INTERNATIONAL JOURNAL OF NUMERICAL MODELLING : ELECTRONIC NETWORKS, DEVICES AND FIELDS,
Vol. 9, 159–165 (1996)

# VERIFICATION OF COMPUTATIONAL ELECTROMAGNETIC PROGRAMS

NATHAN IDA

*Department of Electrical Engineering, The University of Akron, Akron, OH 44325-3904, U.S.A.*

## SUMMARY

Although software testing comprises a large body of knowledge and is extensively used in verification and validation (V&V) of software and software systems, the need to verfiy engineering software posses unique challenges. Electromagnetic programs fall under the category of so-called non-testable programs, a group that includes numerical computation and most other programs that use floating point operations. There are, however, certain techniques that can be used to check for program correctness in the context of expected behaviour and results. The approach to testing by multiple methods, multiple codes and multiple algorithms is well known as an effective tool in testing of scientific software. In addition, testing against canonical and known solutions, evaluation of programs based on properties of the computation performed and error analysis are all common techniques used to verify computer programs. Their use in the context of the TEAM series of problems and workshop is used as an example for a unique method of verification of programs and to underscore the needs of both users and program developers in this important issue of code validation.

## INTRODUCTION

Although it is intuitively understood what the terms testing, verification and validation mean, it is useful to define them here, especially because they are often used interchangeably. Testing is the process by which programs are checked to ensure that they perform the intended functions. Verification is formally defined as the process by which a program is evaluated during each life cycle phase to ensure it satisfies the requirements set forth in the previous phase, while validation is usually understood as testing of the software at the end of the development phase to ensure it satisfies its requirements. It is commonly understood that verification and validation must be an integrated process. For this reason, this work uses the term verification to describe the whole, integrated process.

Verification of computer programs takes many forms and varies with the type of programs that require verification. The intended use of programs often dictates the type, extent and, indeed, the need for verification. Some programs only require the execution of a preset sequence which, once tested, cannot result in faulty output. For example, some computer games or controllers are of this type. Others, because of their critical nature require that all, or many, of the possible of the paths through the software be tested and verified to ensure against catastrophic outcomes. Examples of this type are air traffic control, some financial software, life support systems and the like.

Electromagnetic programs cannot be seen as very large or extensive in comparison with some other software systems. Also, because they often rely on existing subroutines and libraries the development of a new program may actually be an extension of an existing program. It would seem therefore that verification of these programs should be a relatively simple proposition. However, by using numerical methods we have traded in the exact theoretical result for an approximation as a means of extending the range of solvable physical problems. Therefore, our results cannot be 'correct' in the true sense. The only thing we can assume with certainty is that our results are a 'good' approximation, often to an unknown result.

This aspect of scientific computation dictates the main concern in all numerical computation, including electromagnetics: how can we verify the correctness of the programs and the accuracy of the results. These concerns are based on two premises:

1. The result is not in general known. This must be so, otherwise there would be no need for a program in the first place.
2. All floating point computation introduces errors. Because of this, correct programs, used on correct data, produce inexact results. In this context the purpose of verification is to ensure that the results are a good approximation.

CCC 0894–3370/96/020159–07

*Received 15 April 1995*
*Revised 30 August 1995*

These two aspects of scientific computation mean that testing methods must be found which, in the absence of known results, can still provide a measure of program correctness.

This paper discusses the subject of program verification as it relates to computational electromagnetics programs. No attempt is made to dwell on the theoretical subjects of testability but rather to discuss the general subjects of verification of program correctness as it relates to scientific, numerical calculations and the issues associated with it. After a short review of general concepts of verification and testing of software, we discuss general testing methods applicable to general electromagnetic programs followed by discussion of the TEAM experience in testing of electromagnetic software. TEAM (Testing Electromagnetic Analysis Methods) is an ongoing experiment in software and model verification undertaken more than ten years ago for the specific purpose of verifying electromagnetic formulations and computer programs.

