ADVANCED MATHEMATICAL CONCEPTS IN COMPUTER SCIENCE APPLICATIONS

Vladimir V. Riabov
Rivier University
420 S. Main Street ▪ Nashua, NH 03060-5086 ▪ USA
E-mail: vriabov@rivier.edu

All computer science disciplines have deep roots in various areas of mathematics, and students and engineers should be familiar with corresponding mathematical concepts and their implementations in real-world applications. This paper reviews several areas of computer science theory and its applications (e.g., computational fluid dynamics, modern encryption algorithms, code development and testing, and system simulation and modeling), where "traditional" and non-trivial mathematical methods (the analysis of singular differential equations, strange attractors, the group theory, modular arithmetic, the theory of graphs, and statistical modeling of rarefied-gas flows with the Direct Simulation Monte-Carlo technique) play key roles.

1. Some Challenges in Computational Fluid Dynamics

1.1 Solving singular differential equations

Various problems of applied mathematics, thermophysics, and aerodynamics (e.g., stability of mechanical systems and flow boundary layers, fuel combustion, and heat protection of spacecraft) come to solving differential equations with small coefficients at the highest derivatives. This phenomenon leads to the formation of regions with small linear dimensions where gradients of functions are large. The numerical analysis of such problems by traditional box-schemes is restricted by non-uniform convergence or divergence of numerical solutions. In the first case study, the numerical solutions of the model singular ordinary differential equation have been evaluated for the linear boundary value problem [1]. The developed numerical method [2] was used for the analysis of gas flow parameters in boundary and viscous shock layers under the conditions of blowing on the body surface and nonequilibrium chemical reactions [3, 4].

From a mathematical point of view, the increase of the flow rate of blowing gas or chemical-reaction rates is equivalent to the existence of a small coefficient at the highest derivative in the boundary-layer equations [1]. A sublayer (of uncertain location) with large gradients of functions is created. The gas flow in the boundary layer was studied using a two-point exponential box-scheme and an effective regularization algorithm [2]. The uniform second-order convergence was obtained for functions and derivatives in the full range of small parameters such as blowing factors and inverse chemical-reaction rates. The approach is applied to boundary layers with gas injection and combustion [3].

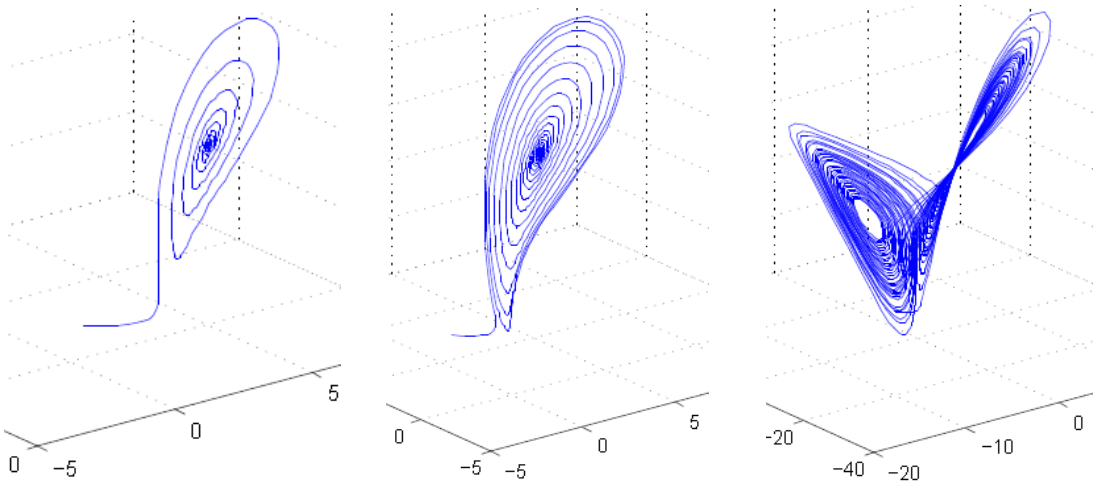
1.2 Strange attractors: an evolution of dynamic systems

The other CFD topic covers mathematical foundations of evolution of dynamic systems that could be described in terms of strange attractors. The case studies examine numerical modeling of chaotic dynamic systems (e.g., turbulence, weather forecast, and economic system development) [5]. They were introduced through classical examples of bifurcations of systems modeling equilibrium in chemical reactions, socio-economy (*Rössler attractor*) and atmospheric dynamics (*Lorenz attractor*) [5].

The Lorenz attractor was first studied by E. N. Lorenz in [6]. It was derived from a simplified model of convection in the Earth's atmosphere. The system is most commonly expressed as the following three coupled non-linear differential equations:

$$dx / dt = a\,(y - x) \quad (1)$$
$$dy / dt = x\,(b - z) - y \quad (2)$$
$$dz / dt = xy - c\,z \quad (3)$$

Here (x, y, z) are the Cartesian coordinates, "t" is the time variable, "*a*" is the Prandtl number, "*b*" is the Rayleigh number, and "*c*" is the system parameter. The series does not form limit cycles nor does it ever reach a steady state (see Fig. 1). Instead, it is an example of deterministic chaos. As with other chaotic systems, the Lorenz system is sensitive to the initial conditions: two initial states no matter how close will diverge.



A) *b*=12, *a*=10, *c*=8/3 B) *b*=16, *a*=10, *c*=8/3 C) *b*=28, *a*=10, *c*=8/3

Figure 1. Solutions of the Lorenz system (Eqs. 1-3) for different values of *b*.

The general assumption is that *a*, *b*, *c* > 0; *a* = 10, and c is varied [7]. The system exhibits chaotic behavior for *b* = 28, but displays knotted periodic orbits for other values of *b*. A saddle-node bifurcation occurs at $c(b - 1) = 0$. When $a \neq 0$ and $c(b - 1) \geq 0$, the equations generate three critical points. The critical points at (0,0,0) correspond to no convection,

and the critical points at $(\pm[c(b-1)]^{0.5}, \pm[c(b-1)]^{0.5}, b-1)$ correspond to steady convection. This pair is stable only if $b < a(a+c+3)/(a-c-1)$. When $a = 10$, $b = 28$, $c = 8/3$, the Lorenz system has chaotic solutions, but not all solutions are chaotic.

The Matlab calculations show the system evolution for different values of *b* (see Fig. 1). For small values of *b*, the system is stable and evolves to one of two fixed point attractors. When *b* is larger than 24.28, the fixed points become repulsors and the trajectory is repelled by them in a very complex way, evolving without ever crossing itself [7].

The sensitive dependence of the solution on initial condition at a=10, b=28, c=8/3 was discussed in [8, 9]. Three time segments (at t = 1, 2, and 3) that have been received with the Java animation [8] and shown in [9, Fig. 7] illustrate the 3-D evolution of two trajectories in the Lorenz attractor starting at two initial points that differ only by 10^{-5} in the x-coordinate. Initially, the two trajectories seem coincident, but, after some time, the divergence is obvious [7-9].

2. Modern Cryptography

Many computer security topics involve Math concepts that are not often taught, or inadequately covered, in college curricula, including *sets, permutations, combinations, and probability*; *number theory* (*divisibility, primes, groups, rings,* and *fields*); *modular arithmetic*; and *computability theory* (*the reasonableness of an algorithm*). The challenge is how to introduce these topics to a typically Math-phobic audience, without eliciting a "deer in the headlights" response. In our classes, we try to motivate coverage based on real-world applications of these concepts.

