and features the code offers the user in terms of model framework geometry, simulated processes, boundary conditions, and analytical and operational capabilities.

3.1.4 *groundwater model application, n*— a nonunique, simplified mathematical description of one or more subsurface components of a local or regional hydrologic system, coded in a computer programing language, together with a quantification of the simulated system in the form of framework geometry, boundary conditions, system and process parameters, and system stresses.

3.1.4.1 *Discussion*—As defined in 3.1.4, a groundwater model application is a representation of an actual hydrologic system; it should not be confused with the generic computer code used in formulating the groundwater model. This guide concerns only the development, testing, and documentation of generic simulation computer codes, not groundwater model applications.

3.1.5 *groundwater modeling, n*—the process of developing groundwater models.

3.1.6 *groundwater modeling code, n*—the nonparametrized computer code used in groundwater modeling to represent a nonunique, simplified mathematical description of the physical framework, geometry, active processes, and boundary conditions present in a reference subsurface hydrologic system.

3.1.6.1 *Discussion*—The term "nonparameterized computer code" refers to a generalized computer program in which values of parameters can be specified by the user.

3.1.7 *quality assurance (QA), n*—in the development of a groundwater modeling code, the procedural and operational framework put in place by the organization managing the code development project, to ensure technically and scientifically adequate execution of all project tasks, and to ensure that the resulting software product is functional and reliable.

3.2 For definitions of other terms used in this guide, see Terminology D653.

[2] For referenced ASTM standards, visit the ASTM website, www.astm.org, or contact ASTM Customer Service at service@astm.org. For *Annual Book of ASTM Standards* volume information, refer to the standard's Document Summary page on the ASTM website.

[3] The last approved version of this historical standard is referenced on www.astm.org.

[4] The boldface numbers in parentheses refer to a list of references at the end of this standard.

4. Significance and Use

4.1 Groundwater modeling has become an important methodology in support of the planning and decision-making processes involved in groundwater management. Groundwater models provide an analytical framework for obtaining an understanding of the mechanisms and controls of groundwater systems and the processes that influence their quality, especially those caused by human intervention in such systems. Increasingly, models are an integral part of water resources assessment, protection, and restoration studies and provide essential and cost-effective support for planning and screening of alternative policies, regulations, and engineering designs affecting groundwater. It is therefore important that before groundwater modeling codes are used as planning and decision-making tools, their credentials are established and their suitability determined through systematic evaluation of their correctness, performance characteristics, and applicability. This becomes even more important because of the increasing complexity of the hydrologic systems for which new modeling codes are being developed.

4.2 Quality assurance in groundwater modeling provides the mechanisms and framework to ensure that the analytic tools used in preparing decisions are based on the best available techniques and methods. A well-executed quality assurance program in groundwater modeling provides the information necessary to evaluate the reliability of the performed analysis and the level to which the resulting advice may be incorporated in decision-making regarding the management of groundwater resources.

4.3 This guide is intended to encourage consistency and completeness in the development and evaluation of existing and new groundwater modeling codes by describing appropriate code development and quality assurance procedures and techniques.

4.4 In the past, some groundwater modeling codes have been developed that have turned out to be quite useful without having been subject to all of the procedures described in this guide. Nonetheless, the procedures described in this guide will give greater assurances that a code does what its developers intended it to do and that a rational basis is available to judge code adequacy and limitations.

5. Code Development Process

5.1 In groundwater modeling, code development consists of the following:

5.1.1 Definition of design criteria and determining applicable software standards and practices,

5.1.2 Development of algorithms and program structure,

5.1.3 Computer programming,

5.1.4 Preparation of documentation,

5.1.5 Code testing, and

5.1.6 Independent review of scientific principles, mathematical framework, software, and documentation.

5.2 Code design criteria should address the following:

5.2.1 The physical system to be modeled in terms of geometry, physical processes and properties, and stresses,

5.2.2 Assumptions made in deriving the mathematical framework,

5.2.3 Dimensionality, and spatial and temporal discretization,

5.2.4 Type and form of computed entities,

5.2.5 Type and form of code operation control,

5.2.6 Code structure, and programming language,

5.2.7 Input/output structure and applicable data exchange formats,

5.2.8 User interface,

5.2.9 Computer platforms for implementation, and

5.2.10 Type, contents, structure, and level of detail of documentation.

5.3 The development of a specific groundwater modeling code may be part of a research or development project, based on an existing mathematical model, or derived from an existing set of modeling codes.

5.3.1 Code development in groundwater modeling is often part of research aimed at acquiring new, quantitative knowledge about nature through observation, hypothesizing, and verifying deduced relationships, leading to the establishment of a credible theoretical framework for the observed phenomena. The resulting research model represents a fundamental understanding of the studied groundwater system.

5.3.2 The object for model research in groundwater is a subset of the hydrologic system, called the reference system. It contains selected elements of the global hydrologic system. The selection of a particular reference system is influenced by regulatory and management priorities, and by the nature of the hydrologic system (Fig. 1). The conceptual model of the selected reference system forms the basis for quantifying the causal relationships among various components of this system, and between this system and its environment. These relationships are defined mathematically, resulting in a mathematical model. If the solution of the mathematical equations is complex or when many repetitious calculations are required, the use of computers is essential. This requires the coding of the

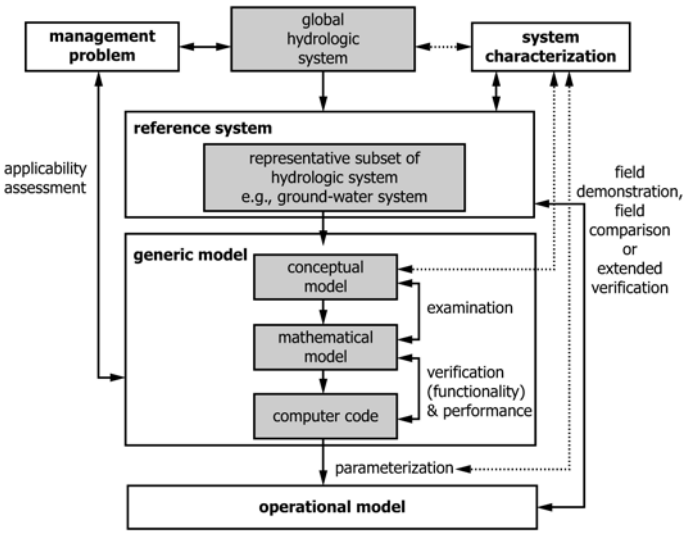

FIG. 1 Concepts and Terms Related to the Development and Testing of a Groundwater Modeling Code (1, modified)

solution to the mathematical problem in a programming language, resulting in a computer code. The conceptual formulations, mathematical descriptions, and computer coding constitute the (generic) model (Fig. 1). Attributing the parameters and stresses in the generic model results in an operational model of the reference system.

5.3.3 To determine the validity of the model, questions need to be answered, such as the following:

5.3.3.1 Does the conceptual model on which the simulation code is based truly represent the reference system?

5.3.3.2 Does the mathematical model closely represent the conceptual model, including the hydrogeologic framework present, and the processes and stresses present? If not, are the simplifications, made to facilitate the formulation of a mathematical model, acceptable and relevant?

5.3.3.3 Does the computer code correctly represent the model's mathematical framework?

5.3.3.4 Will the model be able to represent the responses of the reference system to various stress scenarios?

5.3.4 The two major approaches to achieve acceptance of a groundwater model are: (*1*) the evaluation or (peer) review process covering all phases of the code development process; and (*2*) quantitative comparison with independently obtained data for the reference groundwater system.

NOTE 3—Determining the correctness of the model is basically part of the scientific discovery process and as such is a rather subjective and open-ended process. When will a model, or for that matter a theory, be accepted by the scientific community? Often, this question is replaced by another one: can the model and its underlying theoretical and conceptual assumptions be refuted?