## SOME ASPECTS OF PROGRAM VERIFICATION AND TESTING

Verification of a program means different things to different people. For a programmer who is not involved in the development of the model or the whole software, verification may mean simply that the program is free of syntax and run-time errors. For the designer of a computer game, verification means that the program yields the intended results for intended inputs. Verification of numerical programs has all of the above requirements but also requires that the results obtained are accurate: that is, verification also means that the program produces accurate (but not necessarily exact) results.

Because there are different types of programs and different requirements, verification also varies from one program to the other. However, it is understood that program verification is the process by which the program is deemed to be 'correct' in some, yet undefined, sense. To verify a program's correctness we must test the program, that is, the program must be subjected to a series of tests, that will either show that the program is 'correct' or otherwise detect possible errors in the program which can then be corrected.

There is little disagreement that programs should be tested. Even the best software designer will agree to this, and certainly the user will. But there is wide disagreement as to what constitutes program testing and how extensive testing should be. Should one try to exhaustively test a program to ensure that no errors of any kind exist? If so, are we willing to invest the resources needed to do so? Postulating a tradeoff between exhaustive testing and resources may be a middle of the road solution but then the question is: when do we stop testing, and what are the consequences of incomplete testing? These are subjective arguments. There are also objective arguments: can software be tested to ensure that it is entirely free of errors, are there differences between different types of software, and who should do the testing? In addition, there are other complicatinng factors in software verification. There is an almost universal dislike for testing of software on the part of all involved. Developers and users dislike it mainly becuase it takes time, sometimes more than half of all development time. Many are humbled by the testing experience but, most important, we feel that bugs in software are somehow the result of our own failure. That is, we feel that if we were more careful, thought a little more about it, or were more adept at coding, bugs should have never occurred in the first place. Fortunately, software bugs, while certainly important in electromagnetic software, are almost never the main issue in program verification: the model on which the program is based and numerical errors are the main target.

To answer some of these questions, it is important to first classify both programs and testing methods and to relate these to verification of electromagnetic (or more generally, numerical) software.

## TYPES OF PROGRAM

For the purpose of this paper, programs may be classified as (1) non-numerical and (2) numerical.

### *Non-numerical programs*

These programs are understood to perform operations which lead to known, exact solutions. That is, for any given set of inputs a known output or outputs are obtained. In controlling traffic lights, any sequence of inputs will result in an exact (even if incorrect or catastrophic) sequence of outputs.

The question of accuracy has no meaning. Although there are different types of non-numerical programs, this general description should suffice.

*Numerical and scientific programs*

The main difference between numerical and non-numerical programs is that in numerical programs the results are often unknown and are never exact. This means that testing of numerical programs must rely on different techniques. Firstly, we cannot rely on comparison with exact, known values for two reasons: these are usually not known (otherwise there will be little need for the program); if results are known, such as from an analytic solution devised for comparison, the program, which is almost always based on approximations, cannot be expected to reproduce exact results. However, we can say that if the two results are close (by whatever measure we wish), the numerical program approximates the exact results 'well' or 'very well' or 'satisfactory'. Thus, the test we just performed has served in increasing our confidence in the program. But will it perform similarly well on some other problem for which we do not know the results? Perhaps not, but if we perform a number of such tests, an acceptable level of confidence will have been reached to the point that we may consider the program to be 'correct'.

This classification leads to another: that of testable and non-testable programs.[1,2] Testable programs are those for which the correct results can be determined. Non-testable programs are those for which the exact results either cannot be determined or it is not practical to do so. Clearly numerical programs fall under the second category. This is a revealing point: our purpose in verifying electromagnetic programs is trying to test the non-testable. That is not to say that we cannot verify program correctness and accuracy. We can but we must also remember that because of the inherent numerical errors involved we can never be sure how close our solution is to the exact solution.

For this reason, much of the effort in verification of electromagnetic programs is based on multiple coding and comparison of results with experimental and analytical results and with results from other programs.

