The Role of Problem Solving Environments in Engineering and Mathematics Education[1]

Elias N. Houstis

University of Thessaly

Department of Computer and Communications Engineering

Volos, Greece

September 17, 2003

Abstract: Computers and information technology (IT) are part of every productive organization of the human enterprise, from manufacturing to entertainment, telecommunications, transportation and education. IT is changing the way we deliver, consume, and administer education. Today, we are educating a new generation of engineers raised on "Sesame Street" like programs, where learning through dynamic visual imagery is emphasized. This generation is immersed in an electronic media world surrounded by television, MP3 music boxes, multimedia cellular phones and Bluetooth/WiFi communication devices, digital cameras, internet café, electronic games and toys. Against this background, interactive multimedia based learning is becoming the norm. At the same time, mathematics is increasingly used in almost all areas of human activity. It is the language in which knowledge is expressed for solving problems with computers. However, mathematics as a discipline affecting only a small minority of the so-called *mathematically gifted* people. Most engineering schools are trying to reconfigure the way they educate the next generation of engineers. Computer based environments, referred as problem solving environments (PSEs), that enable users to express their problems in mathematical terms and incorporate the necessary "solving" knowledge are advancing significantly and are part of every scientific and engineering product design and simulation today. Their role in learning engineering and mathematical problem solving is increasing dramatically. In this paper, we review the trends in engineering education and examine the role of PSEs in learning engineering and mathematical problem solving. In addition, we present a review of the history of PSEs and after stating the definition and the goals of a PSE, we summarize the state-of-the-art, the principal components and paradigms for building PSEs and identify some future trends.

1. CURRENT TRENDS IN ENGINEERING AND MATHEMATICS EDUCATION

According to the National Academy of Engineering report, "Computers and Information Technology (IT) are driving an accelerating increase in the productive organization of the human enterprise, from manufacturing to entertainment, telecommunications, transportation and education" [8, 12]. The developments in IT are changing the way we deliver, consume and administer education. Today we are educating a new generation of engineers raised on "Sesame Street" like programs, where learning is based on dynamic visual imagery. This generation is immersed in an electronic media world surrounded by television, MP3 music boxes, multimedia cellular phones and Bluetooth/WiFi communication devices, internet café, electronic games and

[1] Published in the Proceedings of the 6th International Conference on "Technology in Mathematics Teaching"

toys. Interactive multimedia based learning through collaboration, and "co-location" has become the norm in modern educational paradigms. Simultaneously, there have been significant changes in engineering practice. We are moving from process design to product design, from individual projects to team projects, from experience-based design to model-based design, from calculator-based computations to computer simulations, and from paper-based documentation to IT-based archives covering all aspects of design, building and operation from project conception to end of life. Engineering curricula have to adapt to these changes, as we need engineers adept in computer-aided design tools and IT to execute concurrent, collaborative design projects [8].

Advances in computers and Information Technology provide an unprecedented opportunity to reconfigure the way we educate the next generation of engineers [13]. Faculty now has access to a much greater menu of resources to supplement the classroom teaching. Computers and IT help us create and disseminate new, globally accessible instructional materials [8]. The availability of powerful engineering software enables the engineering student to address real world problems with precision and accuracy at a much greater level of detail than in the past. Web-based delivery and distance learning are increasingly used to reach a geographically distributed student body. No significant effort has been undertaken to provide students with hands-on experience in the use of IT-based tools to function effectively in collaborative projects, to manage resources and data, and to become familiar with e-trade/commerce practices. This experience is essential for future engineers who can be anticipated to function in a global market mediated by computers and the Internet. Major educational paradigm shifts are taking place in higher education, as shown below (see [8]):

Old Paradigm	New Paradigm
Rigid Schedule	Courses on Demand
Terminal Degree	Lifelong Learning
Books as Primary Medium	Information on Demand
Delivery in Classroom	Delivery Anywhere
Bricks and Mortar	Bits and Bytes
Technology as an Expense	Technology to Improve Efficiency and Productivity

These paradigms shifts must followed by changes in pedagogy. Following we list some of the expected changes proposed with respect to the current practice.