5.3.5 Since the development of most groundwater modeling codes is based on a mechanistic description of the physical processes, the resulting computer codes can be generalized and applied to other groundwater systems comparable to the reference system. The level of generalization found in the groundwater modeling code is a function of the generality of the model, that is, how common is the reference groundwater system in the real world (allowing for variation in model parameters) and how many different subsets of the model are encountered in solving real-world problems. Determining the applicability of the generalized computer code (that is, code without numerical values for geometry, parameters, boundary conditions, and stresses) requires analysis of the adequateness of the model, the extent of the code's functions and operational characteristics, and the results of code testing.

5.4 Code testing is an integral part of code development. During the programming phase, testing is focussed on individual algorithms, subroutines, functions, and other program elements. At the end of the initial programming phase, the code should be extensively tested using the procedures described in Section 7.

5.5 The preparation of the program documentation starts at the beginning of the code development process and is integral to all stages of code development. Specifically, documentation of theoretical foundation, code design, capabilities, and program structure should be prepared and evaluated during the design and programming phases of the project. Documentation regarding the operation and performance of the code should be prepared before and during initial testing by code developers.

5.6 The final step in code development is independent review and testing.

NOTE 4—Although optimal quality assurance requires the software development project to start with the formulation of code design criteria, this step is often absent in the development of a groundwater modeling code. Therefore, code applicability assessment is crucial in determining the nature of physical systems and management issues that can be addressed by the code. In part, such an assessment is based on detailed evaluation of the functionality description of the code. In-depth analysis of successful site-specific applications provides additional information regarding the utility of the code, enhancing its credibility with users and decision-makers.

6. Code Development Quality Assurance Procedure

6.1 In software engineering terms, software quality assurance (SQA) consists of the application of procedures, techniques, and tools throughout the software life cycle, to ensure that the products conform to prespecified performance requirements (**2**,**5**). The SQA procedures include developing a QA plan, record keeping, and establishing a project QA organization. The SQA techniques include auditing, design inspection, code inspection, error-prone analysis, functional testing, logical testing, path testing, reviewing, and walkthroughs. The SQA tools include text-editors, software debuggers, source code comparitors, and language processors. All of these need to be identified in the initial stage of the software development project as the software design criteria are determined (**2**,**5**). The SQA should be applied to all codes currently in use and yet to be developed, whether for research or water resource management purposes.

6.2 The following code development QA procedures are considered minimum requirements for groundwater modeling codes to be used in support of groundwater management and environmental decision-making (**1**):

6.2.1 Determination and documentation of code design criteria, including functionality (for example, hydrogeologic framework, processes, boundary conditions, and computed variables), input/output requirements (for example, graphics, file handling, and file formats), hardware platform(s), programming language (for example, language type, compiler, industry standard), program structure, and program performance,

6.2.2 Design and documentation of code structure, algorithms, data structures, and input/output characteristics,

6.2.3 Documentation of code development progress (for example, record keeping of implementation strategy for design criteria, problems in coding or performance, and implemented variances of the design),

6.2.4 Testing of program structure and coding (code verification) and subsequent documentation,

6.2.5 Documentation of code characteristics, capabilities, and performance, and preparation of operational instructions (that is user's instructions),

6.2.6 Scientific and technical reviews of code foundation, coding, performance, and documentation, and

6.2.7 Record keeping (reports, paper files, and electronic files) and administrative auditing of adherence to QA plans and QA procedures.

7. Code Testing

7.1 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 dedicated test problems, relevant test data sets, and informative performance measures.

7.2 The results of code testing are expressed in terms of correctness (for example, in comparison with a benchmark), *reliability* (for example, convergence and stability of solution algorithms, and absence of terminal failures), *efficiency* of coded algorithms (in terms of numerical accuracy versus code execution time, and memory and mass storage requirements), and *resources required* for model setup and analysis (for example, input preparation time and effort needed to make output ready for graphic analysis) **(6)**.

7.3 Code testing should sequentially follow the following series of steps **(6)**:

7.3.1 Analysis of the code's functionality in terms of simulation functions, operational features, mathematical framework, and software implementation,

7.3.2 Identification of potential code performance issues based on analysis of simulated processes, mathematical solution methods, computer limitations, and execution environment,

7.3.3 Development of a code testing strategy and test problems that address relevant code performance issues as they are viewed by all stakeholders (for example, researchers, code developers, code users, fund managers, regulatory decision makers, project decision makers, and so forth),

7.3.4 Execution of tests and analysis of results using appropriate comprehensive, informative, and accurate graphic and statistical techniques,

7.3.5 Collection of code information issues and code test problem objectives in overview tables and matrix displays reflecting completeness of testing, as well as correctness, accuracy, efficiency, and field applicability of the tested code,

7.3.6 Identification of performance strengths, and weaknesses of the code and testing procedure,

7.3.7 Documentation of test objectives, model setup for both the tested code, and the benchmark (structure, discretization, and parameters), and results for each test (for both the tested code and the benchmark) in report form and as electronic files including input data, computational results, statistical analysis of all computed results, and graphical representation of key results, and

7.3.8 Preparation of an executive summary of functionality, test strategy and results.

7.4 *Functionality Analysis and Performance Evaluation*

7.4.1 *Functionality Analysis:*

7.4.1.1 Functionality analysis involves the identification and description of the functions of a simulation code in terms of model framework geometry, simulated processes, boundary conditions, and analytical capabilities, and the subsequent evaluation of each code function or group of functions for conceptual correctness and computational accuracy (including convergence for a practical range of parameter values) and consistency (including numerical stability) (see Table 1) **(6)**.

7.4.1.2 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 combining a complete description of the code functions and features with the objectives of targeted test problems (see Fig. 2). The functionality matrix illustrates the extent of the performed functionality analysis **(6)**.

7.4.2 *Performance Evaluation:*

7.4.2.1 Performance evaluation is aimed at characterizing the operational characteristics of the code in terms of: (*1*) correctness, (*2*) overall accuracy; (*3*) reliability; (*4*) sensitivity for grid orientation and resolution, and for time discretization; (*5*) efficiency of coded algorithms (including bandwidth, rate of convergence, memory usage, and disk I/O); and (*6*) level of effort and resources required for model setup and simulation analysis **(6)**.

7.4.2.2 Results of the performance evaluation are expressed both quantitatively and qualitatively in checklists and in tabular form (see Tables 2-5 and Table 6). 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.

7.4.3 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.

7.5 *Code Testing Strategy:*

7.5.1 The code testing strategy represents a systematic, efficient approach to the comprehensive testing of the code. The code testing strategy includes:

7.5.1.1 Formulation of test objectives (as related to code functionality), and of test priorities (based on considerations listed in 7.3.2 and 7.3.3 and on available resources for testing) (see Table 7),

7.5.1.2 Selection or design, or both, of test problems and determination of type and extent of testing for selected code functions or application-dependent combinations of code functions,

7.5.1.3 Determination of level of effort to be spent on sensitivity analysis for each test problem,

7.5.1.4 Selection of the qualitative and quantitative measures to be used in the evaluation of the code's performance, and

7.5.1.5 Determination of the level of detail to be included in the test report and the format of reporting (see Tables 8 and 9, and Table 10).

