2016

Java Computer Animation for Effective Learning of the Cholesky Algorithm with Transportation Engineering Applications

Ivan Makohon
Old Dominion University, imako001@odu.edu

Duc T. Nguyen
Old Dominion University, dnguyen@odu.edu

Mecit Cetin
Old Dominion University, mcetin@odu.edu

Follow this and additional works at: https://digitalcommons.odu.edu/cee_fac_pubs

Part of the Civil Engineering Commons, and the Electrical and Computer Engineering Commons

Java Computer Animation for Effective Learning of the Cholesky Algorithm with Transportation Engineering Applications

Ivan Makohon[1]*, Duc T. Nguyen[2], Mecit Cetin[3]

[1]Modeling, Simulation & Visualization Engineering (MSVE) Department, Old Dominion University, Norfolk, VA, USA
[2]CEE and MSVE Department, Old Dominion University, Norfolk, VA, USA
[3]Civil & Environmental Engineering (CEE) Department, Old Dominion University, Norfolk, VA, USA
Email: *imako001@odu.edu, dnguyen@odu.edu, mcetin@odu.edu

Abstract

In this paper, the well-known Cholesky Algorithm (for solving simultaneous linear equations, or SLE) is re-visited, with the ultimate goal of developing a simple, user-friendly, attractive, and useful Java Visualization and Animation Graphical User Interface (GUI) software as an additional teaching tool for students to learn the Cholesky factorization in a step-by-step fashion with computer voice and animation. A demo video of the Cholesky Decomposition (or factorization) animation and result can be viewed online from the website:
http://www.lions.odu.edu/~imako001/cholesky/demo/index.html. The software tool developed from this work can be used for both students and their instructors not only to master this technical subject, but also to provide a dynamic/valuable tool for obtaining the solutions for homework assignments, class examinations, self-assessment studies, and other coursework related activities. Various transportation engineering applications of SLE are cited. Engineering educators who have adopted "flipped class-room instruction" can also utilize this Java Visualization and Animation software for students to "self-learning" these algorithms at their own time (and at their preferable locations), and use valuable class-meeting time for more challenging (real-life) problems' discussions. Statistical data for comparisons of students' performance with and without using the developed Java computer animation are also included.

Keywords

Cholesky Algorithm, Shortest Path, Linear Programming, Traffic Flows, Decomposition/Factorization, Java Visualization/Animation, Statistical Data

1. Introduction

The Cholesky Decomposition (Factorization) Algorithm was presented by Andre Louis Cholesky in an unknown and unpublished 8-page manuscript on December 2, 1910 entitled: On the numerical solutions of systems of linear equations. The method was unknown outside the French circle of topographers until another French officer published a paper explaining the method in 1924. It wasn't until 1950 when the Cholesky Decomposition Method was widely known after John Todd's lectures, several colleagues and students of him undertook the study of the Cholesky method. Today, the Cholesky Decomposition Method is widely known [1] [2] and is used to solve systems of Symmetric Positive Definite (SPD) simultaneous linear equations (SLE).

Engineering and science applications of SLE have been extensively documented in the literatures [3]-[6]. Cholesky method for solving SLE has been installed in all major commercialized Finite Element Analysis (FEA) software for the analysis and optimal design of Civil/Structural, Mechanical, Aerospace, Electrical engineering systems. Cholesky algorithm has also been considered as a major building block in the revised (matrix) SIMPLEX algorithm for solving Linear Programming (LP) problems [3]. The well-known shortest path (SP) problems [7] in transportation engineering communities can also be formulated/solved as a LP problem, although more efficient algorithms such as the Dijkstra, Label Correction, A*, Shortest Distance Decomposition Algorithms (SDDA) [8] have often be employed to solve such SP problems.

Least square regression models, and SLE has been discussed in Himes-Donnell's published work in "Speed Prediction Models for Multilane Highways: Simultaneous Equations Approach" [9]. Adu and his colleagues have considered "Applications of SLE to Traffic Flow for a Network of Four One-Way Streets in Kumasi, Ghana" [10]. SLE has also been applied in Li's research work in [11]. Intelligent transportation system (ITS) has often collected real-time (big) data from various sources. Thus, data compression, data recovery techniques (where SLE plays a central role) are critically important for ITS applications [12].