## TESTING METHODS

Testing methods can be classified in two very broad categories:[3–6]

1. *exhaustive testing*
2. *partial testing.*

In exhaustive testing all possible routes in the program are tested. Once these are all found to be correct, the program is said to be correct. This is easily done on small, simple programs, but is very difficult and expensive on large programs. In numerical software, extensive testing can only guarantee that the program is correct, not that its results are accurate. In most cases, only partial testing can be performed, in which case, we infer the correctness of the program from the tests performed. Clearly this cannot guarantee program correctness but can increase our confidence in the program.

There are two general methods of testing:[4,7]

1. *Black box testing*: The program is viewed as a black box with inputs and outputs only. The internal structure of the program is either not known or is disregarded in the testing phase. Tests are performed by providing inputs and analysing outputs. The outcome of these tests then tells the user whether the program is 'correct' or 'incorrect'. This type of testing method is also called *specification based testing* or *functional testing* and is most often performed by outside users of the programs rather than by the developer of the program. For example, the user will only have an executable code and therefore the structure of the program itself, its language or, for that matter, the computer itself must all be disregarded and attention focused on inputs and outputs.
2. *White box testing*: The internal structure of the program is known to the user. While black box testing may still be performed, the user will often use knowledge about program structure to design tests which are most likely to reveal errors. This type of testing, also called *program-based testing* or *structural testing*, is often performed by the developer of the program rather than by an outside user. In this type of test, the tester has much more freedom in taking into account program structure, and other parameters.

In general it is believed that black box testing can reveal all errors in a program but this would require an infinite sequence of tests and therefore cannot, in general, ensure program correctness. White-box testing is a finite process but cannot detect all errors in the program.[8] Either way, in the context of numerical programs, neither can guarantee correctness or accuracy.

## VERIFICATION OF ELECTROMAGNETIC PROGRAMS

In light of the preceding discussion, how can we test electromagnetic software? No single answer to the problem exists: once we have traded the exact solution for an approximation, we can only hope for an approximate solution. Therefore, since an exact solution cannot exist it is also not possible to assume that testing will tell us that the solution is correct in the absolute sense. However, there are a number of methods that can be used to decrease the possibility of incorrect results:

1. Standard methods of path traversal should be used to ensure program constructs are correct. These will usually be based on flowcharts of the program and should eleminate all bugs which cause software failure. Indeed, we may view this step as part of the development stage since it is common to all software development. Before this is done, the software is not usable and the special issues associated with the numerical aspects of the software cannot be addressed. On the other hand, software bugs can affect numerical results. This means that throughout the verification process we may have to look for software bugs.
2. There are some additional steps that can be taken by the developer to ensure program correctness, at least to some degree. One is to check physical laws associated with the computation: is the circulation of the magnetic field correct, is the energy in the system correct, does power propagate in the required direction, are the divergence conditions satisfied: These will often point to problems in the program, the mathematical model (formulation) or both, and should give a good indication on how to proceed.

   A second useful indicator is the rate of convergence in iterative algorithms. Often, errors in the program will cause a lower rate of convergence rather than causing a catastrophic fault.

   A third step that may be taken is to use perturbation methods to verify program correctness even in the absence of known results. Perturbation methods assume that the same calculation, done in a perturbed way, should yield the same results if the program is correct. For example, various calculations may be done in a different sequence, or subroutines may be called in different orders. Any variation in results may indicate faults. Different compilers may also detect errors by virtue of their use of different paths through the programs.

