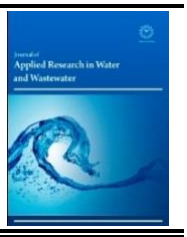
Razi University

Journal of
Applied Research in Water & Wastewater

Journal homepage: www.arww.razi.ac.ir



Original paper

Implementation of the skyline algorithm in finite-element computations of Saint-Venant equations

Reza Karimi[1], Ali Akbar Akhtari[1,*], Omid Seyedashraf[2]

[1]*Department of Civil Engineering, Razi University, Kermanshah, Iran.*
[2]*Department of Civil Engineering, Kermanshah University of Technology, Kermanshah, Iran.*

ARTICLE INFO

Article history:
Received 11February 2014
Received in revised form 3 May 2014
Accepted 23 May 2014
Available online 28 June 2014

Keywords:
Skyline solver
Sparse matrices
Dam-break flow
Saint-Venant equations

ABSTRACT

Solving a large sparse set of linear equations is of the problems widely seen in every numerical investigation in the entire range of engineering disciplines. Employing a finite element approach in solving partial derivative equations, the resulting stiffness matrices would contain many zero-valued elements. Moreover, storing all these sparse matrices in a computer memory would slower the computation process. The objective of this study is to attain insight into Skyline solver in order to store the non-zero valued entries of large linear systems and enhance the calculations. Initially, the Skyline solver is introduced for symmetric or non-symmetric matrices. Accordingly, an implementation of the proposed solver is conducted using various grid form sets and therefore, several stiffness matrices with different sizes to evaluate the solver's capability in solving equation systems with a variety of dimensions. Comparing the obtained numerical results it was concluded that Skyline algorithm could solve the equation systems tens of times faster than a regular solver; especially in conducting iterative mathematical computations like Saint-Venant Equations.

Nomenclature

B	nodal degree of freedom
DIMMA	number of elements in the global matrix
DM	dimension of the stiffness matrix
DMSL	total number of elements in a 1D matrix
EL	number of elements
K	stiffness matrix
NO	number of nodes in an element
TOTVAL	number of non-zero cells in a matrix
TOTVALS	number of non-zero cells in a symmetric matrix
X	unknown vector
ZEVA	the number of zero valued cells in the global matrix

1. Introduction

Employing the numerical methods like finite difference and finite-element methods for numerical analysis of a hydraulic phenomenon; particularly in its three-dimensional form, can result in solving large systems of linear equations like Eq.(1). Regularly, these equation systems are symmetric and include many zero-valued entries in their equivalent matrix system form. Gauss-Jordan elimination, Atomic Triangular matrices, Cholesky-method and Strip-method are the most distinguished mathematical approaches to solve these systems of linear equations. For example, in the Gauss-Jordan elimination method, the determinants and inverse matrices must be calculated (Poole 2002). Moreover, the required numerical stability can be reached by conducting a partial diagonalization of the respective matrix (Golub and Van Loan 1996).

Corresponding author E-mail: Akhtari@razi.ac.ir

$$[K][X]=[B] \quad (1)$$

where, [K] is the coefficients matrix (stiffness matrix in finite-element approach),[X] is the unknown vector, and [B] is the nodal degree of freedom.

The constraints on computer storage requirements and CPU prevent using common solvers for intricate problems, like fluid flow with thousands of equations to be solved. There are two types of solvers, iterative and direct solvers. Below, we will present a concise overview of these solvers.

Iterative methods are found to be capable of solving large linear equation systems since they require less storage room and CPU time. They are inaccurate solvers, but their converged solutions can be close enough to the exact solution. The main process in an iterative method is the matrix-vector multiplication as compared to the matrix reduction in direct methods. A noteworthy benefit of iterative methods is that a given set of equations can be spit up into several subsets of equations and calculations can be performed in parallel on an array of processors. However, convergence characteristics of iterative methods depend on the condition number of the system of linear equations, and to find a suitable preconditioner is an essential to reach convergence. Normally, in Taylor-Galerkin computations of dam-break problems, less than six iterations would be enough to achieve convergence (Quecedo and Pastor 2002).