From	To
Teacher Centric	Learner Centered
Single Medium	Multimodal Interaction
Individual Work	Contextual Activity
Passive learning	Active Learning
Artificial Context	Situational Study
Didactic Approach	Emphasis on tools and technologies
Problem Solving	Problem Formulation

In the meantime, every scientific, economic, social, cultural, and engineering field has increased significantly its dependence on mathematics. Mathematical modeling and simulation is part of optimal resource allocation in production, transportation, banking /trading, waste management, designing of drugs and engineering artifacts from bottles to planes and buildings, entertainment (computer games, movies), predicting the weather, cutting expensive materials such as leather, textile or wood. It is one of the major learning subjects in any education system. However, mathematics and mathematics learning are in a very paradoxical situation since they are regarded as a discipline that it is accessible only to a small minority of the so-called *mathematically gifted* people [11]. Moreover, there is a negative image associated with mathematics common in public opinion that prevents its dissemination to a large public audience. To enable more students to meet the professional requirements of mathematics competence and skills many universities in the world including ours have experimented to teach "symbolic" and "computational" mathematics through the use of "open" and "problem specific" PSEs. In the case of teaching "symbolic" mathematics through the use of PSEs, the verdict is still out. There is an enormous number of students required to take symbolic mathematics i.e. "calculus", linear algebra, and differential equations, most of them with diverse backgrounds. Thus, the scaling of the available computational and PSE resources to these numbers of students is rather difficult. However, the use of PSEs to teach computational mathematics is an overwhelming success. Here, the audience is much smaller. Many text books in computational mathematics today are written having in mind specific PSEs (e.g., *Mathematica*) and the results of learning effectiveness are very positive. The learning of computational mathematics is directly related to "programming" computational resources to execute mathematical algorithms. It's worth noticing that these programs very often consist of thousands to millions lines of code (lines of code could be viewed as equivalent to number of lines in a text document). PSEs are designed to increase the programming productivity and reduce the cost of maintaining and updating such large codes. The most significant activity of any research and development activity is brain storming and prototyping of ideas. PSEs provide the most cost effective way for prototyping and experimentation in a collaborating mode. Moreover, they support effectively most of the new pedagogies identified above.

Today's PSE technology is rather knowledge- and cost- intensive. PSE development requires significant human and computational resources. In order for PSEs to impact the learning engineering problem solving process in a big way, software engineering technologies are required that will allow novel users to build PSEs for any human activity requiring problem solving as easily as writing documents. A trend in this direction is the use of most commercial PSEs as kernels. That is, to use their software structure and facilities to build "tool kits" for a specific problem solving process. Unfortunately, the dissemination of these tool kits is often followed by significant restrictions from the kernel vendors and requires "deep" knowledge of computing.

The purpose of this paper is to advocate the use of existing PSEs in engineering and mathematics education by showing that this is consistent with the current vision for engineering education in 21st century and the background and expectations of current generation of engineers. In addition, we want to summarize the state-of-the-art and the principal components and paradigms for building PSEs and identify some future trends.

2. PROBLEM SOLVING ENVIRONMENTS

Solving problems with computers involves primarily three technologies. First is the *hardware* that executes basic operations (arithmetic, data transfers, logical operations, displays…). Second is the algorithmic technology which takes a problem data as input, and manipulates it mathematically to produce the solution. Algorithms are specified mathematically and often involve manipulations which do not exist exactly as (or even close to) hardware operations. The final technology is *programming* which takes abstract algorithms and expresses them in sufficient detail so they can be executed approximately by computer hardware. We define programming as using common high-level programming languages such as Ada, Fortran 90, C, C++, C# and Java. This class of languages has been in widespread use for 40 years with a modest rate of evolution and the bulk of scientific programming uses them.

Everyone is familiar with the enormous increase in computer hardware and power. People are much less familiar with the enormous increase in algorithmic power over the past 60 years [16]. Before computers (1900–1945), the algorithms known were extremely slow by today's standards. The rate and amount of progress depends on the problem area and is especially dependent on the size of the problem. It is a widely accepted maxim that as computers get faster, the importance of more efficient algorithms only grows, and does not lessen. As problem sizes increase, the effects of algorithm speed up increase, and it is reasonable to expect to speed up the simulation of an entire automobile by a factor of 10^{20} compared to 1945.

There has been little increase in programming power over the past 60 years. The evolution of programming languages and aids might have increased programming power by a factor of 5 or 10 over 60 years. This gives a rate of increase of power of 3 or 4% a year. This slow increase has totally changed the financial nature of computing. In 1960, the cost of buying and operating hardware dominated computing costs. Today the cost of programming dominates computing costs.