   A fourth step is to vary the precision of computation. Some indication on errors may be had by simply reducing or increasing the precision of the calculation. Excessive sensitivity on computer precision may indicate a problem with the program, the algorithms used, or both.
3. After the program seems to be working, the program should be tested on data for which the solution is known: analytical solutions, simplified artificial problems, or perhaps related subject problems which use the same mathematical equations (pressure or temperature instead of electric potential, for example) may be used for this purpose. In this step, correctness and accuracy can be evaluated, although inferences from these simplified problems to others should be made cautiously.
4. One simple method of verifying programs is to use a second program to solve the same problem. This method, sometimes known as *dual coding*,[1] is useful only if the second program was written independently of the program that is being verified and only as a comparison. That is, if both programs provide identical (or more often, very similar) results, then we may assume the program being verified is correct. Just as well, it may be that both programs are wrong. If the results are different, there is nothing that can be said about either program unless one is known to be correct. The latter is a very dangerous position to take: no program can ever be assumed to be correct in absolute terms. Dual coding is seldom practical as a general technique because of the effort involved, but there are instances where it may be done, at least partially. For example, one may already have a previous code which can be used for this purpose. Or, perhaps, in the development process the developer may decide to implement two or three different formulations in the same program. While this cannot guarantee that the results obtained with any one formulation are correct, comparison of the different results may reveal errors and inaccuracies that can then be used to correct or improve the program.

   There is, however, one particular form of dual coding that is useful in electromagnetics, and in

particular in static fields: that of dual formulations. The advantage of these is that they bound the solution in terms of energy. Because dual methods approach the solution from below and from above, it is possible to decide on how accurate the solution is even in the absence of comparison data.[9] Unfortunately it is not always possible to do so.

5. Verification can be done against independently produced results by different programs, different formulations and, if appropriate, against experimental results. This method, while not always possible, ensures against the possibility of bias which can occur in dual coding (if the same person or group develops both codes). It also provides comparison across a wider sample of results, increasing the confidence in the program. Comparison with experimental results, if these are available, provides some measure of absolute accuracy which is not available in comparison with numerically produced results. That is, with experimental results we have a means of evaluating different programs against absolute (if not exact) results. This method is a modified form of a voting system[1] and is the basis for the TEAM approach to testing.

In all of these, either black-or white-box testing may be used. In most cases related to electromagnetic programs, it seems that white-box testing is more common as the developer often does the testing. Although it is normally not advisable that the developer of the program also does the testing, it is often a necessity, especially with white-box testing.

## THE TEAM EXPERIENCE

A successful model for verification of electromagnetic computation methods and software is afforded by the TEAM series of workshops. The TEAM series of workshops originated in 1985 as a means of comparing eddy current codes, but it later expanded to include other aspects of computational electromagnetics including static and high frequency applications.[10] The basic idea behind these workshops is a series of 'problems', each geared towards some aspect of computational electromagnetics. In this model, a person or group defines a completely specified problem. That is, the proposers choose the subject and extent of the problem, compile, generate, measure or compute the results one should expect of a correct computation, write a detailed statement of the problem, and present the final statement to the TEAM group for possible adoption. Although TEAM has a governing board, problem adoption is done in a forum open to all participants in TEAM activities. Once a problem is adopted it is published and available to anyone interested in using the problem, its results and any results that may be generated by participants. The problem definition contains comparison results, tables or plots that the user may compare to, and references to the sources of the problem and available published results. Workshops are scheduled at various intervals, ranging from a few weeks to a few months, in which participants present, discuss, and summarize their experience with their own software. Results and methods are compared in informal presentations which are normally only finalized on the day of the presentation or shortly before. TEAM workshops are normally held in conjunction with other relevant meetings and conferences, and are organized in rounds, each round lasting about two years. The number of workshops in each round varies from about two to five. Each round is concluded with a final or global workshop, normally in conjunction with the Compumag conference. Most decisions regarding TEAM problems, such as adoption of new problems, closure of old problems, location of new workshops and the like are taken at this workshop in an open forum.

To encourage open discussion TEAM publishes all presentations in a proceedings of each workshop but does not review any of the results. Formal papers are discouraged, while discussion of risky, questionable, and incorrect results as well as partial solutions is encouraged. All discussion related to given problems is encouraged including software issues, formulations, presentation of results, and errors. After presentations on a specific problem, the results are summarized, and differences, trends in results and errors are analysed. There will usually be no conclusions associated with these summaries: the conclusions are left to the code user.

