gmESSI Manual

A translator from gmsh to ESSI Files

Sumeet Kumar Sinha

July 18, 2016

Contents

1 Introduction

gmESSI pronounced as *[g-mESSI]* is an acronym for gmsh (a three-dimensional finite element mesh generator with built-in pre- and post-procESSIng facilities) to ESSI (UC Davis Earthquake-Soil-Structure-Interaction Simmulator) Finite Element Program Translator. The translator has been developed in C++ language by Sumeet Kumar Sinha, a PhD student under Prof. Boris Jeremic at University of California Davis. The primary aim is to provide an easy, handy and powerfool pre-procESSIng tool to develop FEA models and make them interface with various ESSI functionalities. The gmESSI Translator Package consist of the *Translator, Python module, a Sublime plugin and its manual.* The whole package can be downloaded from here. The sublime plugin ***[gmESSI-gmsh Tools]*** can be downloaded from Package Control or from GitHub.

Note:- ***However smart a translator is, never fully trust it! Always check out the output files generated to see whether the conversion made was correct or not.***

2 Getting Started

The translator utilizes the **Physical and Entity group** concept of Gmsh, which gets imprinted in the mesh **.msh** file and then manupulates these groups to convert the whole mesh to ESSI commands. Thus, making physical groups is the key to conversion for this Translator. The Translator basically provides some strict syntax for naming these Physical Groups using ***tags and gmESSI commands*** which provides it the information about how and what to convert. *The translator is made so general that any other FEM program can use it with little tweaks to have their own conversion tool.* A quick look at some important features of the program are:

- It has a lot of **predefined commands** which do the conversion at the blink of an eye. These commands make it easier to define elements, boundary conditions, contacts, fixities, loads ....
- It provides a python module **gmESSI** using which the user can program and make their own conversions.
- The **gmESSI sublime plugin** makes it easy by providing syntax colouring and text-completion for gmsh as well as gmESSI commands. The plugin can be installed by the name of **gmESSI-gmsh Tools** from sublime package control. The repository for this plugin can be found at GitHub
- The translator uses a ***mapping.fei*** file to check for its command syntax and conversion. A user can easily add a command in **mapping.fei** and it would get reflected automatically in the translation.

2.1 Installation Process

The Translator have its dependencies on Octave (3.2 or higher), Boost(1.58 or higher), (Python 2.7 or higher) and g++ (4.9 or higher) commpiler. One should make sure to have them before compiling it. A quick reference to these dependencies are

- **Boost**:: It can be downloaded urlhttp://www.boost.org/users/download. Just follow the installation steps as provided on the site. You can also install boost as **sudo apt-get install boost-all-dev.**
- **Octave**:: The package is available at `http://packages.ubuntu.com/search?keywords=octave` for different ubuntu versions. Make sure to comment all the *warnings* of file ***ov-typeinfo.cc*** before compiling the package. **Alternatively**, one can try to install liboctave by running **sudo apt-get install liboctave-dev** but the user would get warnings from octave while running the program.
- **Python**:: The ubuntu users can install it using **sudo apt-get install python-dev**.
- **g++**:: the latest version of the gnu compiler can be installed on linux machine by running the following commands.

  sudo apt-get install build-essential
  sudo add-apt-repository ppa:ubuntu-toolchain-r/test
  sudo apt-get update
  sudo apt-get install gcc-4.9 g++-4.9 cpp-4.9 gfortran-4.9

Installation of the gmESSI translator is really easy. Just follow the steps as shown below. Script

```
# go to folder where you want to store gmESSI software and run this script there

#Download the latest version of the package from the github
git clone https://github.com/SumeetSinha/gmESSI.git
cd ~/gmESSI

# install the package
make
make install

############## For installation of gmESSI plugin in sublime ##############
# open sublime−text
# make sure you have installed package control
# if not then install it first from https://packagecontrol.io/installation
# go to Preferences−>PackageControl−>InstallPackage
# search for gmESSI−gmsh Tools or gmESSI−Tools and install it
# restart sublime
###################################################################
```

2.2 Running gmESSI

gmESSI can be invoked from the bash terminal by typing **gmESSI**. It can take *one or multiple* **XYZ.msh** files as an argument and convert them to ESSI files in their respective folders names as **XYZ__ESSI__Simulation** placed at the execution location. Example1.geo Example1.msh

2.2.1 In Terminal

```
$ gmessi ./Example1.msh
Message::Files converted to  /gmESSI/Example1_Essi_Simulation

Add_Node_Load_Linear{Fx,10*kN} Found!! Sucessfully Converted
Add_All_Node{m,3} Found!! Sucessfully Converted
Fix{all} Found!! Sucessfully Converted
Add_8NodeBrick{1} Found!! Sucessfully Converted


************************ Updated New Tag Numberring *********************
damping = 1
displacement = 1
element = 27
field = 1
load = 4
material = 2
motion = 1
node = 55
nodes = 55
```

2.2.2 In Python

The same translation can also be achieved in python. gmESSIScriptExample1.py

```
$ python
Python 2.7.9 (default, Apr 2 2015, 15:33:21)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import gmessi
>>> PythonExample = gmessi.gmESSIPython("Example1.msh")
WARNING::Directory Allready Present. The contents of the Folder may get changed
Message::Files converted to  /gmESSI/Example1_Essi_Simulation

Add_Node_Load_Linear{Fx,10*kN} Found!! Sucessfully Converted
Add_All_Node{m,3} Found!! Sucessfully Converted
Fix{all} Found!! Sucessfully Converted
Add_8NodeBrick{1} Found!! Sucessfully Converted
>>> PythonExample.DisplayNewTagNumbering()


************************ Updated New Tag Numberring *********************
damping = 1
displacement = 1
element = 27
field = 1
load = 4
material = 2
motion = 1
node = 55
nodes = 55
```

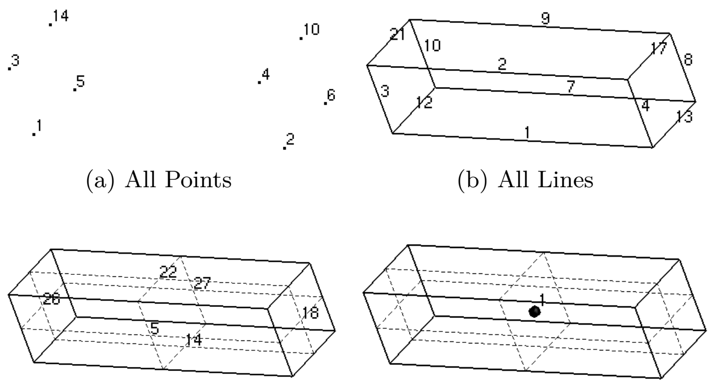
The translator creates a directory **Example1_ESSI_Simulation** and places ***Example_geometry.fei***, ***Example1_load.fei***, ***Example1_analysis.fei*** and ***Example1.msh*** in it. The user now only needs to add solver and tweak the file ***Example1_analysis.fei*** to run his analysis on ESSI. The terminal displays the *warning, error messages and log of command conversions* as shown above.

In the end it shows the **new number tags** for each **ESSITag** of which user can refer and use for further conversion. **ESSITag** is explained later in this manual.

3 Gmsh Physical Groups and Geometrical Entities

Geometrical Entities are the most elimentary objects in **Gmsh**. Each *point, line, surface and volume* is a geometrical entity and possess a unique identification number. Elementary geometrical entities can then be manipulated in various ways, for example using the *Translate, Rotate, Scale or Symmetry* commands. They can be deleted with the *Delete* command, provided that no higher-dimension entity references them. Below ***Example1.geo*** is the description of a **.geo** file in **Gmsh** for creating a *cantilever beam.*

```
$ vim Example1.geo
Point (1) = {0,0,0}; // Creating a point
Extrude (4,0,0) {Point{1}; Layers{5};} // Dividing the beam length in 5 parts
Extrude (0,1,0) {Line{1}; Layers{2};Recombine;} // Dividing the beam width in 2 parts
Extrude (0,0,1) {Surface{5}; Layers{2};Recombine;} // Dividing the beam depth in 2 parts
```



(a) All Points (b) All Lines

(c) All Surface (d) All Volume

Figure 1: Showing Geometrical Entities. Every point, line, surface and volume has an unique identification number assigned to it.

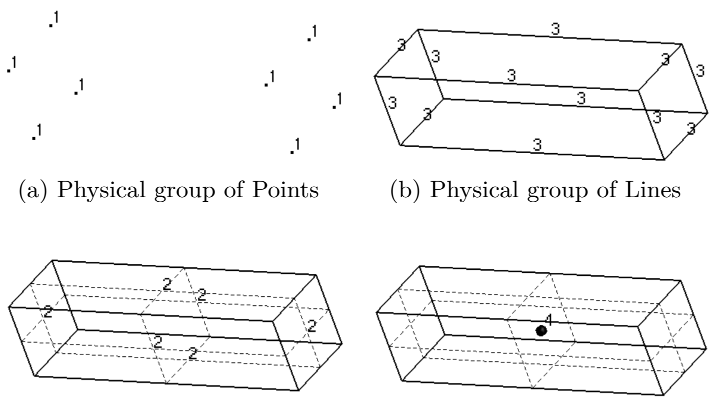
Figure 1 shows, the different unique identification number attached to each of the nodes, lines, surface and volume of the geometry of **cantilever beam**. The user can now **create physical groups** of type *{nodes, lines, surface, volume}* containig one or more geometrical entities of that type i.e. each is either *points, lines, nodes and surface* respectively.

Physical Groups are groups of same type *{nodes, lines, surface, volume}* of elementary geometrical entities. These **Physical Groups** cannot be modified by geometry commands. Their only purpose is to assemble elementary entities into larger groups, possibly modifying their orientation, so that they can be referred to by the mesh module as single entities. As is the case with elementary entities, each *physical point, physical line, physical surface or physical volume* are also assigned a unique identification number.