7.5.2 There are three levels of testing **(1)**:

7.5.2.1 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

TABLE 1 Functions and Features of a Typical Three-Dimensional Saturated Flow and Transport Model (5)

General Model Capabilities:
- Uncoupled Darcian groundwater flow and nonconservative single-component solute transport in saturated porous medium
- Distributed parameter specification

Spatial Orientation:
- 1-D horizontal
- 1-D vertical
- 2-D horizontal
- 2-D vertical
- Quasi 3-D (layered)
- Fully 3-D

Grid Design:
- 1-D, 2-D, or 3-D block-centered finite difference grid with constant or variable cell size

Time Discretization:
- Steady state flow
- Transient flow
- Transient transport
- Variable time step size
- Multiple transport time steps per flow time step
- Multiple flow time steps per stress period
- Variable stress periods

Matrix Solvers:
- SOR
- ADI
- PCG

Aquifer Conditions:
- Confined
- Leaky-confined
- Unconfined

Aquifer Systems:
- Single aquifer
- Single aquifer/aquitard
- Multiple aquifers/aquitards

Variable Aquifer Conditions in Space:
- Variable layer thickness
- Confined and unconfined conditions in same aquifer
- Aquitard pinch out
- Aquifer pinch out

Changing Aquifer Conditions in Time:
- De-saturation of cells at water table
- Re-saturation of cells at water table
- Confined/unconfined conversion

Parameter Representations:
- Hydraulic conductivity: heterogenous (variable in space), anisotropic
- Storage coefficient: heterogeneous
- Longitudinal dispersivity: heterogeneous
- Transverse dispersivity: heterogeneous
- Sorption coefficient homogeneous (single value for total model area)
- Decay coefficient: homogeneous

Fluid Conditions:
- Density constant in time and space
- Viscosity constant in time and space

Boundary Conditions for Flow:
- Fixed head
- Prescribed time-varying head
- Zero flow
- Fixed boundary flux
- Prescribed time-varying boundary flux
- Areal recharge variable in space and time
- Induced recharge from or discharge to stream; stream need not be directly connected to groundwater
- Drains
- Evapotranspiration dependent on distance
- Surface to water table
- Free surface, seepage face

Solute Transport Processes:
- Advection
- Hydrodynamic dispersion
- Molecular diffusion
- Linear equilibrium sorption
- First-order radioactive decay
- First-order chemical/microbial decay

Boundary Conditions for Solute Transport:
- Fixed concentration
- Prescribed time-varying concentration
- Zero solute flux
- Specified constant or time-varying solute flux
- Areal recharge of given (constant or time-varying) concentration
- Induced infiltration of given (constant or time-varying) concentration
- Concentration-dependent solute flux

Sources/sinks:
- Injection/production well with constant or time-varying flow rate
- Injection well with constant or time-varying concentration
- Injection well with constant or time-varying solute flux
- Production well with aquifer concentration-dependent solute outflux
- Springs with head-dependent flow rate and aquifer concentration-dependent solute flux

	functions				
test problem objective	function 1	function 2	function 3	function 4	function 5
test 1		X			X
test 2	X				
test 3		X			
test 4				X	X
test 5		X		X	
test 6				X	

FIG. 2 Generic Model Functionality Matrix; Checked Cells Indicate that the Objective of the Test Problem Corresponds with a Model Function (6)

TABLE 2 Example Performance Evaluation Table—Part 1 (6)

Test Case	Number of Nodes	Number of Time Steps	Time Step, days	Convergence, number of iterations[A]	Central Processor Use, s	Memory Use, Kbytes	Set-up Time, h
1	500	1	10	5	11	550	2
2	500	1	10	50 (maximum; did not converge)	205	550	1.5
3	500	1	10	11	34	550	1.5
4	500	1	10	22	55	550	2
5a	500	1	10	7	21	550	1
5b	5000	1	10	9	309	3880	4
5c	500	10	1	21	80	550	1

[A] Convergence is expressed as the number of iterations needed to reach a convergence criterion. In general, core performance depends on the nature and value of the convergence criterion. Most codes allow the user to specify the value of the specific type of convergence criterion (or types) used in the code.

functions (see Fig. 3); and (*3*) comparing with independent, accurate benchmarks (for example, analytical solutions).

NOTE 5—If the benchmark computations themselves have been made using a computer code, this computer code should in turn be subjected to rigorous testing by comparing computed results with independently derived and published data.

7.5.2.2 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 (that is, comparison between runs with the same code using different functions to represent a particular feature) and intercomparison (that is, comparison between different codes simulating the same problem).

TABLE 3 Example Performance Evaluation Table—Part 2 (6)

Test Case	Sensitivity to Grid Size[A]	Sensitivity to Grid Orientation[B]	Sensitivity to Time Discretization[C]	Stability[D]	Sensitivity to Convergence Criterion[E]
1	0.1	0.01	0.1	satisfactory	1.001
2	0.02	0.007	0.2	unsatisfactory	0.92
3	0.03	0.02	0.1	satisfactory	1.001
4	0.001	0.008	0.3	satisfactory	0.998
5a	0.3	0.04	0.3	satisfactory	0.9998
5b	0.25	0.05	0.25	satisfactory	0.999
5c	0.21	0.045	0.1	satisfactory	0.9997

[A] Sensitivity to grid size is determined by comparing the sum of absolute values of the differences in computed nodal values with the sum of computed nodal values divided by 2, employing two grid designs differing by a factor of 10 in the number of active nodes.
[B] Sensitivity to grid orientation is determined by comparing the sum of absolute values of the differences in computed nodal values with the sum of computed values divided by 2, using two identical grid designs rotated 45° with respect to each other.
[C] Sensitivity to time discretization is determined by comparing the sum of absolute values of the differences in computed nodal values with the sum of computed values divided by 2, using for a constant period two time discretizations differing by a factor of 10.
[D] Stability is rated "unsatisfactory" if in one or more runs stability problems are encountered; otherwise stability is rated "satisfactory."
[E] Sensitivity to convergence criterion is a measure for reproducibility. It is given as the ratio of the sum of the dependent variable computed for two values of the convergence criterion differing one order of magnitude.

Typically, synthetic data sets are used representing hypothetical, often simplified groundwater systems.

7.5.2.3 At Level III, a code (and its underlying theoretical framework) is tested to determine how well a model's theoretical foundation and computer implementation describe 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.

NOTE 6—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 will always retain an element of incompleteness and subjectivity.

7.5.3 First, Level I testing is conducted (often during code development), and, if successfully completed, it is followed by Level 2 testing. The code may gain further credibility and user confidence by being subjected to Level 3 testing (that is, field or laboratory testing).

7.5.4 Although, 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 code testing may be necessary.

7.6 *Code Testing Evaluation Criteria:*

7.6.1 An important aspect of code testing is the definition of informative and efficient measures for use as evaluation or performance criteria. Such measures should characterize quantitatively the differences between the results derived with the simulation code and the benchmark, or between the results obtained with two comparable simulation codes.

7.6.2 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.

7.6.3 Graphical measures are especially significant for 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.

7.6.3.1 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. The selection of a set of representative sample data-pairs may be based on symmetry considerations, model domain areas with potential higher deviations, or on specific interest in subdomains (that is, vertical or horizontal slices of the model domain).

7.6.3.2 There are five types of graphical evaluation techniques particularly suited (see Table 11 and Table 12):

(1) The *X-Y* plots or line graphs of spatial (for example, distance) or temporal behavior of dependent variable and other computed entities (see Figs. 4-6, and Fig. 7),

(2) One-dimensional column plots or histograms (specifically to display test deviations),

(3) Combination plots of line graphs of dependent variable and column plots of deviations (see Fig. 8),

(4) Contour and surface plots of the spatial distribution of the dependent variable and the residuals, and

(5) Three-dimensional, isometric, column plots or three-dimensional histograms (see Fig. 9 and Fig. 10).

7.6.3.3 The conclusions from visual inspection of graphic representations of testing results may be described qualitatively (and subjectively) by such attributes as "poor," "reasonable," "acceptable,"" good," and "very good" (see Fig. 11).

7.6.4 There are three general procedures, coupled with standard linear regression statistics and estimation of error statistics, to provide quantitative goodness-of-fit measures (**7**):

7.6.4.1 *Paired-data Performance*—The comparison of simulated and observed data for exact locations in time and space,

7.6.4.2 *Time and Space Integrated, Paired-data Performance*—The comparison of spatially and temporally integrated or averaged simulated and observed data, and

7.6.4.3 *Frequency Domain Performance*—The comparison of simulated and observed frequency distributions.

