# Parallel Processing for 3-D Slope Stability Problems

Cheng Yung Ming
Department of Civil and Structural Engineering
Hong Kong Polytechnic University, Hong Kong
Hong Kong, China

Li Na
Department of Civil and Structural Engineering
Hong Kong Polytechnic University, Hong Kong
Hong Kong, China

*Abstract*—**Rigorous three-dimensional (3D) slope stability analysis is time consuming and takes major computer resources. To speed up the computation so that engineers can get results within shorter time is a bottleneck problem. In this study, we propose to speed up 3D slope stability analysis program with "OpenMP" which is tested to be effective, by using parallel processing. Further, the authors demonstrate that calculation time by using OpenMP becomes much less than before and this kind of parallel processing is not difficult to be used for general problems. Through the analysis, we have also discovered the number of computers' CPU has a great impact on the performance of the 3-D slope stability program. There are various precautions in using parallel processing, or else there will be conflict of memory address and changes of values by different threads of processes.**

***Keywords-parallel processing; limit equilibrium method; OpenMP; factor of safety.***

## I. INTRODUCTION

Landslip is one of the most serious disasters all over the world. To assess the potential dangerous of slopes, different slope stability analysis methods were developed and improved by researchers and engineers in the past. Among these methods, Limit Equilibrium Method (LEM) has been considered much by researchers in slope stability analysis.

All slope failures are three-dimensional (3-D) in nature. However, 3-D analysis based on limit equilibrium method is still seldom adopted in practice at present, because of the following limitations:

1. Direction of sliding is not considered in most existing slope stability formulations so that the problems under consideration must be symmetrical in geometry and loading;
2. Location of critical nonspherical 3-D failure surface under conditions is a difficult global optimization problem which has not been solved effectively; and
3. Existing methods of analyses are numerically unstable under transverse horizontal forces.

Two dimensional (2-D) modeling is usually adopted to greatly simplify the problem. However, all the actors of afety (FOS) obtained by 2-D slope stability methods are the approximate values, even though within a short computation time. While 3-D LEM can obtain a more reliable FOS with less assumptions, and more accurate consideration on the shape of the slope, but the computation time are highly increased at the same time.

Moreover, computer techniques are improved very fast in the past few decades and multiple CPUs/memory computers are largely applied in computing and engineering technology nowadays. It is a good chance for geotechnical engineers to perform parallel processing technique into slope stability analysis program to shorten the running time of 3-D slope stability programs.

In this research, parallel processing is proposed, and OpenMP, a shared-memory application programming interface (API) was tried. Calculation time, the bottleneck problem confused by most of the civil engineers, is testified to be shortened in 3-D limit equilibrium analysis in this study. There are seldom 3-D slope stability programs applying parallel processing recently. So it is a breakthrough in calculation time efficiency in 3-D slope stability analysis.

In this article, 3-D LEM methods are presented first, then the definition of parallel processing and OpenMP are briefly described. The main part of this research is to speed up for "SLOPE3D" by OpenMP, the authors explain how to do parallel processing in 3-D slope stability program, and the speedup efficiency is also discussed.

## II. THREE-DIMENSIONAL LIMIT EQUILIBRIUM METHOD

The use of three-dimensional slope stability method becomes necessary as increasing requirement for accuracy in slope stability analysis.

Baligh and Azzouz [3], Azzouz and Baligh [1] presented a method that extends the concept of the 2-D circular arc shear failure method to 3-D slope stability problems. The method was just appropriate for a slope in cohesive soil. The results obtained by the method showed that the 3-D effects could lead to 4 to 40 percent increase in the factor of safety.

Chen and Chameau [6] extended Spencer's 2-D method to 3-D. The sliding mass was assumed to be symmetrical and divided into several vertical columns. The inter-column forces had the same inclination throughout the mass, and the shear forces were parallel to the base of the column.

Hunger [10] proposed a 3-D method that is a direct extension of the assumptions associated with Bishop's [5] 2-D simplified method. He assumed that the vertical shear forces acting on both longitudinal and the lateral vertical faces of each column can be neglected in the equilibrium equations. Also the vertical force equilibrium equation of each column and the summary moment equilibrium equation of the entire assemblage of columns are sufficient conditions to determine all the unknown forces.

