# Mathematical and Engineering Tools/Software/Game For Learning STEM Topics

D.T. Nguyen[1], S. Kadiam[1], A. Kaw[2], A. Yalcin[2], G. Lee-Thomas[1], A. Mohammed[1], R. Pendyala[3]
[1]Old Dominion University (ODU); [2]University of South Florida (USF); [3]Arizona State University (ASU)

**Abstract:** Solving large (and sparse) system of simultaneous linear equations has been (and continue to be) a major challenging problem for many real-world engineering/science applications [1-2]. For many practical/large-scale problems, the sparse, Symmetrical and Positive Definite (SPD) system of linear equations can be conveniently represented in matrix notation as $[A]\{x\}=\{b\}$, where the square coefficient (non-singular) matrix $[A]$ and the Right-Hand-Side (RHS) vector $\{b\}$ are known. The unknown solution vector $\{x\}$ can be efficiently solved by the following step-by-step procedures [1-2]: Reordering phase, Matrix Factorization phase, Forward solution phase, and Backward solution phase.

In this research work, online Minimizing Fill-In-Terms Game Based Learning (MFIT-GBL) approach and Unlimited Attempt Self-Assessment Online Quizzes (UASOQs) have been developed/tested to help engineering students to understand crucial details about step-by-step numerical procedures for solving Simultaneous Linear Equations (SLE). The MFIT-GBL tool has been developed and can be played by either a single player, or two players. Through this MFIT-GBL open-ended game, the players/learners will not only understand the key concepts involved in reordering algorithms (based on existing algorithms), but also have the opportunities to "discover new algorithms" which are better than existing algorithms. Implementing the proposed MFIT_GBL for matrix reordering and factorization phases can be enhanced by FLASH [3] computer environments, where computer simulation with animated human voice, sound effects, visual/graphical/colorful displays of matrix tables, score (or monetary) awards for the best game players, etc. can all be exploited. Next, Structural Matrix Method (SMM) software module for helping Structural Engineering undergraduate students to practice their problem solving skills and self-assessment tests are also described. Finally, through the developed UASOQs, on-line Cholesky algorithms for solving simultaneous linear equations (SLE), symbolic input for integration software tools, engineering students will have the necessary (online) tool to practice their problem solving skills, and to have good assessment tool to self-evaluate their learning capabilities.

## I. Introduction

Since 2002, the open courseware resources at Holistic Numerical Methods Institute (HNMI) [7-9] have enhanced instructor preparation and development as well as the student educational experience by facilitating a hybrid educational approach to the teaching of Numerical Methods, a pivotal Science, Technology, Engineering, Mathematics (STEM) course, via

(a) customized textbooks (b) adapted course websites, (c) social networking via blogs, (d) YouTube audiovisual lectures, (e) online multiple-choice question tests (f) worksheets in a computational system of choice, (g) real-life applications based on the choice of one's STEM major

For a search on 'numerical methods', HNMI is ranked #2 on Google and in the last year (December 1, 2009 – November 30, 2010), there were

- 1,001,721 page views (328,699 visits) to Ref. [7],
- 489,541 views of the audiovisual lectures on YouTube [8], and
- 76,435 visits to the numericalmethodsguy blog [9].

Knowing that "knowledge that is taught in only a single context is less likely to support flexible knowledge transfer than is knowledge that is taught in multiple contexts" [*10*] and to further improve the educational experience and performance, a number of new prototype tools have been developed, implemented and assessed for the topic of Simultaneous Linear Equations under this project. These tools are:

- Online Minimizing Fill-In-Terms Game Based Learning (MFIT-GBL) approach [4] for learning re-ordering algorithm and solving Simultaneous Linear Equations (SLE).
- On-line Structural Matrix Method (SMM) software module for problem solving skills and self-assessment tests [6].
- Unlimited Attempt Self-Assessment Online Quizzes (UASOQs).
- On-line problem solving skills and self assessment tests (related to Cholesky algorithms) [14].
- On-line problem solving skills (related to symbolically input for integration topic) [15].

## II. Online Minimizing Fill-In-Terms Game Based Learning (MFIT-GBL) Approach [4, 5] for Learning Re-ordering Algorithm and Solving Simultaneous Linear Equations (SLE) [1, 2, 4, 5].

Solving large (and sparse) system of simultaneous linear equations (SLE) has been (and continue to be) a major challenging problem for many real-world engineering/science applications [1-2]. In matrix notation, the SLE can be represented as:

$$[A]\{x\} = \{b\} \tag{1}$$

where [A] = known coefficient (non-singular) matrix, with dimension NxN
{b} = known right-hand-side (RHS) Nx1 vector
{x} = unknown Nx1 vector.

### 2.1 Symmetrical Positive Definite (SPD) SLE

For many practical SLE, the coefficient matrix [A] (see Eq.1) is SPD. In this case, efficient 3-step Cholesky algorithms [1-2] can be used.