Nevertheless, solving sparse equation systems through classic methods would lead to larger computer storage requirements and therefore higher CPU costs. Nevertheless, there are various methods for re-ordering these sparse matrices. For example, Chin Shen et al. (2002) have proposed a set of parallel preconditioning approaches built up upon a multilevel block incomplete factorization method, which implements an iterative solver to complete the linear system solving process. According to their research, this technique is suitable for

solving large sparse linear systems on distributed memory parallel computers that consist of a collection of independent processors. They have also proposed two new algorithms for constructing preconditioners based on the theory of block autonomous sets. Moreover, in order to solve the large linear equations of the discretized form of shallow water equations, Fang and Sheu (2001) have made use of the bi-conjugate gradient stabilized (Bi-CGSTAB) and the generalized minimum residual (GMRES) methods, which are two Krylov subspace iterative solvers. They have evaluated these methods through comparing the obtained numerical results to the ones obtained from multi-frontal solver, which is a well-known direct solver regularly used in MATLAB software. Based on their investigations, the GMRES performed much faster than the direct solver.

The previously mentioned re-ordering methods can be employed in direct solvers too. Direct methods are those in which simultaneous linear arithmetical equations are solved by successive elimination of variables and back replacement. The Gauss elimination method is a fixed-step procedure. Furthermore, the frontal and skyline solution methods are modifications of Gauss elimination technique, which are frequently used direct solvers. These methods are common procedures in numerical simulations and especially in the finite-element analysis when we face a sparse mass matrix in the computations. The frontal solution method is faster than nearly all direct solvers as it calls for less core space as long as active variables can be kept in the core since the method completes the computational steps related to creating element matrices and assembling the global stiffness matrix in one single step (Irons 1970).

Other novel procedures include the approaches, which deal with only non-zero entries of a stiffness matrix through storing them in a new compacted form. Nour-Omid and Taylor (1984) have proposed an algorithm for assembly of K matrix. According to their studies, the data structure can be extended to be used in conjunction with any solution procedures by simply expanding the compacted form into a form appropriate for the respective solver. The scheme is similar to Skyline solver and results in considerable reduction in the storage needs during the assembly process. Alan Mathison has proposed another novel technique. In this method, the coefficient matrix must be decomposed into two lower and upper triangular matrices. Furthermore, the equation system can be solved by performing Forward Substitution and Back Substitution methods (Poole 2002). While, employing the Cholesky method, a unit valued element must be considered for the diagonal entries of the upper triangle of the matrix.

In this research, a one-dimensional dam-break test case is numerically examined to evaluate the efficiency of the Skyline solver through measuring its run time. The motivation of the investigation was to assess the necessary CPU time for different grid forms used to discretize the computational domain through a classic finite-element method and enhance the calculations. Accordingly, a shock wave induced by a dam failure will be numerically modeled and its results will be compared to the pre-existing numerical data.

2. Materials and methods

2.1. Band matrix solver

Among all conventional directs solvers; this method is more suitable to solve banded matrices. The matrix, A, of size n×n is called a banded matrix if all of its entries except the diagonal ones are zero-valued elements. The width and size of these diagonal stripes can be denoted by k_1 and k_2, which represent the width of left and right half of the diagonal strip, respectively. The global banded matrix can be constructed by storing the diagonal entries while having zeros for matrix cells except the diagonal ones. For example, the matrix, $A_{(4\times4)}$, with the width equal to three would be stored as a matrix like $A^{'}$ of the size 4×3 (Golub and Van Loan 1996). Here, the constant coefficients k_1 and k_2 are equal to one.

$$A=\begin{bmatrix} a_{11} & a_{12} & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ 0 & a_{32} & a_{33} & a_{34} \\ 0 & 0 & a_{43} & a_{44} \end{bmatrix} \tag{2}$$

$$A^{'}=\begin{bmatrix} 0 & a_{11} & a_{12} \\ a_{21} & a_{22} & a_{23} \\ a_{32} & a_{33} & a_{34} \\ a_{43} & a_{44} & 0 \end{bmatrix} \tag{3}$$

2.2. Symmetric Band-matrix solver

This solver is similar to the Band Matrix Method, but more useful for symmetrical matrices. The following strategy can be used for positive definite matrix systems, which are banded and symmetrical. (4) and (5) demonstrates the respective arrangement of the re-ordered matrix.

$$A=\begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 & 0 \\ & a_{22} & a_{23} & a_{24} & 0 & 0 \\ & & a_{33} & a_{34} & a_{35} & 0 \\ & & & a_{44} & a_{45} & a_{46} \\ & sym & & & a_{55} & a_{56} \\ & & & & & a_{66} \end{bmatrix} \tag{4}$$

$$A^{*}=\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{22} & a_{23} & a_{24} \\ a_{33} & a_{34} & a_{35} \\ a_{44} & a_{45} & a_{46} \\ a_{55} & a_{56} & 0 \\ a_{66} & 0 & 0 \end{bmatrix} \tag{5}$$

