REVISING MATHEMATICAL FOUNDATIONS FOR COMPUTER SCIENCE COURSES

Vladimir V. Riabov, Ph.D.*
Professor, Department of Mathematics and Computer Science, Rivier University

Keywords: *computer graphics, code complexity, code error prediction, cryptography, Galois fields, graphs, Java applets, modular arithmetic, primes, quicksort algorithm, strange attractors*

Abstract

The role of advanced mathematical concepts and methods is analyzed in various computing applications, including numerical methods, cryptography, structured testing, atmospheric dynamics, and complex terrain visualization. Students' challenging research projects with implementation of these techniques are reviewed.

1 INTRODUCTION

Mathematics has a vital role in the development of computer science, electronic systems, and numerous applications of computers in various aspects of everyday life of people, social groups, and sciences. The objective of this paper is to review several advanced mathematical concepts and methods (modular arithmetic and Galois fields; game theory; strange attractors; structured testing methodology, computer graphics, and visualization) that contribute into the development of various practical applications in cryptography, atmospheric dynamics, planet exploration, complex geometrical visualization, and other disciplines.

Students' challenging research projects with implementation of these techniques (accompanied with the developed computing tools and programming codes) are reviewed. The mathematical concepts, algorithms, and codes were examined by undergraduate and graduate students in *Discrete Mathematics*, *Algorithms*, *Computer Science Fundamentals*, *Computer Networks*, *Computer Security*, *Numerical Methods*, *Software Engineering*, *Professional Seminar*, and other courses taught by the author.

2 MATHEMATICAL TOPICS IN TRADITIONAL COMPUTER SCIENCE CURRICULA

The opinions on why computer science students need general knowledge of mathematical concepts have been widely discussed in academia [1]. Several scholars [2, 3] even recommended long lists of mathematical methods and formulas (ironically named as "Computer Science Cheat Sheets") that every computer science student should be familiar with. Unfortunately, these "Cheat Sheets" cover mostly basic introductory-level mathematical concepts (e.g., series, function-value order definitions, permutations, combinations, probability, identities, recurrences, geometry, trigonometry, matrices, hyperbolic functions, calculus of derivatives and integrals, finite calculus, partial fractions, Taylor's series and expansions, Stieltjes integration, and Cramer's rule). Only a few complex math methods have been mentioned there [2]: the brief reviews of the Number Theory, Graph Theory, and the Master Method for algorithm analyses. Unfortunately, the advanced concepts (e.g., modular arithmetic and Galois fields; game theory; fuzzy logic; strange attractors; pattern recognition, and the direct simulation Monte-Carlo technique) were not included in those reviews.

ISSN 1559-9388 (online version), ISSN 1559-9396 (CD-ROM version).

Some popular software tools (e.g., MATLAB [4] and FLUENT [5]) are available for students in exploring various mathematical methods in different application, such as linear algebra, calculus, differential equations, special functions, etc. The Java Applets [6] have been developed by the author and his colleague to introduce basic concepts of primes, factorials, permutations, combinations, and probability (see Figs. 1 and 2).

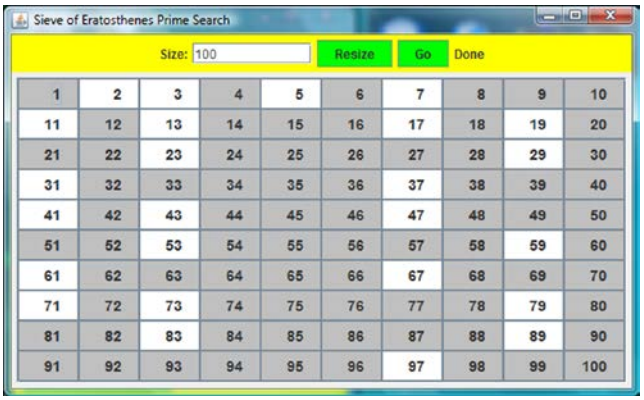


Figure 1. The "Sieve of Eratosthenes" Java applet for the prime numbers search.

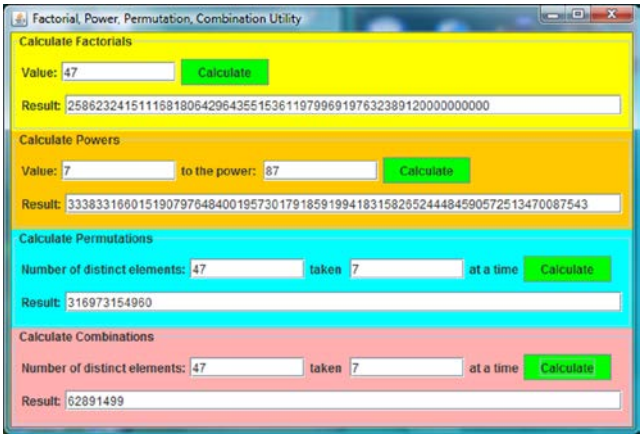


Figure 2. The Java applet for calculating factorials, powers, permutations, and combinations.

