Computer-Aided Structural Engineering: Dangers/Ethics/Quality & a Return to Engineering Common Sense

The significant benefits that can accrue from the use of computers cannot be gained without applying stringent procedures governing their use.

LEROY Z. EMKIN

The continuing rapid growth in the numbers of affordable and easily accessible computers and computer programs has created a particularly difficult dilemma for the practicing structural engineer. On the one hand, every engineer now can easily afford to use a computer and reap well known benefits, some of which include:

- Higher productivity;
- The ability to solve more complex problems with greater accuracy and speed; and,
- The ability to practice with state-of-the-art techniques of analysis and design.

On the other hand, there are very dangerous pitfalls. Some of these dangers include *the lack of:*

- Software quality and reliability;
- Software and hardware maintenance;
- Education and training in the techniques of effectively integrating computers into engineering practice
- Proper and complete software user documentation;
- Training in software use; and,
- Technical support services.

Among these and other problems, by far the most potentially dangerous and the most difficult to identify, avoid or correct are the problems of inadequate structural engineering software quality and reliability, and the gross misuse of computers for engineering analysis and design.

Background

Although computers have been in regular use in the structural engineering profession for 25 or more years, it has only been in the past eight years or so that their use has become so pervasive. In particular, the introduction of workstations and personal computers, combined with the rapid increase in performance and decrease in cost of such computers, has led to a surprising (and possibly disturbing) surge in the use of computers by large numbers of engineers who may not be fully aware of all of the consequences of computer use in professional practice.

Concomitant with the availability of low-cost, high-performance computers is the immense growth of available and affordable engineering computer software. The proliferation of computer hardware and software has not only created outstanding opportunities for the structural engineering professional, but it has also created a particularly difficult dilemma for the practicing structural engineer. While almost every engineer now can easily afford to use a computer, there are some dangerous pitfalls to computer use that include:

- Excessively limited software functionality;
- Shallow technological capabilities of engineering software;
- Software limitations on the size of problems that can be solved;
- Inadequate or non-existent database management facilities of the software;
- Grossly inadequate software quality and reliability;
- Insufficient software and hardware maintenance;
- Non-existent education and training in the techniques of effectively integrating computers into engineering practice (*e.g.*, user qualification, software validation, result validation, software vendor qualification);
- Poor and incomplete software user documentation;
- Training in software and hardware use given by persons with poor qualifications as instructors and/or with inadequate engineering experience;
- Inadequate technical support services from software vendors;
- Gross misuse of engineering computer software; and,
- Improper expectations and beliefs regarding the "problem solving" capabilities of engineering computer software.

The backdrop of inexpensive, powerful and easily available computing tools versus the very real drawbacks associated with their use will be a continuing predicament for the engineering profession.

Among all the problems with computer-aided engineering and computer-aided design (CAE/CAD) that exist today in engineering practice, the most insidious ones are related to the issues of quality of the CAE/CAD world, and the apparent lack of awareness that these problems are as severe as they are. Since the dangers include problems associated with the quality and reliability of engineering design, public safety and liability of engineers, and since the numbers of engineers involved with computers appear now to be increasing rapidly, it is necessary to establish a regular dialog between professionals that addresses these issues, that exposes the shortcomings and that provides guidelines for practice that can be implemented to avoid serious problems.

Myths & Misconceptions About Computers

The computer environment within which an engineer can perform highly iterative engineering modeling, analysis, design and decision-making functions is an imperfect one at best, and a dangerous and error-riddled environment at worst. Numerous myths and misconceptions have evolved over the last 25 years of computer use by engineers. Some of the more disturbing ones include:

- Computers perform numerical computations without error. (The recent problem with the Pentium processor underscores this issue.)
- Some computer programs have no bugs.
- If a computer program has been shown to solve a certain type of problem without error for one set of data, it will solve exactly the same type of problem with different data without error.
- A computer program that has had no changes made to it whatsoever will compute the same results every time it is executed with the same data.
- All computers are equally precise.
- If a computer program runs without error on one computer, it will run without error on another computer of the same or different model.
- Developers of engineering computer software have sufficient education, knowledge, experience and practice qualifications.
- There are widely accepted guidelines pursuant to which computer software quality assurance (QA) and quality control (QC) procedures are performed on a regular basis.
- Engineers use computers best by specifying numerical data to the computer, and by having the computer perform the complex and tedious numerical computations.