#### Step 1: Matrix Factorization Phase

In this step, the coefficient matrix [A] can be decomposed into

$$[A] = [U]^T [U] \tag{2}$$

where [U] is a NxN upper triangular matrix.

The following simple example will illustrate how to find the matrix [U].
Various terms of the factorized matrix [U] can be computed/derived as following (see Eq. 2):

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} = \begin{bmatrix} u_{11} & 0 & 0 \\ u_{12} & u_{22} & 0 \\ u_{13} & u_{23} & u_{33} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix} \tag{3}$$

Multiplying 2 matrices on the right-hand-side (RHS) of Eq. (3), then equating each upper-triangular RHS terms to the corresponding ones on the upper-triangular left-hand-side (LHS), one gets the following 6 equations for the 6 unknowns in the factorized matrix $[U]$.

$$u_{11} = \sqrt{A_{11}}\,; u_{12} = \frac{A_{12}}{u_{11}}\,; u_{13} = \frac{A_{13}}{u_{11}} \tag{4}$$

$$u_{22} = \left(A_{22} - u_{12}^2\right)^{\frac{1}{2}}; u_{23} = \frac{A_{23} - u_{12}u_{13}}{u_{22}}\,;\; u_{33} = \left(A_{33} - u_{13}^2 - u_{23}^2\right)^{\frac{1}{2}} \tag{5}$$

In general, for a general NxN matrix, the diagonal and off-diagonal terms of the factorized matrix $[U]$ can be computed from the following formulas:

$$u_{ii} = \left( A_{ii} - \sum_{k=1}^{i-1} \left(u_{ki}\right)^2 \right)^{\frac{1}{2}} \tag{6}$$

$$u_{ij} = \frac{A_{ij} - \sum_{k=1}^{i-1} u_{ki} u_{kj}}{u_{ii}} \tag{7}$$

As a quick example, one computes:

$$u_{57} = \frac{A_{57} - u_{15}u_{17} - u_{25}u_{27} - u_{35}u_{37} - u_{45}u_{47}}{u_{55}} \tag{8}$$

Thus, for computing $u(i = 5, j = 7)$, one only needs to use the (already computed) data in columns # i(=5), and # j(=7) of [U], respectively.

**Step 2: Forward Solution phase**

Substituting Eq. (2) into Eq. (1), one gets:

$$[U]^T [U]\{x\} = \{b\} \tag{9}$$

Let's define:

$$[U]\{x\} \equiv \{y\} \tag{10}$$

Then, Eq. (9) becomes:

$$[U]^T \{y\} = \{b\} \tag{11}$$

Since $[U]^T$ is a lower triangular matrix, Eq. (11) can be efficiently solved for the intermediate unknown vector $\{y\}$, according to the order $\begin{Bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_N \end{Bmatrix}$, hence the name "forward solution".

In general, one has

$$y_j = \frac{b_j - \sum_{i=1}^{j-1} u_{ij} y_i}{u_{jj}} \tag{12}$$

**Step 3: Backward Solution phase**

Since $[U]$ is an upper triangular matrix, Eq. (10) can be efficiently solved for the original unknown vector $\{x\}$, according to the order $\begin{Bmatrix} x_N \\ x_{N-1} \\ x_{N-2} \\ . \\ x_1 \end{Bmatrix}$, hence the name "backward solution".

In general, one has:

$$x_j = \frac{y_j - \sum_{i=j+1}^{N} u_{ji} x_i}{u_{jj}} \tag{13}$$

## 2.2 Re-Ordering Algorithms for Minimizing Fill-in Terms [1]

During the factorization phase (of Cholesky algorithms), many "zero" terms in the original/given matrix [A] will become "non-zero" terms in the factored matrix [U]. These new non-zero terms are often called as "fill-in" terms (indicated by the symbol F). It is, therefore, highly desirable to minimize these fill-in terms, so that both computational time/effort and computer memory requirements can be substantially reduced. For example, the following matrix [A] and vector $\{b\}$ are given:

$$[A]=\begin{bmatrix} 112 & 7 & 0 & 0 & 0 & 2 \\ 7 & 110 & 5 & 4 & 3 & 0 \\ 0 & 5 & 88 & 0 & 0 & 1 \\ 0 & 4 & 0 & 66 & 0 & 0 \\ 0 & 3 & 0 & 0 & 44 & 0 \\ 2 & 0 & 1 & 0 & 0 & 11 \end{bmatrix} \tag{14}$$

$$\{b\}=\begin{Bmatrix} 121 \\ 129 \\ 94 \\ 70 \\ 47 \\ 14 \end{Bmatrix} \tag{15}$$

By appropriated swapping the equation numbers (by using appropriated reordering algorithms), such as

IPERM (new equation #) = {old equation #}. As an example,:

$$IPERM\begin{Bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{Bmatrix}=\begin{Bmatrix} 6 \\ 5 \\ 4 \\ 3 \\ 2 \\ 1 \end{Bmatrix} \tag{16}$$