The theory of probabilities is introduced through the experimental examination of coin tosses using the corresponding Java applet shown in Fig. 3.

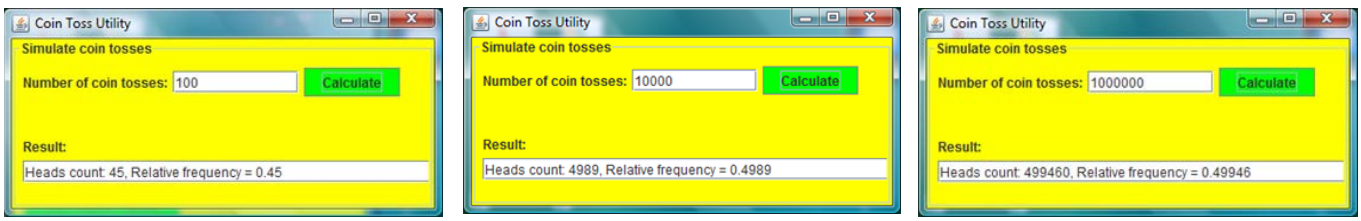


Figure 3. The Java applet for experiments with coin tosses.

Numerous coin-toss experiments run by students indicate that the asymptotic probability of head appearance is 0.5 (at the infinite sample size, see Fig. 3). They also estimate that the error is proportional to $1/N^{½}$, where N is the number of coin tosses.

The systems of linear algebraic equations can be effectively solved by analytical and numerical methods [7, 8] for various engineering applications. As examples of complex case study projects in *Linear Algebra* and *Data Mining* courses, students explore Google™ search algorithms: PageRank [9], Penguin, Panda, and Hummingbird. The PageRank algorithm measures the importance of website pages as the number of links received from other websites [9]. The RankPage computations can be computed either iteratively (by the power iteration method) or algebraically (by construction of a stochastic matrix with an eigenvalue equal to one). The PageRank algorithm is implemented in MATLAB/ Octave™ and Python codes [9].

3 COMPUTER SCIENCE APPLICATIONS ENRICHED WITH THE ADVANCED MATHEMATICAL CONCEPTS

3.1 The Role of Number Theory in Modern Cryptography

The theory of numbers plays probably a unique role in the theoretical computer science and various applications. Traditionally, the related topics (e.g., numerical systems, the Fundamental Theorem of Arithmetic, primes and co-primes) are covered in the *Discrete Mathematics* course. In our pedagogical practice, the more advanced topics (modular arithmetic, abstract groups, Abelian groups, rings, commutative rings, integer domains, and fields) [10] are covered in the advanced elective courses, such as *Computer Security* [11], due to the fact that modern encryption/decryption methods [12, 13] utilize the modular arithmetic and Galois field properties framed with the Fermat's Little Theorem and fundamental properties of Euler's totient function [14, 15]. The relationship between the topics is shown in Fig. 4.

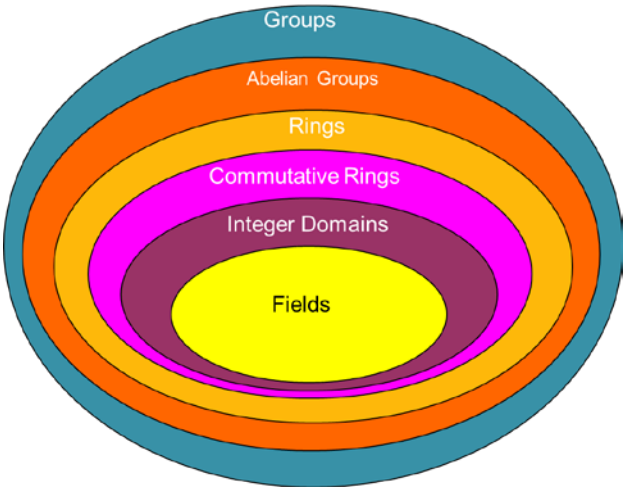


Figure 4. Introducing properties of fields that embed properties of other fundamental sets.

The Java Applets [6, 11, and 16] have been found as an effective tool to introduce the modular arithmetic, properties of Galois Fields, and cryptographically-secure message digest algorithms [17, 18] (see Figs. 5 and 6).