Note:- A **geometrical entity** has only one elemenatry entity number but can be a part of many **physical groups** by sharing their *identification number.*

Below is the continuation of ***Example1.geo*** in **Gmsh** for creating **physical Groups** of *cantilever beam.* Just for the sake of example, **4 physical Groups** are created which consisit of all points, lines, surface and volume respectively of the **cantilever beam model**.

```
$ vim Example1.geo
.....
Physical Point ("All Points") ={1,2,3,4,5,6,10,14};
Physical Surface("All Surface group") = {5,14,22,27,18,26};
Physical Line("All Lines") ={1,2,3,4,12,13,21,17,7,8,9,10};
Physical Volume("All Volume") ={1};
```


(a) Physical group of Points (b) Physical group of Lines

(c) Physical group of Surface (d) Physical group of Volume

Figure 2: Showing all 4 Physical Groups with entities numbered by their physical group id's

All the **geometrical entities** have a **tag list** which **contains the ids of the physical groups to which it belongs or is associated**. In the above example of Figure 2, *every point, line, surface, volume* belongs to only one physical group and thus are showing only one associative number against themselves. In Figure 3 shows **geometrical entities** which are *part of many physical groups.* For example:- the volume in the figure is shown to be a part of two **physical group of volumes** *having id 1 and 7.*

The whole idea of creating a **Physical Group** of *points, lines, surfaces and volumes* and giving it a unique string name is to allow quick **identification and manupulation** during *post-procESSIng.* In **Gmsh** the name of these **Physical Group** along with their corresponding elements and nodes gets transferred to the mesh **.msh** file as shown below. Later in Figure 4 how **Gmsh interpretes** these **Physical groups** in **.msh** file is described.

```
$cat Example1.msh
.....
$PhysicalNames
4
0 1 "All Points"
1 3 "All Lines"
```

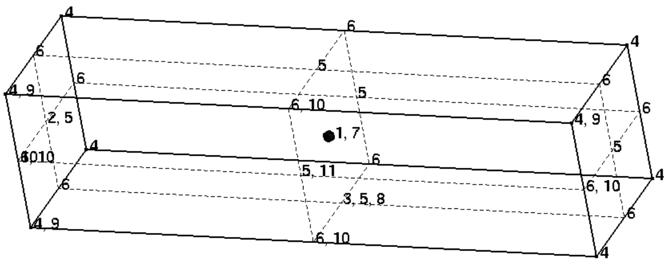


Figure 3: Showing geometrical entities associated with more than one physical group

```
2 2 "All Surface group"
3 4 "All Volume"
$EndPhysicalNames
......
```

Note:- While creating a **physical group** in Gmsh, **only the information (nodes and elements) of that physical group** gets written in the **.msh** file **and rest are not written**, so the user must be carefull to create physical groups of all entities which he needs during post-processing. More information about **Gmsh syntaxing, physical groups, commands, .msh file, save options ... etc** is available on its online documentation `http://geuz.org/gmsh/doc/texinfo/gmsh.html`

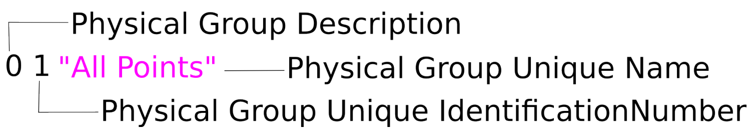

Physical Group Description
0 1 "All Points"
Physical Group Unique Name
Physical Group Unique IdentificationNumber

Figure 4: Description showing how gmsh interprets the Physical Groups.

- **Physical Group Description ::** Gmsh uses it to identify the type of physical group. $0, 1, 2 and 3$ represents the physical group of geometric points, lines, surface and volume respectively.

- **Physical Group Unique Identification Number ::** It is an unique identification number automatically assigned to each physical group by gmsh.

- **Physical Group Unique Name ::** It is also the same as *Physical Group Unique Identification Number* but the difference is that it is not automatic but defined by the user and that too in the form of ***string***.

The gmESSI Translater utilizes the property of naming the physical group as ***string*** to get **gmESSI commands** from the user **written as a part of the Physical group's name** and then operates on the same physical group alongwith which the command was written to carry out the translation to ESSI comands. Below in Example1 which is a *Cantilever analysis*, it is shown how to **create physical groups** with their **string name as gmESSI Syntax** and then carry out translation with gmESSI translator using the **.msh** file produced. Later in Figure 5 how **gmESSI translator interpretes** these **Physical groups** in **.msh** file is described.

```
$ vim Example1.geo
......
Physical Surface ("$FixedSurface$ [Add_All_Node{m,3}] [Fix{all}]") = {26};
Physical Line("$ApplyForce$ [Add_Node_Load_Linear{Fx,10*kN}]") ={8};
Physical Volume("$CantileverVolume$ [Add_8NodeBrick{1}]") ={1};
```

```
$cat Example1.msh
......
$PhysicalNames
3
1 2 "$ApplyForce$ [Add_Node_Load_Linear{Fx,10*kN}]"
2 1 "$FixedSurface$ [Add_All_Node{m,3}] [Fix{all}]"
3 3 "$CantileverVolume$ [Add_8NodeBrick{1}]"
$EndPhysicalNames
......
```

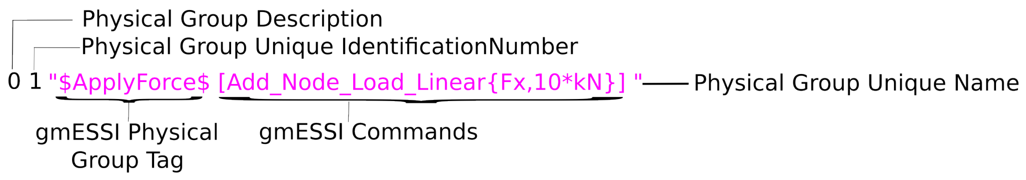

Physical Group Description
Physical Group Unique IdentificationNumber
0 1 "$ApplyForce$ [Add_Node_Load_Linear{Fx,10*kN}] "
Physical Group Unique Name
gmESSI Physical Group Tag
gmESSI Commands

Figure 5: Description showing how gmESSI interprets the Physical Groups.

For gmESSI translator, the whole **Physical Group Unique Name** is basically a combination of two parts as shown below.

- **gmESSI Physical Group Tag ::** As name describes it a tag given to the physical group by the user. It should be quoted between **$** signs. It is just for the user reference to have a description about the physical group.
- **gmESSI Commands ::** It is a list of commands that gmESSI translator understands. Each command is quoted between [ ]. *Multiple commands are separated by space.*

The gmESSI translator reads the command ***[Add_Node_Load_Linear{Fx,10*kN}]*** and applies nodal force **Fx=10*kN** to all the nodes of the physical group i.e (id **2** or **ApplyForce**) alongwith which the command was written.

Note:- For ***Gmsh*** `"$ApplyForce$ [Add_Node_Load_LinearFx,10*kN]"` is nothing but a name for the Physical group having id 2. Whereas, for ***gmESSI*** its a command for textitadding uniform nodal load to that physical group.

4 gmESSI Syntax

gmESSI Translator as said above utilizes the **naming of the physical groups** to get commands from the user and ***carry out the converion by acting on the same physical group elements and nodes***.

4.1 gmESSI Syntax

gmESSI follows strict syntax. gmESSI parses the **physical group name string** to find out the command, so *sticking to the syntax is very important* for it to understand the commands. Let us have a quick look at the syntax

4.1.1 Physical Group Names

Physical group names for gmESSI can follow two situations as described below

- **Only Physical Group Tag with no commands ::** As the name describes it refers to the situation in which, the user is only interested to make physical group for its convinience but do not need to have an gmESSI command associated with it for conversion. In that case the user can follow two syntax either "$Physical Group Tag$" or "Physical Group Tag"

- **Physical Group Tag with commands ::** When the user wants gmESSI command/s associated with the physical group for convesion, in that case the syantax is

"$Physical Group Tag$ [Command1{arg1,arg2}] [Command2{arg1,arg2,arg3}]"
~~"Physical Group Tag [Command1{arg1,arg2}] [Command2{arg1,arg2,arg3}]"~~

4.1.2 Physical Group Tags Syntax

Physical group tags can be any alphanumeric sequence but should not contain any of these [ ]$ literals in their names. The tags can have any spaces in between and must be enclod between **$** sign.

"$Physical Group Tag1$" "$PhysicalGroupTag2$" "$Physical_Group_Tag3$"
~~"$Physical [GroupTag3$"~~ ~~"$Physical GroupTag3"~~ ~~"$Physical$GroupTag3$"~~

4.1.3 gmESSI Command Syntax

gmESSI translator commands are always enclosed between opening/closing square brackets [and] respectively. A typical gmESSI command syntax is shown in Figure 6

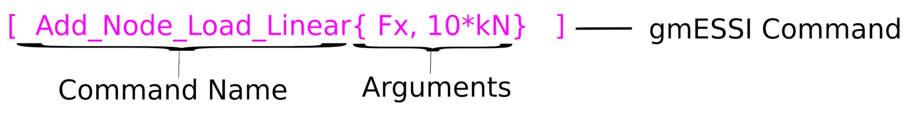


Figure 6: gmESSI command description

- **Command Name ::** Just as regular function gmESSI Commands have a name and take arguments. The names are usually self explanatory of its function like *Add_8NodeBrick{material}, Free{dof}, Var{variable,value} .. etc*

- **Arguments ::** Arguments as always are separated by **comma ','**.

- The arguments of gmESSI commands can also have tags associated with them so that it becomes easy for the user to interpret the argument and make changes in future. The tag and the argument is separaed by **:=**. Tag itself has no meaning but it serves as an important information center for user. An example is shown below to show how tags are applied.

  – gmESSI command having arguments without tags

    1. [Add_Node_Load_Linear{Fx,10*kN}]

  – gmESSI command having arguments with tags

    1. [Add_Node_Load_Linear{ direction:= Fx, magnitude:= 10*kN}]
    2. [Add_Node_Load_Linear{ direction:= Fx, 10*kN}]
    3. [Add_Node_Load_Linear{ Force_Direction:= Fx, Strength:=10*kN}]

  It can be seen from above examples that the tags are optional and also the user can put their own tag names. The sublime plugin **gmESSI-gmsh Tools** comes with elaborative tags for the parameters and a lot more syntax-colouring and text-completionfor gmESSI commands. Users are encouraged to use the plugin and take its advantage.