Using the above results (see Eq. 16), one will be able to construct the following re-arranged matrices:

$$[A^*]=\begin{bmatrix} 11 & 0 & 0 & 1 & 0 & 2 \\ 7 & 44 & 0 & 0 & 3 & 0 \\ 0 & 0 & 66 & 0 & 4 & 0 \\ 1 & 0 & 0 & 88 & 5 & 0 \\ 0 & 3 & 4 & 5 & 110 & 7 \\ 2 & 0 & 0 & 0 & 7 & 112 \end{bmatrix} \tag{17}$$

and

$$\{b^*\}=\begin{Bmatrix} 14 \\ 47 \\ 70 \\ 94 \\ 129 \\ 121 \end{Bmatrix} \tag{18}$$

Now, one would like to solve the following modified system of linear equations (SLE) for $\{x^*\}$,

$$[A^*]\{x^*\} = \{b^*\} \tag{19}$$

rather than to solve the original SLE (see Eq.1). The original unknown vector $\{x\}$ can be easily recovered from $\{x^*\}$ and $\{IPERM\}$, shown in Eq. (16).

It is clearly to recognize, at this moment, the big benefits of solving the SLE shown in Eq. (19), instead of solving the original Eq. (1), since the factorization of matrix [A*] has much less fill-in term, as compared to fill-in-terms occurred in the factorization of the original matrix [A] !

### 2.3 On-line MFIT-GBL For Reordering/Factorized Phase [4, 5].

Based on the discussions presented in the previous sections, one can easily see the similar operations between the symbolic, numerical factorization and reordering phases of sparse SLE.

In this work, MFIT-GBL tool (shown in Figure 1 [4, 5]) has been designed with the following objectives:

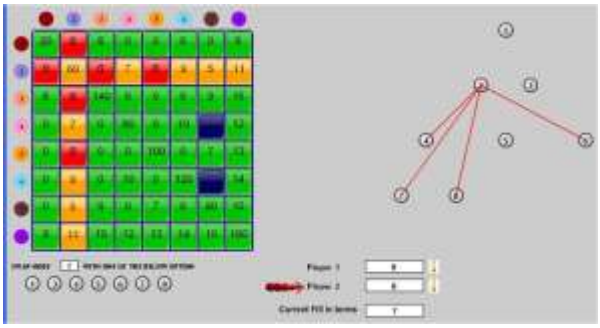


Figure 1: Online MFIT-GBL For Learning to Solve SLE.

(A) Teaching undergraduate students the process how to use the reordering output IPERM(-), see Eq. (16) for converting the original/given matrix [A], see Eq. (14), into the new/modified matrix $[A^*]$, see Eq. (17). This step is reflected in Figure 1, when the "Game Player" decides to swap node (or equation) "i" (say i =2) with another node (or equation) "j", and click the "CONFIRM" icon!

Since node "i = 2" is currently connected to nodes j = 4,6,7,8; hence swapping node i = 2 with the above nodes j will "NOT" change the number/pattern of "Fill-in" terms. However, if node i =2 is swapped with node j=1, or 3, or 5, then the fill-in terms pattern may change (for better or worse) !

(B) Helping undergraduate students to understand the "symbolic" factorization" phase, by symbolically utilizing the Cholesky factorized Eqs. (6-7). This step is illustrated in Figure 1, for which the "game player" will see (and also hear the computer animated sound, and human

voice), the non-zero terms (including fill-in terms) of the original matrix [A] to move to the new locations in the new/modified matrix $[A^*]$.

(C) Helping undergraduate students to understand the "numerical factorization" phase, by numerically utilizing the same Cholesky factorized Eqs. (6-7).

(D) Teaching undergraduate engineering/science students to "understand existing reordering concepts", or even to "discover new reordering algorithms"

## III. Structural Matrix Method (SMM) Software Module [6]

The Stiffness Matrix Method (SMM) module has been developed and evolved for CEE- 310 Structural Analysis [6], a junior-level course required for the BS in Civil Engineering program. The module development and implementation is part of an ongoing transformation of undergraduate education at Old Dominion University (ODU), which seeks to integrate technology-based student learning tools in a number of undergraduate engineering courses. It is important to develop web-tools for undergraduate engineering education that are simulation and visualization based, and that can be used in "any time-any place" mode.

The SMM module [6] includes brief reading sections on various components of the SMM process and the theoretical backgrounds behind the developed formulas adopted for calculations. The reading sections are followed by an interactive application unit, which includes the computation of the structural responses (such as nodal displacements, member-end-actions and support reactions), visualization and animation (such as plots of un-deformed and deformed structures, as shown in Figure 2) [6], with highlighted observations to enhance student learning. Students are then assigned exercises, which require both hand calculations and the use of the interactive unit.

