A Chess-Like Game For Teaching Engineering Students To Solve Large System Of Simultaneous Linear Equations.

Duc T. Nguyen[1]; Ahmed Ali Mohammed[2]; and Subhash Kadiam[3]
[1]Professor, ODU, [2]Structural Engineer,[3]Ph.D Student, ODU.
[1]dnguyen@odu.edu, [2]ahmedali484@yahoo.com, [3]skadi002@odu.edu

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 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, a Game-Based Learning (GBL) approach has been developed to help engineering students to understand crucial details about matrix reordering and factorization phases. A "chess-like" game has been developed and can be played by either a single player, or two players. Through this "chess-like" 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 "chess-like" game 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. Preliminary demonstrations of the developed GBL approach can be viewed/played by any players who have access to the internet web-site [4]!

1. INTRODUCTION

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 matrix, with dimension NxN
{b} = known right-hand-side (RHS) Nx1 vector
{x} = unknown Nx1 vector.

2. 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:

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

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

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".

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, 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".

As a quick example, one has:

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

$u_{44} x_4 = y_4$, hence $x_4 = \dfrac{y_4}{u_{44}}$ (18)

$u_{33} x_3 + u_{34} x_4 = y_3$, hence $x_3 = \dfrac{y_3 - u_{34} x_4}{u_{33}}$ (19)

Similarly: $x_2 = \dfrac{y_2 - u_{23} x_3 - u_{24} x_4}{u_{22}}$ (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:

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

Remarks

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

(b) If the coefficient matrix [A] is symmetrical but not necessary positive definite, then the above Cholesky algorithms will not be valid. In this case, the following LDL^T algorithms can be employed:

$$[A]=[L][D][L]^T \tag{23}$$

For example,

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ L_{21} & 1 & 0 \\ L_{31} & L_{32} & 1 \end{bmatrix} \begin{bmatrix} D_{11} & 0 & 0 \\ 0 & D_{22} & 0 \\ 0 & 0 & D_{33} \end{bmatrix} \begin{bmatrix} 1 & L_{21} & L_{31} \\ 0 & 1 & L_{32} \\ 0 & 0 & 1 \end{bmatrix} \tag{24}$$

Multiplying the 3 matrices on the RHS of Eq. (24), then equating the resulting upper-triangular RHS terms of Eq. (24) to the corresponding ones on the LHS, one obtains the following formulas for the "diagonal" [D], and "lower-triangular" [L] matrices:

$$D_{ii} = A_{ii} - \sum_{k=1}^{i-1} L_{ik}^2 D_{kk} \tag{25}$$

$$L_{ij} = \left(A_{ij} - \sum_{k=1}^{j-1} L_{ik} D_{kk} L_{jk} \right) * \left(\frac{1}{D_{jj}} \right) \tag{26}$$

Thus, the LDL^T algorithms can be summarized by the following step-by-step procedures

Step1: Factorization phase

$$[A]=[L][D][L]^T \tag{23, repeated}$$

Step 2: Forward solution and diagonal scaling phase

Substituting Eq. (23) into Eq.(1), one gets:

$$[L][D][L]^T\{x\}=\{b\} \tag{27}$$

Let's define:

$$[L]^T\{x\}=\{y\} \tag{28}$$

$$[D]\{y\}=\{z\} \tag{29}$$

Then Eq. (27) becomes:

$$[L]\{z\}=\{b\} \tag{30}$$

Eq. (30) can be efficiently solved for the vector $\{z\}$, then Eq. (29) can be conveniently (and trivially) solved for the vector $\{y\}$.

Step 3: Backward solution phase

In this step, Eq. (28) can be efficiently solved for the original unknown vector $\{x\}$.