Cheng [7] proposed a new formulation for 3-D Morgenstern-Price Method. The sliding mass was divided into several vertical columns and three assumptions are

 ISBN: 978-1-61208-154-0 

included in the formulation: (1) Mohr–Coulomb failure criterion is valid; (2) For Morgenstern–Price's method, the factor of safety is determined based on the sliding direction where factors of safety with respect to force and moment are equal; (3) Sliding direction is the same for all soil columns (a unique sliding direction). The Morgenstern-Price formulation can also be simplified to 3-D Bishop and Janbu's simplified method by only consider force or moment equilibrium equation and neglecting all the inter-column vertical and horizontal shear force.

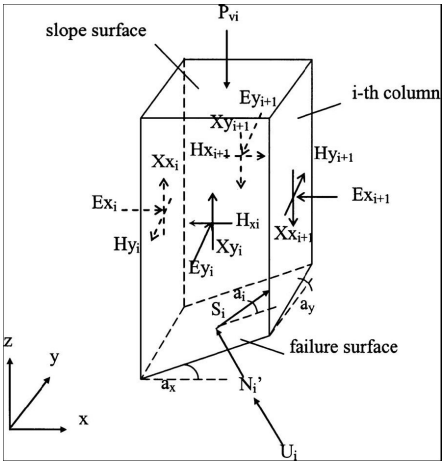


Figure 1. External and internal force acting on a typical soil column

A unique sliding direction shown in Figure 1 can be determined in the present 3D slope stability methods, which can also prevent the spread of the sliding when considering transverse earthquake load. And this method can easily converge. (Cheng, [8])

## III. PARALLEL COMPUTERS – OPENMP PROGRAM

Today's computer systems become highly complex. Multiple CPUs and memory appear. As a result, a computer might be able to fetch a datum from memory, multiply two floating-point numbers, and evaluate a branch condition at the same time. We call this kind of function parallel processing and those computers are shared-memory parallel computers or multiprocessor computers (Chapman, [4]).

Although modern computers can achieve parallel processing, it's impossible to process automatically. Therefore compilers are necessary to identify independent streams of instructions which can be executed in parallel and OpenMP is one of the programming interfaces which can help us to do parallel processing by adding it in the sequential program.

OpenMP is a shared-memory application programming interface (API) but it is not a new programming language. It can be coded by FORTRAN (most of the civil program is written by FORTRAN), C and C++ to describe how work is to be shared among threads which will execute on different processors or cores and to order access for sharing data as needed. OpenMP supports the so-called fork-join programming model illustrated in Figure 2. Under this approach, the program starts as a single thread (initial tread) of execution, just like a sequential program. Once an OpenMp parallel is constructed, the program is executed automatically to create a team of threads (Fork) implementing independent work and only the original thread, or master of the team is continuous at the end of the construction (Join). The space between Fork and Join is called a parallel region. (Chapman, Jost, 2008)

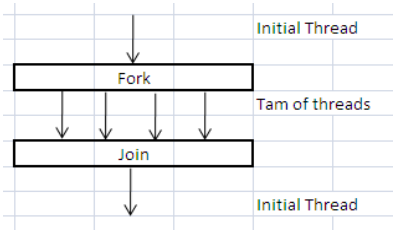


Figure 2. The Fork-join programming mode support by OpenMP

OpenMP have two major advantages, one is its strong emphasis on structured parallel programming and the other is simples to use. Those reasons make it popular and run on many different platforms nowadays (Chapman, [4]).

## IV. SPEEDING UP FOR 3D SLOPE STABILITY ANALYSIS PROGRAM BY "OPENMP"

Due to the large number of unknown and complicated calculation procedure in three-dimension LEM, it is very time consuming to calculate minimum factor of safety with high accuracy for a wide range of possible failure surface with a large scale and complicate slope (usually with more than two soil layers) by 3-D "rigorous" limit equilibrium method. And the calculation time can be counted in number of days.

Parallel processing by shared-memory parallel computers or multiprocessor computers is proposed in the paper to shorten the computation time for three-dimensional limit equilibrium method using OpenMP. Slope stability analysis program "SLOPE3D" is a sequential program written in FORTRAN and was speeded up by OpenMP in this research. The methods and speedup efficiency of using OpenMp in "SLOPE3D" are discussed in the following sections. A 3-D slope model case of "SLOPE3D" is shown in Figure 3.