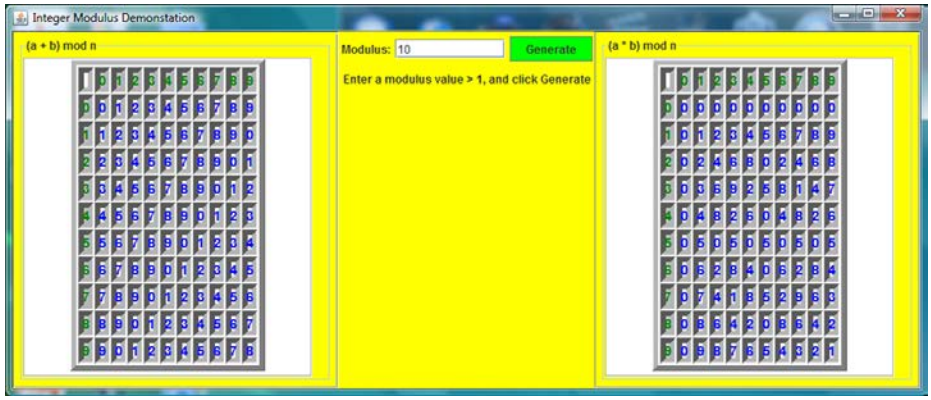


Figure 5. An example of using Java applets for the modular-arithmetic analysis.

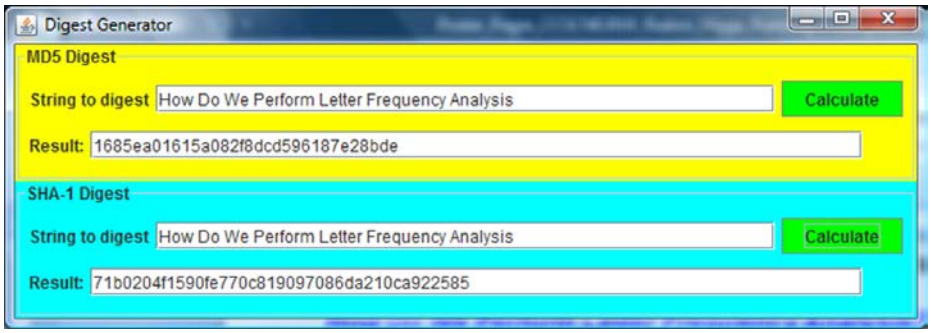


Figure 6. An example of using Java applets for creating message digests in MD5 and SHA-1 standards.

Many students select challenging topics on modern encryption/decryption methods. Their overviews cover the role of number theory in modern cryptography [19], the coding theory [20], the Advanced Encryption Standard [21], the Remote Authentication Dial-In User Service (RADIUS) protocol [22], security issues in Wi-Fi technology [23], and other applications.

3.2 The Graph Theory Application: Reducing the Programming Code Complexity

In the *Software Quality Assurance* course, the structured testing methodology [24, 25] and graph-based metrics (cyclomatic complexity, essential complexity, module design complexity, system design complexity, and system integration complexity) have been reviewed by students and applied for studying the C-code complexity and estimating the number of possible errors and required unit and integration tests for the Carrier Networks Support system [26]. Comparing different code releases, it is found that the reduction of the code complexity leads to significant reduction of errors and maintainability efforts.

For each module (a function or subroutine with a single entry point and a single exit point), an annotated source listing and flowgraph is generated as shown in Fig. 7. The flowgraph is an architectural diagram of a software module's logic.

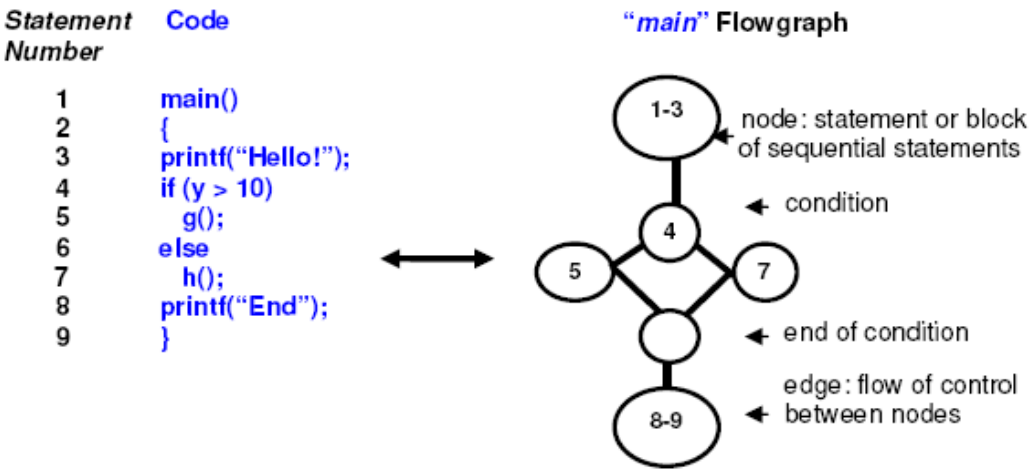


Figure 7: The annotated source listing and the related flowgraph.