The potential for powerful problem solving systems was recognized very early [2, 14]. Inadequate computing power made such systems unfeasible until the 1980s when serious work began [1, 15]. In April 1991, a research conference [3] defined a problem solving environment (PSE) as follows:

> *A PSE is a computer system that provides all the computational facilities necessary to solve a target class of problems*

These facilities include advanced solution methods, automatic or semiautomatic selection of solution methods and ways to easily incorporate novel solution methods. Furthermore, PSEs use a language natural for the problem class and allow users to solve problems without specialized knowledge of the underlying computer hardware or software. Thus the definition

```
PSE = Natural Language + Solvers + Knowledge Base + Software Bus
```

The goal of PSE technology is to exploit hardware and algorithm power to deliver "cheap and fast" problem solutions with minimum knowledge in "computer programming". Like books in libraries, PSEs codify accepted practice in the form of software implementation and in this manner, aid in the preservation of knowledge from one generation to another. PSEs can dramatically increase programming power for science and engineering through a "natural" human interface. Future PSEs will use essentially all the resources of future computing

technologies, e.g., grid and networks of computers, dynamic and collaborative computations, knowledge discovery and learning, software reuse, human-machine interaction, etc. PSEs will be the future repositories of knowledge in specific scientific areas. Already people started calling them “scientific portals” and are advancing their web interfaces.

3. TEACHING MATHEMATICS USING PSES

Engineering schools tend to accept students with reasonable mathematical knowledge and skills after secondary schooling world wide. It appears that most of them have learned mathematical concepts from examples and they have learned to recognize patterns than to use the basic theories associated with various areas of mathematics. They aim to study and learn for the short term i.e. for passing examinations without a deep knowledge and understanding of the concepts and their consequence to later studies of their engineering discipline.

Against this background, one has to ask what PSEs have to offer that will motivate students to change their view and attitude towards mathematics and how they can help to excite the majority of students towards this subject. Some of the advantages of using PSEs in teaching mathematics are the following

- Provide a greater proportion of students with the opportunity to apply sophisticated mathematical methods in problem-solving.
- Enable students to acquire knowledge for life since some of these PSEs are used as kernels to develop special toolkits for advanced engineering subjects and are updated regularly with the latest algorithmic and programming technologies.
- Increase visualization, thus, the attractiveness of mathematics through graphics capabilities and special tools of analyzing the results of mathematical studies and experiments
- Produce logs and information-rich documentation of a particular effort in various publishing standards
- Collaboration with other students and teachers
- Solving and prototyping real engineering problems in student’s time frame
- Provide interaction and self validation during the learning process and increase independence of students
- Support concentration on modeling and problem solving strategies in mathematics
- Allow the possibility to deal with larger and more realistic problems
- Not decrease the theoretical level of the mathematics curriculum
- Not be used instead of traditional textbook methods
- Be applied as a supplement to traditional methods
- Be used to stress the numerical methods in the mathematics curriculum
- Be used for data analysis and to simulate technical processes

Many of these trends are already witnessed; even a cursory glance at the recent issues of *IEEE/AiP Computing in Science and Engineering* journal reveals the persistent role of PSEs and PSE-like environments in imparting important mathematical concepts. There are even competitions designed to recognize innovative uses of scientific education using PSEs (e.g., sponsored by the U.S. Department of Energy).

Several educators fear that PSEs might promote the use of mathematical concepts without understanding them and make parts of the standard mathematics curriculum redundant. This is a worry reminiscent of when scientific calculators were first introduced. Unfortunately, PSEs are not expert systems yet! Thus, PSEs could exaggerate the phenomenon "garbage in – garbage out" in case the user does not understand the underlying concepts. On the other hand it can be argued that not many well educated engineers and scientists know how to compute manually many of the basic mathematical functions – e.g., determine the derivative of a function from first principles, evaluate special functions like trigonometric and logarithmic functions. It appears that as we evolve intellectually, we "operationalize" certain complex derivations, and such tedious processes become "automata". We need to know their definition and usage only though their properties and applicability while their implementation is left to a library/PSE. Thus, PSEs reduce the time to train pupils in extremely specialized techniques and devote more time to mathematical culture (tools, concepts, history) and solving realistic problems.

4. STATE OF THE ART IN 2000

Problem solving environments are now a well established methodology for computational science and engineering. Rather than present detailed review of PSEs, we just make some general comments and present a set of information sources. The PSE web site

http://www.cs.purdue.edu/research/pses

provides a definition of PSEs a reading list, a list of conferences on PSEs, and a list of PSE projects in computational science. The latter have a brief description, contact information and link to the PSE web site. Most of these projects are to create actual PSEs (31 projects), some are to build infrastructure (11 projects) and a few are related symbolic systems (4 projects). The book [5] covers a broad range of the research on PSEs. It has a bibliography of 415 papers related to PSEs and Figure 1 shows a plot of the publication dates, both by year and cumulative. This figure shows how the PSE field blossomed once sufficient computing power became available to make PSEs practical.