### *A. How to Use OpenMP in Three-Dimensional Slope Stability Analysis Program*

First step is to classify the part of the program which is suitable to implement parallelization to gain the maximum efficiency. However, it's quite difficult to parallel all parts of the program. The procedure of 3D slope stability analysis program can be mainly distributed into four parts, input reading, previous calculation (usually for simple calculations which can obtain object values in main calculation by changing the input value), main calculation and output generation. The main calculation occupies large part of computational procedures and it takes most of the computation time, so parallelization was tried in this part to obtain the maximum efficiency. Two ways of parallel

 ISBN: 978-1-61208-154-0 

processing are tried in this research to use in the main calculation part of "SLOPE3D", illustrated in Figure 4 and 5.

In Model 1 of Figure 4, two parallel regions are generated and a new matrix value Temp_TTX(i,j) is developed in 1st region to store all the initial calculating values. Open MP enables program to create team of threads to distribute the Do-Loops according to the number of CPUs, e.g., 4 CPUs, 4 parts of Do-Loops. In 2nd region, all threads will simultaneously read the values stored in TTX, add their value in Temp_TTX (i,j), and write out the results in the single storage location for summation TTX.

Different from Model 1, there was only one parallel region in Model 2 in Figure 5. Both values of Temp_TTX and summation values of TTX are calculated in the same parallel region. Each thread (CPU) will make a copy for the value; read and write will only execute within the thread. 4 different values of Temp_TTX will be stored in each thread (CPU). To prevent "Data Competition", a Critical Clause is added to allow only one thread to do calculation to obtain summation values of TTX from 4 private Temp_TTX. Model 2 reduces the number of parallel regions from 2 to 1, which can reduce the fork and join time.

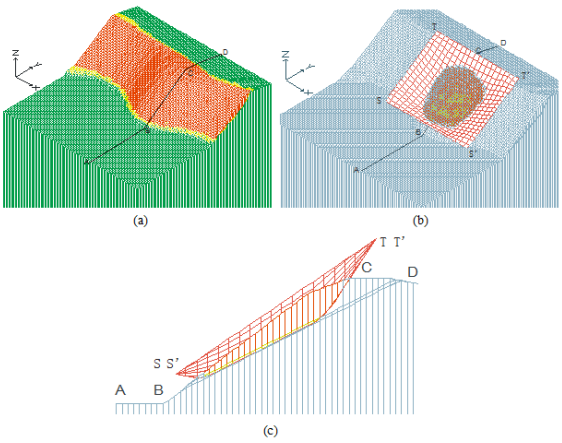


Figure 3. "Slope 3D" model of a concave and convex slope combination: (a) a concave and convex slope highlighted in red region; (b) FEM meshing made in the sliding area; (c) cross-section of the failure surface of the slope

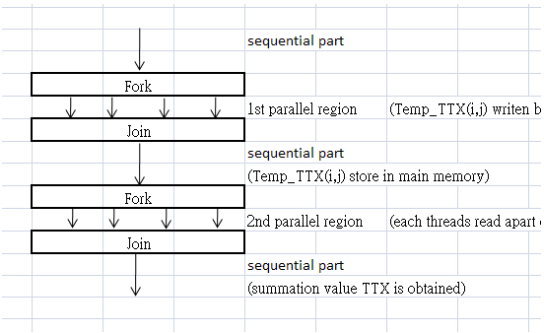


Figure 4. Mine-Map of parallel processing Model 1

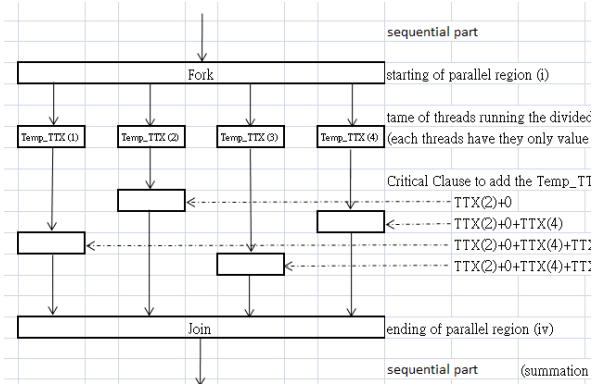


Figure 5. Mine-Map of parallel processing Model 2

## B. *Running Time Comparison*

In order to check the efficiency of those parallel processing models above, a traditional 3-D slope is analyzed by the parallel programs mentioned above with 3-D Morgenstern-Price Method. Two issues are focused on the comparison; one is the running time against number of divided soil columns. The number of soil columns divided in failure mass of slope is an important factor affecting the accuracy of factor of safety. The other issue is the running time of computers with different number of CPUs. The results are shown in Table 1, 2 and 3 with 2 CPUs, 4 CPUs and 8 CPUs computer, respectively.