2.3. Skyline method

This method is useful for banded symmetric matrices, which are commonly seen in the finite-element analysis with an appropriate node numbering from. The scheme stores the non-zero cells of a matrix entry at the beginning and end of each column in the second-line array. However; in the case, the non-zero elements are located around the main diagonal, and the global stiffness matrix is created in a certain numerical order; an arithmetical logic can be proposed, in which the second-line array could be removed. Consequently, one can apply a series of simple mathematic rules when solving linear equations while keeping the self-determining property of the computational process. According to the following example, here the stiffness matrix can be stored in a single-dimensional array or in a single column vector. To do this, the diagonal entries of the matrix must be stored in an array.

$$A=\begin{bmatrix} a_{11} & 0 & a_{13} & 0 & 0 & 0 \\ & a_{22} & 0 & a_{24} & 0 & 0 \\ & & a_{33} & 0 & 0 & 0 \\ & & & a_{44} & a_{45} & 0 \\ & sym & & & a_{55} & a_{56} \\ & & & & & a_{66} \end{bmatrix} \tag{6}$$

$$A^{'}=\begin{bmatrix} a_{11} & a_{22} & a_{13} & 0 & a_{33} & a_{24} & \cdots \\ 0 & a_{44} & a_{45} & a_{55} & a_{56} & a_{66} \end{bmatrix} \tag{7}$$

$$A^{*}=\begin{bmatrix} 1 & 2 & 5 & 8 & 10 & 12 \end{bmatrix} \tag{8}$$

where, the elements of matrix A^{*} represent the element values of each column up to the diagonal position.

Since there could be too many elements in each stiffness matrix, the assembly and storing steps of these sparse matrices are of the most complex steps, which engineers and scientists must take in order to discretize the respective governing equations. As a result, in most cases the analyst can manipulate the structure of the stiffness matrix to a more proper form and have a smooth computational process. For example, the banded matrix storage and the Skyline technique can be employed in order to minimize the space required for non-zero valued elements. Both methods store the non-zero cells in a narrow band near the main diagonal of the matrix. Accordingly, this can significantly reduce the computer memory required to store the matrix entries. The Skyline storage method has an efficient functionality for this; however, its way of storing and solving is complicated. Using Skyline solver, two one-dimensional matrices must be employed while its optimum output is reachable

Please cite this article as: R. Karimi, A. A. Akhtari, O. Seyedashraf, Implementation of the skyline algorithm in finite-element computations of Saint-Venant equations, Journal of Applied Research in Water and Wastewater, 1(2), 2014, 63-67.

when these matrices are dense enough or in other word have fewer zero cells.

A Skyline solver can be used whenever the elements of the entire matrix are placed around the main diagonal based on a mathematical logic. The principal goal here is to store the non-zero elements and the right-hand side terms of Eq. (1). Moreover, the eliminations must be done on the previously mentioned one-dimensional array. Eq. (2) depicts the structure of a global stiffness matrix for three-node elements. According to this equation, the distance between any two consecutive cells of the main diagonal must be equal to the number of nodes in each element. Furthermore, for each column of the stiffness matrix, there is a simple mathematical relationship between the nodes and number of elements in each column, which can be used in the Gauss-Jordan solver in order to minimize the required CPU. This process would be more effective when the computational domain is partitioned into rather small elements, and the stiffness matrix has many entries.

Eq. (9) depicts the structure of the respective global stiffness matrix.

$$\begin{bmatrix} a & a & a & 0 & 0 & 0 & 0 & 0 & 0 \\ a & a & a & 0 & 0 & 0 & 0 & 0 & 0 \\ a & a & b & b & b & 0 & 0 & 0 & 0 \\ 0 & 0 & b & b & b & 0 & 0 & 0 & 0 \\ 0 & 0 & b & b & c & c & c & 0 & 0 \\ 0 & 0 & 0 & 0 & c & c & c & 0 & 0 \\ 0 & 0 & 0 & 0 & c & c & d & d & d \\ 0 & 0 & 0 & 0 & 0 & 0 & d & d & d \\ 0 & 0 & 0 & 0 & 0 & 0 & d & d & d \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix} = \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \\ Q_5 \\ Q_6 \\ Q_7 \\ Q_8 \\ Q_9 \end{bmatrix} \quad (9)$$