- Most of the time these **arguments are dummy** which means that they just get copied to their **equivalent ESSI command** at *their respective places.* These arguments thus have a **string** datatype. For example: the command Add_Node_Load_Linear{Fx,10*kN} is equivalent to the ESSI command add load #{} to node #{} type linear {} = {}. Fx and 10*kN goes to their respective position directly through the translater as shown in the Figure 7 **Load number 1** and **node number 32** are computed by the translator and then inserted in the ESSI command.

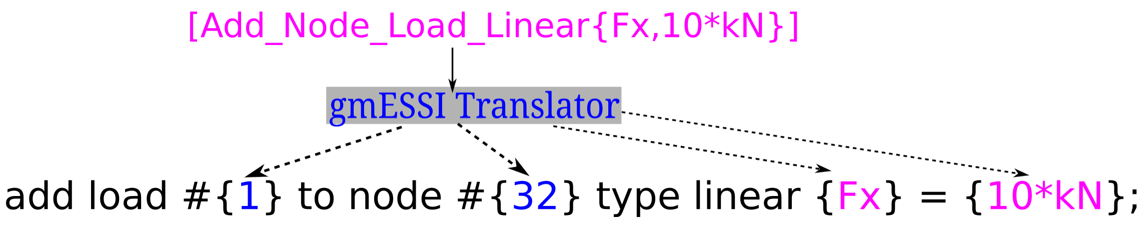

[Add_Node_Load_Linear{Fx,10*kN}]
gmESSI Translator
add load #{1} to node #{32} type linear {Fx} = {10*kN};

Figure 7: gmESSI conversion description

Note:- gmESSI Translator *does not provide syntax checking* for **those dummy arguments**. It means that, whatever you write it gets copied at the respective position in the equivalent ESSI command, so the user must be carefull with what they are writing in these arguments. For Example :- the command Add_Node_Load_Linear{ ForceDirection, Magnitude} based on the arguments can get converted to

1. Add_Node_Load_Linear{Fx,10*kN} $--->$ add load #1 to node #32 type linear Fx = 10*kN
2. Add_Node_Load_Linear{Fx,10} $--->$ add load #1 to node #32 type linear Fx = 10
3. Add_Node_Load_Linear{ft,10*kN} $--->$ add load #1 to node #32 type linear ft = 10*kN

All the above conversions are correct. But conversion **1)** is correct for **ESSI** because **ForceDirection** is one of *Fx,Fy,Fz* and **Magnitude 10*kN** has *proper units*. So the user must be very careful with what they are writing in the arguments.

- Some of the arguments *are not string but represent numerical quantities*, which are manupulated by the translator during conversion. Thus, the user must supply only numbers as strings without any alphabets else it would lead an unexpected termination of program. These arguments corresponds to **Special Commands** such as Connect Command and Material Variational Commands. The manual talks about them later.

4.2 Command's Physical/Entities Group

As stated somewhere else, **gmESSI commands** operates on *physical groups. The physical groups* on which it operates is the one alongwith which the command is written. But **gmESSI** provides a more flexible approach for these commands so *that they can be written anywhere and can* ***operate on any Physical group as well as Entity group***. Let us look at them one by one

- **Default Physical Group to gmESSI Commands ::** The default physical group on which the gmESSI commands operate is the one along with which it is written in the **.msh** file. Let's look at some of them
  - 1 2 "$ApplyForce$ [Add_Node_Load_Linear{Fx,10*kN}]" operates on physical group 2
  - 3 5 "$ApplyForce$ [Add_8NodeBrick{material#1}]" operates on physical group 5
- **Userdefined Physical and Entity Group to gmESSI Commands ::** The flexibility of gmESSI syntax allows the users to operates it's command on specific physical groups or geometric entities. This user specifies the group by including an additional argument Physical_group# or Entity_Group#Tag in front of the gmESSI commands describing the group. The **tag** can be either *Physical_Group_Id, Physical_Group_Name*. Let's look at some of them
  - 1 2 "$ApplyForce$ [Add_Node_Load_Linear{Physical_Group#3,Fx,10*kN}]" operates on physical group 3

- 1 2 "$ApplyForce$ Add_Node_Load_Linear{Entity_Group#5,Fx,10*kN}]" operates on entity gr oup 5
- 3 5 "$ApplyForce$ [Add_8NodeBrick{Entity_Group#9,1}]" operates on entity group 9
- 3 5 "$ApplyForce$ [Add_8NodeBrick{Physical_Group#CantileverVolume, 1}]" operates on physical group which has string_tag as CantileverVolume.

For example in reference to Example1.msh Physical_Group#CantileverVolume or Physical_Group#3 refers the same physical group

A **Entity group** is the same as **Physical group** but contains all the geometrical entities that have the same unique id. While **Physical group** contains many geometrical entities but of only one entity type, **Entity group** can contain **mixture of entity types** but **for each type maximum upto one geometrical entity**. For example:- Entity_Group#5 contains *all elemenntary geometrical nodes, lines, surface and volumes that have unique id as 5 and* ***are present in mesh generated .msh*** *file.* In Figure 1 *Point 1, Line 1 and Volume 1* belongs to **Entity Group** having id 1.

Whereas a **physical group** is a group of point, line, surface or volume defined by the user which contains all the geometrical entities that falls under that domain/group. Figure 2 shows physical groups.

Note:- When the user *explicitly* defines a *physical/entity group* for the command, the command now **can be written anywhere** alongwith any physical group in the **.msh** file.

Note:- Whenever one is applying an **General Elemental Command** or **Nodal Command** on an **entity group**, the user should be very carefully beacuse it might contain nodes from different elements and more than 1 type of elements respectively. Typically, when the meshes are created by gmsh automatically, the chances of this kind of conflicts are very less. But still user must be carefull while using Entity Group.

5 gmESSI Output

gmESSI Translator translates the commands of **.msh** file to different **.fei** files and put them in a folder. It also updates the mesh **.msh** file and puts it in the same folder. The **log of translation, errors and warnings** are displayed on the terminal. Let us demostrate each of them one by one using **Example1.msh** and a *general meshfile name* **XYZ.msh**.

5.1 Folder *XYZ_ESSI_Simulation*

gmESSI Translator creates a folder at the place of execution. All the files are written in that folder only. Typically the name of the folder created for **XYZ.msh** file is XYZ_ESSI_Simulation. In case the folder allready exists a warning message is shown on the terminal.

```
$ gmESSI Example1.msh

Files will be converted to Example/Example1_ESSI_Simulation

WARNING::Directory Allready Present. The contents of the Folder may get changed

Files converted to Example/Example1_ESSI_Simulation
```

By default the translation is allways done in the overwrite mode meaning the previous existing files/folder if any would get changed and updated. But the user can turn off the mode by **non-overwrite mode** option **-o** to create a new non-conflicting folder XYZ_ESSI_Simulation_n where n refers to 1,2,3,...

```
$ gmESSI −o Example1.msh

Files converted to Example/Example1_ESSI_Simulation_1
```

The execution of gmESSI XYZ.msh produces warnings/errors in the following situtaions.

- ERROR:: Please Enter the Gmsh File :: It occurs if the user does not give a filename. The possible situtaion for getting this error is gmESSI and gmESSI -o XYZ.msh
- ERROR:: The program failed to open the file XYZ.msh It occurs *if the given file or one of the files in the argument does not exist or fails to open* because of some reason. Examples:- gmESSI Xyz.msh and gmESSI - XYZ.msh.
- WARNING::Directory Allready Present. The contents of the Folder may get changed :: It occurs *when users translates its file XYZ.msh in overwrite mode and the corresponding folder XYZ_ESSI_Simulation allready exists at the execution location.*
- Files converted to Example/Example1_ESSI_Simulation :: The message refers to the location of the folder where the translations have been saved.

5.2 Translation Log *Terminal*

gmESSI Translator displays the log of translation of **gmESSI commands** to **ESSI commands** on the terminal. Proper Errors Messages and Warnings are **echoed** to the user. The **execution of the commands** are **sequential** which means the commands written first are executed first and similarly their success and failure is also echoed first. Let us look at this aspect with Example1.msh.

```
$cat Example1.msh
......
$PhysicalNames
3
1 2 "$ApplyForce$ [Add_Node_Load_Linear{Fx,10*kN}]"
2 1 "$FixedSurface$ [Add_All_Node{m,3}] [Fix{all}]"
3 3 "$CantileverVolume$ [Add_8NodeBrick{1}]"
$EndPhysicalNames
......
```

Here, the sequence of execution of commands is [Add_Node_Load_Linear{Fx,10*kN}] , [Add_All_Node{m,3}] , [Fix{all}] and [Add_8NodeBrick{1}]. Notice that the same order gets reflected in the translation log on the terminal as shown below.

```
$ gmessi ./Example1.msh
......
Add_Node_Load_Linear{Fx,10*kN} Found!! Sucessfully Converted
Add_All_Node{m,3} Found!! Sucessfully Converted
Fix{all} Found!! Sucessfully Converted
Add_8NodeBrick{1} Found!! Sucessfully Converted
......
```

Apart from the **translation log details** on the terminal in the front of user, similar kind of log gets added for each translation of **gmESSI commands** in their respective files in which they are translated. In these files **each sucessfull translation** is enclosed between corresponding **RespectiveUserCommand Begins** and **RespectiveUserCommand Ends**. The same is shown below through the contents of Example1_geometry.fei. Notice that all the translations are enclosed between **Begins and Ends Tag**.

```
$ cat Example1_geometry.fei
//*****************************************************************
// AddAllNode{m,3}Begins
//*****************************************************************

add node # 1 at (0.000000*m ,0.000000*m ,0.000000*m ) with 3 dofs;
add node # 2 at (1.000000*m ,0.000000*m ,0.000000*m ) with 3 dofs;
add node # 3 at (0.000000*m ,1.000000*m ,0.000000*m ) with 3 dofs;
.................................................................

//*****************************************************************
// AddAllNode{m,3}Ends
//*****************************************************************
```

Note:- The textbftextitordering/sequence of commands in ESSI analysis file is important and so the user must make sure that the translations are made in the same order or if not the user should change it **manually by (cut/copy/paste)** in textbfgeometry.fei, load.fei and analysis.fei files before execution.

Having given a short descripton of the Translation log, let us look more closely one by one and understand the messages, errors and warnings promted in the log on the terminal.

- Found!! :: This message in front of the gmESSI command as shown above on translation log in the terminal means that, *the corresponding command was found in the* ***gmESSI Command Library***.