The best way to understand the Cholesky method is to see it working in practice by performing and solving small examples by hand and working through the algebra [13]. Various teaching philosophies have been proposed, tested and documented by the educational research communities, such as video lectures (YouTube), "flipped" class lectures (where students are encouraged to read the lecture materials at their own time at homes, and problem solving and/or questions/answers sessions are conducted in the usual classroom environments), US Science Technology Engineering Mathematics (STEM) summer camps, game-based-learning (GBL) [14]-[16], virtual laboratories [17] and concept inventory [18].

With the steady decline in US STEM interests and enrollments, the National Science Foundation (NSF) and the White House have developed national strategies and provided the significant budget to STEM education research [15] [16] in the past years, with the ultimate goals to improve both the quality and number of highly trained US educators, student workforce in STEM topics, in today highly competitive global mar-

kets. With the explosions of internet's capability and availability, it is even more critical to effectively train this future USA-STEM work-force and/or to develop effective STEM related teaching tools to reach a maximum possible number of "distance learners/audiences".

The final product from this work provides experimental results that show that this developed software/tool helps both the students and their instructor to not only master this technical subject, but also provide a valuable tool for obtaining the solutions for homework assignments, class examinations, self-assessment tools, etc. The developed "educational version" of a Java Visualization and Animation software tool provides several desirable features, such as:

- A detailed, precise and clear step-by-step algorithm will be displayed in text and human voice during the animation of the algorithm.
- Options to hear animated voice in 2 major languages (English and Spanish).
- Options to input/output terminal container yard layouts (CVS file), or manually edit the layouts using an editor, or "randomly generating" layouts.
- Output of the "final" results can be exported to text, so that the users/learners can check/verify their "hand-calculated" results, which is an important part of the learning process.

In this paper, a simple Cholesky decomposition example is thoroughly explained along with the derived formulas. The focuses are then shifted toward a Java Visualization and Animation software tool that provides step-by-step instructions for learning/teaching the Cholesky decomposition method. This Java Visualization and Animation software tool was used in an Old Dominion University (ODU) 300-Level Civil and Environmental Engineering (CEE) Computation course (CEE 305) in the Fall' 2014 semester. Lecture materials on how to solve system of SPD SLEs along with the Java-based software was handed out, and briefly discussed to the students as a self-study take-home assignment. Students were given an in-class exam on the Cholesky method. The goal was to determine if the Java-based (self-learning) software would improve or worsen the student's exam performance compared to the traditional face-to-face instructor's classroom lecture. The experiment results from this case-study are captured and explained in details in Section 4.

2. Summary of Cholesky's Decomposition Method

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 [5] [6]. In matrix notation, the SLE can be represented as:

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

where

$[A]$ = known coefficient matrix, with dimension $N \times N$

$\{b\}$ = known right-hand-side (RHS) $N \times 1$ vector

$\{x\}$ = unknown $N \times 1$ vector.

Symmetrical Positive Definite (SPD) SLE

For many practical SLE, the coefficient matrix $[A]$ (see Equation (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

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

where $[U]$ is a $N \times N$ 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 Equation (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 Equation (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 $N \times N$ 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 factorized) data in columns # $i(=5)$, and # $j(=7)$ of [U], respectively.

Step 2: Forward Solution phase

Substituting Equation (2) into Equation (1), one gets:

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

Let's define:

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

Then, Equation (9) becomes:

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

Since $[U]^{\mathrm{T}}$ is a lower triangular matrix, Equation (11) can be efficiently solved for

the intermediate unknown vector $\{y\}$, according to the order $\begin{Bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{Bmatrix}$, hence the name

"forward solution".