We start every class with a brief discussion of an unusual non-trivial topic that is called a "warm-up" exercise [10]. After these "warm-up" exercises, the instructor offers a discussion on the main topic and asks students for a feedback on lecture materials and their arguments on selecting a competitive strategy for the problem analysis and development. These discussions help students to focus on the main point of the class session and stay active in class. Here is an example of the "warm-up" exercise that opens an introductory discussion of the theory of large numbers, which leads to the applied theory of encryption algorithms, such as the RSA Public-Key encryption algorithm [11]. At the same time, it illustrates a strong bond between mathematics and computer science. A student (even if he/she is not familiar with the theory of numbers) can solve the problem by a simple experimentation.

2.1 Warm-up case study: What is the last digit of the number 6387^{5927} [mod(10)]?

We are interested in the last digit only of this number. Following the Newton's Binomial Theorem, it is absolutely enough to consider the last digit of a simpler number 7^{5927}. Doing experiments with powers of number 7, we find that the last digit can only be 7, 9, 3, or 1, and therefore, it is a cycle of *four* cases. The power, 5927 can be represented as $5927 = 4\times1481+3$. Therefore, the last digit of 7^{5927} (and 6387^{5927}) is the same as the last

digit of $7^3 = 343$, which is "3". Knowing two key parameters [e.g., the base (10) and the power (5927)], we can now restore all digits of the given huge number.

2.2 Modular arithmetic and public key cryptography

Modern encryption algorithms are based on applications of modular arithmetic [12] and prime numbers. To explore different topics of the number theory (e.g., divisibility, prime numbers, groups, rings, and fields) and modular arithmetic, students used Java applets [10, 13] that were created using recommendations from [14, 15].

Sometimes the modular multiplicative inverse has a solution, and sometimes it does not. For example, the inverses of 2, 4, 5, 6, and 8 (mod 10) do not exist. It turns out that $a^{-1} \equiv x \pmod{p}$ that has a solution iff a and p are relatively prime. In the considered case, the multiplicative inverses exist for the relative primes (to $p = 10$) of 1, 3, 7, and 9. This example could be used in introducing the finite field of order p, known as the *Galois Field* [10, 12], GF(p), which is defined as the set $\mathbf{Z}_p$ of integers $\{0, 1, ..., p - 1\}$, together with the arithmetic operations modulo p. The subset $\mathbf{Z}_p^*$ is defined as the set of (mod p) integers that are relatively prime to p. In this case study, $p = 10$ and $\mathbf{Z}_p^* = \{1, 3, 7, 9\}$. Every element in $\mathbf{Z}_{10}^*$ is present in the multiplicative table [10] based on these four elements only, and no other elements other than those are present. Furthermore, every element in $\mathbf{Z}_{10}^*$ is present in every row of the table. It turns out that this is true for all p; therefore, $\mathbf{Z}_p^*$ is closed under multiplication (mod p).

This fundamental property of relative primes allows introducing Euler's totient function, $\varphi(p)$ [10, 12] (the number of positive integers less than p, that are relative primes to p) with the following properties (Eqs. 4-6):

$$\varphi(1) = 1 \quad (4)$$

$$\varphi(p) = p - 1 \text{ (for } p \text{ prime)} \quad (5)$$

$$\varphi(m) < m - 1 \text{ (for } m \text{ composite)} \quad (6)$$

In other words, the totient function $\varphi(p)$ is the number of elements in $\mathbf{Z}_p^*$ (see Fig. 2).

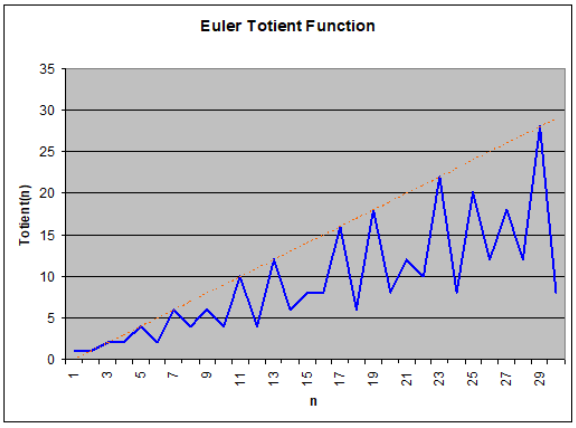


Figure 2. The values of Euler's totient function $\varphi(n)$ at various values of n.

The Euler's totient function $\varphi(n)$ has the unique "composition" property [10-12] (Eq. 7, below): assume we have two distinct prime numbers, p and q, and an integer $n = pq$, then

$$\varphi(n) = \varphi(pq) = \varphi(p) \times \varphi(q) = (p - 1) \times (q - 1) \qquad (7)$$

This fact laid the foundation to various modern encryption algorithms [10, 15], including the RSA public key encryption [11].

2.3 The Advanced Encryption Standard (AES)

In January 1997, the National Institute of Standards (NIST) announced a contest to select a new encryption standard to be used for protecting sensitive, non-classified, U.S. government information. After rigorous reviews of 5 final proposals, NIST chose a submission called *"Rijndael"* by two Belgian cryptographers – Joan Daemen and Vincent Rijmen [16]. *Rijndael* uses arithmetic in the *Galois Field* GF(2^8), the finite field of order 256. It can be shown [12] that the order of a finite field (number of elements in the field) must be a power of a prime, p^n, where n is a positive integer. Therefore, in *Rijndael* $n = 8$, and each element of the field can be represented by an octet. The bits in the octet are the coefficients of a polynomial over $\mathbf{Z}_2$ modulo the *irreducible* $\mathbf{Z}_2$ *polynomial* [12].

Byte values are represented as polynomials with the least significant bit being the coefficient of x^0, and the most significant bit the coefficient of x^7, e.g., {10100011} identifies the specific field element: $x^7 + x^5 + x + 1$. Some finite field operations involve one additional bit to the left of an 8-bit byte. When this extra bit is present, it appears as {01} to the left of the other 8 bits: {01} {00011011}. Addition in a finite field is achieved by "adding" the coefficients for the corresponding powers in the polynomials for the two elements. This operation of addition is performed using an XOR operation denoted by $\oplus$. For example, all notations below are equivalent:

$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2 + 0$ [polynomial notation];
$\{01010111\} \oplus \{10000011\} = \{11010100\}$ [binary notation].

Multiplication in *Rijndael* is the multiplication of polynomials *modulo the irreducible polynomial* [12]. For example, in the polynomial notation:

$(x^6 + x^4 + x^2 + x + 1) \bullet (x^7 + x + 1) = x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$, and
$(x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1) \bmod (x^6 + x^4 + x^2 + x + 1) = x^7 + x + 1$.

The modular reduction by $m(x)$ ensures that the result will be a binary polynomial of degree less than 8, and thus can be represented in a byte. This multiplication is *associative*, and the element {01} is the *multiplicative identity*. For any non-zero binary polynomial $b(x)$ of degree less than 8, the multiplicative inverse of $b(x)$, denoted by $b^{-1}(x)$ can be found using the Extended Euclidean algorithm [12]. As it follows from the above, the set of 256 possible byte values, with XOR used as addition, and the multiplication

defined as above, has the structure of the finite field GF(2^8). The detail description of the AES algorithm can be found in [16, 17].