- Sucessfully Converted :: As the message itself describes, it occurs *if the command has been sucessfully translated.*

- Not Found!! :: It occurs if the gmESSI Translator *could not find the arbitraty command XYZ in the **gmESSI Command library***. Example:- **Loading{Fx,10*kN} NotFound!!**

- WARNING:: Execuation of the command escaped. The Gssi command XYZ could not be found :: The gmESSI Translator does not terminate the translation if a command is not found, instead gives this warning message following the Not Found!! Error.

- Error:: The command XYZ has a syntaxERROR in Entity_Group/Physical_Group# tag :: It occurs if there is a syntax error in Entity_Group/Physical_Group# argument. The correct represention for **Physical and Entity group Tags** is Physical_Group#n and Entity_Group#n respectively, where n is the **group id as 1,2,3..**. Example of improper representation are Phy#2, Physical#2, Ent#2, Entiti#2 ..

- Warning:: The command XYZ failed to convert as there is no such Physical/Entity Group :: It occurs if one of the arguments in the command is Enty/Physical_Group# and *the specified entity or physical group by the user does not exists in the **.msh** file.*

- Warning:: The command XYZ could not find any nodes/elements on which it operates :: It occurs *if for a specified command, the required element types for translation could not be found in the specified Physical/Entity group.* For Examples:- [8NodeBrick{Physical_Group#1,1}] would give this warning as the Physical_Group#1 being a Physical line group **does not contain any 8-Noded Brick elements on which this command operates**.

- ERROR:: Gmsh File has invalid symbols in Node Section. Unable to convert string to integer in Gmsh File :: It occurs *if there is perhaps a string inside the Nodes section of **.msh** file.*

- ERROR:: The command XYZ has a syntax errors :: It occurs *if the specified command by the user contain any stntax errors caught while parsing the command.*

- ERROR:: Gmsh File has invalid symbols in Element Section. Unable to convert string to integer in Gmsh File :: It occurs *if there is perhaps a string inside the Element section of **.msh** file.*

5.3 Geometric File ***XYZ_geometry.fei***

Geometric file *XYZ_geometric.fei* is one of three parts of ESSI Analysis file and it contains the translation of commands related to the geometry of the structure, like ***declaration of nodes and elements***.

5.4 Load File *XYZ_load.fei*

Load file *XYZ_load.fei* contains the translation of commands related to the load and boundary conditions on the structure, like ***declaration of fixities, boundary conditions, master slave, nodal loads, surface loads etc...***.

5.5 Analysis File *XYZ_analysis.fei*

Analysis file *XYZ_analysis.fei* is the main file which is run on ESSI. The main file ***includes load file and geometry file*** through textbfinclude command. A typical analysis file after conversion looks like the following.

```
$ cat Example1_analysis.fei

  model name "Example1.msh";

  include "/Examples/Example1_ESSI_Simulation/Example1_geometry.fei";

  new loading stage "Stage_1 Loading";

  include "/Examples/Example1_ESSI_Simulation/Example1_load.fei";
```

The user can now add solver, time steps and even rearrange the file structure accordingly to **ESSI synatx**. Please not that *ESSI Interpreter* is sequential and *follows certain ordering in commands*, **like materials should be declared before assigning to elements, master-slave can be assigned only when both nodes are declared .. etc.**. The user should be carefull with the order in which conversions are made and if necessary should **change it manually by (cut/copy/paste)** later in the **files geometry.fei, load.fei and analysis.fei** or use the python module discussed later before running in ESSI. Please refer to the ESSI manual for more details on the ordering of the commands.

5.6 Mesh File *XYZ.msh*

Mesh file *XYZ.msh* for most of times is the input XYZ.msh file. But if **textcolorvioletConnect Command** is used, the file contains textbftextitadditional physical group, nodes and 2-noded elements. The textbftextcolorvioletConnect Command is discussed in the more detail later in this manual.

5.7 Updated ESSI Tags *Terminal*

Updated ESSI Tags refers to the new Tag numbering reference associated with ESSI Tags. textbfESSI has tag numberings associated for damping, displacement, element, field, load, material, motion and node/nodes. For example in textbfESSI Command **textcolorvioletadd node # 1 at (x,y,z) with 3 dofs**, **node** is an **Tag** and requires a new number textitike 1 to be associated with that node. **gmESSI translator** displays the textitnew numberings available for each ESSI Tag so that the user is made aware of new numberings while manually specifying an ESSI command after the translation.

```
$ gmessi Example1.msh
  .....................
```

```
******* Updated New Tag Numberring ********
damping = 1
displacement = 1
element = 27
field = 1
load = 4
material = 2
motion = 1
node = 55
nodes = 55
```

6 gmESSI Commands Library

Having the knowledge about the syntax, output files, errors and warnings, its time to move on to different types of commands that gmESSI offers. **gmESSI Translator** offers conversion for all the commands of **ESSI**. *The equivalent ESSI command to which gmESSI command translates is also shown in forthcomming sections.* Apart from that there are some special command that gmESSI supports. Just for simplicity the commands are categorised on *the basis of their operation on nodes/elemnts.* As stated earlier, the commands are translated to one of three file **XYZ_geometry.fei, XYZ_load.fei and XYZ_analysis.fei**. Let us look at them closely one by one alongwith all its supported commands.

6.1 Nodal Commands

Nodal commands operates on all the textitnodes of a physical/entity group. Their translations are written in **XYZ_load.fei** file. For example:- [Fix{Physical_Group#1,ux}] would fix ux dof for all the nodes of physical group 1. The different commands under this category and their corresponding ESSI commands are

- gmESSI :: [Add_Node_Mass{mx,my,mz}]
  ESSI :: add mass to node #{} mx ={} my ={} mz ={}
  Description :: $add\ mass\ to\ node\ \# < . > mx = < mass > my = < mass > mz = < mass >$

- gmESSI :: [Add_Node_Mass{mx,my,mz,Imx,Imy,Imz}]
  ESSI :: add mass to node #{} mx ={} my ={} mz ={} Imx ={} Imy ={} Imz ={}
  Description :: $add\ mass\ to\ node\ \# < . > mx = < mass > my = < mass > mz = < mass > Imz = < mass * length^2 > Imy = < mass * length^2 > Imz = < mass * length^2 >$

- gmESSI :: [Add_Node_Damping{damping_no}]
  ESSI :: add damping #{} to node #{}
  Description :: $add\ damping\ \# < . > to\ node\ \# < . >$

- gmESSI :: [Fix{dofs}]
  ESSI :: fix node #{} dofs {}
  Description :: *add damping # < . > to node # < . >*

- gmESSI :: [Free{dofs}]
  ESSI :: free node #{} dofs {}
  Description :: *free node # < . > dofs < . >*

- gmESSI :: [Add_Master_Slave{master_node, dof}]
  ESSI :: add constraint equal dof with master node #{} and slave node #{} dof to constrain {}
  Description :: *add constraint equal dof with master node # < . > and slave node # < . > dof to constrain < . >*

- gmESSI :: [Remove_Master_Slave{}]
  ESSI :: remove constraint equal dof node #{}
  Description :: *remove constraint equaldof node # < . >*

- gmESSI :: [Add_Single_Point_Constraint{constrain_dof, constrain_value}]
  ESSI :: add single point constraint to node #{} dof to constrain {} constraint value of {}
  Description :: *add single point constraint to node # < . > dof to constrain < dof_type > constraint value of < correspondingunit >;*

- gmESSI :: [Add_Node_Load_Linear{direction, magnitude}]
  ESSI :: add load #{} to node #{} type linear {}= {}
  Description :: *add load # < . > to node # < . > type linear FORCETYPE = < force|moment > //FORCETYPE = Fx Fy Fz Mx My Mz F_fluid_x F_fluid_y F_fluid_z*

- gmESSI :: [Add_Node_Load_Path_Series{direction, magnitude, series_file}]
  ESSI :: add load #{} to node #{} type path_series {} = {} series_file ={}
  Description :: *add load # < . > to node # < . > type path_series FORCETYPE = < forceormoment > time_step = < time > series_file = "STRING"*

- gmESSI :: [Add_Node_Load_Path_Time_Series{direction, magnitude, time_step, series_file}]
  ESSI :: add load #{} to node #{} type path_time_series {} = {} time_step = {} series_file ={}
  Description :: *add load # < . > to node # < . > type path_time_series FORCETYPE = < forceormoment > series_file = "STRING"*

- gmESSI :: [Add_Imposed_Motion_Time_Series{dof, time_step, disp_scale_unit, disp_file, vel_scale_unit, vel_file, acc_scale_unit, acc_file}]
  ESSI :: add imposed motion #{} to node #{} dof {} time_step

= {} displacement_scale_unit = {} displacement_file = {} velocity_scale_unit = {} velocity_file = {} acceleration_scale_unit = {} acceleration_file ={}
Description :: *add imposed motion # < . > to node # < . > dof DOFTYPE time_step = < t > displacement_scale_unit = < length > displacement_file = "filename" velocity_scale_unit = < velocity > velocity_file = "filename" acceleration_scale_unit = < acceleration > acceleration_file = "filename";*

- gmESSI :: [Add_Imposed_Motion{dof, disp_scale_unit, disp_file, vel_scale_unit, vel_file, acc_scale_unit, acc_file}]
  ESSI :: add imposed motion #{} to node #{} dof {} displacement_scale_unit = {} displacement_file = {} velocity_scale_unit = {} velocity_file = {} acceleration_scale_unit = {} acceleration_file ={}
  Description :: *add imposed motion # < . > to node # < . > dof DOFTYPE displacement_scale_unit = < length > displacement_file = "filename" velocity_scale_unit = < velocity > velocity_file = "filename" acceleration_scale_unit = < acceleration > acceleration_file = "filename";*

- gmESSI :: [Print_Node{}]
  ESSI :: print node #{}
  Description :: *print node # < . >*

- gmESSI :: [Add_Node_Load_From_Reaction{}]
  ESSI :: add load #{} to node #{} from_reactions
  Description :: *add load # < . > to node # < . > from_reactions*

- gmESSI :: [Add_Node_Self_Weight{field_no}]
  ESSI :: add load #{} to node #{} type self_weight use acceleration field #{}
  Description :: *add load # < . > to node # < . > type self_weight use acceleration field # < . >*

Note:- As earlier said, that **the arguments of gmESSI commands are dummy and gets copied directly to the ESSI equivant command**, so the user must be very much attentive/aware to what he is writing. *The arguments should be filled with values of the corresponding ESSI command along with required units if any.* **For more details about the values to the arguments, please refer to ESSI Manual**.
Note:- Every Nodal command has the option of adding Physical_Group# or Entity_Group# argument in the front.