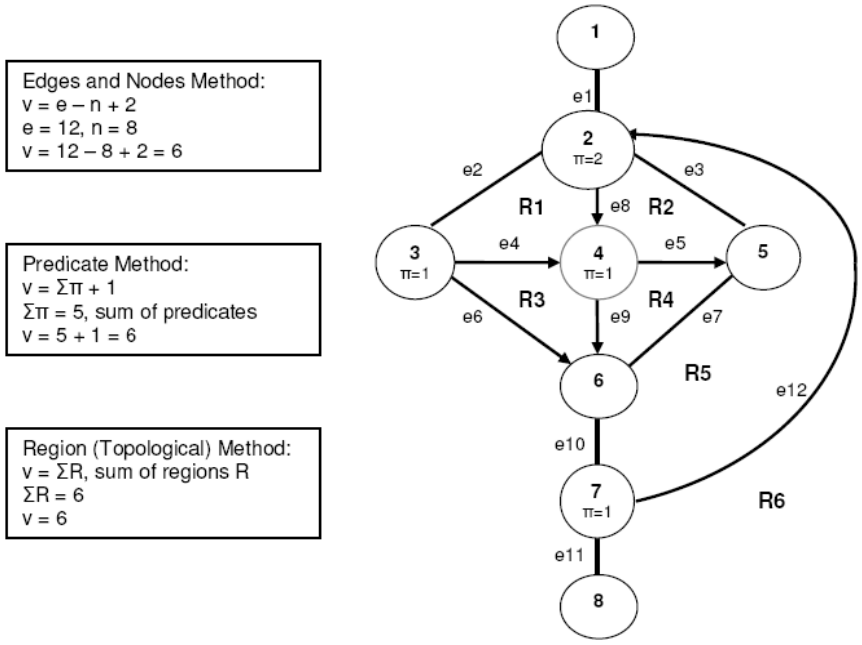


Figure 8: Three methods of evaluating the cyclomatic complexity of the graph.

Cyclomatic complexity, *v*, is a measure of the complexity of a module's decision structure [24, 25]. It is the number of linearly independent paths and, therefore, the minimum number of paths that should be tested to reasonably guard against errors. A high cyclomatic complexity indicates that the code may be of low quality and difficult to test and maintain. In addition, empirical studies have established a correlation between high cyclomatic complexity and error-prone software [26]. The results of

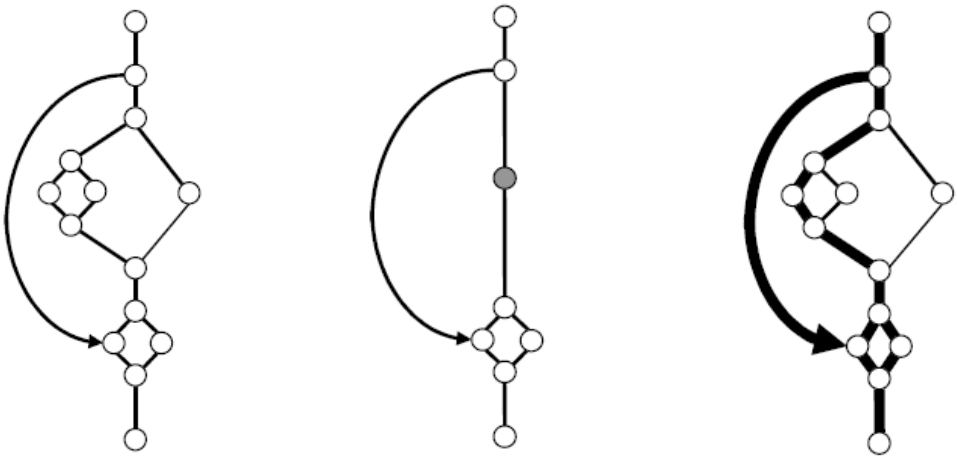
psychological experiments by Miller [27] suggest that modules approach zero defects when the McCabe's cyclomatic complexity is within 7 ± 2. Therefore, the threshold of *v*-metric is chosen as 10. Miller's experiments have led to the reduction of the programming-code complexity.