3. RE-ORDERING ALGORITHMS FOR MINIMIZING FILL-IN TERMS [1,2].

During the factorization phase (of Cholesky, or LDL^T 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:

$$[A]=\begin{bmatrix} 112 & 7 & 0 & 0 & 0 & 2 \\ 7 & 110 & 5 & 4 & 3 & 0 \\ 0 & 5 & 88 & 0 & 0 & 1 \\ 0 & 4 & 0 & 66 & 0 & 0 \\ 0 & 3 & 0 & 0 & 44 & 0 \\ 2 & 0 & 1 & 0 & 0 & 11 \end{bmatrix} \tag{31}$$

$$\{b\}=\begin{Bmatrix} 121 \\ 129 \\ 94 \\ 70 \\ 47 \\ 14 \end{Bmatrix} \tag{32}$$

The Cholesky factorization matrix [U], based on the original matrix [A] (see Eq. 31) and Eqs. (6-7), can be symbolically computed as:

$$[U]=\begin{bmatrix} \times & \times & 0 & 0 & 0 & \times \\ 0 & \times & \times & \times & \times & F \\ 0 & 0 & \times & F & F & \times \\ 0 & 0 & 0 & \times & F & F \\ 0 & 0 & 0 & 0 & \times & F \\ 0 & 0 & 0 & 0 & 0 & \times \end{bmatrix} \tag{33}$$

In Eq. (33), the symbols "x", and "F" represents the "non-zero" and "Fill-in" terms, respectively.

In practical applications, however, it is always a necessary step to send the original matrix [A] through re-ordering algorithms (or subroutines) [Refs 1-2] and produce the following integer mapping array

IPERM (new equation #) = {old equation #} (34)

such as, for this example:

$$IPERM\begin{Bmatrix}1\\2\\3\\4\\5\\6\end{Bmatrix}=\begin{Bmatrix}6\\5\\4\\3\\2\\1\end{Bmatrix} \qquad (35)$$

Using the above results (see Eq. 35), one will be able to construct the following re-arranged matrices:

$$[A^*]=\begin{bmatrix}11&0&0&1&0&2\\7&44&0&0&3&0\\0&0&66&0&4&0\\1&0&0&88&5&0\\0&3&4&5&110&7\\2&0&0&0&7&112\end{bmatrix} \qquad (36)$$

and

$$\{b^*\}=\begin{Bmatrix}14\\47\\70\\94\\129\\121\end{Bmatrix} \qquad (37)$$

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

$$[A^*]\{x^*\}=\{b^*\} \qquad (38)$$

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. (35).

The factorized matrix $[U^*]$ can be "symbolically" computed from $[A^*]$ as:

$$[U^*]=\begin{bmatrix}\times&0&0&\times&0&\times\\0&\times&0&0&\times&0\\0&0&\times&0&\times&0\\0&0&0&\times&\times&F\\0&0&0&0&\times&\times\\0&0&0&0&0&\times\end{bmatrix} \qquad (39)$$

It is clearly to recognize, at this moment, the big benefits of solving the SLE shown in Eq. (38), instead of solving the original Eq. (1), since the factorized matrix $[U^*]$ has only 1 fill-in term (see the symbol "F" in Eq. 39), as compared to 6 fill-in-terms occurred in the factorized matrix [U] (shown in Eq. 33)!

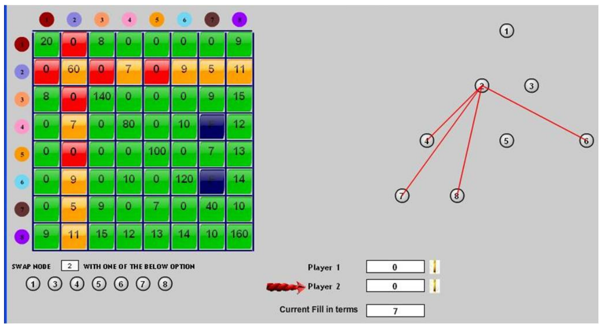
4. ON-LINE CHESS-LIKE GAME FOR REORDERING/FACTORIZED PHASE [4].

Based on the discussions presented in the previous section 2 (about factorization phase), and section 3 (about reordering phase), one can easily see the similar operations between the symbolic, numerical factorization and reordering phases of sparse SLE.

In practical computer implementation for the solution of SLE, the reordering phase is usually conducted first (to produce the mapping between "old↔new" equation numbers, as indicated in the integer array IPERM(-) in Eqs. 34-35).