6.2 General Elemental Commands

General Elemental Commands operates on all the *elements of a physical/entity group*. Their translations are written in **XYZ_load.fei** file. For example:-

[Add_SelfWeight{Physical_Group#1,1}] would add self-weight along the field#1 direction on all the elemnets of physical group 1. The different commands under this category and their corresponding ESSI commands are

- gmESSI :: [Add_Element_Self_Weight{acc_field_no}]
  ESSI :: add load #{} to element #{} type self_weight use acceleration field #{}
  Description :: *add load # < . > to element # < . > type self_weight use acceleration field # < . >*

- gmESSI :: [Add_Element_Damping{damping_no}]
  ESSI :: add damping #{} to element #{}
  Description :: *add damping # < . > to element # < . >*

- gmESSI :: [Check_Element{}]
  ESSI :: check element #{}
  Description :: *check element # < . >*

- gmESSI :: [Print_Element{}]
  ESSI :: print element #{}
  Description :: *print element # < . >*

Note:- Every General Elemental command has the option of adding Physical_Group# or Entity_Group# argument in the front.

6.3 Elemental Commands

Elemental Commands operates only to textit specific elements of a physical/entity group. Their translations are written in **XYZ_geometry.fei** file. For example:- [Add_8NodeBrick{Physical_Group#1,1}] textcolorviolet would initialize only 8 noded bricks with material 1 of the physical group 1. The different commands under this category, the element types on which they operate and their corresponding ESSI commands are shown below.
Note:- Every Elemental command has the option of adding Physical_Group# or Entity_Group# argument in the front.

6.3.1 2-Noded Line Elements

- gmESSI :: [Add_Beam_Elastic_Lumped_Mass{ cross_section, elastic_modulus, shear_modulus, torsion_Jx, bending_Iy, bending_Iz, mass_density, xz_plane_vector_x, xz_plane_vector_y, xz_plane_vector_z, joint_1_offset_1, joint_1_offset_2, joint_1_offset_3, joint_2_offset_1, joint_2_offset_2, joint_2_offset_3}]
  ESSI :: add element #{} type beam_elastic_lumped_mass with nodes ({},{}) cross_section={} elastic_modulus={} shear_modulus={}

torsion_Jx={} bending_Iy={} bending_Iz={} mass_density={} xz_plane_vector = ({}, {}, {}) joint_1_offset = ({}, {}, {}) joint_2_offset= ({}, {}, {})
Description :: *add element # < . > type beam_elastic_lumped_mass with nodes (< . >, < . >) cross_section = < area > elastic_modulus = $< F/L^2 >$ shear_modulus = $< F/L^2 >$ torsion_Jx = $< length^4 >$ bending_Iy = $< length^4 >$ bending_Iz = $< length^4 >$ mass_density = $< M/L^3 >$ xz_plane_vector = (< . >, < . >, < . >) joint_1_offset = (< L >, < L >, < L >) joint_2_offset = (< L >, < L >, < L >)*

- gmESSI :: [Add_Truss{material_no,cross_section, mass_density}]
  ESSI :: add element #{} type truss with nodes ({},{}) use material #{} cross_section ={} mass_density={}
  Description :: *add element # < . > type truss with nodes (< . >, < . >) use material # < . > cross_section = $< length^2 >$ mass_density = $< M/L^3 >$*

- gmESSI :: [Add_Frictional_Penalty_Contact{normal_stiffness, tangential_stiffness, normal_damping, tangential_damping, friction_ratio, contact_plane_vector_x, contact_plane_vector_y, contact_plane_vector_z}]
  ESSI :: add element #{} type FrictionalPenaltyContact with nodes ({}, {}) normal_stiffness={} tangential_stiffness={} normal_damping={} tangential_damping={} friction_ratio={} contact_plane_vector= ({}, {}, {})
  Description :: *add element # < . > type FrictionalPenaltyContact with nodes (< . >, < . >) normal_stiffness = < F/L > tangential_stiffness = < F/L > normal_damping = < F/L > tangential_damping = < F/L > friction_ratio = < . > contact_plane_vector = (< . >, < . >, < . >)*

- gmESSI :: [Add_Beam_Elastic{ cross_section,elastic_modulus, shear_modulus, torsion_Jx, bending_Iy, bending_Iz, mass_density, xz_plane_vector_x, xz_plane_vector_y, xz_plane_vector_z, joint_1_offset_1, joint_1_offset_2, joint_1_offset_3 , joint_2_offset_1, joint_2_offset_2, joint_2_offset_3}]
  ESSI :: add element #{} type beam_elastic with nodes ({},{}) cross_section= {} elastic_modulus= {} shear_modulus={} torsion_Jx={} bending_Iy={} bending_Iz={} mass_density={} xz_plane_vector= ({}, {}, {}) joint_1_offset= ({}, {}, {}) joint_2_offset= ({}, {}, {})
  Description :: *add element # < . > type beam_elastic with nodes (< . >, < . >) cross_section = < area > elastic_modulus = $< F/L^2 >$ shear_modulus = $< F/L^2 >$ torsion_Jx = $< length^4 >$ bending_Iy = $< length^4 >$ bending_Iz = $< length^4 >$ mass_density = $< M/L^3 >$ xz_plane_vector = (< . >, < . >, < . >) joint_1_offset = (< L >, < L >, < L >) joint_2_offset = (< L >, < L >, < L >)*

- gmESSI :: [Add_Beam_Displacement_Based{ material_no, section_no,

mass_density, integration_rule, xz_plane_vector_x, xz_plane_vector_y, xz_plane_vector_z, joint_1_offset_1, joint_1_offset_2, joint_1_offset_3, joint_2_offset_1, joint_2_offset_2, joint_2_offset_3}]
ESSI :: add element #{} type beam_displacement_based with nodes ({}, {}) with #{} integration_points use section #{} mass_density={} IntegrationRule={} xz_plane_vector=({}, {}, {}) joint_1_offset= ({}, {}, {}) joint_2_offset= ({}, {}, {})
Description :: *add element # < . > type beam_displacement_based with nodes (< . >, < . >) with # < . > integration_points use section # < . > mass_density* $= < M/L^3 >$ *IntegrationRule = "" xz_plane_vector = (< . >, < . >, < . >) joint_1_offset = (< L >, < L >, < L >) joint_2_offset = (< L >, < L >, < L >)*

- gmESSI :: [Add_Beam_9dof_Elastic{ cross_section, elastic_modulus, shear_modulus, torsion_Jx, bending_Iy, bending_Iz, mass_density, xz_plane_vector_x, xz_plane_vector_y, xz_plane_vector_z, joint_1_offset_1, joint_1_offset_2, joint_1_offset_3, joint_2_offset_1, joint_2_offset_2, joint_2_offset_3}]
ESSI :: add element #{} type beam_9dof_elastic with nodes ({}, {}) cross_section={} elastic_modulus={} shear_modulus={} torsion_Jx={} bending_Iy={} bending_Iz={} mass_density={} xz_plane_vector=({}, {}, {}) joint_1_offset= ({}, {}, {}) joint_2_offset= ({}, {}, {})
Description :: *add element # < . > type beam_9dof_elastic with nodes (< . >, < . >) cross_section = < area > elastic_modulus* $= < F/L^2 >$ *shear_modulus* $= < F/L^2 >$ *torsion_Jx* $= < length^4 >$ *bending_Iy* $= < length^4 >$ *bending_Iz* $= < length^4 >$ *mass_density* $= < M/L^3 >$ *xz_plane_vector = (< . >, < . >, < . >) joint_1_offset = (< L >, < L >, < L >) joint_2_offset = (< L >, < L >, < L >)*

- gmESSI :: [Add_ShearBeamLT{ cross_section, material_no}]
ESSI :: add element #{} type ShearBeamLT with nodes ({},{}) cross_section={} use material #{}
Description :: *add element # < . > type ShearBeamLT with nodes (< . >, < . >) cross_section* $= < l^2 >$ *use material # < . >*

6.3.2 3-Noded Traingular Elements

- gmESSI :: [Add_3NodeShell_ANDES{material_no, thickness}]
ESSI :: add element #{} type 3NodeShell_ANDES with nodes ({},{},{}) use material #{} thickness= {}
Description :: *add element # < . > type 3NodeShell_ANDES with nodes (< . >, < . >, < . >) use material # < . > thickness = < l >*

6.3.3 4-Noded Quadrangle Elements

- gmESSI :: [Add_4NodeShell_ANDES{material_no, thickness}]
  ESSI :: add element #{} type 4NodeShell_ANDES with nodes ({}, {}, {}, {}) use material #{} thickness= {}
  Description :: *add element # < . > type 4NodeShell_ANDES with nodes (< . >, < . >, < . >, < . >) use material # < . > thickness = < l >*

- gmESSI :: [Add_4NodeShell_MITC4{material_no, thickness}]
  ESSI :: add element #{} type 4NodeShell_MITC4 with nodes ({}, {}, {}, {}) use material #{} thickness= {}
  Description :: *add element # < . > type 4NodeShell_MITC4 with nodes (< . >, < . >, < . >, < . >) use material # < . > thickness = < L >*

- gmESSI :: [Add_4NodeShell_NewMITC4{material_no, thickness}]
  ESSI :: add element #{} type 4NodeShell_NewMITC4 with nodes ({}, {}, {}, {}) use material #{} thickness= {}
  Description :: *add element # < . > type 4NodeShell_NewMITC4 with nodes (< . >, < . >, < . >, < . >) use material # < . > thickness = < L >*

6.3.4 8-Noded Hexahedron Elements

- gmESSI ::[Add_8NodeBrick{material_no}]
  ESSI :: add element #{} type 8NodeBrick with nodes ({}, {}, {}, {}, {}, {}, {}, {}) use material #{}
  Description :: *add element # < . > type 8NodeBrick with nodes (< . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >) use material # < . >*