NOTE 7—The organization and evaluation of code intercomparison results can be cumbersome due to the potentially large number of data-pairs to be analyzed if every computational node is included. This can be mitigated by analyzing smaller, representative subsamples 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 on two lines parallel to the flow direction, one in the center of the model domain and one at the edge (see Fig. 12). Another example is the simulation of the Theis problem; here, two lines of data pairs should be chosen parallel to the two horizontal principal hydraulic conductivity axes, while a third set of data pairs should be on a line at 45° to these axes (see Fig. 13). Test cases that are symmetrical can be analyzed for a smaller portion of domain based upon

TABLE 4 Functionality Issues for Confined/Unconfined Conditions (6)

Functionality Issue	Test Objective	Type of Test
In unconfined aquifers, transmissivity is dependent on the computed heads.	To determine if the code correctly represents the water table under steady-state conditions. How sensitive are the results for the difference between initial conditions and final heads, or boundary conditions? Does the number of model layers make a difference?	Steady-state benchmark Level 1B
In unconfined aquifers, a rising water table might rise above the top of the initial model layer, invading dry cells (saturation/wetting).	To determine if the code functions properly when water invades dry model cells, both under steady-state conditions (initial condition set below final water-bearing model cells) and transient conditions.	Steady-state, transient benchmark Level 1B
In unconfined aquifers, a falling water table might drop below the bottom of the initial (partially) water-filled cells (de-saturation).	To determine if the code functions properly when water evacuates wet model cells and fully water-filled cells become partially water-filled, both under steady-state conditions (initial condition set above final water bearing model cells) and transient conditions.	Steady-state, transient benchmark Level 1B
Cyclic variations of the water table position over more than one model layer require repeated desaturation and resaturation of model layers.	To determine if accuracy (in terms of heads and mass balance) is maintained over multiple desaturation and rewetting cycles, and if no stability problems occur.	Transient benchmark Level 1B
For unconfined conditions, transmissibility is a function of saturated thickness. Various schemes exist to treat the resulting nonlinear terms, including (damped) corrections at each iteration or time step, or both.	To determine the accuracy for water table conditions for various steady-state and transient conditions (for example, poor initial conditions, and small hydraulic conductivity or storativity).	Steady-state transient conceptual test intercomparison Level 1A
When the head in a confined layer drops below the top of that layer, conditions become unconfined. This phenomenon typically occurs in areas of the model domain where discharge is significant. If the discharge diminishes or is reversed, conditions may become confined again.	To determine proper assignment of storativity and other code settings when conditions change between confined and unconfined (in both quasi and fully 3-D mode), and to determine stability under these conditions.	Transient benchmark intra-comparison Level 1B and 2A
Most 2-D and 3-D codes include an option to simulate groundwater flow in a quasi three-dimensional mode.	To determine if quasi three-dimensional mode works properly for unconfined and semi-confined multilayer systems.	Transient benchmark Level 1A

TABLE 5 Functionality Issues for Advective and Dispersive Solute Transport (6)

Functionality Issue	Test Objective	Type of Test
Advection-dominated transport often creates numerical problems in the vicinity of the solute front.	To determine accuracy in terms of concentrations and mass balance, to evaluate stability and the occurrence of oscillations and numerical dispersion, and to perform sensitivity analysis with respect to transport parameter values, and spatial and temporal discretization.	Steady-state uniform flow transient transport benchmark Level 1B
Accuracy of simulation of dispersive transport is dependent on grid orientation. Inclusion of cross terms of the dispersion coefficient may improve accuracy.	To determine sensitivity of concentration distribution and mass balance for grid orientation.	Steady-state uniform flow transient transport benchmark Level 1B
Accuracy of dispersive transport may be influenced by the contrast in the main directional components of the dispersivity, especially when using non-optimal grid orientation.	To determine accuracy of concentration distribution and mass balance for different ratios for the dispersivity components.	Steady-state uniform flow transient transport benchmark Level 1B
Sometimes, advective-dispersive transport is negligible and molecular diffusion is prominent.	To determine accuracy in terms of concentrations and mass balance when molecular diffusion is important.	Transient benchmark Level 1A

TABLE 6 Functionality Issues for Solute Fate (Retardation and Decay) (6)

Functionality Issue	Test Objective	Type of Test
Sorption is often represented as a linear or nonlinear reversible equilibrium reaction, represented by a retardation coefficient. Some codes implicitly maintain mass balance in both the dissolved and solid phases, other codes display mass balance problems under certain scenarios.	To evaluate correctness of reversible sorption function and to determine accuracy in terms of concentrations and mass balance for various sorption rates (check for reversibility).	Steady-state uniform flow transient transport hand calculations (mass balance) benchmark (concentrations) Level 1A, 1B
Some codes include zero-order production or removal in the source/sink term of the governing equation.	To evaluate correctness and accuracy of this function in terms of concentrations and mass balance.	Steady-state uniform flow transient transport hand calculations (mass balance) benchmark (concentrations) Level 1A, 1B
Many codes include first-order production or decay in the source/sink term of the governing equation. Some codes display instabilities or inaccuracies when half-life times are about the same order of magnitude as or smaller than the time steps.	To evaluate correctness and accuracy of this function in terms of concentrations and mass balance for both large and small values of the decay coefficient (including zero).	Steady-state uniform flow transient transport hand calculations (mass balance) benchmark (concentrations) Level 1A, 1B

the type of symmetry present. For example, test cases that have radial symmetry can be divided into four equal representative radial slices; this significantly decreases the required number of data pairs in the analysis and considerably reduces the evaluation effort.

TABLE 7 Major Test Issues for Three-dimensional Finite-difference Saturated Groundwater Flow and Solute Transport Codes (6)

General Features:
Mass balances (regular versus irregular grid)
Variable grid (consistency in parameter and stress allocation)
Hydrogeologic Zoning, Parametrization, and Flow Characteristics:
Aquifer pinch out, aquitard pinch out
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

TABLE 8 Elements of a Test Report

Introduction
Program name
Program title
Tested version
Release date
Author/custodian
Reviewer (name, organization)
Review date
Short description
Computer and software requirements
Test environment (computer, operating system, and so forth
Reviewed materials/documentation
Installation review
Discussion of general operation (batch, interactive, graphics)
Terms of availability (legal status, and so forth)
Type/level of support
Testing
Analysis of code functions and preparation of functionality description
Overview and discussion and reevaluation of testing performed by code authors
Overview and detailed description of additional test performed
Presentation and discussion of functionality analysis matrix
Presentation and discussion of performance tables
Optional discussion of applicability issues both from a theoretical point of view, as well as based on applicability testing
Conclusions
Testing (performance, limitations, cautions)
Documentation (completeness and correctness of functionality description, correctness of theory, consistency of mathematical description and coded functionality, correctness and completeness of user's instructions)
Installation and general operation
Code setup (how easy/difficult it is to run the code)
Specific hints/tricks learned during testing, not present in documentation

TABLE 9 Test Details to be Discussed in Test Report

General problem description (including assumptions, limitations, boundary conditions, parameter distribution, time-stepping, figures depicting problem situation)
Test objectives (features of simulation code, specifically tested by test problem)
Benchmark reference
If feasible, benchmark solution (for example, analytical solution)
Reference to benchmark implementation (hand calculation, dedicated software, generic mathematical software, and so forth)
Test data set
Model setup, discretization, implementation of boundary condition, representation of special problem features (for both tested code and benchmark code; electronic input files)
Results (table of numerical and benchmark results (if available) for the dependent variable at selected locations and times; mass balances; statistical measures and supporting figures; electronic results files)
Sensitivity analysis strategy and results
Discussion of results

7.6.5 Useful quantitative evaluation measures for code testing include the following (6):

7.6.5.1 *Mean Error* (ME), defined as the mean difference (that is, deviation) between the dependent variable calculated by the numerical model h_c and the benchmark value of the dependent variable h_b for n data pairs:

$$ME = \frac{\sum (h_c - h_b)}{n} \quad (1)$$

7.6.5.2 *Mean Absolute Error* (MAE), defined as the average of the absolute values of the deviations:

$$MAE = \frac{\sum | (h_c - h_b)}{n} \quad (2)$$

7.6.5.3 *Positive Mean Error* (PME) and *Negative Mean Error* (NME), defined as the ME for the positive deviations and negative deviations, respectively;

