User Manual

# Program Purpose

The purpose of this program is to determine stress and strain distributions within a dry gravity loaded rock mass based on specified model geometry, overburden unit weight, ground surface elevation, applied boundary conditions, and excavations within the mesh.

# Directory Layout

The “GeoSolver.m” main function calls sub-functions and opens input files from directory paths relative to the main function. Therefore, the program folder with all of its contents can be located in any folder, but the directory structure within the main program folder must not be altered. The program folder must consist of “Geosolver.m”, a folder entitled “Input\_Files” which contains all input text files, and a folder entitled “Functions” which contains all sub-functions called from the main function. This directory structure is used in the zip file containing all of the program contents, so directory issues should not be incurred.

# Input Files

The program reads in a single “.txt” file containing all the necessary information to generate the matrices required to perform the finite element analysis. The format for the input files is defined by the output from ‘*Zazu’*. There are four important criteria that must be met for the input file to be read correctly.

**Specification of Text File:**

The name of the text file must be specified in the main function of the code in line 8. The name of the file should be entered **WITHOUT** the “.txt” extension, and the corresponding text file must be saved in the “Input\_Files” directory.

**Section Headers:**

There will be three sections that will be read by our code: the node list, the element (incidence) list, and the excavation coordinate list. The first defines the x-y location of the nodes, the second constrains the connectivity between nodes, and the third defines the nodes along excavation boundaries. The program code will define the start of each section using keywords.

Start of node list: ***BEGIN\_NODES***

Start of element list: ***BEGIN\_ELEMENTS***

Start of excavation list:  ***BEGIN\_EXCAVATIONS***

The former 2 lists must be included in the text input file, whereas the latter need only be included if the problem has excavations.

To define the end of a section, there must be at least one blank line between the end of one section and the start of another, or the end of the text file must follow. The code uses these strings as a search cursor to begin reading in values. If these are not present, then the program will not run. These 3 sections **MUST** be in this order or the data will not be properly read in to Matlab.

**Delimiters:**

The columns in the input file are intended to be tab-delimited. Any other delimiters (comma, period, etc) could introduce errors into the input matrices.

**Columns:**

There is a specific column structure that is output by ‘*Zazu’*, and the input file must match this structure. Column headers / title can **NOT** be included in the input file, and each column is separated by a single tab. The required columns are shown below in the proper order. Both node, element, and excavation lists have unique indices starting from 1. Three nodes are identified in the Element list by referring to the index of each respective node.

Node List:

*Index X Y*

Element List:

*Index ‘Quad’ ‘EL1’ ‘ECTri1’ Node1 Node2 Node3*

Excavation List:

*Excavation# X Y*

For the first Node and Element Lists, the first column should be a numbered list beginning at 1. For the Excavation List, the first column represents the excavation number.

**Example input file**

The following is an example of a text file that can be read by the program:

%================================================

BEGIN\_NODES

1 0 0

2 0.866 1.5

3 1.732 0

4 2.598 1.5

5 3.464 0

BEGIN\_ELEMENTS

1 Quad EL1 ECTri1 1 2 3

2 Quad EL1 ECTri1 2 3 4

3 Quad EL1 ECTri1 3 4 5

%================================================

The following is an example of the excavation section of a larger input text file that can be read by the program:

%================================================

BEGIN\_EXCAVATIONS

1 30 30

1 40 30

1 40 80

1 30 80

2 0 0

2 0 20

2 6 19.0788

2 10 17.3205

2 12 16

2 14 14.2829

2 16 12

2 18 8.7178

2 20 0

%================================================

# Data Structures

The code will read the text file into Matlab, and store it in two matrices. The node matrix will be named *node\_coords* and will have the exact same format as the input text.

node\_coords = [X-Coordinate Y-Coordinate]

The element matrix has been simplified from the text input to improve program speed and simplicity, so that on the index and node references are preserved. The element matrix will be named *incidences*.

incidences = [node1Index node2Index node3Index]

Once these two matrices are defined in Matlab the mesh that will be used in the finite element analysis is drawn.

The element stiffness matrices are calculated by the ElementStiffnessMatrix() function sequentially and are stored in the three-dimensional matrix named *element\_stiffness\_matrices.* The dimensions of this matrix are as follows:

element\_stiffness\_matrices(6,6,n\_el)

The global stiffness matrix is calculated by the GlobalStiffnessMatrix() function and is stored in the two-dimensional matrix named *global\_stiffness\_matrix.* The dimensions of this matrix are as follows:

global\_stiffness\_matrices(2\*n\_nodes,2\*n\_nodes)

The resultant elemental stresses and strains as well as nodal forces and displacements are stored in Matlab in the following variables:

*u\_solved –* vector of length 2\*n\_nodes which stores all x and y displacements at each node, and is in the same order as the node list *node\_coords*

*f\_solved* – vector of length 2\*n\_nodes which stores all x and y forces at each node, and is in the same order as the node list *node\_coords*

*strains* – matrix of dimensions 3 rows by n\_el columns which stores all strains of each element. Row 1 contains x-strains, row 2 contains y-strains, and row 3 contains shear strains.

*stresses* – matrix of dimension 5 rows by n\_el columns which stores all stresses of each element. Row 1 contains x-stresses, row 2 contains y-stresses, row 3 contains shear stresses, row 4 contains the maximum principle stress (sigma 1), and row 5 contains the minimum principle stess (sigma 3).

The error in calculated displacements is calculated within the Solve\_With\_Partitioning() function and is stored in the vector named *err.* This error is the root-mean-square-error.

# Results Interpretation

# Sign Convention

The positive x- and y-directions are to the right, and upwards, respectively, and therefore x- and y-forces and displacements are positive to the right and upwards. Stresses are defined as positive in compression. Strains follow the same sign convention as displacements.

# Interpolation Methods

There are three interpolation methods for the stress and strain contours: *Element Centre Points, Nodal Averages,* and *Constant Strain Elements. Element Centre Points* interpolates between the actual elemental values at the centroids of the elements, and thus provides the technically most accurate representation of the finite element results. *Nodal Averages* interpolates between stress and strain values at each node, and each nodal stress or strain value is an average of the elemental values of the elements which share that node. *Nodal Averages* thus provides a more visually appealing image since contours extend to the model boundaries, however in coarse meshes the results may not be as accurate as results using the *Element Centre Points* method. *Constant Strain Elements* plots the same values as *Element Centre Points,* except the elements are plotted as constant strain triangles instead of using interpolation functions.

# Output Files

Output files can be saved as a \*.txt file by the user from the click of a command button on the *Results* form. There are 3 sections of any output file – the incidence list and corresponding element-specific values, the node list and corresponding node-specific values, and the root-mean-squared error calculated on the unknown displacements. A sample output files is shown on the following page.

# Verification

All verification files are contained within the *Verification* folder. The following Excel files should be viewed for verifications:

*Element Stiffness Matrix Verification.xlsx*

*Global Stiffness Verification.xlsx*

*Primary and Secondary Unknown Verification.xlsx*

*Secondary Unknowns – Principle Stress Verification.xlsx*

Sample output file (pasted into a Microsoft Excel spreadsheet):