A node is the smallest unit of code in a program. Edges on a flowgraph represent the transfer of control from one node to another [24]. Given a module whose flowgraph has *e* edges and *n* nodes, its cyclomatic complexity is $v = e - n + 2$. This complexity parameter equals the number of topologically independent regions of the graph and correlates with the total number of logical predicates in the module [24, 25] (see Fig. 8).

Branching out of a loop. Branching into a loop. Branching into a decision. Branching out of a decision.

Figure 9: Examples of the unstructured logical constructs.

Essential complexity, *ev*, is a measure of unstructuredness, the degree to which a module contains unstructured constructs [25, 26] (see Fig. 9), which decrease the quality of the code and increase the effort required to maintain the code and break it into separate modules. When a number of unstructured constructs is high (essential complexity is high), modularization and maintenance is difficult. In fact, during maintenance, fixing a bug in one section often introduces an error elsewhere in the code [25, 26].



Original flowgraph, v = 5. Reduced flowgraph, v = 3, ev = 3. Superimposed flowgraph, v = 5, ev = 3.

Figure 10: Evaluation of the essential complexity of the flowgraph.

Essential complexity is calculated by removing all structured constructs from a module's flowgraph and then measuring the cyclomatic complexity of the reduced flowgraph [25, 26] shown in Fig. 10. The

reduced flowgraph gives you a clear view of unstructured code. When essential complexity is 1, the module is fully structured. When essential complexity is greater than 1, but less than the cyclomatic complexity, the module is partly structured. When essential complexity equals cyclomatic complexity, the module is completely unstructured. The unstructured modules should be recommended for redesigning.

3.3 The Role of Physiology and Linguistics in Estimating the Number of Errors in the Programming Code

The following case study has been designed to introduce Halstead's metrics [28] and explore the physiological phenomenon of human-brain restrictions [29] in estimating the number of projected errors in the programming code. The McCabe IQ™ tool has been used in producing Halstead metrics [28] for codes written in selected programming languages. Supported by numerous industry studies [25], the B-metric of Halstead represents the estimated number of errors in the program.

The concept of event-discriminations was introduced by John M. Stroud, a psychologist, in "The Fine Structure of Psychological Time" [29], where he defined a "moment" as "the time required by the human brain to perform the most elementary discrimination" [28, p. 48]. He reported that, for all waking, conscious time, these "moments" occurred at a rate of "from five to twenty or a little less" per second, which is known nowadays as the Stroud number, S [28]. His study was based on the analysis of the internal processing rate of the brain that correlates with the range of the number of frames per second which a motion picture should have to appear as a continuous picture rather than as single frames. Halstead [28] applied this concept to the evaluation of discriminations in input/output program events. Finally, the Stroud number, S, was used by Halstead in estimating the programming time, $\check{T} = E/S$, where E is the number of elementary mental discriminations required for the program implementation.

The linguistic complexity of various languages (English, PL/I, Algol, Assembly, and others) was studied by Halstead [28, pp. 62-70] for estimating the language level parameter ($\lambda = 2.16$) that was used in calculating the mean number of elementary discriminations between potential errors in programming ($E_0 = 3000$), and, finally, the number of "delivered" bugs, $B = E^{2/3}/E_0$.

Following Halstead's procedures [28], students in *Software Quality Assurance* and *Software Engineering* classes identified all operators and operands, their frequencies, and estimated the program length, vocabulary size, volume, difficulty and program levels, the effort and time amounts to implement and understand the program, and the number of delivered bugs (possible errors), B. They compared their findings with values calculated by using SQA tools (McCabe™ IQ, Java Source Metric™, CCCC™, SourceMonitor™, and COCOMO-II™), and found that the results are sensitive to the programming language type (procedural or object-oriented) [26]. In particular, students found that efforts to implement and understand the program were higher for procedural languages (FORTRAN and C) than for the object-oriented language (Java), even for simple algorithms, like Euclid's algorithm for calculating the Greatest Common Divisor. They also found that large C/C++ source files [30] contain more actual errors than the number of delivered bugs (B) suggested [28].

4 STUDENTS' RESEARCH PROJECTS

Many students selected challenging topics for their research projects in various computer science courses. Here we only make overviews of a few outstanding projects that have been performed using the advanced mathematical concepts briefly described in the previous sections.