After a particular problem has been solved by many groups, using as many formulations and programs as is practical, and over the course of perhaps ten or more workshops, the participants may deem a problem to be adequately solved. At this point it is understood that little is to be gained in additional solutions and the problem is closed. A complete summary is prepared including all relevant results, comparisons, errors, and any experience that has accumulated, and the summary is submitted for publication in a journal for the benefit of the computational electromagnetics community. All problems remain in a sense open since anyone may decide to revisit a particular problem at any point in time, especially if there is something new to be learned about the problem or the results.

Table I. TEAM problems

| Prob. | Description/type | Status | Reference |
|---|---|---|---|
| 1 | Cylinder in uniform, transient field (3-D, transient) | S | 11 |
| 2 | Infinite cylinder in magnetic field (2-D, steady state) | S | 11 |
| 3 | Coil over a conducting plate with holes (3-D, steady state, multiply connected) | S | 11 |
| 4 | Rectangular aluminum block, with rectangular hole in exponential field | S | 11 |
| 5 | Four aluminum cubes enclosed within an iron box under laminated iron pole | S | 11 |
| 6 | Hollow sphere in sinusoidal field (axisymmetric, steady state) | S | 11 |
| 7 | Plate with hole (like problem 3 but thicker conductors) | S | 12 |
| 8 | Coil above a crack (3-D, low frequency, low-level fields) | S | 12 |
| 9 | Moving coil in a cylindrical tube (axisymmetric, velocity effects) | S | 12 |
| 10 | Plate over a coil (3-D, transient, non-linear) | S | 13 |
| 11 | Sphere in step field (similar to problem 6, but transient excitation) | S | 12 |
| 12 | Cantilevered beam in crossed field (3-D, moving conductor) | S | 12 |
| 13 | Non-linear steel channels (non-linear, 3-D, static) | A | 14 |
| 14 | Eddy current losses in Euratom LCT coil (pulsed excitation) | S | 12 |
| 15 | Rectangular slot in a thick plate: a problem in non-destructive testing (3-D, steady state) | A | |
| 16 | Magnetic damping in torsional mode (3-D, coupled problem) | S | 14 |
| 17 | The jumping ring (transient, coupled problem) | A | |
| 18 | Waveguide loaded cavity (2- or 3-D, high frequency) | A | |
| 19 | Microwave field in a loaded cavity (3-D, lossy materials) | A | |
| 20 | 3-D static force problem (3-D, non-linear) | A | |
| 21 | An engineering oriented loss model (3-D, eddy current, non-linear) | A | |

S = solved and closed; A = active.

The purposes of the problems in TEAM are:

1. to test both formulations (that is, the mathematical models) used to build the program and the programs themselves. Although the stated goal is the testing of analysis methods through comparison of results, they also verify program correctness and, more importantly, program accuracy. For this reason, problems are chosen to encompass all or most of the important aspects of computational electromagnetics. Participation in TEAM activity, which does not carry any 'reward' such as formal refereed publications is, to a very large extent, based on the value participants place on their ability to gain confidence in their own software or software supplied by software vendors. In the latter case, TEAM problems supply independent testing of the programs under realistic conditions
2. to detect formulation and program characteristics which must be modified and improved. Often these characteristics are not necessarily errors but have, perhaps, slow convergence, inefficient algorithms or methods of presentation, for example
3. to encourage unbiased testing with multiple formulations and multiple programs by as many different programs as is practical
4. to share experience gained in solving various problems or in the development of programs
5. to create a repository of solved problems which can then be used by developers to verify new programs, and extensions and modifications of existing programs.

## TEAM PROBLEMS

To ensure a balanced approach to testing of various electromagnetic problems, TEAM has adopted a variety of problems. Some are intended to test static fields programs. Others are useful for quasistatic applications and still others for time-dependent fields. Within these, there are various levels of difficulty. Results provided with the problem are either experimental or analytical, and in some cases numerical. Emphasis is on verifiable results, and some problems have been verified experimentally

or analytically by more than one source. To date a total of 21 different problems have been adopted, some that have been closed and some that are still open. Table I summarizes the TEAM problems, their levels and topics, and their status, together with some reference material.