There are many more myths and misconceptions about computers in engineering. In general, not one of these assumptions is always true; nor are they always false. The only sure thing is that the computer world is saturated with low-quality hardware and software, incorrect and misleading information, untruths, lack of knowledge, poor accuracy, and much error — adding up to great risk for the engineer and for the constituency that is served by the engineer.

More disturbingly, however, is that few, if any, engineering activities include in-depth discussions of these myths and misconceptions. The lack of such discussion simply reinforces the misinformation. How then can engineering professionals properly use CAE/CAD technology in engineering practice if its very use implicitly endorses ideas about computers that are wrong or not fully developed?

Engineering Software Quality Assurance, Quality Control & Verification

The two most disturbing problems in the engineering use of computers today are the problems of a rapidly growing use of large numbers of low-quality and improperly validated engineering software products, as well as the widespread lack of knowledge of both the fundamental issues of the quality of engineering use of computers and the impact of computer use on engineering liability and on the reliability of engineering designs.

The lack of awareness and the lack of knowledge of the seriousness of the impact of the computer environment (including the full range of small to large computers) on engineering practice pervades the entire engineering profession.

For example, a typical letter from an engineering company to a software vendor often takes the following form:

Dear Sir:

My company is currently searching for computer programs for civil/structural engineering analysis and/or *design for our XYZ personal computer. Please mail to us your current price list and a list of available programs together with a brief summary description of each program. If available, we would also like to receive a floppy disk that contains a demonstration version of your program(s). Example of program output formats would also be appreciated.*

Sincerely,
Engineer

Following the receipt of the requested information, the engineer will run the demonstration version of the program and, if the demonstration and the output formats are deemed satisfactory, the engineer typically will send the software vendor a letter similar to the following:

Dear Sir:

The demonstration version of your engineering computer program appears to satisfy our needs. Enclosed please find a check for the full version of your program and associated user manual. Please send the program and manual to us immediately.

Sincerely,
Engineer

The above correspondence is fairly typical (it is based on numerous actual letters from well known and respected engineering companies). However, the exchange is woefully inadequate. There is no reflection whatsoever of any awareness or concern on the part of the engineer of the need to know a lot more about the computer environment and vendor qualifications than simply a successful observation of a demonstration version of a program (probably prepared by marketing, not technical, specialists), and its "pretty" output formats.

Why aren't engineers exercising better professional judgment and simple common sense? Engineering software purchases must not be conducted like buying the family car — that is, a pretty picture, bright colors, great advertising, and a smooth test ride are simply not sufficient justification for purchasing a software product that could cause unsafe engineering designs to be produced. The professional engineer needs to apply the full range of expertise, experience, knowledge and concern for excellence to the evaluation, selection and use of computer technologies for engineering analysis and design.

Unfortunately, it is not fully appreciated by engineers that the maker of the tools used by engineers (*e.g.*, computers, operating system software, compilers, run-time libraries, application software) do not consider themselves liable for bad engineering, nor do they consider themselves liable for unsafe, poorly performing or uneconomically engineered products whose bad design may, in part, be due to incorrect information produced by the tool (*i.e.*, computer technology). Only the engineer of record is responsible and is considered to be liable since the engineer and not the tool maker:

- Makes the appropriate simplifications, assumptions, and approximations;
- Establishes the mathematical models of the products to be designed;
- Defines the data to be used for analysis and design;
- Decides which algorithms and solution methods are appropriate;
- Interprets results;
- Verifies the correctness and/or relevancy of results;
- Implements the results into a design decision-making process; and
- Ultimately approves an engineering design for manufacture or construction.

The tool maker is not involved in any of these activities whatsoever.