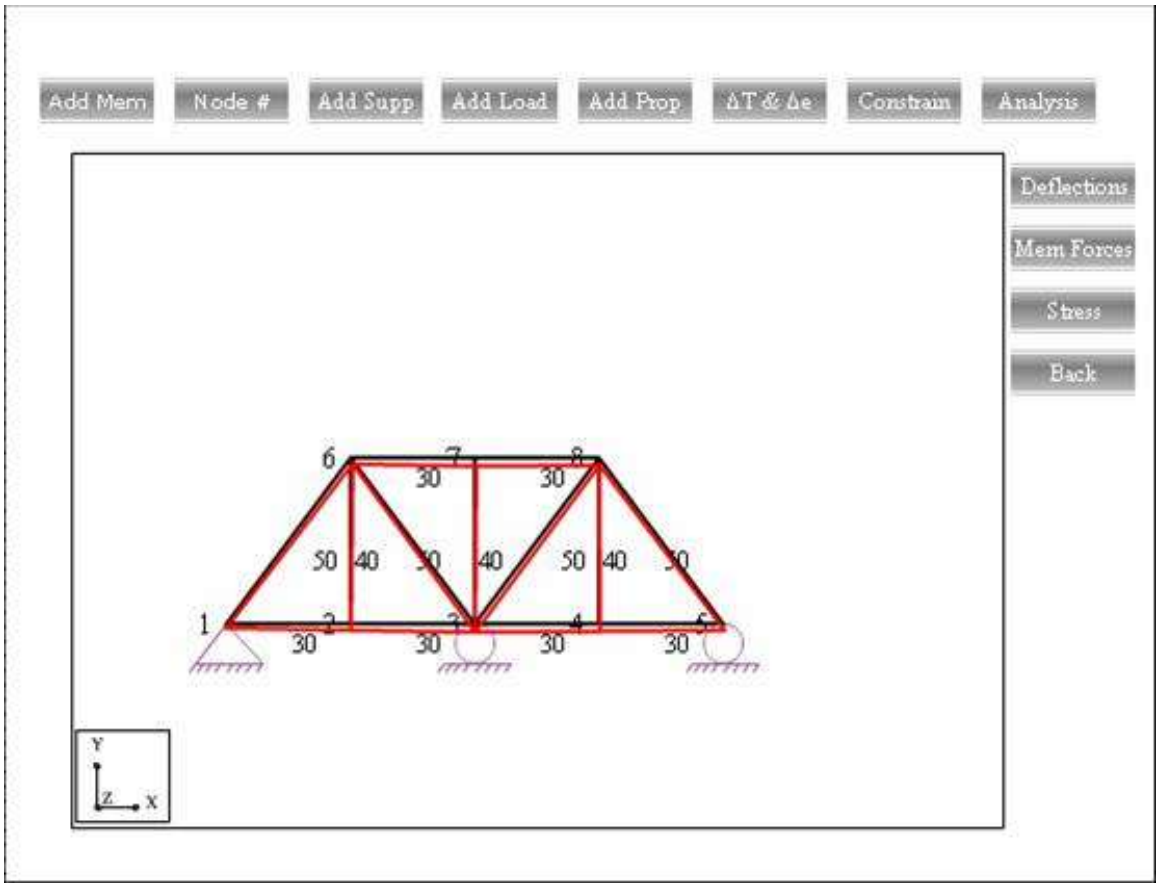


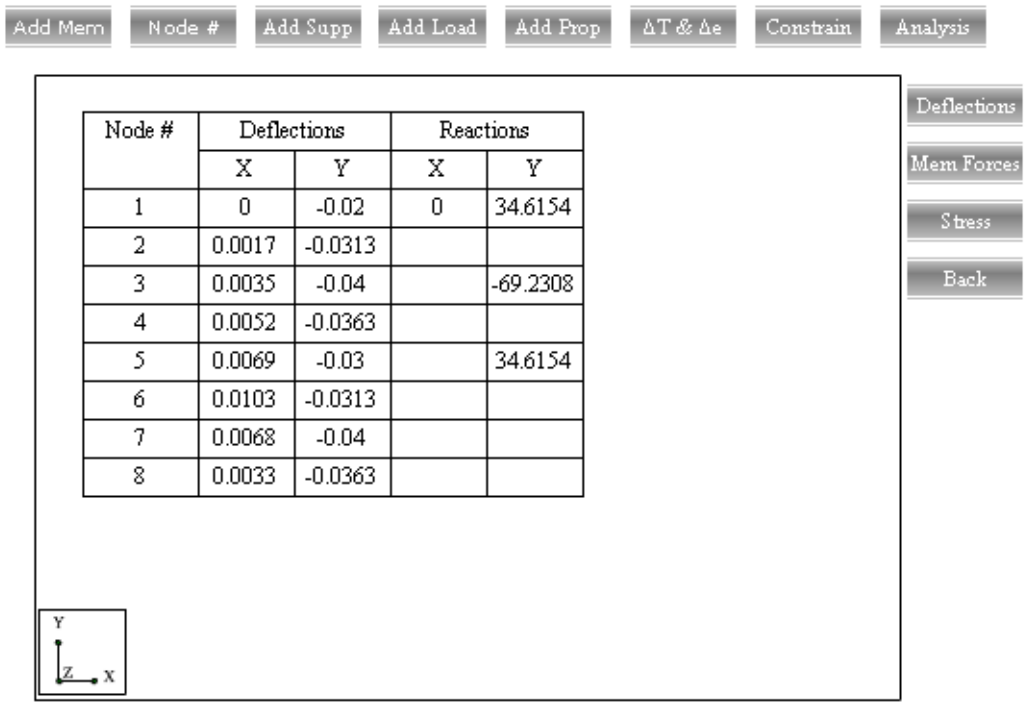

Figure 2: Deflected Shape for Support "Settlements" Bridge (Truss) Example.