Here, the dimension of the matrix can be calculated through a simple mathematical calculation using NO and EL parameters as the number of nodes in each element and number of elements, respectively. The following equations show the relationships between the various components of the matrix system.

$$DM = \left(EL \times (NO-1)\right)+1 \quad (10)$$

$$DIMMA = DM^2 \quad (11)$$

$$TOTVAL = \left(EL * NODE^2\right) - (EL-1) \quad (12)$$

$$TOTVALS = \left(EL * NO * \frac{NO+1}{2}\right) - (EL-1) \quad (13)$$

$$DIMSL = TOTVALS + DM \quad (14)$$

$$ZEVA = DIMMA - TOTVAL \quad (15)$$

where DM is the dimension of the stiffness matrix; DIMMA is the total number of elements in the global quadratic matrix. TOTVAL and TOTVALS are the number of non-zero valued cells in a regular and symmetric matrix formation, respectively. DMSL is the total number of elements in a one-dimensional matrix, and ZEVA is the number of zero valued cells in the global matrix.

3. Results

Here an incompressible fluid is consider, which is assumed to have no viscosity (inviscid fluid) that is homogeneous with a constant and uniform density (ρ). Subjected to the characteristics of the flow and computational domain, some approximations are imposed to the Navier-Stokes system of equations to obtain the efficient equations capable of simulating Dam-break shock waves. Accordingly, the Saint-Venant Equations can be expressed as follows:

$$\frac{\partial A}{\partial t} + \frac{\partial Q}{\partial x} = 0 \quad (16)$$

$$\frac{\partial Q}{\partial t} + \frac{\partial}{\partial x}\left(\frac{Q^2}{A}\right) + gA\frac{\partial h}{\partial x} = gA(S_0 - S_f) \quad (17)$$

where A is the cross-sectional area; t denotes the time (s); h is the water elevation (m); g is the gravity acceleration, and x is the flow direction. Sf and S0 correspond to friction and bottom slopes, respectively.

In this study, a homogeneous case is considered. Consequently, the source terms are neglected (S=0). A standard finite-element approach was used in order to have an approximate solution to the incompressible and shock-dominated flow problem. The numerical method employs the weight functions and the new parameter, α, to solve the system in an implicit approach. The numerical method approximately solves the governing equations in an iterative way so that the results obtained in the step s would be close enough to the results obtained in the computational step s+1. The written computer code uses the following equations to solve the momentum and continuity equations in an implicit approach (Karimi 2012).

$$\left[M^1\right]\left[\frac{\partial U}{\partial t}\right] + [K][U] = [F] \quad (18)$$

$$\left[\hat{K}\right]_{s+1}[U]_{s+1} = \left[\bar{K}\right]_s[U]_s + \left[\hat{F}\right]_{s,s+1} \quad (19)$$

$$\left[\hat{K}\right]_{s+1} = \left[M^1\right] + \alpha\,\Delta t\,[K]_{s+1} \quad (20)$$

$$\left[\bar{K}\right]_s = \left[M^1\right] + (1-\alpha)\,\Delta t\,[K]_s \quad (21)$$

$$\left[\hat{F}\right]_{s,s+1} = \Delta t(\alpha[F]_{s+1} + (1-\alpha)[F]_s) \quad (22)$$

where [M1] is the coefficient matrix of time-dependent terms; [K] is the coefficient matrix of U, and [F] is the Right Hand Side (RHS) vector.

In these set of equations, the parameter,α ,is varied within the range [0,1]. Accordingly, one may define α=0, α=1/2, α=2/3 while they act as a Backward Difference, Crank–Nicolson, Galerkin or Forward Difference schemes, respectively (Reddy 2006).

1D Dam-break Problem

In order to assess the effectiveness of the proposed numerical scheme, results obtained from the previously mentioned mathematical approach is put side by side a finite difference model of a one-dimensional dam-break problem. Tseng and Chu (2000) have employed a predictor–corrector Total Variation Diminishing (TVD) scheme for the computation of unsteady one-dimensional dam-break flows. The presumption of an immediate and complete breach is chosen to simplify the simulation, while we are applying certain arithmetical methods for analyzing the Skyline's efficiency in solving large linear equation systems. Moreover, the experiment's presumptions illustrate a reinforced concrete arch dam failure (Seyedashraf 2012).