Consequently, the practicing engineer bears the full burden of assuring that all processes, algorithms and tools employed in product engineering are correct to the extent that the profession accepts and endorses them as current state-of-practice. Unfortunately, when the tool consists of a computer and all its associated components, it is absolutely impossible for any engineer to investigate each component separately in order to determine that component's accuracy, precision, performance, relevance or any other attribute that will influence the final outcome of an engineering design process. The factors involved are simply too numerous or complex, or they are simply inaccessible to the engineer.

For example, some of the factors that can greatly influence the above attributes are:

- Computer hardware architecture such as word size, word structure, instruction set, hardware-software interfaces, numerical processor design, chip design, memory design, or any one or combination of thousands of components in the computer.
- Operating system software such as computer resource managers, linkers, loaders, compilers and run-time libraries.
- The quality of the algorithm embedded in the computer's run-time library or application software subprograms that actually solves the engineering problem. For

example, a mathematical function such as a logarithm or a cosine may be implemented in such a subprogram so that the accuracy is inadequate (*i.e.*, wrong results due to an incorrect algorithm or the bad implementation of a correct algorithm). Another example is where the computational precision is insufficient, leading to large errors that are due to the accumulation of a large number of very small precision errors.

- Inconsistencies between the problem to be solved and the particular algorithms implemented in the application software used to solve the problem.
- Computer hardware errors that sometimes produce incorrect numerical results that are difficult to detect.
- Errors (bugs) in one or more components of the software.
- Engineering modeling techniques that may be inconsistent with problem solving assumptions embedded in the application software.
- Errors in data provided by the engineer.
- Innumerable other sources of error.

A fact of life is that all (there are no exceptions) commercially available computers and computer software are subject to many of the above factors with varying degrees of influence on their ability to produce correct solutions to engineering problems. Of even greater concern is the fact that when incorrect solutions are produced, they are often not so "wrong" that they can be immediately recognized as incorrect. Furthermore, sometimes the results are grossly incorrect, but if the engineer has little feeling for what the "correct" results should look like (either due to ignorance or lack of experience), it becomes almost impossible to recognize wrong results. The danger of computers is that many engineers always assume (and almost all engineers surely expect) that computers will produce "correct" solutions to problems. Such assumptions and expectations often dull the awareness and sensitivity of the engineer to potential, and often likely, errors produced by the computer.

The problems associated with the quality of the computer environment may appear to be insurmountable. Clearly, the engineer user of computers has no control over the design of computer hardware or its associated operating system/run-time library software. In addition, the engineer user has little, if any, control over the design of the application software products purchased. Even if the engineer develops custom application software, all the factors described above are still relevant and can cause such self-written software to produce incorrect results, where the wrong answers can be just as difficult to detect.

Does all this mean that the engineer should avoid using computers? Should the use of computers be limited to only those applications where the safety of society (*i.e.*, loss of life and/or property due to failures of engineered products) is not affected by wrong computer results (such as in business and office automation software)? Or, is there a way that the problems can be solved with a sufficiently high degree of confidence so that the continued use of computers by more and more engineers can be justified?

Clearly, the problems must be recognized, addressed and solved. If these problems cannot be solved in a satisfactory manner, then two of the most obvious solutions would be to:

- Stop using computers for engineering design; or,
- Subject society served by engineers to the possibly unacceptably high risks that are associated with the problems posed by the use of computers that could result in poor and unsafe engineering.

Neither of these two options is acceptable, especially because society itself makes greater and greater demands on engineers for ever more complex, reliable, safe and economically engineered products. And the computer is the only environment within which the engineer can solve the increasingly complex problems that need to be solved and still deliver such solutions in a profitable manner, in the time required and in a safe, reliable and economical form.