Then, sparse "symbolic" factorization phase is followed by using either Cholesky Eqs. 6-7, or the LDL^T Eqs. 25-26 (without requiring the actual/numerical values to be computed). The reason is because during the "symbolic factorization" phase, one only wishes to find the number (and the location) of non-zero "fill-in terms". This "symbolic" factorization process is necessary for allocating the "computer memory" requirement for the "numerical factorization" phase which will actually compute the exact numerical values of $[U^*]$, based on the same Cholesky Eqs. (6-7) (or the LDL^T Eqs. (25-26)).

In this work, a chess-like game (shown in Figure 1 [4]) has been designed with the following objectives:



	1	2	3	4	5	6	7	8
1	20	0	8	0	0	0	0	9
2	0	60	0	7	0	9	5	11
3	8	0	140	0	0	0	9	15
4	0	7	0	80	0	10	F	12
5	0	0	0	0	100	0	7	13
6	0	9	0	10	0	120	F	14
7	0	5	9	0	7	0	40	10
8	9	11	15	12	13	14	10	160

SWAP NODE 2 WITH ONE OF THE BELOW OPTION

① ③ ④ ⑤ ⑥ ⑦ ⑧

Player 1: 0

Player 2: 0

Current Fill in terms: 7

Figure 1: A Chess-Like Game For Learning to Solve SLE.

(A) Teaching undergraduate/HS students the process how to use the reordering output IPERM(-), see Eqs. (34-35) for converting the original/given matrix [A], see Eq. (31), into the new/modified matrix $[A^*]$, see Eq. (36). 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/HS 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/HS students to understand the "numerical factorization" phase, by numerically utilizing the same Cholesky factorized Eqs. (6-7).

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

5. FURTHER EXPLANATION ON THE DEVELOPED GAME

1. In the above Chess-Like Game, which is available on-line [4], powerful features of FLASH computer environments [3], such as animated sound, human voice, motions, graphical colors etc… have all been incorporated and programmed into the developed game-software for more appealing to game players/learners.

2. In the developed "Chess-Like Game", fictitious monetary (or any kind of 'scoring system") is rewarded (and broadcasted by computer animated human voice) to game players based on how he/she swaps the node (or equation) numbers, and consequently based on how many fill-in "F" terms occurred.

3. Based on the original/given matrix [A], and existing re-ordering algorithms (such as the Reverse Cuthill-Mckee, or RCM algorithms [1-2]) the number of fill-in ("F") terms can be computed (using RCM algorithms). These internally generated information will be used to judge how good the players/learners are, and/or broadcasting "congratulation message" to a particular player who discovers a new (swapping node) strategies which are even better than RCM algorithms!

4. Initially, the player(s) will select the matrix size (8x8, or larger is recommended), and the percentage (50%, or larger is suggested) of zero-terms (or sparsity of the matrix). Then, "START Game" icon will be clicked by the player.

5. The player then CLICK one of the selected node "i" (or equation) number appeared on the computer screen. The player will see those nodes "j" which are connected to node "i" (based on the given/generated matrix [A]). The player then has to decide to swap node "i" with which of the possible node "j". After confirming the player's decision, the outcomes/results will be announced by the computer animated human voice, and the money-award will (or will NOT) be given to the players/learners, accordingly. In this software, a maximum of $1,000,000 can be earned by the player, and the "exact dollar amount" will be INVERSELY proportional to the number of fill-in terms occurred (as a consequence of the player's decision on how to swap node "i" with another node "j").

6. The next player will continue to play, with his/her move (meaning to swap the ith node with the jth node) based on the current best non-zero terms pattern of the matrix.

6. ACKNOWLEDGMENTS

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

7. REFERENCES

[1] Duc T. Nguyen, "Finite Element Methods: Parallel-Sparse Statics and Eigen-Solutions", Springer Publisher (2006).
[2] Duc T. Nguyen, "Parallel-vector Equation Solvers for Finite Element Engineering Applications", Kluwer Academic/Plenum Publishers (2002).
[3] www.brothersoft.com/downloads/flash-animation-software.html.
[4] http://www.lions.odu.edu/~amoha006/Fillinterms/FILLINTERMS.html