Another indication of the state of the PSE field is the number of government research initiatives that either focus on PSEs or have PSE research as a large component. In the United States there have been such initiatives from the National Science Foundation and the Department of Energy.

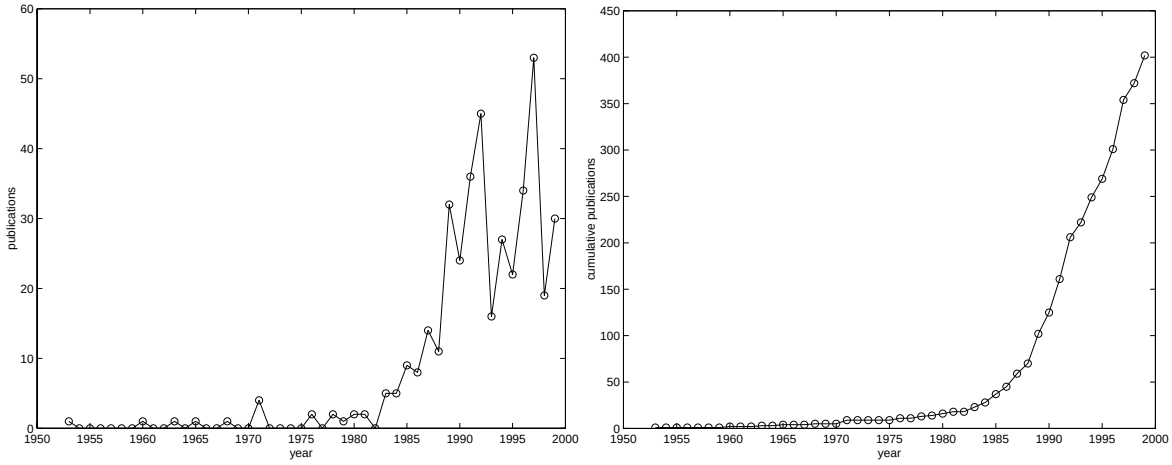

Figure 1: Publication dates from the bibliography of Gallopoulos (a) by year, (b) cumulative from 1963 to 1999.

5. PSE TECHNOLOGIES

There are five key enabling technologies for PSEs, see [5]. We summarize them here.

- *Software Reuse*: One design objective of future PSEs is to use scalable libraries and systems as building blocks. The PSEs can enable users to define the problem via a multi-media user interface that uses a geographically distributed computing infrastructure. This objective requires tools that enhance reuse and enable layered approaches to application development. In the case of sequential applications, CORBA and OLE help integrate software components and give interoperability between packages and languages. Java delivers them with its Write Once, Run Anywhere capability. More general and powerful tools are needed in this area.

- *Natural Languages*: Attempts to raise the language level for PSEs include specialized language interfaces associated with ALPAL, ELLPACK, MatLab, Mathematica, Maple, CAPSE, etc. The dream here is to develop a language that allows the user to specify an "outline" of the problem and the associated computations. The integration of natural language interfaces with PSEs is required. The current efforts to create better and more natural computer languages must continue.

- *Collaboratory Modeling and Problem Solving*: Computational modeling will shift from the current single physical component design to the design of a whole physical system with a large number of components that have different shapes, obey different physical laws and manufacturing constraints, and interact with each other through geometric and physical interfaces. For example, the analysis of an engine requires components from thermodynamics, mechanics, structures and geometry. The design of the engine requires that these different domain-specific analyses interact in order to find the final solution. The different domains share common parameters and interfaces, but each has its own parameters and constraints. The process requires that development of new algorithmic strategies and software for managing the complexity and harvesting the power of future computing resources. The MPSE (Multidisciplinary PSE) is the software implementation of this framework which combines discipline-specific problem solving environments. The MPSE design objective is to allow the natural specification of multi-physics applications and their simulation with interacting PSEs through mathematical and software interfaces across networks of computational resources.

- *Netcentric Computing*: Rapid advances in networking technologies and commodity high performance computing systems have created a new paradigm of network-based computing (NC). It involves a large number of geographically distributed computer resources such as PCs, workstations, and massively parallel processors (MPP) connected through a high speed network as a single meta-computer or computational grid. The NC paradigm promises to put the PSE technology at the fingertips of any scientist or engineer, anytime and anywhere. It will enable the development of the virtual scientific library and computational server concepts as part of software reuse.