As a quick example, one has:

$$\begin{bmatrix} u_{11} & 0 & 0 \\ u_{12} & u_{22} & 0 \\ u_{13} & u_{23} & u_{33} \end{bmatrix} \begin{Bmatrix} y_1 \\ y_2 \\ y_3 \end{Bmatrix} = \begin{Bmatrix} b_1 \\ b_2 \\ b_3 \end{Bmatrix} \tag{12}$$

$$u_{11} y_1 = b_1 \rightarrow y_1 = \frac{b_1}{u_{11}} \tag{13}$$

$$u_{12} y_1 + u_{22} y_2 = b_2 \rightarrow y_2 = b_2 - u_{12} y_1 / u_{22} \tag{14}$$

Similarly

$$y_3 = \frac{b_3 - u_{13} y_1 - u_{23} y_2}{u_{33}} \tag{15}$$

In general, one has

$$y_j = \frac{b_j - \sum_{i=1}^{j-1} u_{ij} y_i}{u_{jj}} \tag{16}$$

Step 3: Backward Solution phase

Since $[U]$ is an upper triangular matrix, Equation (10) can be efficiently solved for

the original unknown vector $\{x\}$, according to the order $\begin{Bmatrix} x_N \\ x_{N-1} \\ x_{N-2} \\ \vdots \\ x_1 \end{Bmatrix}$, hence the name

"backward solution".

As a quick example, one has:

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{Bmatrix} = \begin{Bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{Bmatrix} \tag{17}$$

$$u_{44} x_4 = y_4, \text{ hence } x_4 = \frac{y_4}{u_{44}} \tag{18}$$

$$u_{33} x_3 + u_{34} x_4 = y_3, \text{ hence } x_3 = \frac{y_3 - u_{34} x_4}{u_{33}} \tag{19}$$

Similarly:

$$x_2 = \frac{y_2 - u_{23}x_3 - u_{24}x_4}{u_{22}} \tag{20}$$

$$x_1 = \frac{y_1 - u_{12}x_2 - u_{13}x_3 - u_{14}x_4}{u_{11}} \tag{21}$$

In general, one has:

$$x_j = \frac{y_j - \sum_{i=j+1}^{N} u_{ji}x_i}{u_{jj}} \tag{22}$$

Note: Amongst the above 3-step Cholesky algorithms, factorization phase in step 1 consumes about 95% of the total SLE solution time.

3. Java Computer Animated Software Tool for Teaching the Cholesky Algorithm

The developed software is written in Java [19]-[21] and is meant to create 2D animations and provide voice step-by-step explanations for the Cholesky Algorithm. The software is developed from scratch using several open-source software. The software uses various third-party libraries, such as Java Swing library for the animations, the text-to-speech library (Google API Translate) for voice, and the matrix library (Efficient Java Matrix) for data storage.

Applying open-source libraries to this software allowed lots of Graphical User Interface (GUI) functionality and features to be developed. The software GUI is developed with JFC/Swing which is included within the Java Development Kit (JDK). The Main GUI consists of seven main components as shown in **Figure 1**.
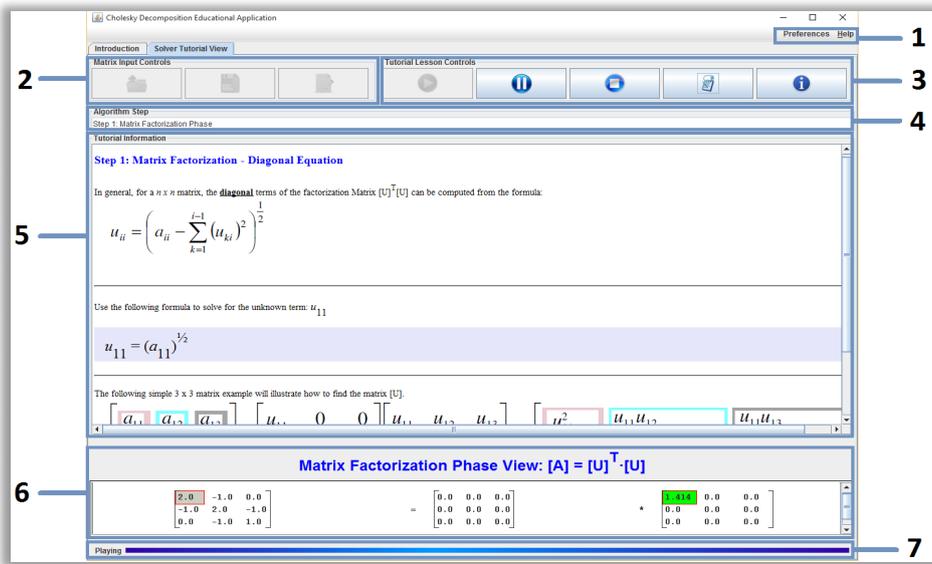

Figure 1. Cholesky decomposition educational application.