- gmESSI :: [Add_8NodeBrick_gauss_points{material_no}]
  ESSI :: add element #{} type 8NodeBrick using Gauss points each direction with nodes ({}, {}, {}, {}, {}, {}, {}, {}) use material #{}
  Description :: *add element # < . > type 8NodeBrick Gauss points each direction with nodes (< . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >) use material # < . >*

- gmESSI :: [Add_8NodeBrickLT{material_no}]
  ESSI :: add element #{} type 8NodeBrickLT with nodes ({}, {}, {}, {}, {}, {}, {}, {}) use material #{}
  Description :: *add element # < . > type 8NodeBrickLT with nodes (< . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >) use material # < . >*

- gmESSI :: [Add_8NodeBrick_elastic{material_no}]
  ESSI :: add element #{} type 8NodeBrick_elastic with nodes ({}, {}, {}, {}, {}, {}, {}, {}) use material #{}
  Description :: *add element # < . > type 8NodeBrick_elastic with nodes (< . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >) use material # < . >*

- gmESSI :: [Add_8NodeBrick_up{material_no}]
  ESSI :: add element #{} type 8NodeBrick_up with nodes ({}, {}, {}, {}, {}, {}, {}, {}) use material #{} porosity = {} alpha = {} rho_s = {} rho_f = {} k_x = {} k_y = {} k_z = {} K_s = {} K_f= {}
  Description :: *add element # < . > type 8NodeBrick_up with nodes (< . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >) use material # < . > porosity = < . > alpha = < . > rho_s =* $< M/L^3 >$ *rho_f =* $< M/L^3 >$ *k_x =* $< L^3T/M >$ *k_y =* $< L^3T/M >$ *k_z =* $< L^3T/M >$ *K_s = < stress > K_f = < stress >*

- gmESSI :: [Add_8NodeBrick_upU{material_no}]
  ESSI :: add element #{} type 8NodeBrick_upU with nodes ({}, {}, {}, {}, {}, {}, {}, {}) use material #{} porosity = {} alpha = {} rho_s = {} rho_f = {} k_x = {} k_y = {} k_z = {} K_s = {} K_f= {}
  Description :: *add element # < . > type 8NodeBrick_upU with nodes (< . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >) use material # < . > porosity = < . > alpha = < . > rho_s =* $< M/L^3 >$ *rho_f =* $< M/L^3 >$ *k_x =* $< L^3T/M >$ *k_y =* $< L^3T/M >$ *k_z =* $< L^3T/M >$ *K_s = < stress > K_f = < stress >*

- gmESSI ::[Add_Variable_8-27NodeBrick{material_no}]
  ESSI :: add element #{} type variable_node_brick_8_to_27 using Gauss points each direction with nodes ({}, {}, {}, {}, {}, {}, {}, {}) use material #{}
  Description :: *add element # < . > type variable_node_brick_8_to_27 Gauss points each direction with nodes (< . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >) use material # < . >*

6.3.5 20-Noded Hexahedron Elements

- gmESSI :: [Add_20NodeBrick{material_no}]
  ESSI :: add element #{} type 20NodeBrick with nodes ({}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}) use material #{}
  Description :: *add element # < . > type 20NodeBrick with nodes (< . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >) use material # < . >*

- gmESSI :: [Add_20NodeBrick_gauss_points{material_no}]
  ESSI :: add element #{} type 20NodeBrick using Gauss points each direction with nodes ({}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}) use material #{}
  Description :: *add element # < . > type 20NodeBrick Gauss points each direction with nodes (< . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >) use material # < . >*

- gmESSI :: [Add_20NodeBrick_elastic{material_no}]
  ESSI :: add element #{} type 20NodeBrick_elastic with nodes ({}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}) use material #{}
  Description :: *add element # < . > type 20NodeBrick_elastic with nodes (< . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >) use material # < . >*

- gmESSI :: [Add_20NodeBrick_upU{material_no}]
  ESSI :: add element #{} type 20NodeBrick_upU with nodes ({}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}) use material #{} porosity = {} alpha = {} rho_s = {} rho_f = {} k_x = {} k_y = {} k_z = {} K_s = {} K_f= {}
  Description :: *add element # < . > type 20NodeBrick_upU with nodes (< . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >) use material # < . > and porosity = < . > alpha = < . > rho_s =* $< M/L^3 >$ *rho_f =* $< M/L^3 >$ *k_x =* $< L^3T/M >$ *k_y =* $< L^3T/M >$ *k_z =* $< L^3T/M >$ *K_s = < stress > K_f = < stress >*

- gmESSI ::[Add_Variable_8-27NodeBrick{material_no}]
  ESSI :: add element #{} type variable_node_brick_8_to_27 using Gauss points each direction with nodes ({}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}) use material #{}
  Description :: *add element # < . > type variable_node_brick_8_to_27 Gauss points each direction with nodes (< . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >) use material # < . >*

6.3.6 27-Noded Hexahedron Elements

- gmESSI :: [Add_27NodeBrick{material_no}]
  ESSI :: add element #{} type 27NodeBrick with nodes ({}, {}) use material #{}
  Description :: *add element # < . > type 27NodeBrick with nodes (< . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >) use material # < . >*

- gmESSI :: [Add_27NodeBrick_gauss_points{material_no}]
  ESSI :: add element #{} type 27NodeBrick using Gauss points each direction with nodes ({}, {}) use material #{}
  Description :: *add element # < . > type 27NodeBrick Gauss points each direction*

with nodes (< . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >) use material # < . >

- gmESSI :: [Add_27NodeBrickLT{material_no}]
  ESSI :: add element #{} type 27NodeBrickLT with nodes ({}, {}) use material #{}
  Description :: *add element # < . > type 27NodeBrickLT with nodes (< . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >) use material # < . >*

- gmESSI :: [Add_27NodeBrick_elastic{material_no}]
  ESSI :: add element #{} type 27NodeBrick_elastic with nodes ({}, {}) use material #{}
  Description :: *add element # < . > type 27NodeBrick_elastic with nodes (< . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >) use material # < . >*

- gmESSI ::[Add_Variable_8-27NodeBrick{material_no}]
  ESSI :: add element #{} type variable_node_brick_8_to_27 using Gauss points each direction with nodes ({}, {}) use material #{}
  Description :: *add element # < . > type variable_node_brick_8_to_27 Gauss points each direction with nodes (< . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >) use material # < . >*

6.3.7 Surface Loads

- gmESSI :: [Add_8NodeBrick_SurfaceLoad{PhyEntSurfaceTag,m1}]
  ESSI :: add load #{} to element #{} type surface at nodes ({}, {}, {}, {}) with magnitude ({})
  Description :: *add load # < . > to element # < . > type surface at nodes (< . > , < . > , < . > , < . >) with magnitudes (< . >)*

- gmESSI :: [Add_8NodeBrick_SurfaceLoad{PhyEntSurfaceTag, m1, m2, m3, m4}]
  ESSI :: add load #{} to element #{} type surface at nodes ({}, {}, {}, {}) with magnitudes ({}, {}, {}, {})
  Description :: *add load # < . > to element # < . > type surface at nodes (< . > , < . > , < . > , < . >) with magnitudes (< . > , < . > , < . > , < . >)*

- gmESSI :: [Add_20NodeBrick_SurfaceLoad{PhyEntSurfaceTag,magnitude}]
  ESSI :: add load #{} to element #{} type surface at nodes ({}, {}, {}, {}, {}, {}, {}, {}) with magnitude ({})
  Description :: *add load # < . > to element # < . > type surface at nodes (< . > , < . > , < . > , < . >, < . >, < . >, < . >, < . >) with magnitudes (< . >)*

- gmESSI :: [Add_20NodeBrick_SurfaceLoad{PhyEntSurfaceTag, m1, m2, m3, m4, m5, m6 ,m7, m8}]
  ESSI :: add load #{} to element #{} type surface at nodes ({}, {}, {}, {}, {}, {}, {}, {}) with magnitudes ({}, {}, {}, {}, {}, {}, {}, {})
  Description :: *add load # < . > to element # < . > type surface at nodes (< . > , < . > , < . > , < . >, < . >, < . >, < . >, < . >) with magnitudes (< . > , < . > , < . > , < . >, < . >, < . >, < . >, < . >)*

- gmESSI :: [Add_27NodeBrick_SurfaceLoad{PhyEntSurfaceTag,magnitude}]
  ESSI :: add load #{} to element #{} type surface at nodes ({}, {}, {}, {}, {}, {}, {}, {}, {}) with magnitude ({})
  Description :: *add load # < . > to element # < . > type surface at nodes (< . > , < . > , < . > , < . >, < . >, < . >, < . >, < . >, < . >) with magnitudes (< . >)*

- gmESSI :: [Add_27NodeBrick_SurfaceLoad{PhyEntSurfaceTag, m1, m2, m3, m4, m5, m6, m7, m8, m9}]
  ESSI :: add load #{} to element #{} type surface at nodes ({}, {}, {}, {}, {}, {}, {}, {}, {}) with magnitudes ({}, {}, {}, {}, {}, {}, {}, {}, {})
  Description :: *add load # < . > to element # < . > type surface at nodes (< . > , < . > , < . > , < . >, < . >, < . >, < . >, < . >, < . >) with magnitudes (< . > , < . > , < . > , < . >, < . >, < . >, < . >, < . >, < . >)*

6.4 AddNode Commands

AddNode Commands adds either all the nodes of the model *[AddAllNode{unit,nof_dofs}]* or nodes of only specific physical/entity group *[AddNode{unit,nof_dofs}]*

- gmESSI :: [Add_Node{unit,nof_dofs}]
  ESSI :: add node # {} at ({},{},{}) with {} dofs
  Description :: *add node # < . > at (< length >,< length >,< length >) with < . > dofs*

- gmESSI :: [Add_All_Node{unit,nof_dofs}]
  ESSI :: add node # {} at ({}, {}, {}) with {} dofs
  Description :: *add node # < . > at (< length >,< length >,< length >) with < . > dofs*

Note:- Obviously only [Add_Node{unit,nof_dofs}] has the option of adding Physical_Group# or Entity_Group# argument in the front.