7.6.5.4 *Mean Error Ratio* (MER), a composite measure indicating systematic overpredicting or underpredicting by the code:

$$MER = \frac{|ME|}{ME} \times \frac{|NME}{PME} \quad (\text{for } PME < |NME|) \quad (3)$$

$$MER = \frac{|ME|}{ME} \times \frac{PME}{|NME|} \quad (\text{for } PME > |NME|) \quad (4)$$

7.6.5.5 *Maximum Positive Error* (MPE) and *Maximum Negative Error* (MNE), defined as the maximum positive and negative deviation, respectively, indicating potential inconsistencies or sensitive model behavior;

7.6.5.6 *Root Mean Squared Error* (RMSE), defined as the square root of the average of the squared differences between the dependent variable calculated by the numerical model and its benchmark equivalent:

$$RMSE = \sqrt{\frac{\sum (h_c - h_b)^2}{n}} \quad (5)$$

TABLE 10 Elements of the Executive Summary of the Test Report

Program name, title, version, release date, authors, custodian
Reviewer (name, organization)
Detailed program description (functionality)
Computer/software requirements
Terms of availability and support
Overview of testing performed by authors
Overview of additional testing performed
Discussion of specific test results (illustrating strengths and weaknesses)
Discussion of completeness of testing (functionality matrix)
Representative performance information
Main conclusions on test results
Comments on installation, operation, and documentation
List of main documentation references
Tables providing overview of performed tests and performance information
Figures illustrating key results

7.6.6 Various computed variables may be the focus of graphic or statistical comparison:

7.6.6.1 *Saturated Flow Codes*—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 locations of free surfaces and seepage surfaces,

7.6.6.2 *Unsaturated Flow Codes*—Hydraulic heads or suction heads, water contents or saturations, head gradients, global water balance, and internal and boundary fluxes, and

7.6.6.3 *Solute Transport Codes*—Concentrations, mass fluxes global mass balance (per species), and breakthrough curves at observation points and sinks (wells and streams).

8. Documentation of Code Design and Code Development

8.1 The audit trail for QA in model development consists of reports and files on the development of the model and should include the following:

8.1.1 . Report on the development of the code including the (standardized and approved) programmer's notebook, and the items listed in 6.2,

8.1.2 Test report including items listed in 7.3.7,

8.1.3 Changes and verification of changes made in code after baselining, and

8.1.4 Any special conditions, operational restrictions, or other limitations for code use.

8.2 Various files should be retained (in hard copy and, at higher levels, in digital form) including the following:

8.2.1 Executable image and source code of baselined version of the tested code,

8.2.2 Run-time version and data files or spreadsheet files representing the benchmark for each test, and

8.2.3 Input and output of the tested code for each test run.

9. Software Documentation

9.1 Software or computer code documentation can be described as the information recorded during the design, development, and maintenance of a computer program for the purpose of explaining pertinent features of the program and aspects of the data processing system, including purposes, methods, logic, relationships, capabilities, limitations, and operational instructions.

9.2 Code documentation is the principal instrument for those involved in a modeling effort, such as the code developer, code maintenance staff, computer system operators, and code users, to communicate regarding all aspects of the software.

9.3 The main purposes of software documentation are as follows (**8**):

9.3.1 To record technical information that enables system and program changes to be made quickly and effectively,

9.3.2 To enable programmers and system analysts, other than software originators, to use and to work on the programs,

9.3.3 To assist the user in understanding what the program is about and what it can do,

9.3.4 To increase program sharing potential,

9.3.5 To facilitate auditing and verification of program operations, that is, code evaluation,

9.3.6 To provide managers with information to review at significant developmental milestones so that they may determine that project requirements have been met and that resources should continue to be expended,

9.3.7 To reduce the disruptive effects of personnel turnover,

9.3.8 To facilitate understanding among managers, developers, programmers, operators, and users by providing information about maintenance, training, and changes in and operation of the software, and

9.3.9 To inform other potential users of the functions and capabilities of the software, so that they can determine whether it serves their needs.

9.4 A complete set of code documentation consists of three types of manuals providing code information for managers, users, and programmers.

9.4.1 The manager's manual is a summary containing: (*1*) description of code functions, underlying principles and assumptions, and application limitations; (*2*) code development history; (*3*) summary test report with an overview of the performed testing and key findings; and (*4*) a discussion of current and future applications.

9.4.2 The user's manual should include an in-depth treatment of the equations on which the code is based, of the underlying assumptions, of the boundary conditions that are incorporated in the code, of the method and algorithms used to solve the equations, and of the limiting conditions resulting from the chosen approach. The documentation must include user's instructions for implementing and operating the code, and for preparing data files. It should present examples of model formulation (for example, grid design, assignment of boundary conditions), complete with input and output file descriptions and a trouble-shooting guide. Finally, user's documentation should include an extensive code testing report.

9.4.3 The programmer's manual should include code specifications, code description, flow charts, description of routines, data base description, source listing and error messages, and should provide instructions for code modification and maintenance.

9.5 The code itself should be well-structured and internally well-documented; where possible, self-explanatory parameter, variable, subroutine, and function names should be used.

NOTE 8—While documentation should commence at the very beginning

Program Name:	HOTWTR
Program Title:	Simulating Coupled Three-Dimensional Steady-State Groundwater Flow and Heat Transport in Saturated Media
Version:	1.1
Release Date:	September 1993
IGWMC Number:	FOS 67
Institution of Development:	U.S. Geological Survey, Denver, Colorado

TEST 03D:

Geometry: Multi-layer profile model (2-D cross-sectional); homogeneous aquifer of 13 by 1 cells horizontally, and 10 layers

Processes: Internal heat conduction; heat conduction through overburden to land surface; no groundwater flow

Boundary conditions: Given heat flux condition at lower boundary (natural geothermal gradient at bottom boundary; second-type b.c.); fixed temperature at opposite lateral boundaries (first-type b.c.); given temperature at surface boundary (third-type b.c.); no areal ground-water recharge from precipitation; no pumping or injection of water in wells; zero ground-water flux at lower, lateral, and upper boundaries.

Objective: To qualitatively evaluate conductive heat flow through aquifer resulting from first-, second- and third-type heat flow boundary conditions.

Results: Problem has zero ground water flow; heat in-flux occurs along lower and upper boundaries, and along upper part of high temperature boundary; heat out-flux occurs along lower part of high temperature boundary and along low temperature boundary (see contour graph).

Evaluation: Results conform to expected behavior (qualitative conceptual test).