**Theoretical background for the Stiffness Matrix Method [6]:**

The entire Stiffness Matrix Method (SMM) will involve with the following major components:

(a) Element local matrices
(b) Element global matrices
(c) Assembly process
(d) Boundary conditions
(e) Solution of system of linear equations
(f) Structural responses (see Figure 3)

Details of the above key components has been explained and presented in the ODU website [6] (then click on the theoretical development module). More advanced treatments of the above item (e) can be found in Refs. [1, 2, 6].




| Node # | Deflections | | Reactions | |
|---|---|---|---|---|
| | X | Y | X | Y |
| 1 | 0 | -0.02 | 0 | 34.6154 |
| 2 | 0.0017 | -0.0313 | | |
| 3 | 0.0035 | -0.04 | | -69.2308 |
| 4 | 0.0052 | -0.0363 | | |
| 5 | 0.0069 | -0.03 | | 34.6154 |
| 6 | 0.0103 | -0.0313 | | |
| 7 | 0.0068 | -0.04 | | |
| 8 | 0.0033 | -0.0363 | | |

Figure 3: Deflections and reactions for support "settlements" example.

**Students Self-Assessment Test (in none_multiple choice style) [6]:**

Self-Assessment module is a friendly (and critically important) module where students can assess their performance themselves. In this module, separate set of (randomly generated) questions were designed for 2-D Truss, Beam and/or Frame problems. The student has the choice to take the self-assessment test on either 2-D truss, beam, or frame problems. The results of the test are automatically graded and sent to the instructor and student by E-mail. The main advantage of this module is that the student has to compute some "detail, intermediate" variables (see Figure 4) before getting their "final" answers. While the *final answers* could be obtained by the student through the developed "Interactive Simulation & Visualization Module" [6], the *intermediate answers* are "intentionally" unrevealed, for student's self-assessment test purpose! The grading policy adopted in this module is as follows: the student will be awarded 100% for each correct answer, otherwise only 35% for partial credit will be given to him/her for a wrong answer. At the end, the student's average test score is calculated and sent to his/her email address, along with the student's answers and the correct answers (automatically generated/printed by the computer, as shown in Figure 5). The grading policy of this module can be changed according to the instructor's choice. This is also a very helpful module for instructor, since he/she does not have to "painfully grade" students' tests (especially for those classes with high student enrollments!).

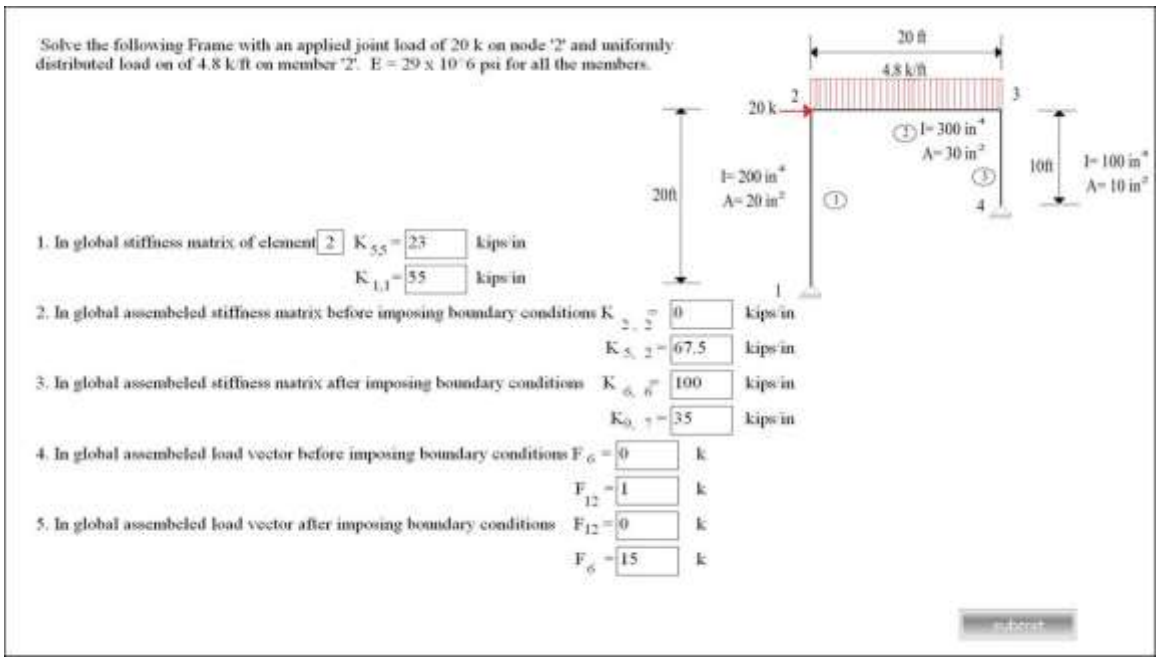



Figure 4: Self-Assessment Test: Frame problem (Students will enter his/her answers in the textbox provided).