Several issues can be observed from the results shown below. First, for the third lines in the nethermost part of these three curves which is the running time for original sequential program are quite linear in Figure 7 and 8. It can be assumed that the running time of sequential 3-D slope analysis program is directly proportional to the number of soil column divided in the same slope analysis problem.

Moreover, there are apparent differences on running time between the original program and the two parallel processing programs when the numbers of soil columns are higher than 100,000. Also the difference is increasing when the numbers of divided soil columns are increasing. This is reasonable, because in respect to the typical calculation element, the increasing numbers of soil columns are referring to the size of the Do-Loop. Therefore, the calculation time for the Do-Loop will be increase and the performances of parallel processing should also be enhanced.

Although the difference between the running time in two parallel processing program are not large, the performance of program using parallel processing Model 2 can be noticed being better than Model 1, for the reason that extra running time can be reduced by decreasing the number of parallel regions and the matrix value Temp_TTX(i,j) shown in Figure 5.

On the other hand, if we focus on the speedup percentage in these three groups of results, negative value is found when the numbers of soil columns are lower than 40,000. This can be explained by the time using in forking and joining the paralleled threads. OpenMP supports the so-called fork-join programming model, illustrated in Figure 2, and extra

 ISBN: 978-1-61208-154-0 

programming time is used to this model. If the reduced calculation time we gain due to parallel processing are lower than the forking and joining time we use for the paralleled threads, negative speedup will occur. In order to obtain a positive speedup percentage, a boundary can be set to limit the use of parallel processing which will only occur when the number of divided are higher than 40,000.

The maximum speedup percentage for 2 CPUs, 4CPUs and 8 CPUs computer are 27.31%, 29.95% and 36.5%, respectively. Higher speedup percentage can be obtained for computer with more CPUs. This is because, for the same size of original Do-Loop in the program, more Do-Loops with relative smaller size can be divided when more number of threads (number of CPUs) is provided. Much running time can be reduced due to the Do-Loops can run in once time.

TABLE I. RUNNING RESULT FROM 3-D MORGENSTERN-PRICE METHOD IN 2 CPUS COMPUTER

| **Analysis method :** | | **Morgenstern-Price's Method** | | | | |
|---|---|---|---|---|---|---|
| **No. of Cycles :** | | 100 | | | | |
| **No. of CPU** | | 2 | | | | |
| **Spacing** | **No. of soil columns** | **Original running time (s)** | **OpenMP running time (s)** | | **Speed Up (%)** | |
| | | | OMP 1 | OMP 2 | OMP 1 | OMP 2 |
| 0.5 | 1,600 | 8.391 | 21.297 | 16.922 | -153.81 | -101.67 |
| 0.4 | 2,500 | 12.75 | 24.985 | 20.453 | -95.96 | -60.42 |
| 0.25 | 6,400 | 32.969 | 42.547 | 38.125 | -29.05 | -15.64 |
| 0.2 | 10,000 | 51.781 | 59.969 | 54.906 | -15.81 | -6.04 |
| 0.1 | 40,000 | 288.641 | 260.656 | 246.734 | 9.70 | 14.52 |
| 0.05 | 160,000 | 1561.656 | 1327.985 | 1307.937 | 14.96 | 16.25 |
| 0.04 | 250,000 | 2637.532 | 2196.875 | 2028.203 | 16.71 | 23.10 |
| 0.03 | 443,556 | 4819.532 | 3805.922 | 3503.328 | 21.03 | 27.31 |
| 0.025 | 640,000 | 11352.75 | 9282.125 | 8642.453 | 18.24 | 23.87 |

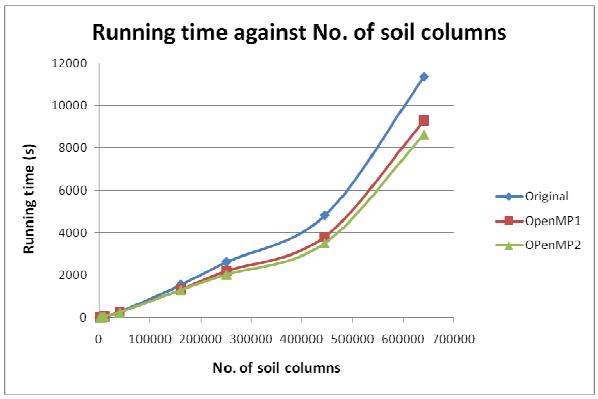


Figure 6. Running time against number of divided column in 2 CPUs computer.