2.4 Deciphering with the linguistic letter frequency analysis

For several decades the substitution cipher approach with the Letter Frequency Analysis [13, 18, 19] has been also effectively used. To explore this approach, which is based on applying the linguistic properties of an original plaintext [20, 21], students make some assumptions about the plaintext:

- That the plaintext consists of characters, not some kind of binary code.
- That it is written in some natural language with known linguistic properties (e.g., English).
- That we know the frequency of letters in a typical piece of text in that language.
- That the plaintext is typical of normal English text, and so we expect the same frequencies of letters (approximately, within statistical fluctuations).

As long as we know that there is a 1-to-1, unique mapping from plaintext to ciphertext (and, therefore, from ciphertext to plaintext), we can employ our knowledge of those letter frequencies to crack a substitution cipher. It is important to note that we need a large enough piece of text to give us some expectation that we have a large statistical sample. The longer the message, the better statistical sample we are likely to have.

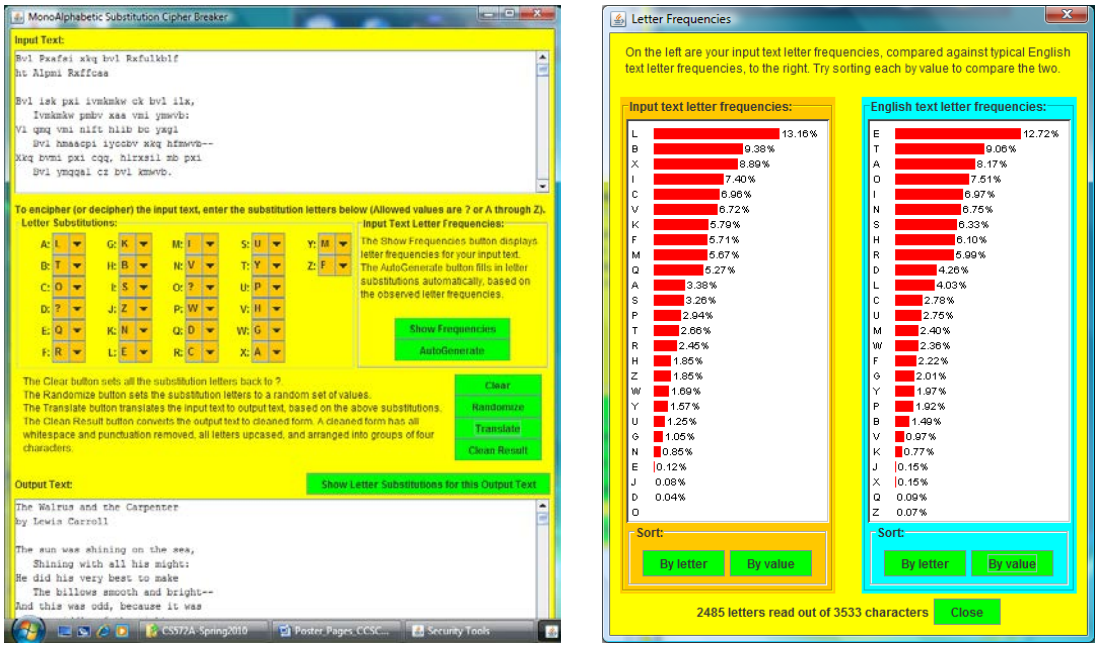


Figure 3. Deciphering the structured ciphertext. MonoAlphabetic Cipher Breaker (*left*) and letter frequencies in typical English (*right*).

Known letter frequencies in typical English text may be found on the web [19]. A typical representation of the letter frequencies in traditional English (E, T, A, O, I, N, S …) is

shown on the bar chart (see Fig. 3, right). The Java tool [10] allows a student to view the letter frequencies for the ciphertext being examined (see Fig. 3, center). Students may display letter frequencies in alphabetic order, or in order by frequency. If one of the characters has a 20% then the language may be German since it has a very high percentage of E. Italian has 3 letters with a frequency greater than 10% and 9 characters are less than 1% [20].

The linguistic analysis [22] also shows that common pairs in English are consonants TH and vowels EA. Others are OF, TO, IN, IT, IS, BE, AS, AT, SO, WE, HE, BY, OR, ON, DO, IF, ME, MY, UP. Common pairs of repeated letters are SS, EE, TT, FF, LL, MM and OO. Common triplets of text are THE, EST, FOR, AND, HIS, ENT or THA.

The MonoAlphabetic Cipher Breaker Java applet [10] (see Fig. 3, left) was used for deciphering the structured ciphertext (620 words; 2,485 letters out of 3,533 characters), where the original word spacing, punctuation, and style have been retained. Travis Brant, a CS graduate student, wrote in the assignment report: "…The solution was reached by only using the statistical distribution for the first three characters. Once those were in place, the text was long enough that searching for uncommon words with only one missing character was easily done. Once this practice was put into place, decoding the bulk of the message was reduced to an iterative process of searching for the next nearly-complete word. Decoding this message took about fifteen minutes."

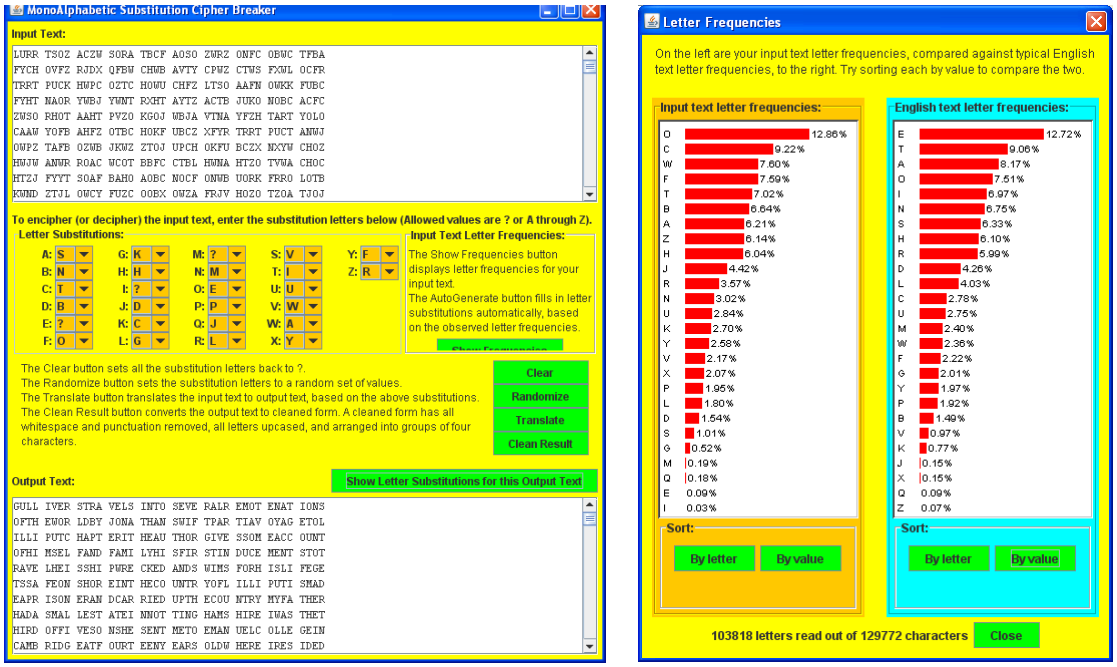


Figure 4. Deciphering the ciphertext organized in groups of four letters. MonoAlphabetic Cipher Breaker (*left*) and letter frequencies in typical English (*right*).