In his project "Designing an Easily Modifiable Cipher for Educational Purposes" [31], Michael Jeffords introduced two easily-implemented ciphers that can be used to teach the basics of crypto-analysis using the character frequency analysis. In this approach, each cipher builds from the previous one, and the implementer can choose to keep contextual clues such as spacing and capitalization. These ciphers were designed to both hide letter frequencies and to be broken easily. The goal was that this approach could be used to excite students about the field of crypto-analysis and steer them toward open standards of encryption. Michael explored the ideas of the Vernam cipher [32] and worked on its implementation with the RC4 algorithm [33].

David Snogles developed the Personal Encrypted Talk system for his final capstone project. Its primary goal was to secure Instant Messaging communications between two parties on the Internet. Secondary objectives were Java Cryptography Architecture research and the practical experience gained by the student in the development of a scalable Java-based Graphical User Interface application. His article [34] summarizes the software engineering steps followed during the project implementation.

Robert Marceau studied C.A.R. Hoare's quicksort algorithm that has become a popular sorting algorithm due to the average performance of $\Theta(n \bullet \log n)$, limited use of extra storage (typically $\Theta(\log_2 n)$ recursive calls) and better performance on average compared to heapsort, which is another $\Theta(n \bullet \log n)$ sorting algorithm. The algorithm code may be found in several standard libraries supporting C, C++, and Java. The major drawback in the quicksort algorithm is the $\Theta(n^2)$ worse case performance. This worse performance is exhibited for some rather common initial permutations. The student studied this performance of the quicksort algorithm, and offered its particular modifications to minimize the probability that the worst-case performance will be exhibited [35].

In his research project, Douglas Selent used the McCabe IQ™ tool to analyze source code complexity of the Light-up Puzzle program [36]. The source code was converted into a graph, which then was used to analyze the source code complexity. The features of the tool that were used in this project are the Battlemap, System Complexity metrics, method flowgraphs, scatter diagrams, Halstead metrics, Class metrics, and Object-oriented metrics. The tool allowed him to determine if any parts of the source code are unreliable or unmaintainable, and make proper code corrections. This approach helped him identify vulnerable code areas, reduce error rates, shorten testing cycles, improve maintainability, and maximize reusability. In order to verify the effectiveness of the McCabe™ IQ tool, he re-factored the program in the areas reported to be highly complex and error-prone. After this, he compared the McCabe's Metrics reports on the initial analysis to the reports on the re-factored analysis and charted results to clearly indicate the improvements made to decrease complexity. This technique improved significantly the quality of his software code.

Maxim Sukharev-Chuyan, a junior from Souhegan High School, Amherst, NH, who took our *Numerical Methods* course, studied a simple basic model of chaotic behavior in atmospheric layers known as the Lorenz system [37]. He developed a Java code for an animation of the waterwheel model of the strange attractors for the Lorenz system. The visualized simulations demonstrate chaotic behavior of the numerical solution of the Lorenz system of nonlinear ordinary differential equations [38].

Kevin Gill developed the Living Mars image project [39] that included topics related to computer graphics, software development, astronomy, and planetary science. The purpose of the project was to create a visualization of the planet Mars as could look with a living biosphere. This makes no distinction as to whether this biosphere would represent an ancient or future, possibly terraformed planet. The algorithms and methods used in generating shadows on digital elevation models were developed in his previous study [40]. These include formulas that are common in computer graphics applications and are often provided by specific frameworks (i.e., OpenGL). The basics of model rendering are covered from

the structure of the source data to the interpolation of hypsometric/bathymetric tint colors. The primary algorithm presented in [40] is the calculation of shadows using ray tracing. The methods are based on the code from the jDem846 open source project managed by the student. Code listings are provided in [40] and available in full online.

5 CONCLUDING REMARKS ON STUDENTS' INVOLVEMENT

The author has described several computing applications that involve the advanced mathematical concepts. These theoretical concepts, algorithms, practical applications, and the developed computing tools and codes were thoroughly examined by undergraduate and graduate students in *Discrete Mathematics*, *Algorithms*, *Numerical Methods*, *Computer Science Fundamentals*, *Computer Networks*, *Computer Security*, *Software Engineering*, *Software Quality Assurance*, *Professional Seminar*, and other courses taught by the author. The concepts paved the roads for students' research projects on various practical applications in cryptography, numerical methods, code complexity reduction, atmospheric dynamics, computational visualization, planet exploration, and other disciplines. After brief in-class discussions of the case studies, each student continued working on a selected case analyzing algorithms, creating computer codes (in Python, MATLAB, C/C++ or Java), running them at various parameters, comparing numerical results with known data, and presenting the findings to classmates. In the course evaluations, students stated that they became deeply engaged in course activities through examining the challenging problems related to the applications of the advanced mathematical concepts.