Overcoming Drawbacks

So, what can be done? The answer is simple, but

is not necessarily easy to implement. The solution requires that the following actions be taken:

- No computer software product should be considered for purchase without properly qualifying and certifying the vendor of the software.
- No computer software product should be used for engineering computations unless it has been fully and properly validated and certified pursuant to industry accepted standards of engineering software QA and QC.
- The engineer must apply the same high degree of care and detail when validating computer software accuracy and checking actual computer results as is applied when using time-tested and traditional procedures for checking hand computations (*e.g.*, every data value, assumption and computation must be checked and rechecked prior to use).
- The engineer must use best judgment and full depth of knowledge and range of experience when using the computer. Inexperienced persons without sufficient engineering knowledge that would qualify them to be responsible for engineering designs should not be allowed to use the computer software without careful and complete supervision by a competent, experienced, knowledgeable and responsible engineer. In other words, *a computer program does not make a good engineer, only a good engineer should use a computer program.*
- Engineering management must be more concerned about the quality of engineering computation than ever before when a computer is involved. Engineering managers must set examples and provide incentives and comprehensive training programs for the proper use of computers in the engineering design decision-making process. Those managers who trade away quality of engineering in return for assumed increases in productivity and profitability through the use of a computer are only fooling themselves, their companies and their clients. It takes a major commitment and the up-front investment of people, time and money in order to reap the benefits of computers. There is no easy path to success and profit.
- The engineering education community must recognize the urgent need to include in their curricula the issues of how the computer environment impacts the areas of engineering liability, quality of engineering computations, procedures for assuring software quality and the qualification of engineering computer professionals.
- Professional engineering societies must establish and aggressively promote guidelines for the proper use of engineering tools such as computers (which can have the potential for causing a serious degradation of engineering quality on the one hand, but which are tools, on the other hand, that the engineering professional cannot afford to ignore).
- Government regulatory agencies that are charged with assuring the safety of the general public must develop regulations to protect the public from the dangers of the improper use of computers in engineering design.

While these actions may appear to be fairly obvious responses to the problem, they are by no means easy to apply or enforce. The brief history of the use of computers in engineering (less than a full generation) is replete with examples of the computer élite inside engineering companies who exempt themselves from careful scrutiny by the responsible engineers within their companies. Examples abound of the use of the mystique of computer programming and of computer programmers that assumes that computer programmers and their programs must be correct if they can be shown to correctly solve the "test" problems supplied by those same mystical programmers.

It is more often the case than not that those responsible for developing engineering software are not engineers, or are engineers with little or no experience in "real world" engineering practice, and are almost never engineers with professional engineer's licenses to practice. How then can such programmers be suffi-

ciently aware of the highly sensitive issues associated with quality, reliability, liability and practice of engineering? The programmer programs with the very comforting and sedating knowledge that the professional engineer of record, rather than the computer programmer, is legally liable for the final engineering design.

Furthermore, the actions suggested above are also difficult to implement and enforce because:

- There are simply little or no precedents for implementing or enforcing regulations that control procedures for qualifying, validating and certifying engineering computer software, software programmers and software users; and,
- There is very little available knowledge to help guide engineers and programmers in the *proper* QA, QC, validation and certification of engineering computer software, software programmers, software users and software results.

So, the question must be asked again. What can be done? At this time, probably the best beginning is to focus on the issue of software vendor and software product qualification and certification. Clearly, no other aspect of the solution makes a lot of sense unless the engineering software product and its vendor are properly qualified, validated and certified as to their ability to produce accurate solutions and professional qualifications, respectively. The first requirement must be to assure the quality of software and vendor prior to committing to use the vendor's software.

How can the software publisher and the software itself undergo proper QA inspection with a high level of confidence in the absence of industry standards for QA and QC, and in view of all the problems noted above?

Quality Assurance & Quality Control Solutions

At least two industries within the United States have addressed that question. They are the defense and nuclear power industries. The nuclear power industry has focused on the software products that are more likely to be used by the general engineering profession, such as finite element analysis software. Consequently, some of the solutions to the software and vendor QA and QC problems in the nuclear industry that have evolved over the past thirteen years are probably more applicable.

In 1982, the United States Nuclear Regulatory Commission (NRC) imposed regulations 10CFR50 and 10CFR21, as well as other related ones, on the nuclear power design industry with respect to certain classifications of computer software. These regulations were originally intended to regulate QA and QC procedures related to the design, construction and operation of nuclear power plants in the United States. However, up until that year, these regulations were not required to be applied to computer software used by the nuclear design industry.