The second ciphertext (25,955 words; 103,818 letters out of 129,772 characters) was organized in groups of four letters and word spacing and punctuation have been

removed. The absence of the content clues (word spacing and punctuation) makes it more difficult to decipher the ciphertext, while the larger sample allows greater use of letter frequency analysis (see Fig. 4). Travis Brant reported:

> "… Deciphering this example was much more difficult than anticipated. The lack of preserved whitespace and punctuation made searching for possible word separation difficult. Once statistical analysis was performed on the text, there was little exposed that seemed correct. The only word that stuck out was the end of the first line, "ENAT IONS". I figured that this was as good of a start as any. Next, I caught some word pairings on the first three sections of line thirty, "OCIM ONTI NUED". At this point, I swapped M for C to create "O?IC ONTI NUED". At this point swapping Z for R brought more words forth. A large breakthrough was reached when L was swapped for G, spelling "GULL IVER" as the first words. From here, I looked up a sample of the text from this story "Gulliver's Travels" and saw that the message was the text from Jonathan Swift's work. I used the text from the book to identify and fix the remaining glitches in the decoded text, and was finished. The message was indeed the story of "Gulliver's Travels" by Jonathan Swift. This problem took about one-and-a-half hour of analysis to decipher".

To reduce the time of deciphering this unstructured ciphertext, one student even wrote the customized UNIX scripts and a standard UNIX dictionary to help with the mechanics of the solution [10].

3. The Graph Theory Implementation in the Structured Testing Methodology

In this case study, the structured testing methodology [23] and graph-based metrics (cyclomatic complexity *v*, essential complexity *ev*, module design complexity *iv*, system design complexity S_0, and system integration complexity S_1) [24] were reviewed 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 [25]. Comparing different code releases, it is found that the reduction of the code complexity leads to significant reduction of errors and maintainability efforts. Students worked on the selected cases analyzing algorithms, creating computer codes (in C/C++ or Java), running them at various parameters, comparing numerical results with known data, and presenting the findings to classmates [26, 27].

3.1 Software complexity metrics overview

The McCabe metrics [23, 24] are based on graph theory and mathematically rigorous analyses of the code structure, which explicitly identify high-risk areas. For each module (a subroutine with a single entry point and a single exit point), an annotated code listing and flowgraph is generated as shown in Fig. 5. The flowgraph is an architectural diagram of a software module's logic.

Cyclomatic complexity, *v*, is a measure of the complexity of a module's decision structure [24]. It is the minimum number of independent 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. The results of Miller's psychological experiments [28, 29] suggest that modules approach zero defects when *v* is within 7 ± 2. Therefore, the threshold of *v*-metric is chosen as 10. 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]. A module flowgraph with *e* edges and *n* nodes has the cyclomatic complexity $v = e - n + 2$ that is the number of topologically independent regions of the graph [23, 24, 26] (see Fig. 6).

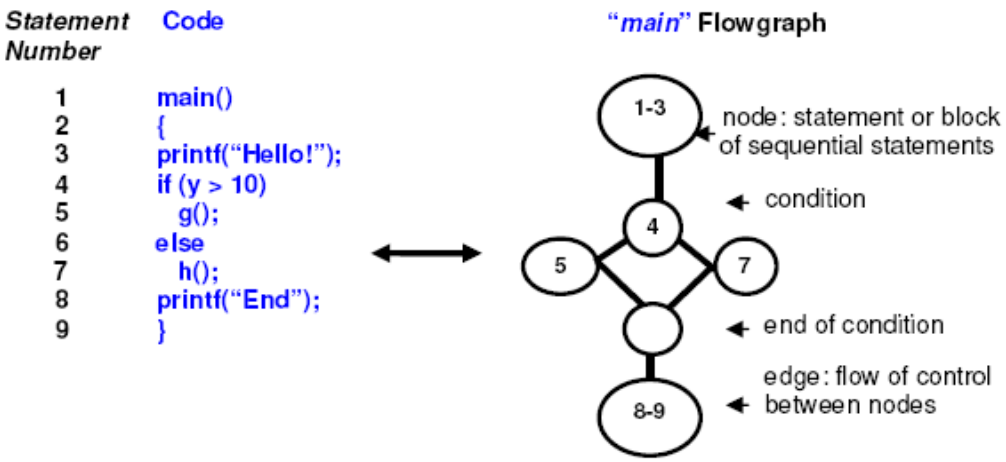


Figure 5. The annotated source listing and the related flowgraph.

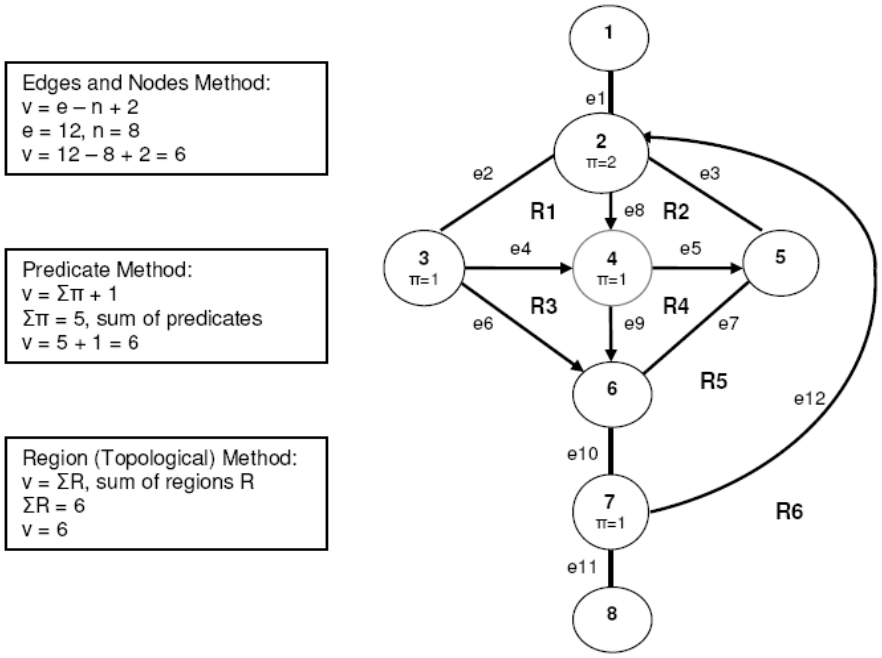


Figure 6. Three methods of evaluating the cyclomatic complexity of the sample graph.

Essential complexity, *ev*, is a measure of unstructuredness [24], the degree to which a module contains unstructured logical constructs [23], 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 (*ev* is high), modularization and maintenance is difficult. These modules should be recommended for redesigning.

Module design complexity, *iv*, is a measure of its decision structure as it relates to calls to other modules [23]. This quantifies test efforts of a module with respect to integration with other modules. Software with high values of *iv* tends to have a high degree of control coupling, which makes it difficult to isolate, maintain, and reuse software components.