The test case exhibits the development of the flow in the computational domain from time $t = 0s$ to $t = 60s$. The computational domain is a straight, rectangular channel, and the dam is located in the middle of a horizontal channel whose Manning roughness coefficient is assumed zero. Originally, the water body is motionless with uneven depths on both sides of the hypothetical barrier, 10m in the upstream and 5m in the downstream region of the dam. The obtained result is depicted in Fig. 1 and compared to its respective finite difference solution.

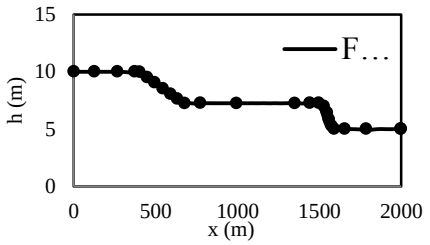


Fig. 1. Water depths at t=60(s) in a 1D dam-break problem.

At a distance of some 650m from the beginning of the channel in the upstream section of the broken dam, a rarefaction wave is visible at t=60s, which is moving toward the upstream side. Both numerical models show the same results for the water elevation in the horizontal channel from x=650m to x=1500m. However, some discrepancies are noticeable. For example, at x=1500m, an almost 90º shock wave is captured in the downstream section of the broken dam, which is associated with a trivial difference in comparing to Ming's model.

Please cite this article as: R. Karimi, A. A. Akhtari, O. Seyedashraf, Implementation of the skyline algorithm in finite-element computations of Saint-Venant equations, Journal of Applied Research in Water and Wastewater, 1(2), 2014, 63-67.

Table (1) depicts characteristics of the grid form and the matrices used in solving the linear equation systems.

Table 1. Global matrix characteristics for different element types used to discretize the computational domain.

Number of nodes on each element	3	4	5	6	7	8
Number of elements	100	100	100	100	100	100
Matrix dimension	201	301	401	501	601	701
Number of cells in the global matrix	40401	90601	160801	251001	361201	491401
Number of non-zero valued cells	801	1501	2401	3501	4801	6301
Skyline's matrix dimension	1002	1802	2802	4002	5402	7002
Number of zero valued cells	39600	89100	158400	247500	256400	485100

3. Discussion

As observed in table (1), extra zero-valued entries are produced in the global stiffness matrix, when the number of nodes in each element is increasing.

To conduct the numerical computations, a QBasic code was developed. The computer code was implemented on a Pentium 4, 2.40 GHz personal computer. Tables (2) and (3) depict the numerical results obtained from different types of elements employed to discretize the computational domain while two distinctive solvers were used to solve the linear equation system, which are Gauss-Jordan and Skyline, respectively.

Table 2. Analogy of the results obtained from a finite element simulation of dam-break flow using different elements. The linear equations are solved by a Skyline technique.

Node	Δx(m)	Δt (s)	Time
3	50	1	0 min , 7 sec
3	40	1	0 min , 8 sec
3	30	1	0 min , 9 sec
3	20	0.5	0 min , 26 sec
3	10	0.25	1 min , 36 sec
3	2.5	0.1	14 min , 57 sec
5	100	1	0 min , 10 sec
5	50	0.5	0 min , 33 sec
5	40	0.5	0 min , 40 sec
5	30	0.4	1 min , 3 sec
5	20	0.2	2 min , 58 sec
5	10	0.1	11 min , 15 sec
7	100	0.5	0 min , 33 sec
7	40	0.25	2 min , 22 sec
7	30	0.25	3 min , 4 sec
7	25	0.1	9 min , 9 sec
7	60	0.5	0 min , 50 sec
7	70	0.5	0 min , 43 sec
7	50	0.4	1 min , 13 sec
7	50	0.2	2 min , 25 sec
4	50	1	0 min , 12 sec
4	40	0.5	0 min , 27 sec
4	30	0.5	0 min , 33 sec
4	20	0.25	1 min , 29 sec
4	10	0.2	3 min , 27 sec
4	5	0.1	13 min , 20 sec
6	100	1	0 min , 16 sec
6	50	0.4	0 min , 54 sec
6	40	0.4	1 min , 4 sec
6	30	0.2	2 min , 45 sec
6	20	0.2	4 min , 0 sec
6	10	0.1	15 min , 57 sec
8	100	0.5	0 min , 41 sec
8	50	0.25	2 min , 32 sec
8	40	0.25	3 min , 7 sec
8	30	0.2	4 min , 59 sec

Table 3. Analogy of the results obtained from a finite element simulation of a dam-break flow using various elements. The linear equations are solved by a Gauss-Jordan technique.