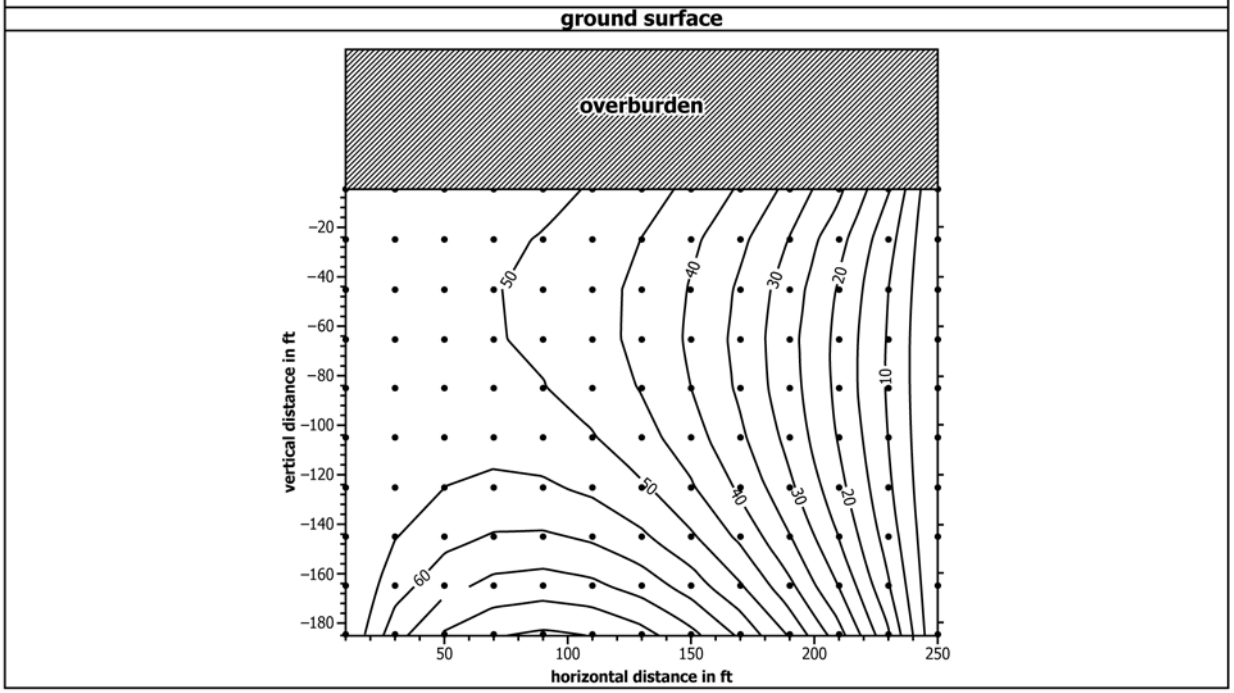

FIG. 3 Example of a Conceptual Test Problem: Temperature Distribution (°C) in a Homogeneous Aquifer (6)

of a software development project, it is often left until the project is otherwise complete. This practice makes documenting the code difficult because it requires searching old (often incomplete and incomprehensible) records prepared by personnel that may have left the project, or relying on fading staff recollections.

10. Scientific and Technical Reviews

10.1 A complete scientific and technical review of a groundwater modeling code comprises the following:

10.1.1 Examination of underlying concepts, governing equations, and algorithms chosen,

10.1.2 Evaluation of documentation and general ease of use,

10.1.3 Inspection of program structure and program logic,

10.1.4 Error analysis,

10.1.5 Examination of the computer coding, and

10.1.6 Evaluation of the performed code testing.

10.2 *Model Examination:*

10.2.1 Model examination determines whether anything fundamental was omitted in the initial conceptualization of the reference system. Such a procedure determines whether the

TABLE 11 Overview of Graphical Code Testing Evaluation Techniques (6)

Type of Variable	Type of Graph	Optional Graph
Distribution of dependent variable in space and time	Line graph versus distance for selected times, line graph versus time for selected locations, two- dimensional contour plot, two- dimensional histograms	Two- and three- dimensional iso-surfaces
Deviations in dependent variable in space and time	Line graph versus distance for selected times, line graph versus time for selected locations, two- dimensional contours (for large number of nodes), two-dimensional histograms	
Combination graphs	Line graph of dependent variable and deviations versus distance/ time	
Global mass balance	Line graph versus time	
Iteration error	Line graph versus number of iterations for selected times	

concepts underlying the model adequately represent the nature of the system under study, and identifies the processes and actions pertinent to the model's intended use. The examination also determines whether the equations representing the various processes are valid within the range of the model's applicability, whether these equations conform mathematically to the intended range of the model's use, and whether these equations conform mathematically to the intended range of the model's use, and whether the selected solution approach is the most appropriate. Finally, model examination determines the appropriateness of the selected initial and boundary conditions, and establishes the applicability range of the model.

10.2.2 For complex modeling codes, detailed examination of the implemented algorithms is required to determine whether appropriate numerical schemes have been adopted. This step should disclose any inherent numerical problems such as nonuniqueness of the numerical solution, inadequate definition of numerical parameters, incorrect or non-optimal values used for these parameters, numerical dispersion, numerical instability such as oscillations or divergent solution, and problems regarding conservation of mass.

10.3 *Computer Code Inspection*—In this inspection, attention is given to the manner in which modern programming principles have been applied with respect to code structure, compliance with programming standards, efficient use of programming languages, and internal documentation. This step might reveal undetected programming or logic errors that are hard to detect in verification runs.

10.4 *Evaluation of Code Documentation*— The documentation is evaluated through visual inspection, comparison with existing documentation standards and guidelines, and through its use as a guide in preparing for and executing code testing runs and example problems.

10.5 *Evaluating Ease of Use*—The data files provided by the model developer are used to evaluate the operation of the code and the user's guide through a test-run process. At this stage, special attention is given to the rules and restrictions necessary to operate the code ("tricks," for example, to overcome restrictions in applicability), to the code's input file structure, and to the user-interface, if present.

11. Keywords

11.1 benchmarking; computer code; groundwater; groundwater modeling; model testing; quality assurance; verification

TABLE 12 Use of Graphical Evaluation Techniques (6)

Test Problem Dimensionality	Graph Type				
	Contours of Spatial Distribution	Line Graph of Spatial Distribution	Line Graph of Behavior in Time	1-D Histogram of Spatial Distribution	2-D Histogram of Spatial Distribution
1-D	...	yes	at selected locations	yes	...
2-D horizontal	areal	for selected lines parallel to axes in middle of model domain and at edges and for lines under 45° with axes (separate graphs for each data-pair set)	at selected locations (dependent variable)	at same locations as line graph (deviations; combine with line graph for data-pair set)	for rectangular grids only
2-D vertical	profile	for selected lines parallel to axes in middle of model domain and at edges and for lines under 45° with axes (separate graphs for each data pair set)	at selected locations (dependent variable)	at same locations as line graph (deviations; combine with line graph for data-pair set)	for rectangular grids only
Radial-symmetrical	areal	for 2 axes and for a line under 45° with the axes (combination plot of all three data-pair sets in separate graphs for variable and deviation)	at selected locations	at same locations as line graph (deviations; combine in separate graph for each data- pair set, with line graph)	for rectangular grids only
3-D	selected slices and profiles	for selected lines parallel to the axes and at a 45° angle with the axes	at selected locations	at same locations as line graph (deviations; combine with line graph for each data- pair set)	for rectangular grids only; same slices and profiles as used for contours
Transient	at selected times	at selected times	for linear, logarithmic, or user-defined time- stepping	at selected times	at selected times

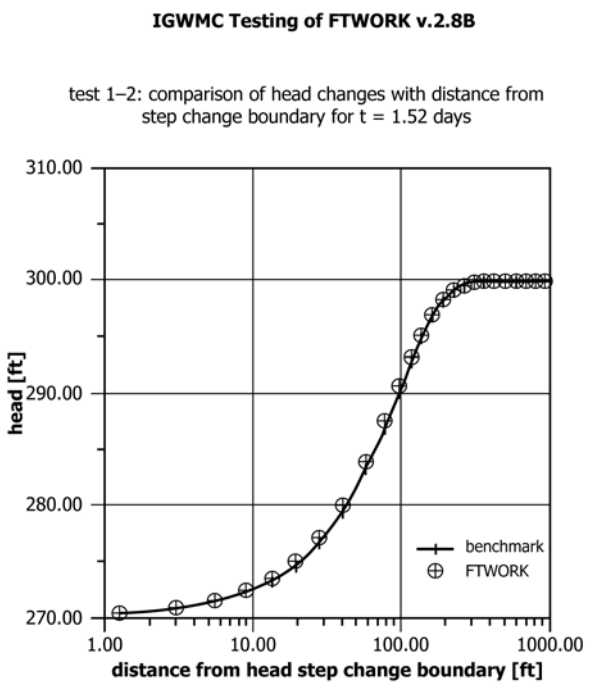

FIG. 4 Graphic Comparison of Heads versus Distance Computed by Code and Generated by Benchmark (6)