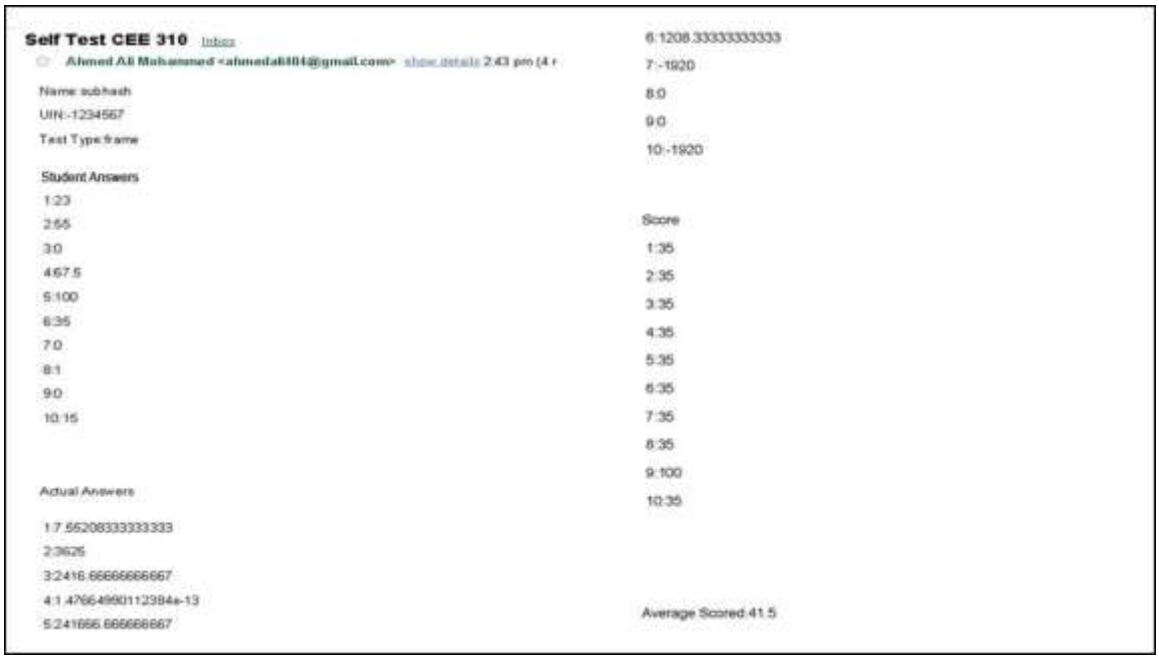



Figure 5: Self-Assessment results and graded score are included in each student's email

## IV. Unlimited Attempt Self-Assessment Online Quizzes (UASOQs)

In an effort to find the best use of limited teaching assistant's time in today's economy , a study reported at last year's ASEE 2010 conference by two of the authors of this poster indicated that there is no statistically significant difference in student performance when homework is collected and graded versus assigned and not graded.
Because of the use of course management systems such as Blackboard [11] in most universities and the availability of quiz makers such as Respondus [12], we wanted to determine if using graded online quizzes would improve the student performance, while minimizing grading time. Although other online homework and grading systems such as WebAssign [13] are available, there are substantial recurring costs associated with the use of such systems both for students and faculty, and at this time availability of engineering topics in these systems is very limited.

As a pilot study, using Respondus, we developed the UASOQs for the three subtopics of Simultaneous Linear Equations topic taught in the class – Background of Matrix Algebra, Gaussian Elimination and LU Decomposition. All quizzes had 6-7 questions, and were of algorithmic form. This form allows the instructor to choose some or all variables to take values within a range, and develop a formula for the correct answer. These formulas were developed by writing scripts with symbolic computation in MATLAB and Maple.

When a student takes the quiz, the system randomly chooses the values of the selected variables, and he/she answers the question by filling in the answer field (see Fig. 6). The student's answer is checked against the correct value. Feedback, including the answer and the correctness of the answer, is given immediately.