Node	Δx(m)	Δt (s)	Time
3	50	1	6 min , 4 sec
3	40	1	11 min , 25 sec
3	30	1	25 min , 28 sec
3	20	0.5	>60 min
3	10	0.25	>60 min
3	2.5	0.1	>60 min
5	100	1	6 min , 6 sec
5	50	0.5	>60 min
5	40	0.5	>60 min
5	30	0.4	>60 min
5	20	0.2	>60 min
5	10	0.1	>60 min
7	100	0.5	40 min , 30 sec
7	40	0.25	>60 min
7	30	0.25	>60 min
7	25	0.1	>60 min
7	60	0.5	>60 min
7	70	0.5	>60 min
7	50	0.4	>60 min
7	50	0.2	>60 min
4	50	1	19 min , 25 sec
4	40	0.5	>60 min
4	30	0.5	>60 min
4	20	0.25	>60 min
4	10	0.2	>60 min
4	5	0.1	>60 min
6	100	1	11 min , 39 sec
6	50	0.4	>60 min
6	40	0.4	>60 min
6	30	0.2	>60 min
6	20	0.2	>60 min
6	10	0.1	>60 min
8	100	0.5	>60 min
8	50	0.25	>60 min
8	40	0.25	>60 min
8	30	0.2	>60 min

According to these tables, the Skyline technique has enhanced the solving process, which has led to a 10-times faster computational process.

4. Conclusion

The Skyline method was discussed in this research and was employed to solve the spare linear equation systems for numerical simulation of a shock wave propagation emanating from a one-dimensional dam-break problem. Skyline is a novel approach to deal with the problems associated with sparse matrices in numerical computations. The technique only deals with the non-zero valued cells of the global stiffness matrix, and stores them in a one-dimensional matrix, which was later solved using an elimination method. The numerical results obtained from the proposed procedure were evaluated through a test case and compared to the ones acquired from a regular Gauss-Jordan solver. It has been shown that Skyline is an appropriate scheme for reducing the CPU time required for processing instructions of a computer, and therefore, computer storage costs. Moreover, using this solver, the obtainable numerical data would be the same as the ones acquired from well-known and regularly used solvers. Consequently, employing this method is recommended for problems that lead to large linear equation systems.

Reference

Fang C., Sheu T., Two element-by-element iterative solutions for shallow water equations, SIAM Journal on Scientific Computing 22 (2001) 2075-2092.

Golub G.H., Van Loan C.F., Matrix Computations, Johns Hopkins University Press, Baltimore, 1996.

Irons, B.A., A frontal solution program for finite element analysis, International Journal for Numerical Methods in Engineering 2 (1970) 5-32.

Karimi R., Numerical Solution of the Dam Break Problem With Finite Element Method, Thesis, Razi University, 2012.

Please cite this article as: R. Karimi, A. A. Akhtari, O. Seyedashraf, Implementation of the skyline algorithm in finite-element computations of Saint-Venant equations, Journal of Applied Research in Water and Wastewater, 1(2), 2014, 63-67.

Nour-Omid B., Taylor R.L., An algorithm for assembly of stiffness matrices into a compacted data structure, Engineering Computations 1 (1984) 312-317.

Poole D. Linear Algebra: A Modern Introduction, Cengage Learning, Thomson Brooks Cole, 2006.

Quecedo M., Pastor M., A reappraisal of Taylor–Galerkin algorithm for drying–wetting areas in shallow water computations, International Journal for Numerical Methods in Fluids 38 (2002) 515-531.

Reddy J. N., An introduction to the finite element method, McGraw-Hill Higher Education, New York City, New York, 2006.

Seyedashraf O., Development of Finite Element Method for 2D Numerical Simulation of Dam-Break Flow Using Saint-Venant Equations, MSc. Thesis, Razi University, 2012.

Shen C., Zhang J., Parallel Two Level Block ILU Preconditioning Techniques for Solving Large Sparse Linear Systems, Parallel Computing, 28 (2002) 1451-1475.

Tseng M.H., Chu C.R., The simulation of dam-break flows by an improved predictor–corrector TVD scheme, Advances in Water Resources, 23 (2000) 637-643.

Please cite this article as: R. Karimi, A. A. Akhtari, O. Seyedashraf, Implementation of the skyline algorithm in finite-element computations of Saint-Venant equations, Journal of Applied Research in Water and Wastewater, 1(2), 2014, 63-67.