REFERENCES

[1] Beaubouef, T. Why Computer Science Students Need Math. *SIGCSE Bulletin*, 2002, Vol. 34, No. 4, pp. 57-59.

[2] Seiden, S. Theoretical Computer Science Cheat Sheet. *ACM SIGACT News*, 1996, Vol. 27, No. 4, pp. 52-61, doi=10.1145/242581.242585. [Online] https://www.tug.org/texshowcase/cheat.pdf

[3] Levet, M. Dream.in.Code Computer Science Cheat Sheet. August 30, 2013. [Online] http://www.dreamincode.net/forums/index.php?app=core&module=attach&section=attach&attach_id=33834&s=364b4f7796351660ba11b255878aadca

[4] MATLAB, MathWorks, Inc. [Online] https://www.mathworks.com/products/matlab.html

[5] FLUENT, Fluent, Inc. [Online] https://www.fluentco.com/

[6] Riabov, V. V., and Higgs, B. J. Algorithms and Software Tools for Teaching Mathematical Fundamentals of Computer Security. In *Proceedings of the 23rd Annual International Conference on Technology in Collegiate Mathematics* (Denver, CO, March 17-20, 2011), Prentice-Hall, 2011, Paper S062, pp. 208-217. [Online] http://archives.math.utk.edu/ICTCM/VOL23/S062/paper.pdf

[7] Poole, D. *Linear Algebra: A Modern Introduction*. Pacific Grove, CA: Brooks/Cole, Thompson Learning, Inc., 2003.

[8] Chapra, S. C. *Applied Numerical Methods with MATLAB for Engineers and Sciences*, 2nd edit., New York, NY: McGraw-Hill, 2008.

[9] Google PageRank Algorithm. [Online] https://en.wikipedia.org/wiki/PageRank

[10] Graham, R. L., Knuth D. E., and Patashnik, O. *Concrete Mathematics*. Reading, MA: Addison-Wesley, 1994.

[11] Riabov, V. V., and Higgs, B. J. Running a Computer Security Course: Challenges, Tools, and Projects. In *Journal of Computing Sciences in Colleges* 25(6), 245-247, June 2010.

[12] Rivest, R. L., Shamir, A., and Adleman, L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 1978; 21(2): 120-126.
[13] Federal Information Processing Standard (FIPS) for the Advanced Encryption Standard, FIPS-197. [Online] http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
[14] Ferguson, N., and Schneier, B. *Practical Cryptography*. Hoboken, NJ: Wiley, 2002.
[15] Kaufman, C., Perlman, R., and Speciner, M. *Network Security: Private Communication in a Public World*, 2nd edition. Upper Saddle River, NJ: Prentice Hall, 2002.
[16] Bishop, D. *Introduction to Cryptography with Java Applets*. Sudbury, MA: Jones & Bartlett Learning, 2002.
[17] The MD5 Message-Digest Algorithm. *Request For Comments*, No. 1321. [Online] http://www.faqs.org/rfcs/rfc1321.html
[18] Secure Hash Standard. *Federal Information Processing Standards Publication 180-1*. NIST. [Online] http://www.itl.nist.gov/fipspubs/fip180-1.htm
[19] Marceau, R. The Role of Number Theory in Modern Cryptography. *Rivier Academic Journal*, 2012, 8(2). [Online] http://www.rivier.edu/journal/ROAJ-Fall-2012/J680-Marceau-Theory-of-Numbers.pdf
[20] Grossman, J. Coding Theory: Introduction to Linear Codes and Applications. *Rivier Academic Journal*, 2008, 4(2). [Online] http://www.rivier.edu/journal/ROAJ-Fall-2008/J201-Grossman-Coding-Theory.pdf
[21] Selent, D. Advanced Encryption Standard. *Rivier Academic Journal*, 2010, 6(2). [Online] http://www.rivier.edu/journal/ROAJ-Fall-2010/J455-Selent-AES.pdf
[22] Szilagyi, D., Sood, A., and Singh, T. RADIUS: A Remote Authentication Dial-In User Service. *Rivier Academic Journal*, 2009, 5(2). [Online] http://www.rivier.edu/journal/ROAJ-Fall-2009/J286-RADIUS-Sood.pdf
[23] Wekhande, V. Wi-Fi Technology: Security Issues. *Rivier Academic Journal*, 2006, 2(2). [Online] http://www.rivier.edu/journal/RCOAJ-Fall-2006/J62-Wekhande.pdf
[24] McCabe, T. J. A Complexity Measure. In *IEEE Transactions on Software Engineering* SE-2(4), 308-320, December 1976.
[25] Watson, A. H., McCabe, T. J. Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric, *NIST Special Publication*, No. 500-235. Gaithersburg, MD: National Institute of Standards and Technology, September 1996.
[26] Riabov, V. V. Methodologies and Tools for the Software Quality Assurance Course. In *Journal of Computing Sciences in Colleges* 26(6), 86-92, June 2011.
[27] Miller, G. The Magical Number of Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. In *Psychological Review*, 63 (2), 81-97, 1956.
[28] Halstead, M. H. *Elements of Software Science*. New York: Elsevier North Holland, 1977.
[29] Stroud, J. M. The Fine Structure of Psychological Time. In *Annals of the New York Academy of Sciences*, Vol. 138, Interdisciplinary Perspectives of Time, 1967, pp. 623–631.
[30] Riabov, V. V. Networking Software Studies with the Structured Testing Methodology. In *Computer Science and Information Systems*. Athens, Greece: ATINER, 2005, pp. 261-276.
[31] Jeffords, M. Designing an Easily Modifiable Cipher for Educational Purposes. *Rivier Academic Journal*, 2012, 8(1). [Online] http://www.rivier.edu/journal/ROAJ-Spring-2012/J603-Jeffords.pdf
[32] Wikipedia, *Gilbert Vernam*. [Online] http://en.wikipedia.org/wiki/Gilbert_Vernam/
[33] RSA Laboratories. *What is RC4?* [Online] http://www.rsa.com/rsalabs/node.asp?id=2250