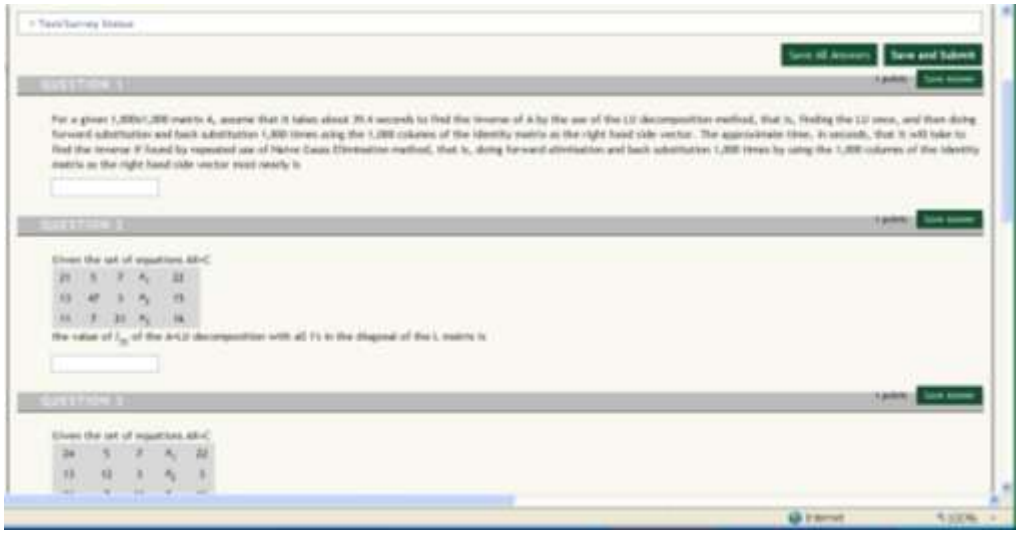


Figure 6: A Screen Shot of the Developed UASOQs

## V. On-Line Software (Numerical Method Topics) for Problem Solving Skills [14, 15]

The developed on-line software has the following important/nice features:

- No software license is required.
- The users/learners are not even required to download any software to his/her desktop/laptop computer.
- For numerical methods topics, it was NOT easy/convenient to write the appropriated program for self-assessment tests/quizzes. This is especially true if the students are asked to find "many numbers" (instead of finding just one number) within a question.
- Most (if not all) of self-assessment computer software system do use "random generations" to randomly create "DIFFERENT" numerical values for the "SAME" style of question(s). The users usually are NOT allowed to provide his/her own input data. This is especially true for the "SYMBOLIC" data (such as function expression) in addition to the usual "NUMERICAL" input data.

### 5.1 Cholesky Method with Self-Assessment Tests [14]

A screen shot for the developed Cholesky self-assessment test is shown in Figure 7

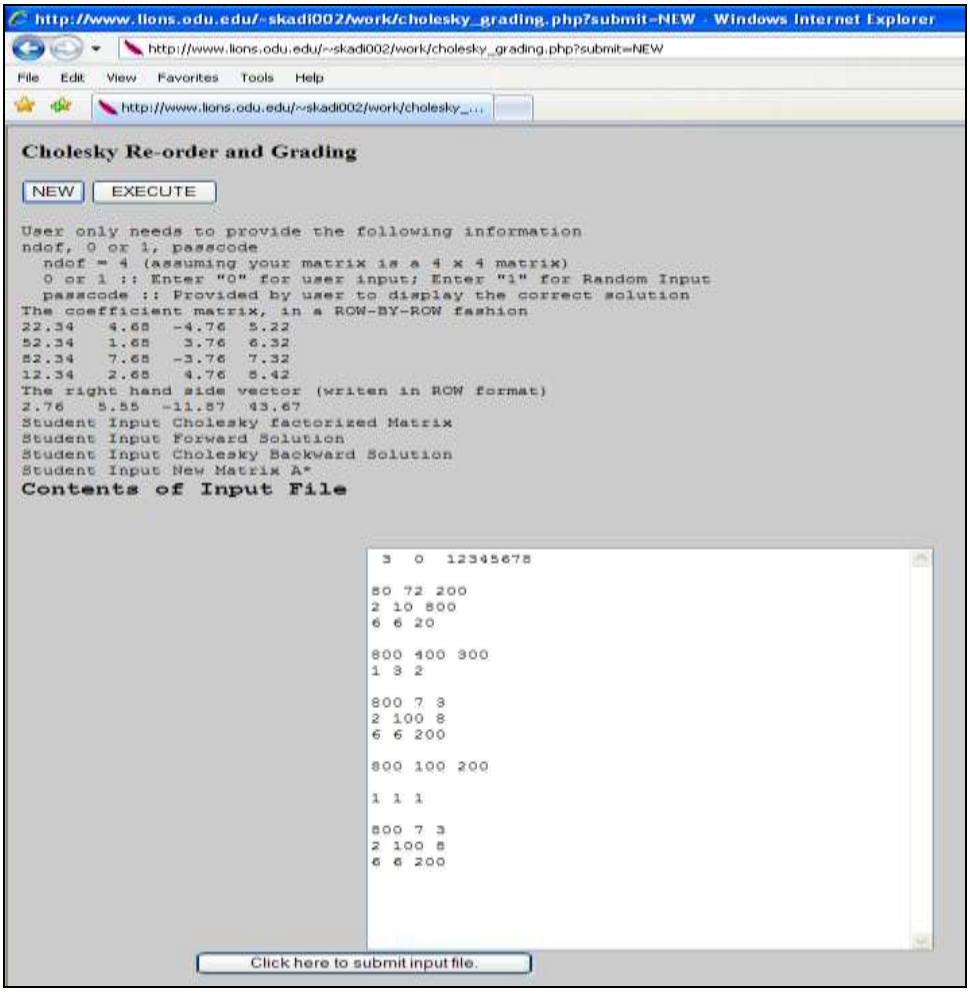



Figure 7: Advanced Endless Quiz for Cholesky Solver

### 5.2 On-Line Symbolic Input for Integration (by multiple trapezoidal rule) Problem Solving Skills [15]

A screen shot for the developed on-line "multiple trapezoidal rule for integration" with user's specified/input "symbolic function" is shown in Figure 8.

