



Standard Guide for Developing and Evaluating Groundwater Modeling Codes¹

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 programming 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

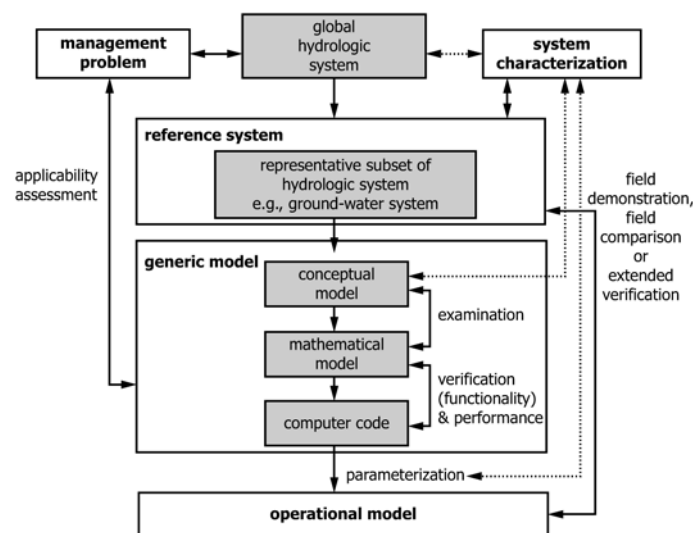


FIG. 1 Concepts and Terms Related to the Development and Testing of a Groundwater Modeling Code

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 walk-throughs. 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 and simulations to

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

General Model Capabilities:	Parameter Representations:
Uncoupled Darcian groundwater flow and nonconservative single-component solute transport in saturated porous medium	Hydraulic conductivity: heterogenous (variable in space), anisotropic
Distributed parameter specification	Storage coefficient: heterogeneous
Spatial Orientation:	Longitudinal dispersivity: heterogeneous
1-D horizontal	Transverse dispersivity: heterogeneous
1-D vertical	Sorption coefficient homogeneous (single value for total model area)
2-D horizontal	Decay coefficient: homogeneous
2-D vertical	Fluid Conditions:
Quasi 3-D (layered)	Density constant in time and space
Fully 3-D	Viscosity constant in time and space
Grid Design:	Boundary Conditions for Flow:
1-D, 2-D, or 3-D block-centered finite difference grid with constant or variable cell size	Fixed head
Time Discretization:	Prescribed time-varying head
Steady state flow	Zero flow
Transient flow	Fixed boundary flux
Transient transport	Prescribed time-varying boundary flux
Variable time step size	Areal recharge variable in space and time
Multiple transport time steps per flow time step	Induced recharge from or discharge to stream; stream need not be directly connected to groundwater
Multiple flow time steps per stress period	Drains
Variable stress periods	Evapotranspiration dependent on distance
Matrix Solvers:	Surface to water table
SOR	Free surface, seepage face
ADI	Solute Transport Processes:
PCG	Advection
Aquifer Conditions:	Hydrodynamic dispersion
Confined	Molecular diffusion
Leaky-confined	Linear equilibrium sorption
Unconfined	First-order radioactive decay
Aquifer Systems:	First-order chemical/microbial decay
Single aquifer	Boundary Conditions for Solute Transport:
Single aquifer/aquitard	Fixed concentration
Multiple aquifers/aquitards	Prescribed time-varying concentration
Variable Aquifer Conditions in Space:	Zero solute flux
Variable layer thickness	Specified constant or time-varying solute flux
Confined and unconfined conditions in same aquifer	Areal recharge of given (constant or time-varying) concentration
Aquitard pinch out	Induced infiltration of given (constant or time-varying) concentration
Aquifer pinch out	Concentration-dependent solute flux
Changing Aquifer Conditions in Time:	Sources/sinks:
De-saturation of cells at water table	Injection/production well with constant or time-varying flow rate
Re-saturation of cells at water table	Injection well with constant or time-varying concentration
Confined/unconfined conversion	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

test problem objective	functions				
	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)

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

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.

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 feature).