[34] Snogles, D. Personal Encrypted Talk - Securing Instant Messaging with a Java Application. *Rivier Academic Journal*, 2005, 1(1). [Online] http://www.rivier.edu/journal/ROAJ-2005-Fall/J14-SNOGLES.pdf
[35] Marceau, R. R. An Analysis of the Worst-Case Performance of Quicksort. *Rivier Academic Journal*, 2011, 7(1). [Online] http://www.rivier.edu/journal/ROAJ-Spring-2011/J515-Marceau.pdf
[36] Selent, D. The Design and Complexity Analysis of the Light-Up Puzzle Program. *Journal of Computing Sciences in Colleges*, Vol. 26, No. 6, 2011, pp. 187-189.
[37] Sukharev-Chuyan, M. The Lorenz System Simulation. *Rivier Academic Journal*, 2013, 9(2). [Online] https://www2.rivier.edu/faculty/vriabov/J788-Sukharev-Chuyan_Lorenz_System_Fall2013.pdf
[38] Lorenz, E. N. *The Essence of Chaos*. Seattle, WA: University of Washington Press, 1993.
[39] Gill, K. M. Putting Life on Mars: Using Computer Graphics to Render a Living Mars. *Rivier Academic Journal*, 2013, 9(1). [Online] http://www.rivier.edu/journal/ROAJ-Spring-2013/J754_Kevin-Gill_Living-Mars.pdf
[40] Gill, K. M. Casting Shadows: Shading Digital Elevation Models Using Ray Tracing. *Rivier Academic Journal*, 2012, 8(1). [Online] http://www.rivier.edu/journal/ROAJ-Spring-2012/J614-Gill.pdf

* **Dr. VLADIMIR V. RIABOV**, Professor of Computer Science and Department Coordinator at Rivier University, teaches Algorithms, Networking Technologies, Computer Security, Software Engineering, Software Quality Assurance, Object-Oriented System Design, System Simulation and Modeling, Numerical Methods, Introduction to Computing, and Professional Seminar in Computer Science. He received a Ph.D. in Applied Mathematics and Physics from Moscow Institute of Physics and Technology and M.S. in Computer Information Systems from Southern New Hampshire University. Vladimir published about 130 articles in encyclopedias, handbooks, journals, and international and national conference proceedings, including *The Internet Encyclopedia*, *The Handbook of Information Security*, *The Handbook of Computer Networks*, *International Journal of Computers and Structures*, *Journal of Spacecraft and Rockets*, *Journal of Aircraft*, *Journal of Thermophysics and Heat Transfer*, *Congress Proceedings of International Council of the Aeronautical Sciences*, *International Symposia on Rarefied Gas Dynamics and Shock Waves*, *International Conferences on Computer Science and Information Systems*, *International Conferences on Technology in Collegiate Mathematics*, *Conferences of American Institute of Aeronautics and Astronautics*, and others. He is a senior member of ACM and AIAA, and a member of IEEE and MAA.