System design complexity, S_0, measures the amount of interaction between modules in a program [23]. The S_0 metric is calculated as the sum of the module design complexities of all modules in a program. It reveals the complexity of the module calls in a program and measures the effort required for bottom-up integration testing. Integration complexity, S_1, measures the number of integration tests necessary to guard against errors [23]. It is the number of linearly independent sub-trees in a program. A sub-tree is a sequence of calls and returns from a module to its descendant modules. The S_1 metric quantifies the integration testing effort. It is calculated by using a simple formula [23], $S_1 = S_0 - N + 1$, where N is the number of modules in the program. Modules with no decision logic do not contribute to S_1. This fact isolates system complexity from its total size.

The McCabe IQ tool produces Halstead metrics [29, 30] for selected languages. Supported by numerous industry studies [23, 25], the *B*-metric of Halstead represents the estimated number of errors in the program.

3.2 Results of the project code analysis

The structured testing methodology [23] and McCabe's IQ tools were used in the C code analyses of different internetworking systems. As an example, the Support Carrier Networks system [25, 26] is studied. It provides both services of conventional Layer-2 switches and the routing and control services of Layer-3 devices. The code (3400 modules, about 300,000 code lines written in C language) was examined with the McCabe metrics [23, 24]. Nine protocol-based sub-trees of the code (for BGP, DVMRP, FR, ISIS, IP, MOSPF, OSPF2, PIM, and PPP networking protocols) were analyzed. This code analysis identified areas with potential errors and modules to be reviewed. This experience significantly improved the code implementation practices of our students.

It was found that 38% of the code modules have the cyclomatic complexity *v* more than 10. About 48% of modules have the essential cyclomatic complexity *iv* more than 4. Only two parts of the code (for FR and ISIS protocols) have low *v* and *ev* metrics. Totally 1147 modules (34%) are unreliable and unmaintainable. Among 3400 modules considered, 1447 modules (42%) are fully structured with $ev = 1$, and 500 modules (15%) are

completely unstructured with $ev = v$. 1066 code modules (31%) have the module design complexity more than 5. Only four protocol-based branches of the code (FR, ISIS, IP, and PPP) have low *iv*-metrics. BGP, MOSPF, and PIM protocol implementations have the worst characteristics (42% of modules require more than 7 integration tests per module). The system design complexity (S_0) is 19417, which is a top estimation of the number of unit tests that are required to fully test the program. The system integration complexity (S_1) is 16026, which is a top estimation of the number of integration tests.

The study of Halstead metrics [29, 30] indicates that the code potentially contains 2920 errors; 203 code modules (6%) have the number of delivered bugs $B > 3$ per module. Only five parts of the code (for FR, ISIS, IP, OSPF2, and PPP protocols) have relatively low error metrics ($B << 1$). In other branches (for BGP, DVMRP, MOSPF, and PIM protocols), $B > 1$.

3.3 Comparison of two customer software releases: Redesign efforts

Based on this analysis of the code, we recommended 271 modules of the old Release 1.2 for redesigning by the software development team. As a result, 16 old modules were deleted and 7 new modules were added for issuing the new Release 1.3. Analyzing the deleted modules, we found that 7 deleted modules were unreliable ($v > 10$) and 6 deleted modules were unmaintainable ($ev > 4$). Also, 19% of the deleted code was both unreliable and unmaintainable. All seven newly added modules were reliable and maintainable.

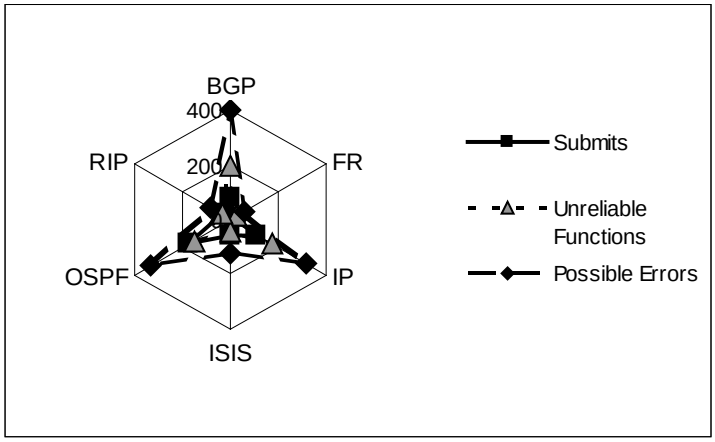


Figure 7. Correlation between the number of error submits, number of unreliable functions (with $v > 10$), and the number of possible errors for six protocols.

After redesigning, the code cyclomatic complexity was reduced by 115 units; 70 old modules (41% of the code) were improved, and only 12 modules (7% of the code) become worse. This analysis demonstrates a robustness of the structured testing methodology and mutual successful efforts of design and test engineers, which allow improving the quality of code releases sent to customers. Studying the relationship between software defect corrections and cyclomatic complexity [25, 26], we found a

good correlation between the numbers of possible errors, unreliable functions (with $v > 10$), and the error submits from the Code Releases (see Fig. 7).

4. Statistical Modeling with the Direct Simulation Monte-Carlo Technique

The direct simulation Monte Carlo (DSMC) method [31] and the two-dimensional DS2G code [32] have been used in this study as a numerical simulation technique for low-density hypersonic gas flows. The DSMC method is a computer-simulating technique for the modeling of real-gas effects by a sample of randomly-selected molecules (thousands or even millions). The position coordinates and velocity components of these molecules are stored in the computer memory and are modified with time as the molecules are concurrently followed through representative collisions and boundary interactions in the simulated physical space [31]. Intermolecular collisions in dilute gases are overwhelmingly likely to be binary collisions involving just two molecules. Given the physical properties of the molecules and the orientation of the trajectories, the post-collision velocities are determined from the equations of linear momentum and energy that must be conserved in the collision.

This direct simulation of the physical processes [31] contrasts with the "traditional" approach of computational fluid dynamics (CFD), which is based on obtaining numerical solutions of the fundamental mathematical equations and proper boundary conditions that model the processes [1-3]. In the cases of rarefied gas flows, when the gas density is sufficiently low, the direct physical simulation becomes a valuable simulating approach without any recourse to the conventional mathematical models of the flow. Under these conditions, the DSMC method becomes a unique adequate tool, because the full set of the Navier-Stokes "continuum-flow" equations does not provide a valid model for rarefied-gas flows, and conventional CFD methods are unable to handle the large number of independent variables that are involved in applications of the Boltzmann equation to realistic multidimensional problems [31]. The validation of the DSMC simulation approach was tested in comparing numerical results [33-36] with experimental data [37-39]. The results of the aerodynamic studies for plates, wedges, and disks are discussed below.

4.1 Aerodynamics of a blunt plate

The comparison of the DSMC numerical results for a drag coefficient of a plate (thickness $\delta = 0.1L$) with experimental data [38] in air (specific heat ratio $\gamma = 1.4$) is studied for Knudsen numbers $Kn_{\infty,L}$ from 0.02 to 3.2, Mach number $M_\infty = 10$, and temperature factor $t_w = T_w/T_0 = 1$. Numerical results [34, 36] correlate well with experimental data [38] at $0.02 < Kn_{\infty,L} < 1$. The free-molecular limit [40] is approached at $Kn_{\infty,L} > 3$.

The study of the influence of Mach number M_∞ on aerodynamic characteristics of bodies of simple shape was conducted at moderate values of the Knudsen number and at constant values of similarity parameters: $Kn_{\infty,L}$, t_w, and γ. The hypersonic stabilization regime [33] occurs at $M_\infty\theta >> 1$ in the case of streamlining of thin bodies when the angle

θ between the generatrix of the body surface and the upstream-flow direction becomes small enough. This regime is realized at smaller values of M_∞, if the angle θ increases.