1) **Menu Buttons (Preferences and Help):** Provides basic options, such as an option to open/save Container Layouts, an option to provide terse, verbose, or no voice step-by-step instructions, an option to change the voice language to English, Spanish, and Chinese, and an option to display a user manual.

2) **Matrix Input Control Buttons:** Input control buttons are provided from the main window to open, save or edit the left-hand-side (LHS) Matrix [A] and right-hand-side (RHS) Vector {b} data.

3) **Tutorial Lesson Control Buttons:** Lesson control buttons are provided from the main window to play, pause, or stop a lesson (an algorithm). The buttons are enabled when the LHS Matrix [A] and RHS Vector {b} is given. Additional buttons (the step-by-step and information button) are provided. The step-by-step button launches a GUI that provides step-by-step instructions taking place during the play session. These step-by-step instructions can be saved to an output text file. The information buttons provide a user guide for the Main GUI.

4) **Algorithm Step:** Textbox view of the current algorithm step during play session.

5) **Tutorial Information:** Textbox view of instructions, equations, or steps (scrollable) is being done within the algorithm. The equations are dynamically provided based on the input data given.

6) **Animation View:** Animation view of the matrices and vectors being updated during the play session.

7) **Status View:** Provides a status view which displays meaningful status: ready, play, or paused.

4. Experimental Results

Civil and Environmental Engineering (CEE) Computation course (CEE 305) has been offered in every Fall semester. In the Fall-2013 semester, the Cholesky method was explained in the traditional (face-to-face) class-room instruction format. However, in the Fall-2014 semester, lecture materials on how to solve system of SPD SLEs along with the Java-based software was handed out to the students (with minimum instructor's lecture time) as a "self-study" take-home assignment. In both semesters (Fall' 2013, and Fall' 2014) students were given an in-class exam on the Cholesky method.

The goal was to determine if the Java-based software would improve or worsen the students' exam performance as compared to the traditional (face-to-face) instructor's classroom lecture. The resulting data was collected by the same professor that taught the same course using the normal (face-to-face) class-room instruction and using this new Java software tool. The results are explained below. Figure 2 shows the Student Percentage (Exam) Scores for the Fall 2013 semester using the conventional (face-to-face) class-room lectures. Figure 3 shows the Student Percentage (Exam) Scores for the Fall 2014 semester using the Java Visualization and Animation software tool for teaching the Cholesky Algorithm. Figure 3 clearly shows that a majority of students scored within the 76.67% or above; whereas, in Figure 2, student scores were scattered with more students on/below the 53.33% mark than in Fall 2014.
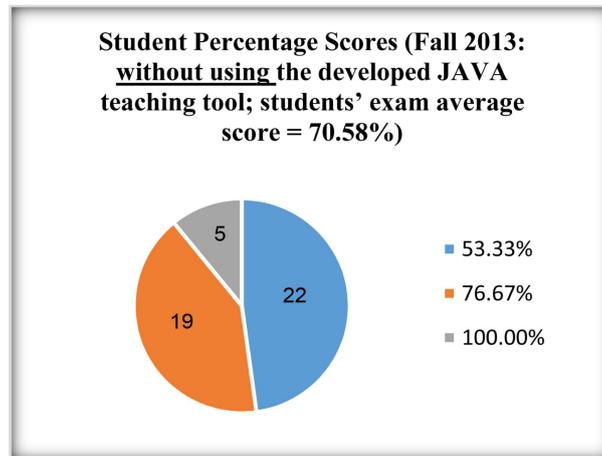
Student Percentage Scores (Fall 2013: <u>without using</u> the developed JAVA teaching tool; students' exam average score = 70.58%)

Figure 2. Student percentage scores (Fall 2013).