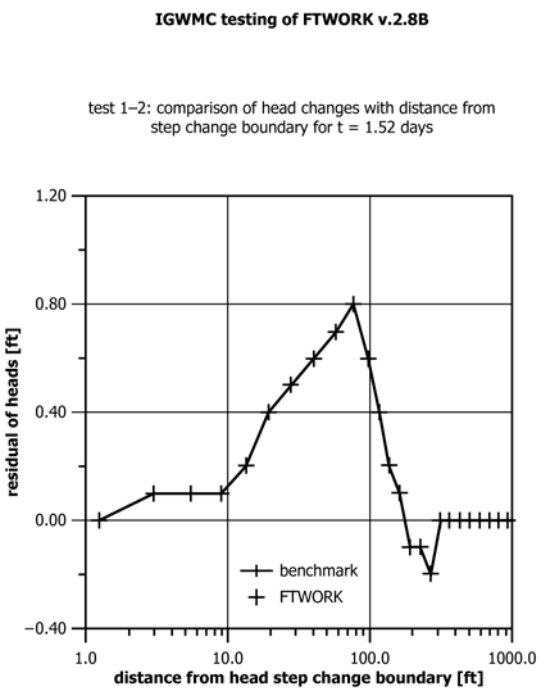

FIG. 5 Graphic Comparison of Head Residuals versus Distance (6)

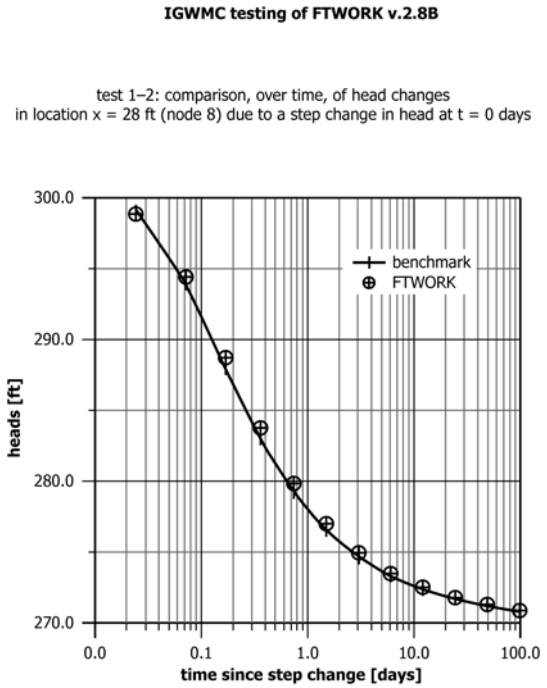

FIG. 6 Graphic Comparison of Heads versus Time (at given location) Computed by Code and Generated by Benchmark (6)

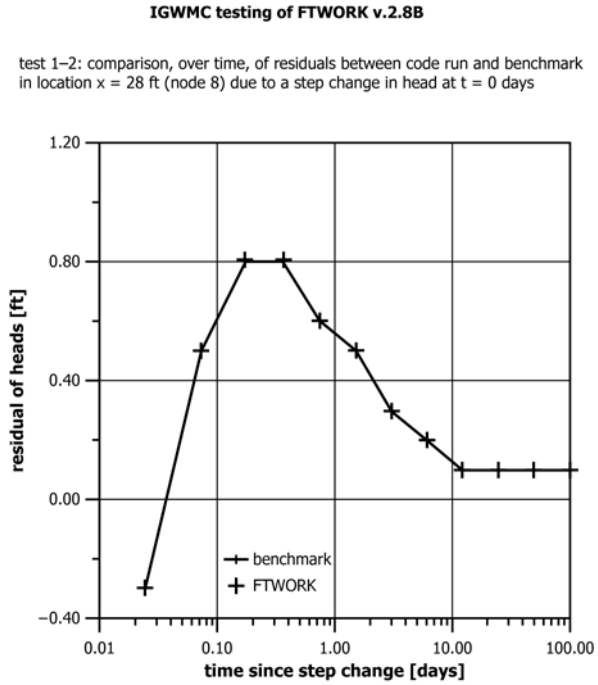

FIG. 7 Graphic Comparison of Head Residuals (at given location) versus Time (6)

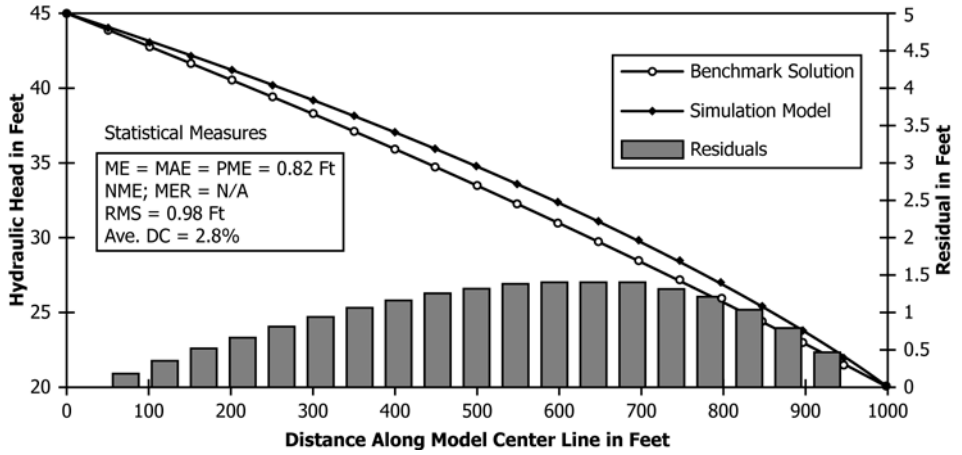

FIG. 8 Combination Plot of *X-Y* Graph of Dependent Variable and Column Plot of Residuals (6)

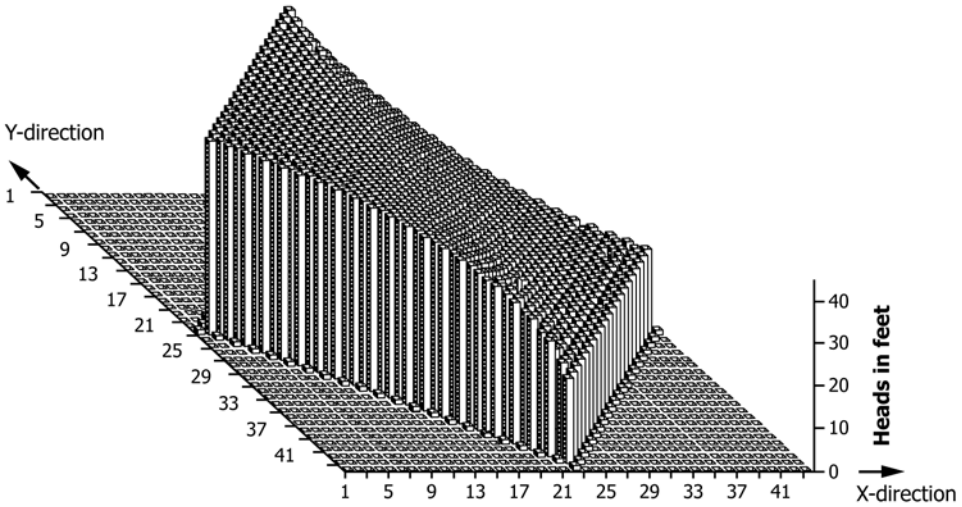

FIG. 9 Quasi-three-dimensional Graphic Representation of Computed Heads (6)