The results of previous studies [37-39] indicate that the hypersonic flow independency principle [33] is realized in the transition rarefied-flow regime at $K = M_\infty \times \sin\theta > 1$. As was found in experiments [37-39], this principle is not true for thin bodies at small angles of attack in rarefied gas flows under the conditions $K < 1$.

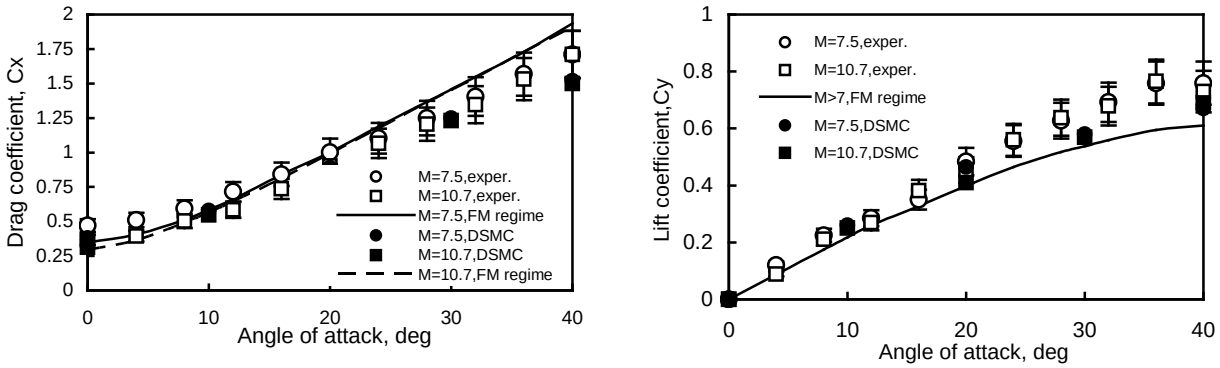


Figure 8. Drag and lift coefficients C_x, C_y for a blunt plate ($\delta = 0.06L$) at $Kn_{\infty,L} = 0.6$ and $M_\infty = 7.5$ (circles) and 10.7 (squares) in helium. Experimental data from Refs. 37-39.

At small angles of attack $\alpha < 12$ deg, the drag coefficient of a blunt plate having relative thickness $\delta = 0.06L$ becomes sensitive to the magnitude of the freestream Mach number in helium flow (see Fig. 8, *left*, $M_\infty = 7.5$ and $M_\infty = 10.7$). The results calculated by the DSMC technique (filled markers) correlate well with the experimental data [37-39] (empty markers). For the lift coefficient, the free-molecular flow data [40], as well as computational and experimental results presented in Fig. 8 (*right*), are independent of the Mach number, and the value $C_{y,FM}$ is less by approximately 15% than the value C_y for the transitional flow regime at $\alpha > 16$ deg. This phenomenon was discussed in Refs. 33, 37-39.

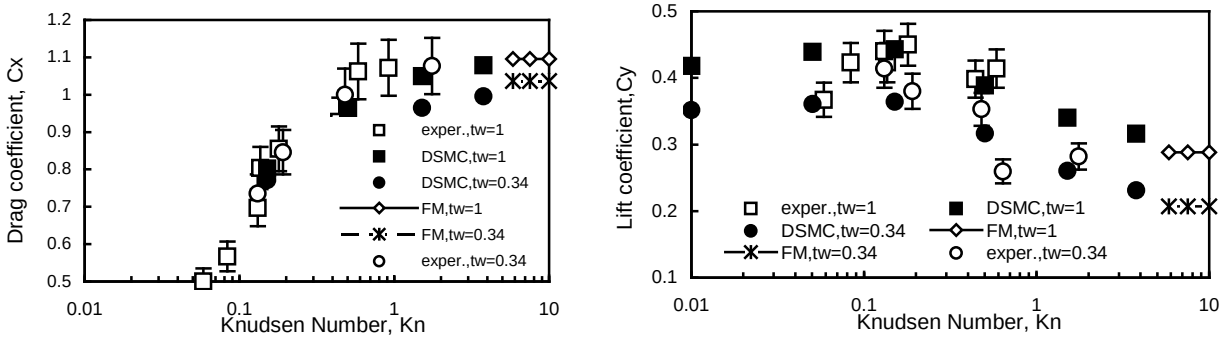


Figure 9. Drag and lift coefficients C_x, C_y for a blunt plate ($\delta = 0.1L$; $\alpha = 20$ deg) in air vs. Knudsen number $Kn_{\infty,L}$ at various temperature factors t_w. Experimental data from [38].

The temperature factor, t_w, is other important similarity parameter [33, 37-49, 41], which effects pressure at the body surface. Numerical data for a plate ($\delta = 0.1L$) at angle attack α = 20 deg and various $Kn_{\infty,L}$ has been studied (see Fig. 9). The lift coefficient, C_y changes non-monotonically from the continuum to the free-molecular flow regime. Maximum values occur in the transition flow regime. The influence of t_w can be estimated as 25% for C_y. The results correlate well with the experimental data [38].

4.2 Aerodynamics of a wedge

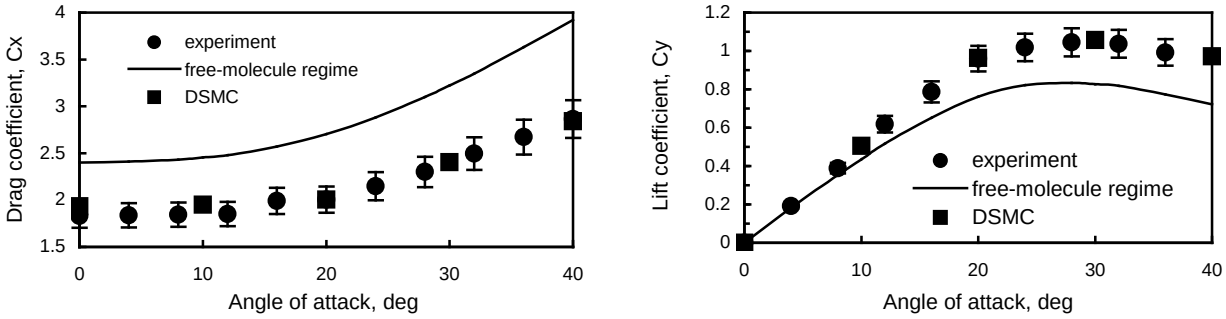


Figure 10. Drag and lift coefficients C_x, C_y for a wedge (θ = 20 deg) in helium flow at $Kn_{\infty,L}$ = 0.3 and M_∞ = 11.8. Experimental data from Refs. 6-9.

The dependence of drag and lift coefficients for a wedge (θ = 20 deg) on the angle of attack has been studied in numerical simulations of helium flow at $Kn_{\infty,L}$ = 0.3, t_w = 1, and the freestream Mach number M_∞ = 11.8. The DSMC results (squares) are shown in Fig. 10 for drag and lift coefficients. The base area of the wedge and its length were taken as the reference area and length. The numerical results correlate well with the experimental data [37-39] (circles), which were obtained in a vacuum wind tunnel at the same flow parameters. In both transitional and free-molecular [40] regimes, the characteristics are not sensitive to changes in upstream flow parameters at $M_\infty > 9$. Another interesting fact is that the lift-drag ratio in the transitional flow regime is larger by 50% than the corresponding parameter in the free-molecular regime [33, 38, 39, 41].