6.5 Singular Commands

Singular Commands does not require any physical/entity group to operate. They are just textitone-line translation and are written in **XYZ_analysis.fei** file. Because of the same reason stated, they **don't have** the option of adding Physical_Group# or Entity_Group# argument in the front. The different commands under this category and their corresponding ESSI commands are

6.5.1 Damping Commands

- gmESSI :: [Add_Rayleigh_Damping{a0,a1,stiffness}]
  ESSI :: add damping #{} type Rayleigh with a0={} a1={} stiffness_to_use={}
  Description :: *add damping # < . > type Rayleigh with a0 = < time > a1 = <* 1/*time > stiffness_to_use = < Initial_Stiffness* | *Current_Stiffness* | *Last_Committed_Stiffness >*

- gmESSI :: [Add_Caughey3rd_Damping{a0,a1,a2,stiffness}]
  ESSI :: add damping #{} type Caughey3rd with a0={} a1={} a2={} stiffness_to_use={}
  Description :: *add damping # < . > type Caughey3rd with a0 = < time > a1 = <* 1/*time > a2 = < . > stiffness_to_use = < Initial_Stiffness* | *Current_Stiffness* | *Last_Committed_Stiffness >*

- gmESSI :: [Add_Caughey4th_Damping{a0,a1,a2,a3,stiffness}]
  ESSI :: add damping #{} type Caughey4th with a0={} a1={} a2={} a3={} stiffness_to_use={}
  Description :: *add damping # < . > type Caughey4th with a0 = < time > a1 = <* 1/*time > a2 = < . > a3 = < . > stiffness_to_use = < Initial_Stiffness* | *Current_Stiffness* |*Last_Committed_Stiffness >*

6.5.2 Variables, Fileds And Includes

- gmESSI :: [Var{variable,value}]
  ESSI :: $< variable >=< value >$
  Description :: $< variable_name >=< valuewithorwithoutunits >$

- gmESSI :: [Include{filename}]
  ESSI :: Include " "
  Description :: *Include "file name"*

- gmESSI :: [Add_Acceleration_Field{field_no, ax, ay, az}]
  ESSI :: add acceleration field #{} ax={} ay={} az={};
  Description :: *add acceleration field # < . > ax = < accel > ay = < accel > az = < accel >*

- gmESSI :: [Bye{}]
  ESSI :: bye;
  Description :: *End ESSI Analysis*

- gmESSI :: [Comment{comment}]
  ESSI :: // comment
  Description :: *// comment followed in XYZ_analysis.fei file*

- gmESSI :: [New_Line{}]
  Description :: *prints a blank line in XYZ_analysis.fei file*

6.5.3 Definig Materials and Section

- gmESSI :: [Add_Linear_elastic_isotropic_3d{material_no, mass_density, elastic_modulus, poisson_ratio}]
  ESSI :: add material #{} type linear_elastic_isotropic_3d mass_density={} elastic_modulus={} poisson_ratio={}
  Description :: *add material # < . > type linear_elastic_isotropic_3d mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$

- gmESSI :: [Add_Linear_elastic_crossanisotropic{material_no, mass_density, elastic_modulus_horizontal, elastic_modulus_vertical, poisson_ratio_h_v, poisson_ratio_h_h, shear_modulus_h_v}]
  ESSI :: add material #{} type linear_elastic_crossanisotropic mass_density={} elastic_modulus_horizontal={} elastic_modulus_vertical={} poisson_ratio_h_v={} poisson_ratio_h_h={} shear_modulus_h_v={}
  Description :: *add material # < . > type linear_elastic_crossanisotropic mass_density = < mass_density > elastic_modulus_horizontal* $= < F/L^2 >$ *elastic_modulus_vertical* $= < F/L^2 >$ *poisson_ratio_h_v = < . > poisson_ratio_h_h = < . > shear_modulus_h_v* $= < F/L^2 >$

- gmESSI :: [Add_Vonmises_perfectly_plastic{material_no, mass_density, elastic_modulus, poisson_ratio, von_mises_radius, initial_confining_stress, initial_confining_stress, algorithm, number_of_subincrements, maximum_number_of_iterations, tolerance_1, tolerance_2}]
  ESSI :: add material #{} type vonmises_perfectly_plastic mass_density= {} elastic_modulus= {} poisson_ratio= {} von_mises_radius= {} initial_confining_stress= {} algorithm= {} number_of_subincrements={} maximum_number_of_iterations= {} tolerance_1= {} tolerance_2={}
  Description :: *add material # < . > type vonmises_perfectly_plastic mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio = < . > von_mises_radius* $= < F/L^2 >$ *initial_confining_stress* $= < F/L^2 >$

algorithm $= <$ *explicit|implicit* $>$ *number_of_subincrements* $= < . >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Add_Vonmises_perfectly_plastic_accelerated{material_no, mass_density, elastic_modulus, poisson_ratio, von_mises_radius, initial_confining_stress, maximum_number_of_iterations, tolerance_1, tolerance_2}]
  ESSI :: add material #{} type vonmises_perfectly_plastic_accelerated mass_density={} elastic_modulus={} poisson_ratio={} von_mises_radius={} initial_confining_stress={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
  Description :: *add material #* $< . >$ *type vonmises_perfectly_plastic_accelerated mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *von_mises_radius* $= < F/L^2 >$ *initial_confining_stress* $= < F/L^2 >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Add_Vonmises_isotropic_hardening{material_no, mass_density, elastic_modulus, poisson_ratio, von_mises_radius, isotropic_hardening_rate, initial_confining_stress, algorithm, number_of_subincrements, maximum_number_of_iterations, tolerance_1, tolerance_2}]
  ESSI :: add material #{} type vonmises_isotropic_hardening mass_density={} elastic_modulus={} poisson_ratio={} von_mises_radius={} isotropic_hardening_rate={} initial_confining_stress={} algorithm={} number_of_subincrements={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
  Description :: *add material #* $< . >$ *type vonmises_isotropic_hardening mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *von_mises_radius* $= < F/L^2 >$ *isotropic_hardening_rate* $= < F/L^2 >$ *initial_confining_stress* $= < F/L^2 >$ *algorithm* $= <$ *explicit|implicit* $>$ *number_of_subincrements* $= < . >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Add_Vonmises_isotropic_hardening_accelerated{material_no, mass_density, elastic_modulus, poisson_ratio, von_mises_radius, isotropic_hardening_rate, initial_confining_stress, maximum_number_of_iterations, tolerance_1, tolerance_2}]
  ESSI :: add material #{} type vonmises_isotropic_hardening_accelerated mass_density={} elastic_modulus={} poisson_ratio={} von_mises_radius={} isotropic_hardening_rate={} initial_confining_stress={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}

Description :: *add material # < . > type vonmises_isotropic_hardening_accelerated mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *von_mises_radius* $= < F/L^2 >$ *isotropic_hardening_rate* $= < F/L^2 >$ *initial_confining_stress* $= < F/L^2 >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Add_Vonmises_kinematic_hardening{material_no, mass_density, elastic_modulus, poisson_ratio, von_mises_radius, armstrong_frederick_ha, armstrong_frederick_cr, initial_confining_stress, algorithm, number_of_subincrements, maximum_number_of_iterations, tolerance_1, tolerance_2}]
  ESSI :: add material #{} type vonmises_kinematic_hardening mass_density={} elastic_modulus={} poisson_ratio={} von_mises_radius={} armstrong_frederick_ha={} armstrong_frederick_cr={} initial_confining_stress={} algorithm={} number_of_subincrements={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
  Description :: *add material # < . > type vonmises_kinematic_hardening mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *von_mises_radius* $= < F/L^2 >$ *armstrong_frederick_ha* $= < F/L^2 >$ *armstrong_frederick_cr* $= < F/L^2 >$ *initial_confining_stress* $= < F/L^2 >$ *algorithm* $= <$ *explicit|implicit* $>$ *number_of_subincrements* $= < . >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Add_Vonmises_linear_kinematic_hardening{material_no, mass_density, elastic_modulus, poisson_ratio, von_mises_radius, kinematic_hardening_rate, initial_confining_stress, algorithm, number_of_subincrements, maximum_number_of_iterations, tolerance_1, tolerance_2}]
  ESSI :: add material #{} type vonmises_linear_kinematic_hardening mass_density={} elastic_modulus={} poisson_ratio={} von_mises_radius={} kinematic_hardening_rate={} initial_confining_stress={} algorithm={} number_of_subincrements={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
  Description :: *add material # < . > type vonmises_linear_kinematic_hardening mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *von_mises_radius* $= < F/L^2 >$ *kinematic_hardening_rate* $= < . >$ *initial_confining_stress* $= < F/L^2 >$ *algorithm* $= <$ *explicit|implicit* $>$ *number_of_subincrements* $= < . >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Add_Vonmises_kinematic_hardening_accelerated{material_no,

mass_density, elastic_modulus, poisson_ratio, von_mises_radius, armstrong_frederick_ha, armstrong_frederick_cr, initial_confining_stress, initial_confining_strain, maximum_number_of_iterations, tolerance_1, tolerance_2}]
ESSI :: add material #{} type vonmises_kinematic_hardening_accelerated mass_density={} elastic_modulus={} poisson_ratio={} von_mises_radius={} armstrong_frederick_ha={} armstrong_frederick_cr={} initial_confining_stress={} initial_confining_strain={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
Description :: *add material # < . > type vonmises_kinematic_hardening_accelerated mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *von_mises_radius* $= < F/L^2 >$ *armstrong_frederick_ha* $= < F/L^2 >$ *armstrong_frederick_cr* $= < F/L^2 >$ *initial_confining_stress* $= < F/L^2 >$ *initial_confining_strain* $= < . >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Add_Vonmises_linear_kinematic_hardening_accelerated{material_no, mass_density, elastic_modulus, poisson_ratio, von_mises_radius, kinematic_hardening_rate, initial_confining_stress, initial_confining_strain, maximum_number_of_iterations, tolerance_1, tolerance_2}]
ESSI :: add material #{} type vonmises_linear_kinematic_hardening_accelerated mass_density={} elastic_modulus={} poisson_ratio={} von_mises_radius={} kinematic_hardening_rate={} initial_confining_stress={} initial_confining_strain={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
Description :: *add material # < . > type vonmises_linear_kinematic_hardening_accelerated mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *von_mises_radius* $= < F/L^2 >$ *kinematic_hardening_rate* $= < . >$ *initial_confining_stress* $= < F/L^2 >$ *initial_confining_strain* $= < . >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Add_Vonmises_perfectly_plastic_LT{material_no, mass_density, elastic_modulus, poisson_ratio, von_mises_radius,initial_confining_stress, algorithm, number_of_subincrements, maximum_number_of_iterations, tolerance_1, tolerance_2}]
ESSI :: add material #{} type vonmises_perfectly_plastic_LT mass_density= {} elastic_modulus= {} poisson_ratio= {} von_mises_radius= {} initial_confining_stress= {} algorithm= {} number_of_subincrements= {} maximum_number_of_iterations= {} tolerance_1= {} tolerance_2= {}