Student Percentage Scores (Fall 2014: <u>using</u> the developed JAVA teaching tool; students' exam average score = 81.53%)
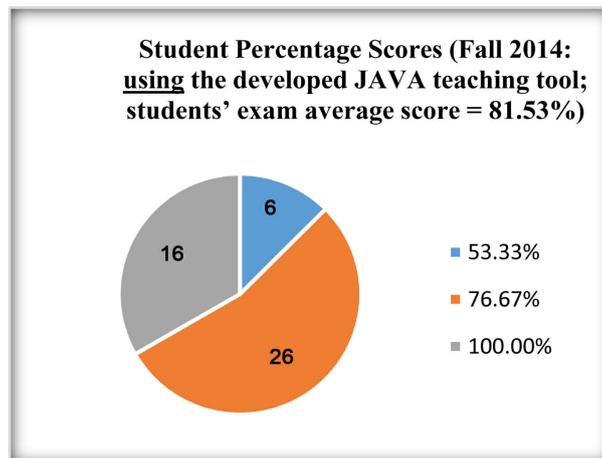
Figure 3. Student percentage scores (Fall 2014).

Using the Java Visualization and Animation software tool for instructions and animation in learning the Cholesky Method for solving SLE, and with the user's input problems' data capability for practicing exercises, the student's average performance/exam scores have improved from 70.58% to 81.53% (as shown in **Figure 2** and **Figure 3**, respectively).

5. Conclusions

In this paper, the Cholesky Method is once again revisited to show a new and different approach for teaching/learning the Cholesky decomposition algorithm, for solving the Simultaneous Linear Equations (SLE). Various transportation engineering applications of SLE have also been highlighted. The developed Java computer animated software has shown that it can be used as an additional/effective tool for teaching and learning the Cholesky method. The students' average scores from in-class exams in the Fall'2013 (traditional face-to-face class lecture instruction mode, without using the developed JAVA animated teaching tool), and in the Fall'2014 (students' self-learning mode [with

minimum instructor's lecture time], using the developed JAVA animated teaching tool) were 70.58% and 81.53%, respectively. Thus, an improvement of more than 10% has been observed! The software tool is developed in Java language and integrated with open-source libraries. It provides a means to be platform independent and can be run on several operation systems. It provides desirable teaching/learning features for the Cholesky algorithm, such as:

- Software tool with a user friendly GUI.
- 2D Graphical visualization and animation for displaying data update.
- 2D Graphical visualization to edit input data.
- Ability to allow the user/learner to open/save the data for later use.
- Ability to allow the user/learner to output/save the step-by-step results.
- Provide a clearly and attractive computer animated voice that provides step-by-step instructions of the algorithms.
- Animated voice can be configurable to translate text-to-speech into another language, such as English, Spanish and Chinese.
- Provide the final and intermediate results of the Cholesky solution process.

A demo video of the Cholesky Decomposition Method's animation and result can be viewed online from any web browser using the website provided in Reference [1].

Acknowledgements

References

[1] Cluster (Parallel) Computing for Large-Scale Engineering & Science Applications. http://www.lions.odu.edu/~skadi002/001-075.pdf http://www.lions.odu.edu/~imako001/

[2] Brezinski, C. (2007) The Life and Work of Andre Cholesky.

[3] Nguyen, D.T., Shen, Y., Mohammed, A.A. and Kadiam, S. (2010) Tossing Coin Game and Linear Programming Big M Simplex Algorithms. Global Conference on Learning and Technology, Penang (Island), 17-20 May 2010. http://www.learntechlib.org/noaccess/34365

[4] Nguyen, D.T., Kaw, A.K., Pendyala, R. and Lee-Thomas, G. (2011) Collaborative Research: Development of New Prototype Tools, and Adaptation and Implementation of Current Resources for a Course in Numerical Methods. NSF Funded Educational Grant (Proposal # 0836916; DUE-CCLI Phase 1: Exploratory); (Funding Period: 1 January 2009-30 July 2011).