4.3 Aerodynamics of a disk

In the free-molecular flow regime, the influence of the specific heat ratio γ on the aerodynamic characteristics of bodies depends on the normal component of the momentum of the reflected molecules, which is a function of γ [33, 38, 39]. The same phenomenon can be observed at the transitional conditions in the case of the disk at α = 90 deg. The nitrogen-argon pair was the most acceptable one for testing [38, 39]. The dependencies of C_x of the disc for Ar (filled triangles) and N_2 (filled squares) are shown in Fig. 11 for a wide range of Knudsen numbers ($Kn_{\infty,D}$). At the same parameters of the upstream flow, numerical data obtained by the DSMC technique for different models of molecules are compared with experimental data [37-39]. This analysis could be applied to the design of a disk ballute [42].

The influence of specific heat ratio on the drag coefficient is more significant for large values of $Kn_{\infty,D} > 1$. In the free molecular regime ($Kn_{\infty,D} > 7$), an increase of C_x is observed as γ increases [38-40]. This increase is caused by the dependence on γ of the reflected momentum of the molecules at $t_w = 1$. The degree of this influence has been evaluated as 8% at $Kn_{\infty,D} > 2$. As the number $Kn_{\infty,D}$ decreases, this influence decreases, and at $Kn_{\infty,D} < 0.4$, the drag coefficient of the disk in a diatomic gas becomes larger than that for a monatomic gas. In the continuum flow regime, the dependence of the drag-coefficient on γ difference is insignificant.

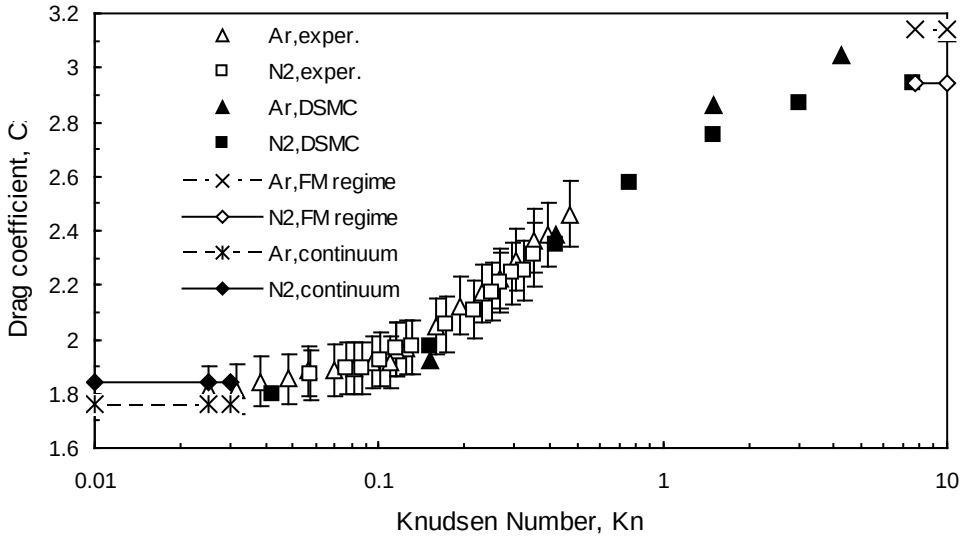


Figure 11. Drag coefficient C_x for a disk (α = 90 deg) vs. Knudsen number $Kn_{\infty,D}$ in argon and nitrogen. Experimental data from Refs. 37-39.

5. Concluding Remarks on Students' Involvement

The author has described some algorithms, tools, and experience of using the Java Applets in computer security courses for seniors and graduate students. The experience has been in general a very positive one, while at the same time providing useful lessons learned. The author believe that this algorithm-exploration and project-based approach with the Java Applets can be effectively applied to courses of a similar nature in academia, and the model can be extended to other areas of applied mathematics.

After brief in-class discussions of the case studies, each student continued working on a selected case analyzing algorithms, creating computer codes (in C/C++, Java, FORTRAN, or MATLAB), running them at various parameters, comparing numerical results with known data, and presenting the findings to classmates.

In particular, students' projects have revealed new information about hypersonic rarefied-gas flows near simple-shape bodies (plates, wedges, and disks) that can be effectively used for investigation and prediction of aerothermodynamic characteristics of hypersonic probes and vehicles during the design of their missions under the complex rarefied

atmospheric conditions of the Earth, Mars, Venus, and other planets. Fundamental insight into the probe characteristics and similarity parameters of these flows was obtained. For conditions approaching the hypersonic limit at $M_\infty >> 1$, the Knudsen number $Kn_{\infty,L}$ (or the equivalent Reynolds number $Re_{0,L}$) and temperature factor t_w are the primary similarity parameters. The influence of other parameters (the specific heat ratio γ, viscosity parameter n, and Mach number M_∞) is significant at $M_\infty\theta << 1$ and $Re_0 < 10$.

In the course evaluations, students stated that they became deeply engaged in course activities through examining the challenging problems related to the advanced concepts in applied mathematics (including the analysis of singular differential equations, strange attractors, the group theory, modular arithmetic, the theory of graphs, and statistical modeling of rarefied-gas flows with the Direct Simulation Monte-Carlo technique) and their practical applications.

Acknowledgements

The author would like to acknowledge his colleague, Dr. Bryan J. Higgs, who have played a direct role in modifying the Computer Science curriculum and designing the Java Applets for the Computer Security course.

Also the author would like to express gratitude to Dr. Graeme A. Bird for the opportunity of using the DS2G computer program, and to Dr. James N. Moss and Dr. Mikhail S. Ivanov for valuable discussions of the DSMC technique.

References

[1] Riabov, V.V. Exploring Singular Differential Equations with Exponential Box-Scheme. *Proceedings of the 19th Annual International Conference on Technology in Collegiate Mathematics* (Boston, MA, February 15-18, 2007), Englewood Cliffs, NJ: Prentice-Hall, 2007, pp. 173-177. [Online] http://archives.math.utk.edu/ICTCM/VOL19/C032/paper.pdf

[2] Riabov, V.V., & Provotorov, V.P. Exponential Box Schemes for Boundary-Layer Flows with Blowing. *Journal of Thermophysics and Heat Transfer*, 1996; 10(1): 126-130.

[3] Riabov, V.V. Applications of Exponential Box Schemes for Viscous Flows with Combustion and Blowing, In: *Computational Fluid and Solid Mechanics*, Proceedings of the Second MIT Conference on Computational Fluid and Solid Mechanics, June 17-20, 2003, edited by K.J. Bathe. Boston, MA: Elsevier, 2003, Vol. 1, pp. 1102-1105.

[4] Riabov, V.V. Advanced Study Cases for Numerical Analysis. *Journal of Computing Sciences in Colleges*, 2012; 27(6): 58-60.

[5] Lorenz, E.N. *The Essence of Chaos*. Boca Raton, FL: CRC Press, 1995.

[6] Lorenz, E.N. Deterministic Nonperiodic Flow. *Journal of the Atmospheric Sciences*, 1963; 20(2): 130–141.