The classifications of software to which the regulations now apply include finite element analysis and pipe stress analysis software, among others. Unfortunately, since the regulations were not written to recognize any special characteristics of software, they were not directly applicable. Therefore, it has been necessary to interpret the regulations in the context of computer software and formulate specific procedural requirements that are directly applicable. This process of interpretation and procedure development has been evolving since 1982 by those software vendors delivering software to the nuclear design industry together with the nuclear design firms using the software. In the finite element analysis software field, there are only a handful of products whose QA, QC and validation procedures are in full conformance to the applicable provisions of the NRC's QA/QC regulations. There are numerous other software developers' finite element programs worldwide whose development procedures do not satisfy the NRC's or any other national or international standard's QA/QC regulations.

The NRC 10CFR21, 10CFR50 and associated regulations – as well as other standards such as the American Society of Mechanical Engineers (ASME) NQA 2.7 software quality standards and the International Standards Organization ISO 9000 Part 3 software quality standards – include a vast amount of detailed, often complex, extremely time-consuming and

highly labor-intensive requirements. Given the evolution over the years to their interpretation in an engineering context, these standards and regulations are very expensive to conform to. However, they represent some of the most complete regulations and standards, and are the current state-of-practice for large-scale, state-of-the-art engineering software QA, QC and verification.

While the current interpretations of the regulations and standards, as well as their detailed procedural requirements, are too extensive to describe herein, it is useful to note the fundamentals of the regulations and their requirements. Such a summary can at least provide a basis upon which an engineering company can begin to understand the breadth and depth of detailed QA and QC activities that should be required of any and all vendors of engineering software under consideration for use. The QA/QC standards require the following procedures and activities:

QA/QC Procedural Manual. A document must be prepared, and continuously updated as necessary, which fully describes in detail all procedures, flow of information, approval forms, responsible personnel, acceptance criteria, records management as well as any other matters relating to the process by which the engineering software product is quality assured, quality controlled, verified and certified.

Vendor Qualifications. Records must be kept that document the qualifications of the personnel who are directly involved in any aspect of software development or validation. Qualifications must note education level, fields of expertise, scope and depth of knowledge, range of professional experience and professional engineering licenses to practice.

Documentary Evidence. Separate and complete evidence of each and every change made to the software must be maintained, including all bug fixes, functional improvements, feature enhancements, performance improvements and any other modifications whatsoever.

Base Verification. Each new release of software, as well as each computer type upon which the software is intended to be executed, must have a separate base verification process performed. This process involves the execution of a collection of test problems (for example, approximately 150 such problems) that are intended to verify the baseline computational accuracy of the software. All test results must be fully documented in a separate document often referred to as the base verification manual.

General QA. Although the base verification process is appropriate and necessary for purposes of detailed documentary evidence of computational accuracy, it is not sufficient to provide the highest level of confidence in such accuracy for engineering software used in production engineering design. Therefore, for each new release of the software, and for each computer type upon which the software is intended to be used, a much larger set of test problems (for example, 2,000 such problems) must be executed. This larger set is also used to verify computational accuracy. In addition, it is used to verify the consistency of results from one release of the software to the next, the conformance of the execution of the software to the user manual documentation and the performance of the software from one release to the next.

Archiving. For each new release of the software, and for each computer type upon which the software is intended to be used, a full copy must be prepared, in both machine readable and human reproducible form, of the complete source code, object libraries, base verification results, general QA results, documentary evidence of all software modifications, changes to user manual documentation, and all procedural forms and approvals. All this material must be stored in proper containers and archived in a vault off-site from where the software is developed. Access to the vault must be properly controlled and limited to only authorized personnel. Furthermore, the archived materials must be stored for a period of time equal to 40 years.