TABLE II. RUNNING RESULT FROM 3-D MORGENSTERN-PRICE METHOD IN 4 CPUS COMPUTER

| **Analysis method :** | | **Morgenstern-Price's Method** | | | | |
|---|---|---|---|---|---|---|
| **No. of Cycles :** | | 100 | | | | |
| **No. of CPU** | | 4 | | | | |
| **Spacing** | **No. of soil columns** | **Original running time (s)** | **OpenMp running time (s)** | | **Speed Up (%)** | |
| | | | OMP 1 | OMP 2 | OMP 1 | OMP 2 |
| 0.5 | 1,600 | 12.67 | 26.875 | 24.312 | -112.12 | -91.89 |
| 0.4 | 2,500 | 17.97 | 44.312 | 28.516 | -146.59 | -58.69 |
| 0.25 | 6,400 | 47.58 | 57.203 | 51.484 | -20.22 | -8.21 |
| 0.2 | 10,000 | 73.56 | 76.891 | 71.906 | -4.53 | 2.25 |
| 0.1 | 40,000 | 338.7 | 293.641 | 280.172 | 13.30 | 17.28 |
| 0.05 | 160,000 | 1472.52 | 1181.703 | 1107.156 | 19.75 | 24.81 |
| 0.04 | 250,000 | 2416.39 | 1829.391 | 1838.923 | 24.29 | 23.90 |
| 0.03 | 443,556 | 4069.52 | 3044.063 | 2850.516 | 25.20 | 29.95 |
| 0.025 | 640,000 | 6082.75 | 4909.578 | 4743.078 | 19.29 | 22.02 |

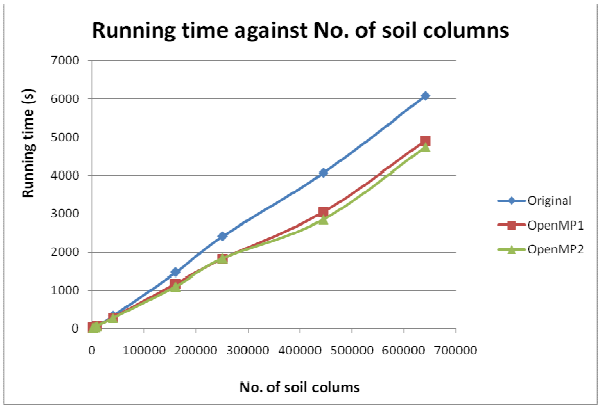


Figure 7. Running time against number of divided column in 4 CPUs computer.

TABLE III. RUNNING RESULT FROM 3-D MORGENSTERN-PRICE METHOD IN 8 CPUS COMPUTER

| **Analysis method :** | | | **Morgenstern-Price's Method** | | | |
|---|---|---|---|---|---|---|
| **No. of Cycles :** | | | 100 | | | |
| **No. of CPU** | | | 8 | | | |
| **Spacing** | **No. of soil columns** | **Original running time (s)** | **OpenMP running time (s)** | | **Speed Up (%)** | |
| | | | **OMP 1** | **OMP 2** | **OMP 1** | **OMP 2** |
| 0.5 | 1,600 | 8.094 | 39.297 | 33.609 | -385.51 | -315.23 |
| 0.4 | 2,500 | 11.922 | 42.5 | 36.453 | -256.48 | -205.76 |
| 0.25 | 6,400 | 33.5 | 59.063 | 53.203 | -76.31 | -58.81 |
| 0.2 | 10,000 | 51.765 | 74.469 | 71.89 | -43.86 | -38.88 |
| 0.1 | 40,000 | 247.313 | 227.906 | 214.7973 | 7.85 | 13.15 |
| 0.05 | 160,000 | 1063.859 | 843.578 | 835.391 | 20.71 | 21.48 |
| 0.04 | 250,000 | 1803.406 | 1298.266 | 1308.688 | 28.01 | 27.43 |
| 0.03 | 443,556 | 3229.969 | 2108.297 | 2050.922 | 34.73 | 36.50 |
| 0.025 | 640,000 | 4854.312 | 3502.86 | 3265.375 | 27.84 | 32.73 |

 ISBN: 978-1-61208-154-0 

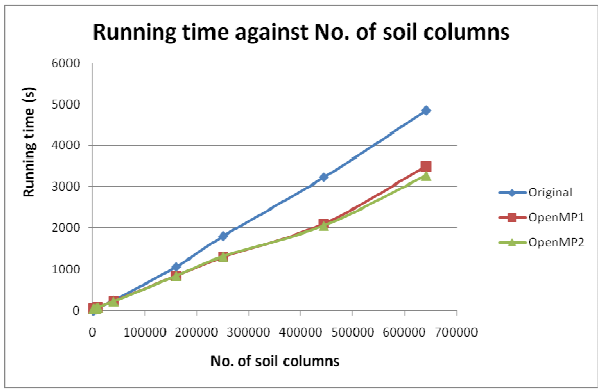


Figure 8. Running time against number of divided column in 8 CPUs computer.