[7] The Lorenz System. [Online] http://en.wikipedia.org/wiki/Lorenz_system/

[8] Lorenz Attractor. [Online] http://to-campos.planetaclix.pt/fractal/lorenz_eng.html
[9] Riabov, V.V. Exploring Mathematical Aspects of Algorithms for Challenging Topics in Numerical Analysis. *Proceedings of the 25th Annual International Conference on Technology in Collegiate Mathematics* (Boston, MA, March 21-24, 2013), Englewood Cliffs, NJ: Prentice-Hall, 2013, pp. 242-251. [Online] http://archives.math.utk.edu/ICTCM/VOL25/S132/paper.pdf
[10] Riabov, V.V., & Higgs, B.J. Algorithms and Software Tools for Teaching Mathematical Fundamentals of Computer Security. *Proceedings of the 23rd Annual International Conference on Technology in Collegiate Mathematics* (Denver, CO, March 17-20, 2011), Englewood Cliffs, NJ: Prentice-Hall, 2011, pp. 208-217. [Online] http://archives.math.utk.edu/ICTCM/VOL23/S062/paper.pdf
[11] 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.
[12] Graham, R.L., Knuth D.E., & Patashnik, O. *Concrete Mathematics*. Boston, MA: Addison-Wesley, 1994.
[13] Riabov, V.V., & Higgs, B.J. Running a Computer Security Course: Challenges, Tools, and Projects. *Journal of Computing Sciences in Colleges*, 2010; 25(6): 245-247.
[14] Bishop, D. *Introduction to Cryptography with Java Applets*. Burlington, MA: Jones & Bartlett Learning, 2002.
[15] Ferguson, N., & Schneier, B. *Practical Cryptography*. Hoboken, NJ: Wiley, 2002.
[16] Federal Information Processing Standard (FIPS) for the Advanced Encryption Standard, FIPS-197. [Online] http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
[17] Selent, D. Advanced Encryption Standard. *InSight: Rivier Academic Journal*, 2010, 6(2): 1-14. [Online] https://www2.rivier.edu/journal/ROAJ-Fall-2010/J455-Selent-AES.pdf
[18] Kaufman, C., Perlman, R., and Speciner, M. *Network Security: Private Communication in a Public World*, 2nd edition. Upper Saddle River, NJ: Prentice Hall, 2002.
[19] Frequencies. [Online] http://www.simonsingh.net/The_Black_Chamber/frequencyanalysis.html
[20] Riabov, V.V. The Role of Linguistics, Psychology, and Physiology in Various Computing Applications. *InSight: Rivier Academic Journal*, 2017; 13(2): 1-10. [Online] https://www2.rivier.edu/journal/ROAJ-Fall-2017/J1010_Riabov_Scientific_Phenomena_in_CS_Applications.pdf
[21] Riabov, V.V. Computing Applications Enriched by Scientific Phenomena. *Journal of Computing Sciences in Colleges*, 2018; 33(6): 189-191.
[22] Decrypting Text - code breaking software: Frequency analysis. Online: http://www.richkni.co.uk/php/crypta/freq.php
[23] 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, 1996.

[24] McCabe, T.J. A Complexity Measure. *IEEE Transactions on Software Engineering*, 1976; 2(4): 308-320.
[25] Riabov, V.V. Networking Software Studies with the Structured Testing Methodology. In: *Computer Science and Information Systems: Selected Papers from the 1st International Conference on Computer Sciences & Information Systems*, (Athens, Greece, 16-18 June 2005), Athens, Greece: Athens Institute for Education and Research, 2005, pp. 261-276.
[26] Riabov, V.V. Methodologies and Tools for the Software Quality Assurance Course. *Journal of Computing Sciences in Colleges*, 2011; 26(6): 86-92.
[27] Riabov, V.V. Exploring Algorithms for Effective Applications of the Graph Theory. *Proceedings of the 26th Annual International Conference on Technology in Collegiate Mathematics* (San Antonio, TX, March 20-23, 2014), Englewood Cliffs, NJ: Prentice-Hall, 2014, pp. 288-297. [Online] http://archives.math.utk.edu/ICTCM/VOL26/S086/paper.pdf
[28] Miller, G. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *Psychological Review*, 1956; 63 (2): 81-97.
[29] Riabov, V.V. Exploring Unique Computing Applications Enriched by Scientific Phenomena and Mathematical Concepts. *Proceedings of the 29th Annual International Conference on Technology in Collegiate Mathematics* (Chicago, IL, March 9-12, 2017), Englewood Cliffs, NJ: Prentice-Hall, 2017.
[30] Halstead, M.H. *Elements of Software Science*. New York, NY: Elsevier North Holland, 1977.
[31] Bird, G.A., *Molecular Gas Dynamics and the Direct Simulation of Gas Flows*, 1st ed., Oxford, England, UK: Oxford University Press, 1994.
[32] Bird, G.A. The DS2G program User's Guide. Version 3.2. Killara, Australia: G. A. B. Consulting Pty, 1999, pp. 1-56.
[33] Riabov, V.V. Comparative Analysis of Hypersonic Rarefied Gas Flows near Simple-Shape Bodies. *Journal of Spacecraft and Rockets*, 1998; 35(4): 424-433.
[34] Riabov, V.V. Numerical Study of Interference between Simple-Shape Bodies in Hypersonic Flows. *Computers and Structures*, 2009; 87: 651-663.
[35] Riabov, V.V. Rarefaction Effects in Hypersonic Aerodynamics. In: *27th International Symposium on Rarefied Gas Dynamics, AIP Conference Proceedings*. Washington, DC: American Institute of Physics, 2011, Vol. 1333, Part II, pp. 1331-1336.
[36] Riabov, V.V. Mathematical Modeling and Simulation of Complex Phenomena in Rarefied Gases. *Proceedings of the 29th Annual International Conference on Technology in Collegiate Mathematics* (Chicago, IL, March 9-12, 2017), Englewood Cliffs, NJ: Prentice-Hall, 2017.
[37] Gusev, V.N., Klimova, T.V., & Riabov, V.V. The Basic Characteristics of the Variation of Aerodynamic Parameters in the Transition Regime at Hypersonic Flow Velocities. *Uchenyye Zapiski TsAGI*, 1976; 7(3): 47-57 (in Russian).
[38] Gusev, V.N., Erofeev, A.I., Klimova, T.V., Perepukhov, V.A., Riabov, V.V., & Tolstykh A.I. Theoretical and Experimental Studies of Flow over Bodies of Simple Shapes by a Hypersonic Stream of Rarefied Gas. *Trudy TsAGI*, 1977; 1855: 3-43 (in Russian).

[39] Riabov, V.V., Aerodynamic Applications of Underexpanded Hypersonic Viscous Jets. *Journal of Aircraft*, 1995; 32(3): 471-479.
[40] Kogan, M.N. *Rarefied Gas Dynamics*, New York, NY: Plenum Press, 1969, pp. 345-390.
[41] Gusev, V.N., Kogan, M.N., & Perepukhov, V.A. The Similarity and Aerodynamic Measurements in Transitional Regime at Hypersonic Velocities. *Uchenyye Zapiski TsAGI*, 1970; 1(1): 24-31 (in Russian).
[42] McRonald, A. A Light-Weight Inflatable Hypersonic Drag Device for Planetary Entry. AIAA Paper 99-0422, Association Aeronautique de France Conference, Arcachon, France, March 16-18, 1999.