The rapid prototyping of devices and systems with numerous inter-related elements require the fast, accurate simulation of physical processes and design optimization using knowledge and computational models from multiple disciplines. The required software is usually

distributed geographically and the execution of such computations on an internet *computational grid* involving hundreds of heterogeneous nodes distributed across enterprises is now at the proof of concept stage. For complex adaptive computations, we will see the utilization of mobile computing or code shipping paradigms where the network is used for actual computations. This change will have enormous impact as the old idea that "The Network is the Computer" becomes a reality. It will not only change the way we work and communicate, but it will change the way we do problem solving [6, 9].

- *Computational Intelligence*: The existing computational science software is characterized by having many essential parameters that must be specified by the user. This complexity is significantly increased by the many parameters of the execution environment. Furthermore, there are many alternative solutions of the same problem by selecting different software for the various phases of the computation. Thus, the task of selecting the best software and the associated algorithmic/hardware parameters for a particular problem or computation is often very difficult and sometimes impossible. An approach for dealing with this task by processing performance data obtained from testing software has been developed [4, 5, 6]. This includes *a knowledge discovery in data bases* methodology for recommending scientific software which uses existing database recommendation techniques. This system is the beginning of applying computational intelligence technologies to PSEs. These will improve enormously and become a basic part of the PSE technologies.

- *Recommender Systems*: A preferred embodiment of computational intelligence in PSEs is as a "recommender system" that serves an advisory role in a PSE. A recommender system interacts with the user/student through a dialog, culling information about the problem at hand. It then uses past experience to suggest a suitable solution process (algorithm and architecture configurations). Such recommendations can serve important purposes, because the PSE can provide phenomenological explanations of why a suggestion was made and can also support "what-if" scenarios. Already recommender systems for important targeted mathematical domains have been built and fielded. Their pedagogical implications are an ongoing area of study.

6. FUTURE PSE SOFTWARE ARCHITECTURE

We consider the problem solving process from both the user's viewpoint and the "systems" viewpoint. Initially, the user must define the problem to the system. In a PSE, this specification is declarative (i.e., only indicates the required information and not what to do with it or how to do something with it), symbolic (i.e., in some abstract form) and in terms that are natural to the problem domain. Only the essential features of the problem are specified; there is no indication of how it is to be solved.

Suppose that there is no existing PSE for solving the problem, but there is a large collection of problem solving components, including those that are needed to solve it. Then, the user must first combine some of these components to form a custom PSE and then apply it to solve the problem at hand. In this case, the user must be able to "browse" the available components, "select" the appropriate ones, and "connect" them to form a custom PSE. Then, to solve the problem, the user transfers the declarative problem specification to the PSE and interacts with the PSE appropriately.

The PSE finally assembled to solve the problem has five stages:

1. *Declarative problem specification*: As described above.

2. *Computational algorithm*: The problem specification is transformed to an algorithm which, when run, solves the problem. This is stated in a high-level, pseudo-code specification of this algorithm.

3. *High-level programming language program*: The algorithm is executed by translating it to a program in some traditional high-level language.

4. *Problem solvers* (libraries, servers): The problem solvers do the real work and we are invoked from the high-level language.

5. *OSs/networks/utilities*: The lowest level is the traditional computing platforms on which the problem solvers execute.

PSEs share many properties with other large integrated software environments such as Microsoft Office. There already exist many software frameworks which support the development of such systems and there is also a relevant body of work on distributed communication environments. However, none of these systems completely addresses the unique infrastructure needs of PSEs. The kernel PPK [17] has been proposed as a kernel for building PSEs (akin to the interfaces released by Wolfram research for extending *Mathematica*). The PSEWare project by a consortium let by Indiana University also is developing kernels for building PSEs. The goal of such a software framework is to build PSEs that support the above problem solving process. The key is to integrate the various pieces to form the comprehensive system that provides problem solving facilities to the user. Such integration software is called ‘middleware’ and PPK is a middleware system for PSEs. Building a PSE using PPK requires one to customize it by configuring its core components (software bus, notebook and object manager) appropriately to create a customized framework into which application-specific components can be integrated easily.