External Audits. In order to assure that the NRC QA/QC regulations are being adhered to in a proper manner, it is required that the vendor of the engineering software be

audited on a regular basis by professional technical auditors from outside the vendor's organization. Audits are generally conducted by the company using the software. Such audits are intended to:

- Discover any deficiencies in the vendor's QA, QC and verification procedures;
- Review the technical qualifications of the vendor's personnel;
- Verify the truthfulness of the vendor's representations;
- Review base verification and general QA test results;
- Review archiving security and completeness;
- Verify the thoroughness and accuracy of QA test problems; and,
- Prepare recommendations for improvements, changes, deletions and additions to the vendor's QA/QC procedures and technical personnel.

Audits may occur at a frequency of about once every three or four months.

Letter Notification. Upon confirmation that a bug (*i.e.*, error) exists in the software that causes an incorrect result to be displayed, the software vendor must provide letter notification regarding the existence of the bug within 72 hours (sometimes within 48 hours) to those companies regulated by the NRC and that use the vendor's engineering software.

Configuration Control. In order to assure the integrity of software and the proper management of QA, QC and verification procedures, a comprehensive configuration control mechanism (that is as automated as possible) must be established by the software vendor. Such a mechanism manages the execution of QA test problems, files of source and object code, audit trails of modifications made to software, historical records and other necessary management and control information.

There are many more requirements related to the NRC 10CFR21 and 10CFR50 QA/QC regulations, and the ASME NQA 2.7 and ISO 9000 Part 3 standards. The above outline provides some insight for engineering organizations that would assist them in formulating questions and demands that should be made to vendors of engineering software. The goal is to substantially increase the likelihood of having access to high-quality software and, thus, provide lower risk to both the engineering software user and the general public whose safety depends on the quality of the engineered products delivered.

Other Issues & a Warning

There are many other issues related to the impact that computers have on the practice of structural engineering. Some of these issues include:

- The characteristics and severe deficiencies of conventional structural engineering computer programs in common use today;
- The benefits of highly integrated structural engineering software systems;
- The dangers of using highly integrated CAD and engineering application software systems;
- The cost-effective and proper use of computers as tools for engineering design;
- The role of natural language processing and expert systems in engineering design;
- Productivity and profitability improvements in engineering through the effective use of computers;
- The use of computers as database-oriented environments for engineering design; and,
- Structural engineering analysis, design and decision-making aided by general-purpose engineering information processing systems.

While it is impossible to argue against the fact that computers are an important computational and data management tool for the engineer, it would be folly to ignore the problems associated with their use. With sufficient standards and regulations within the computing industry as it is applied to engineering, it would be advisable to be wary when using computers and to be especially wary of the developers of

engineering software. The world is full of hundreds of persons and companies claiming to have written, and who peddle, engineering software. But how many of them are sufficiently qualified and understand the issues of engineering software QA/QC? Can they really afford to develop, deliver and support the engineering software upon which they have performed proper QA/QC? Computers, and especially the application software products that run on them, are far from perfect. The safety and satisfaction of the general public that we as professional engineers serve depend on our ability to deliver safe, economical and functional engineered products. Our professional ethical obligations dictate that we care. We, as professional engineers, must not succumb to the blind and uncritical use of computers. We must use our best professional engineering judgment, be guided by our knowledge, experience and common sense, and practice according to the highest level of professional integrity.

ACKNOWLEDGMENTS — *This article is an update of a paper originally published in the Conference Proceedings of the Fourth Conference on Computing in Civil Engineering, ASCE, Boston, Massachusetts, October 1986, and presented as part of the 1995 BSCES Structural Group Lecture Series.*

NOTE — *In the finite element analysis software field, GTSTRUDL, MSC/NASTRAN, ANSYS and a few other programs are in full conformance to the applicable provisions of NRC's QA/QC regulations.*

LEROY Z. EMKIN *received his B.C.E. and M.S.C.E. from the Georgia Institute of Technology in 1965 and 1967, respectively, as well as a doctorate from the Massachusetts Institute of Technology in 1970. He has been with Georgia Tech for 26 years, where he is Professor in the School of Civil & Environmental Engineering. He is also the founder of the Computer-Aided Structural Engineering Center at Georgia Tech. He lectures worldwide on issues of the quality, reliability and cost-effective use/misuse of computers, and on the engineering ethics of using computers in structural engineering practice.*