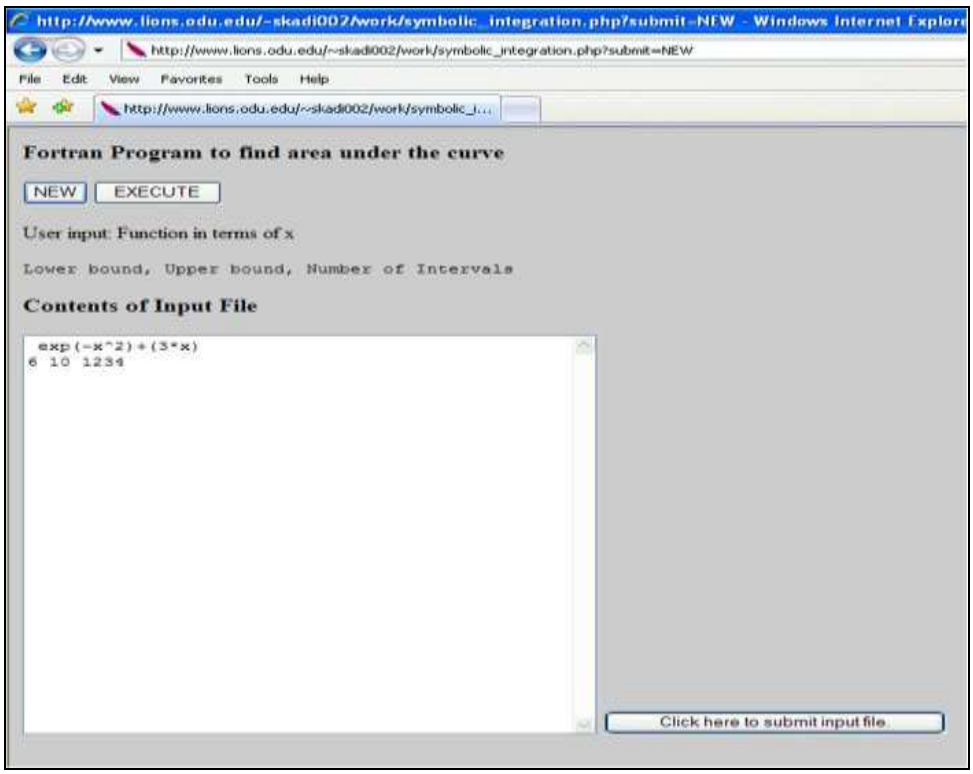



Figure 8: Advance Endless Quiz for "Multiple-Trapezoidal Rule for Integration" (with user's input symbolic function)

## VI. Conclusions

A fairly long lists of new prototype tools/software/game have been developed/tested to enhance the educational experience and examination performance of STEM (Engineering) students (for a "Structural Analysis" course), and (Engineering/Mathematical) students (for a "Numerical Methods" course). The developed software/tools/game is expected to be helpful to both STEM students and educators.

## VII. Acknowledgements

The authors would like to acknowledge the partial supports, provided in this work through the National Science Foundation (NSF Grant #0836916).

## VIII. References


1. Duc T. Nguyen. (2006). *Finite Element Methods: Parallel-Sparse Statics and Eigen-Solutions*, Springer Publishers.
2. Duc T. Nguyen. (2002). *Parallel-vector Equation Solvers for Finite Element Engineering Applications*, Kluwer Academic/Plenum Publishers
3. www.brothersoft.com/downloads/flash-animation-software.html.
4. http://www.lions.odu.edu/~amoha006/Fillinterms/FILLINTERMS.html
5. www.lions.odu.edu/~skadi002 (then CLICK/select CEE-305 course materials, and see YOUTUBE video items #23, #25).
6. http://www.lions.odu.edu/~amoha006
7. **http://numericalmethods.eng.usf.edu**
8. **http://youtube.com/numericalmethodsguy**
9. **http://autarkaw.wordpress.com**
10. *How People Learn: Brain, Mind, Experience and School* , J. D. Bransford, et al, 1999, p. 66.
11. http://blackboard.com
12. http://respondus.com
13. http://www.webassign.net/
14. http://www.lions.odu.edu/~skadi002/work/cholesky.html
15. http://www.lions.odu.edu/~skadi002/work/symbolic_integration.php