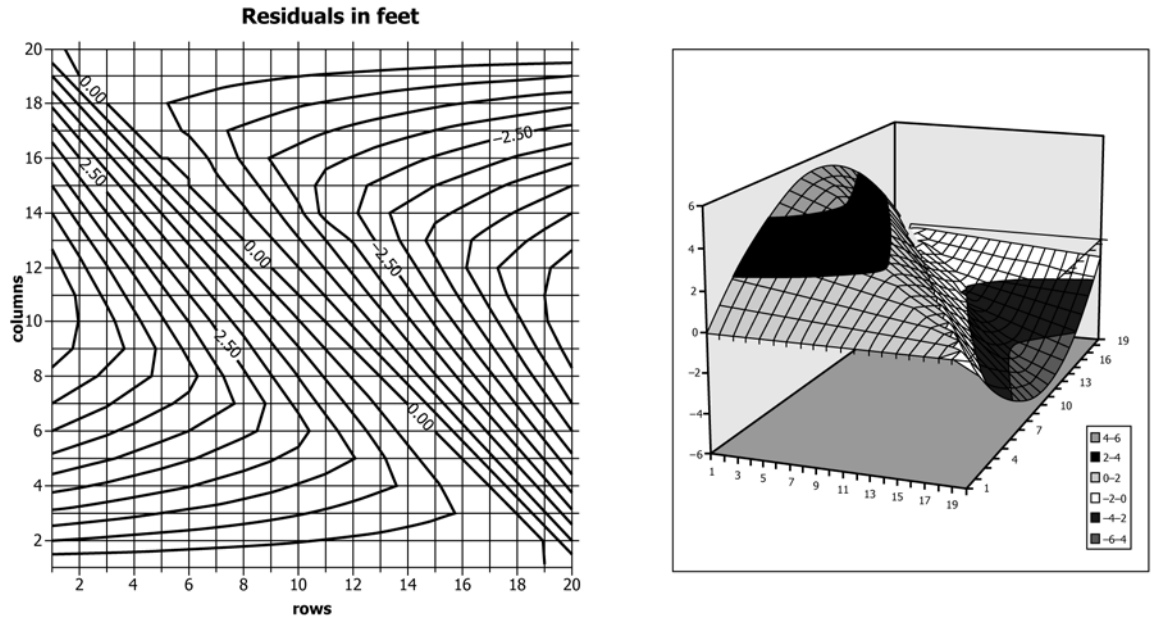

FIG. 10 Three-dimensional Graphic Representation of Head Deviations (Residuals) Between Code Result and Benchmark (6)

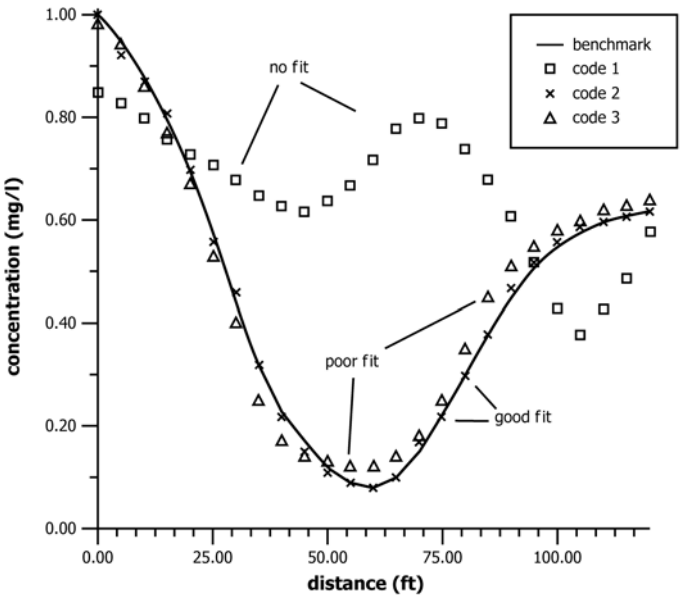

FIG. 11 Qualitative Evaluation of Graphic Comparison (6)

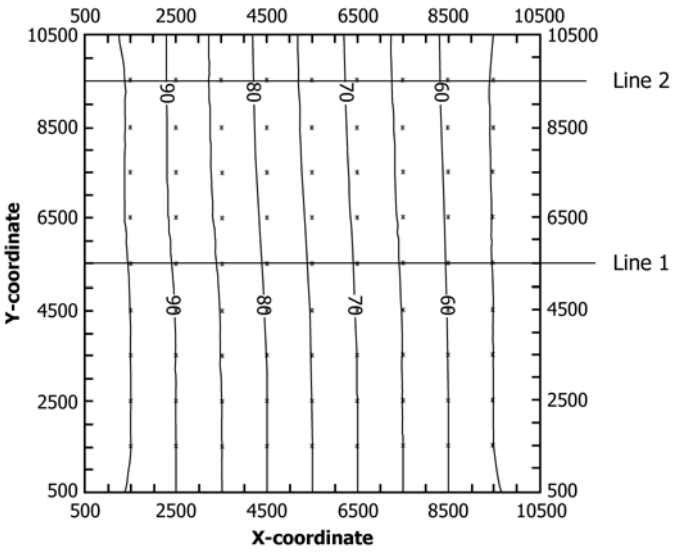

FIG. 12 Representative Sets of Spatially Defined Data Pairs for Intercomparison: One-dimensional, Uniform Flow Case (6)

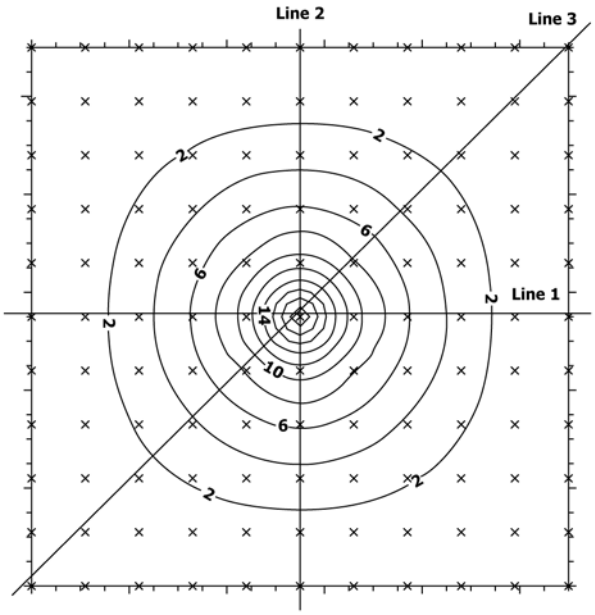

FIG. 13 Representative Sets of Spatially Defined Data Pairs for Intercomparison: Radial, Confined Flow Case (6)

REFERENCES

(1) van der Heijde, P, K. M., and Elnawawy, O. A., *Quality Assurance and Quality Control in the Development and Application of Ground-Water Models*, EPA/600/R-93/011, R. S. Kerr Environmental Research Laboratory, U.S. Environmental Protection Agency, Ada, OK, 1992.

(2) Adrion, W. R., Branstad, M. A., and Cherniasky, J. C., "Validation, Verification and Testing of Computer Software," *Software Validation, Verification, Testing and Documentation*, Petrocelli Books, Princeton, NJ, 1986.

(3) National Research Council (NRC), Committee on Ground-Water Modeling Assessment, Water Science and Technology Board, *Ground-Water Models: Scientific and Regulatory Applications*, National Academy Press, Washington, DC, 1990.

(4) Bredehoeft, J. D., and Konikow, L. F., "Ground-Water Models: Validate or Invalidate," *Ground Water*, 31(2), 1993, pp. 178–179.

(5) Bryant, J. L., and Wilburn, N. P. *Handbook of Software Quality Assurance Techniques Applicable to the Nuclear Industry*, NUREG/CR-4640, Office of Nuclear Reactor Regulation, U.S. Nuclear Regulatory Commission, Washington, DC, 1987.

(6) van der Heijde, P. K. M., and Kanzer, D. A.," Ground-Water Model Testing: Systematic Evaluation and Testing of Code Func-tionality, Performance, and Applicability to Practical Problems," EPA/600/R-97/007, Office of Research and Development, U.S. Environmental Protection Agency, Ada, OK, 1996.

(7) Donigian, Jr., A. S., and Rao, P. S. C., "Example Model Testing Studies," *Vadose Zone Modeling of Organic Pollutants*, S. C. Hern, and S. M. Melancon, eds., Lewis Publishers, Chelsea, MI, 1986.

(8) Gass, S. I.," Computer Model Documentation: A Review and an Approach," *NBS Special Publication 500-39*, Institute for Computer Science and Technology, National Bureau of Standards, U.S. Department of Commerce, Washington, DC, 1979.