## V. CONCLUSIONS AND DISCUSSIONS

Based on the present study, the theoretical backgrounds for 3-D limit equilibrium methods and OpenMP were reviewed. The basic concept of parallel processing and OpenMP are also studied above. Furthermore, how to do parallel processing in 3-D limit equilibrium program "SLOPE3D" is described and its efficiency is also discussed. The following conclusions are summarized and presented for parallel processing in slope stability program:

- OpenMP, a shared-memory application programming interface (API) is a suitable interface for civil engineers to do parallel processing in slope analysis program to reduce the calculation time. It is relatively easy for use and can be added into a sequential program coded in FORTRAN.
- The efficiency of using OpenMP is highly affected by the reduction time obtained from paralleled Do-Loop and the forking & joining time used in the teams of thread.
- About one-third of the calculation time can be reduced by using OpenMP and the larger numbers of CPUs exist in the analyzed computer, the better performance of the program is gained.

Comparing to other shared-memory programming interfaces, OpenMP has better compatibility than them. Take MPI as an example, for those researchers and engineers who do not have enough computer skills, OpenMp not only can be easily used to solve general problems, but also runs well in every computer without modifying the code, while MPI code can only face one single server. Moreover, using the server with higher number of processors to speed up the efficiency of numerical program is not widely applied in Civil engineering. And the 30% time-shortening implementation in this article is acceptable to geotechnical engineers. There is no need to spend so much time on a shared-memory programming interface as complicated as MPI.

At future work, the specific code for the application can be further parallelized. The authors will consider trying some different shared-memory API, further parallelizing the specific code, and experimenting with a higher size experiment with higher number of processors if the equipment is available.

## REFERENCES

[1] Anderson, M. G. and Richards, K. S. (1987), Slope Stability: Geotechnical Engineering and Geomorphology, John Wiley and Sons, N.Y.

[2] Azzouz, A. S. and Baligh, M. M. (1978). Discussion on Three-dimensional slope stability analysis method. J. Geotech. Engng Div. ASCE, 104,GT9, 1206-1208.

[3] Baligh, M. M., and Azzouz, A. S. (1975). End effects on stability of cohesive slopes. J. Geotech. Engrg. Div., ASCE, 101(11), 1105–1117.

[4] Chapman B., Jost G., and Ruud v. d. Pas (2008), Using OpenMP : portable shared memory parallel programming, Cambridge, Mass. : MIT Press, pp. 1-3, 8-24

[5] Bishop, A. W. (1955). The use of the slip circle in the stability analysis of slopes. Ge´otechnique, London, V(1), 7–17.

[6] Chen, R. H. and Chameau, J. L. (1983), Three-dimensional limit equilibrium analysis of slopes, Geotechnique, 33(1), pp. 31-40.

[7] Cheng Y.M. and C. J. Yip (2007), Three-Dimensional Asymmetrical Slope Stability Analysis Extension of Bishop's, Janbu's, and Morgenstern–Price's Techniques, Journal of Geotechnical and Geoenvironmental Engineering, ASCE ,Vol. 133, No. 12, December 2007, pp. 1544-1555

[8] Cheng Y.M. and C.K. Lau (2008), Slope stability analysis and stabilization: new methods and insight , Routledge Publishers, pp. 17-21

[9] Fredlund, D. G., and Krahn, J. (1977), Comparison of Slope Stability Methods of Analysis, Canadian Geotechnical Journal, 14, pp. 429-439

[10] Hungr, O. (1987), An extension of Bishop' Simplified Method of slope stability analysis to three dimensions, Geotechnique, Vol. 37, No. 1, pp. 113-117

[11] Chandra R., Dagum L., Kohr D., Maydan D., McDonld J., and Menon R. (2001), Parallel programming in OpenMP, Morgan Kaufmann Publishers, pp. 33-35

 ISBN: 978-1-61208-154-0 