[5] Nguyen, D.T. (2006) Finite Element Methods: Parallel-Sparse Statics and Eigen-Solutions. Springer Publishers. http://www.springer.com/us/book/9780387293301

[6] Nguyen, D.T. (2002) Parallel-Vector Equation Solvers for Finite Element Engineering Applications. Kluwer Academic/Plenum Publishers. http://www.oracle.com/technetwork/java/javase/downloads/index.html

[7] Makohon, I., Nguyen, D.T., Sosonkina, M., Shen, Y. and Ng, M. (2016) JAVA Based Visualization and Animation for Teaching the Dijkstra Shortest Path Algorithm in Transportation Networks. *International Journal of Software Engineering & Applications* (*IJSEA*), **7**, 11-25.

[8] Johnson, P., Nguyen, D.T. and Ng, M. (2014) An Efficient Shortest Distance Decomposition Algorithm for Large-Scale Transportation Network Problems. *TRB* 2014 *Annual Meeting*, Washington DC, January 2014. https://trid.trb.org/view.aspx?id=1288547

[9] Himes, S.C. and Donnell, E.T. (2010) Speed Prediction Models for Multilane Highways: Simultaneous Equations Approach. *Journal of Transportation Engineering*, **136**, 855-862. http://dx.doi.org/10.1061/(ASCE)TE.1943-5436.0000149

[10] Adu, I.K., *et al.* (2014) Applications of SLE to Traffic Flow for a Network of Four One-Way Streets in Kumasi, Ghana. *International Journal of Contemporary Mathematical Sciences*, **9**, 653-660. http://dx.doi.org/10.12988/ijcms.2014.410109

[11] Li, Q., *et al.* (2007) A Flow Volume Data Compression Approach for Traffic Network Based on Principal Component Analysis. *Intelligent Transportation Systems Conference*, 125-130. http://ieeexplore.ieee.org/document/4357668/?reload=true&arnumber=4357668

[12] Asif, M.T., *et al.* (2013) Data Compression Techniques for Urban Traffic Data. *IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems* (*CIVTS*), 16-19 April 2013, 44-49. http://dx.doi.org/10.1109/civts.2013.6612288

[13] Claxton, K. and Briggs, A. (2006) Decision Modelling for Health Economic Evaluation.

[14] Nguyen, D.T., Mohammed, A.A., Kadiam, S. and Shen, Y. (2010) Internet Chess-Like Game and Simultaneous Linear Equations. *Global Conference on Learning and Technology*, Penang (Island), 17-20 May 2010. http://www.learntechlib.org/noaccess/34365

[15] National Science Foundation (NSF), National Science Board (NSB) (2007) A National Action Plan for Addressing the Critical Needs of the US Science, Technology, Engineering, and Mathematics (STEM) Education System.

[16] Executive Office of the President of the United States (2013) Federal Science, Technology, Engineering, and Mathematics (STEM) Education 5-Year Strategic Plan. A Report from the Committee on STEM Education, National Science and Technology Council (May 2013).

[17] Nguyen, D.T. (2009) Implementation Grant: Simulation and Visualization Enhanced Engineering Education. National Science Foundation (NSF), Funding Period: September 2005-September 2009.

[18] Kaw, A., *et al.* (2016) Improving and Assessing Student Learning in an Inverted STEM Classroom Setting. NSF (Division of Undergraduate Education) Awarded Grant # 1322586 (September 2013-September 2016).

[19] Efficient Java Matrix Library. http://code.google.com/p/efficient-java-matrix-library/wiki/EjmlManual

[20] Google Translate Java. http://code.google.com/p/google-api-translate-java/

[21] Java Standard Edition. http://www.oracle.com/technetwork/java/javase/downloads/index.html

Scientific Research Publishing

Submit or recommend next manuscript to SCIRP and we will provide best service for you:

Accepting pre-submission inquiries through Email, Facebook, LinkedIn, Twitter, etc.

A wide selection of journals (inclusive of 9 subjects, more than 200 journals)

Providing 24-hour high-quality service

User-friendly online submission system

Fair and swift peer-review system

Efficient typesetting and proofreading procedure

Display of the result of downloads and visits, as well as the number of cited articles

Maximum dissemination of your research work

Submit your manuscript at: http://papersubmission.scirp.org/

Or contact jsea@scirp.org