Although the TEAM approach to verification is very broad it may be viewed as a modified form of white-box testing, combined with multiple coding. Unlike many dual coding techniques, which are often written by the same designer or design group, the multiple codes participating in TEAM are written by independent groups, ensuring unbiased testing. The very longevity of TEAM points to the considerable success of its stated purpose.

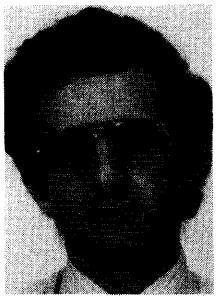
## CONCLUSION

Testing of electromagnetic programs has unique challenges resulting primarily from the numerical nature of the algorithms involved and the unavailability of reliable test data for verification. Although algorithm correctness can be ensured by standard testing techniques, the accuracy of the solutions cannot be tested using these methods. The latter challenge is met by comparison of results produced by multiple codes and multiple formulations. The role of benchmark problems available through the TEAM series of problems and workshops was emphasized as a successful model for program verification.

## REFERENCES

1. E. J. Weyuker, 'On testing nontestable programs,' *Comput. J.*, **25**, (4), 465–470 (1982).
2. J. H. Fetzer, 'Program verification: The very idea,' *Commun. ACM*, **31**, (9), 1048–1063 (1988).
3. J. Voas, L. Morell and K. Miller, 'Predicting where faults can hide from testing,' *IEEE Softw.*, **8**, 41–48 (1991).
4. B. Marick, *The craft of software testing*, Prentice Hall, Englewood Cliffs, NJ, 1995.
5. B. Beizer, *Software system testing and quality assurance*, Van Nostrand Reinhold Company, NY, 1984.
6. N. Harrington and M. Popper, *Understanding software testing*, Ellis Horwood Limited, Chichester, 1989,
7. P. Gilbert, *Software design and development*, Science Research Associates, Chicago, IL, 1983.
8. B. Beizer, *Software testing techniques*, Van Nostrand Reinhold Company, NY, 1983.
9. J. Penman and J. R. Fraser, 'Dual and complentary energy methods in electromagnetism,' *IEEE Trans. Magn.*, **19**, 2311–2316 (1983).
10. L. R. Turner, K. Davey, C. R. I. Emson, K. Miya, T. Nakata, and A. Nicolas, 'Problems and workshops for eddy currennt code comparison,' *IEEE Trans. Magn.*, **24**, 431–434 (1988).
11. *COMPEL*, **7**, (1), (1988), special issue.
12. *COMPEL*, **9**, (3), (1990), special issue.
13. T. Nakata, N. Takahashi and K. Fujiwara, 'Summary of results of banchmark problem 10 (steel plates around a coil)', *COMPEL*, **11**, (3), 335–344 (1992).
14. T. Nakata, N. Takahashi and K. Fujiwara, 'Summary of results for TEAM workshop problem 13 (3-D nonlinear magnetostatic model)', *COMPEL*, **11**, (3), 345–369 (1992).

**Author's biography:**



**Nathan Ida** was born in Rumania in 1949. He is currently Professor of Electrical Engineering at the University of Akron where he has been since 1985. His current research interests are in the areas of numerical modelling of electromagnetic fields, electromagnetic wave propagation, non-destructive testing of materials at low and microwave frequencies and in algorithms. Dr. Ida received his B.Sc. in 1977 and M.S.E.E. in 1979 from the Ben-Gurion University in Israel and his Ph.D. from Colorado State University in 1983. Dr. Ida has published extensively on electromagnetic field comupation, parallel and vector algorithms and computation and on non-destructive testing of materials. He has written three books, two on computation of electromagnetic fields and the third on non-destructive testing with microwaves.