7. FUTURE TRENDS IMPACTING PSE TECHNOLOGY

We have already noted the enormous increases over the past 60 years in computer hardware technologies. Increases have been roughly exponential in computer speeds and memory size. It seems unlikely that these rates will change dramatically; if anything, these will be even larger over the next few decades. Further, the increases in algorithm power have also been roughly exponential provided the problem to be solved is very, very large. This rate is also expected to continue.

Computational intelligence is another area where the increases in power will have an enormous impact. Artificial intelligence has long been a disappointment to computer researchers. Computers began by being simple-minded and stayed that way for decades. But it is the comparison with arithmetic speed that is misleading, not the lack of progress in computer intelligence. Humans are very, very poor at arithmetic and very, very good at thinking. We are very impressed that computers can do arithmetic a billion times faster than they could 50 years ago; they were already better than people when they started. But, machines being a billion times more intelligent than they were 50 years ago is not impressive, because they were incredibly stupid then. Now computers are competitive with people in many areas of intelligence. In a decade or two, they will be a hundred or a thousand times more intelligent than they are now.

That intelligence will be easy to use in PSEs and dramatically affect their performance and increase their value [7].

8. REFERENCES

1. B. Ford and F. Chatelin. *Problem Solving Environments for Scientific Computing*. North Holland, Amsterdam, 1987.
2. C. Fried. An On-Line Computing Center for Scientific Problems. *IEEE Pacific Computer Conference*, p. 221, 1963.
3. E. Gallopoulos, E.N. Houstis, and J.R. Rice. Future Research Directions in Problem Solving Environments. Technical Report, Department of Computer Sciences, Purdue University, pp. 92–132, 1992.
4. E. N. Houstis, A.C. Catlin, J.R. Rice, V. Verykios, N. Ramakrishnan, and C. Houstis. Pythia-II: A Knowledge/Database System for Managing Performance Data and Recommending Scientific Software. *ACM Trans. Math. Software*, **26**, pp. 227–253, 2000.
5. E.N. Houstis, J.R. Rice, E. Gallopoulos, and B. Bramley. *Enabling Technologies for Computational Science: Frameworks, Middleware and Environments*. Kluwer Academic Publishers, 2000.
6. Elias N. Houstis, Ann Christine Catlin, Nitesh Dhanjani, John R. Rice, Naren Ramakrishnan, Vassilios S. Verykios. MyPYTHIA: a recommendation portal for scientific software and services, *Concurrency and Computation: Practice and Experience*, Vol. 14, Nos. 13-14, pp. 1481-1505, 2002.
7. Elias N. Houstis and John R. Rice, On the Future of Problem Solving Environments, *Computational Science, Mathematics, and Software* (R. F. Boisvert, E. N. Houstis, eds), Purdue University Press, 2002.
8. B. Joseph, R. Sureshkumar, R. Motard, Project ELITE: Enhanced Learning via Information Technology in Engineering Education, http://it.che.wustl.edu
9. G. Fox, J. Dongarra, D. Arnold, H. Casanova, A. Catlin, T. Haupt, E. Houstis, and J R. Rice, Problem-Solving Environments, chapter 14, in *Sourcebook of Parallel Computing*, Morgan Kaufmann Publishers, 2003.
10. J. Fox ed., Special Issue on Grid 2002: Architectures, Environments and Applications, *Concurrency and Computation: Practice and Experience*, Vol. 14, Nos. 13-14, 2002.
11. N. Grunwald, A. Kossow, T. Pawletta, R-P Tiedt, Co-operation across disciplines in engineering education using technical and scientific computing environments, *Global J. of Engng. Educ.*, Vol. 3, No. 3, pp. 209-213, 1999.
12. National Research Council, *Engineering Education: Designing an Adaptive System*, National Academy Press, Washington, D.C., 1995.
13. National Academy of Engineering, *Frontiers of Engineering*, National Academy Press, Washington, D.C., 1998.

14. J.R. Rice and S. Rosen. NAPSS – A Numerical Analysis Problem Solving System. *Proceedings ACM National Conference*, pp. 51–56, 1966.

15. J.R. Rice and R.F. Boisvert. *Solving Elliptic Problems using ELLPACK*. Springer, New York, 1985.

16. J.R. Rice. *Numerical Methods, Software and Analysis*. Second edition, Section 10.2C, Academic Press, 1992.

17. S. Weerawarana, E.N. Houstis, A.C. Catlin, J.R. Rice, R. Gaitatzes, C. Crabill, and T. Drashansky. PPK: Towards a Kernel for Building PSEs. Technical Report. Department of Computer Sciences, Purdue University, 1994.