Description :: *add material # < . > type vonmises_perfectly_plastic_LT mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *von_mises_radius* $= < F/L^2 >$ *initial_confining_stress* $= < F/L^2 >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Add_VonmisesLT{material_no, mass_density, elastic_modulus, poisson_ratio, vonmises_radius, kinematic_hardening_rate, isotropic_hardening_rate}]
  ESSI :: add material #{} type VonMisesLT mass_density={} elastic_modulus={} poisson_ratio={} von_mises_radius={} kinematic_hardening_rate={} isotropic_hardening_rate={}
  Description :: *add material # < . > type VonMisesLT mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *von_mises_radius* $= < F/L^2 >$ *kinematic_hardening_rate* $= < F/L^2 >$ *isotropic_hardening_rate* $= < F/L^2 >$

- gmESSI :: [Add_Druckerprager_perfectly_plastic{material_no, mass_density, elastic_modulus, poisson_ratio, druckerprager_angle, initial_confining_stress, algorithm, number_of_subincrements, maximum_number_of_iterations, tolerance_1, tolerance_2}]
  ESSI :: add material #{} type druckerprager_perfectly_plastic mass_density={} elastic_modulus={} poisson_ratio={} druckerprager_angle={} initial_confining_stress={} algorithm={} number_of_subincrements={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
  Description :: *add material # < . > type druckerprager_perfectly_plastic mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *druckerprager_angle* $= < F/L^2 >$ *initial_confining_stress* $= < F/L^2 >$ *algorithm* $= <$ *explicit|implicit* $>$ *number_of_subincrements* $= < . >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Add_Druckerprager_perfectly_plastic_accelerated{material_no, mass_density, elastic_modulus, poisson_ratio, druckerprager_angle, initial_confining_stress, maximum_number_of_iterations, tolerance_1, tolerance_2}]
  ESSI :: add material #{} type druckerprager_perfectly_plastic_accelerated mass_density={} elastic_modulus={} poisson_ratio={} druckerprager_angle={} initial_confining_stress={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
  Description :: *add material # < . > type druckerprager_perfectly_plastic_accelerated mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *druckerprager_angle* $= < F/L^2 >$ *ini-*

tial_confining_stress $= < F/L^2 >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Add_Druckerprager_isotropic_hardening{material_no, mass_density, elastic_modulus, poisson_ratio, druckerprager_angle, isotropic_hardening_rate, initial_confining_stress, algorithm, number_of_subincrements, maximum_number_of_iterations, tolerance_1, tolerance_2}]
ESSI :: add material #{} type druckerprager_isotropic_hardening mass_density={} elastic_modulus={} poisson_ratio={} druckerprager_angle={} isotropic_hardening_rate={} initial_confining_stress={} algorithm={} number_of_subincrements={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
Description :: *add material* $\# < . >$ *type druckerprager_isotropic_hardening mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *druckerprager_angle* $= < F/L^2 >$ *isotropic_hardening_rate* $= < F/L^2 >$ *initial_confining_stress* $= < F/L^2 >$ *algorithm* $= < explicit|implicit >$ *number_of_subincrements* $= < . >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Add_Druckerprager_isotropic_hardening_accelerated{material_no, mass_density, elastic_modulus, poisson_ratio, druckerprager_angle, isotropic_hardening_rate, initial_confining_stress, maximum_number_of_iterations, tolerance_1, tolerance_2}]
ESSI :: add material #{} type druckerprager_isotropic_hardening_accelerated mass_density={} elastic_modulus={} poisson_ratio={} druckerprager_angle={} isotropic_hardening_rate={} initial_confining_stress={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
Description :: *add material* $\# < . >$ *type druckerprager_isotropic_hardening_accelerated mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *druckerprager_angle* $= < F/L^2 >$ *isotropic_hardening_rate* $= < F/L^2 >$ *initial_confining_stress* $= < F/L^2 >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Add_Druckerprager_kinematic_hardening{material_no, mass_density, elastic_modulus, poisson_ratio, druckerprager_angle, armstrong_frederick_ha, armstrong_frederick_cr, initial_confining_stress, algorithm, number_of_subincrements, maximum_number_of_iterations, tolerance_1, tolerance_2}]
ESSI :: add material #{} type druckerprager_kinematic_hardening mass_density={} elastic_modulus={} poisson_ratio={}

druckerprager_angle={} armstrong_frederick_ha={} armstrong_frederick_cr={} initial_confining_stress={} algorithm= {} number_of_subincrements= {} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
Description :: *add material # $< . >$ type druckerprager_kinematic_hardening mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *druckerprager_angle* $= < F/L^2 >$ *armstrong_frederick_ha* $= < F/L^2 >$ *armstrong_frederick_cr* $= < F/L^2 >$ *initial_confining_stress* $= < F/L^2 >$ *algorithm* $= <$ *explicit|implicit* $>$ *number_of_subincrements* $= < . >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Add_Druckerprager_kinematic_hardening_accelerated{material_no, mass_density, elastic_modulus, poisson_ratio, druckerprager_angle, armstrong_frederick_ha, armstrong_frederick_cr, initial_confining_stress, maximum_number_of_iterations, tolerance_1, tolerance_2}]
ESSI :: add material #{} type druckerprager_kinematic_hardening_accelerated mass_density={} elastic_modulus={} poisson_ratio={} druckerprager_angle={} armstrong_frederick_ha={} armstrong_frederick_cr={} initial_confining_stress={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
Description :: *add material # $< . >$ type druckerprager_kinematic_hardening_accelerated mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *druckerprager_angle* $= < F/L^2 >$ *armstrong_frederick_ha* $= < F/L^2 >$ *armstrong_frederick_cr* $= < F/L^2 >$ *initial_confining_stress* $= < F/L^2 >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Add_DruckerpragerLT{material_no, mass_density, elastic_modulus, poisson_ratio, druckerprager_k, kinematic_hardening_rate, isotropic_hardening_rate}]
ESSI :: add material #{} type DruckerPragerLT mass_density={} elastic_modulus={} poisson_ratio={} druckerprager_k={} kinematic_hardening_rate={} isotropic_hardening_rate={}
Description :: *add material # $< . >$ type DruckerPragerLT mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *druckerprager_k* $= < F/L^2 >$ *kinematic_hardening_rate* $= < F/L^2 >$ *isotropic_hardening_rate* $= < F/L^2 >$

- gmESSI :: [Add_Camclay{material_no, mass_density, reference_void_ratio, critical_stress_ratio_M, lambda, kappa, poisson_ratio, minimum_bulk_modulus, pressure_reference_p0, initial_confining_stress, algorithm, number_of_subincrements, maximum_number_of_iterations, tolerance_1, tolerance_2}]

ESSI :: add material #{} type camclay mass_density={} reference_void_ratio={} critical_stress_ratio_M={} lambda={} kappa={} poisson_ratio={} minimum_bulk_modulus={} pressure_reference_p0={} initial_confining_stress={} algorithm={} number_of_subincrements={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
Description :: *add material # < . > type camclay mass_density* $= < M/L^3 >$ *reference_void_ratio* $= < . >$ *critical_stress_ratio_M* $= < . >$ *lambda* $= < . >$ *kappa* $= < . >$ *poisson_ratio* $= < . >$ *minimum_bulk_modulus* $= < F/L^2 >$ *pressure_reference_p0* $= < F/L^2 >$ *initial_confining_stress* $= < F/L^2 >$ *algorithm* $= <$ *explicit|implicit* $>$ *number_of_subincrements* $= < . >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Add_Camclay_accelerated{material_no, mass_density, reference_void_ratio, critical_stress_ratio_M, lambda, kappa, poisson_ratio, minimum_bulk_modulus, pressure_reference_p0, initial_confining_stress, maximum_number_of_iterations, tolerance_1, tolerance_2}]
ESSI :: add material #{} type camclay_accelerated mass_density={} reference_void_ratio={} critical_stress_ratio_M={} lambda={} kappa={} poisson_ratio={} minimum_bulk_modulus={} pressure_reference_p0={} initial_confining_stress={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
Description :: *add material # < . > type camclay_accelerated mass_density* $= < M/L^3 >$ *reference_void_ratio* $= < . >$ *critical_stress_ratio_M* $= < . >$ *lambda* $= < . >$ *kappa* $= < . >$ *poisson_ratio* $= < . >$ *minimum_bulk_modulus* $= < F/L^2 >$ *pressure_reference_p0* $= < F/L^2 >$ *initial_confining_stress* $= < F/L^2 >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Add_Sanisand2004{material_no, mass_density, e0, sanisand2004_G0, poisson_ratio, sanisand2004_Pat, sanisand2004_p_cut, sanisand2004_Mc, sanisand2004_c, sanisand2004_lambda_c, sanisand2004_xi, sanisand2004_ec_ref, sanisand2004_m, sanisand2004_h0, sanisand2004_ch, sanisand2004_nb, sanisand2004_A0, sanisand2004_nd, sanisand2004_z_max, sanisand2004_cz, initial_confining_stress, algorithm, number_of_subincrements, maximum_number_of_iterations, tolerance_1, tolerance_2}]
ESSI :: add material #{} type sanisand2004 mass_density= {} e0= {} sanisand2004_G0= {} poisson_ratio= {} sanisand2004_Pat= {} sanisand2004_p_cut= {} sanisand2004_Mc= {} sanisand2004_c= {} sanisand2004_lambda_c= {} sanisand2004_xi= {} sanisand2004_ec_ref= {} sanisand2004_m= {} sanisand2004_h0= {} sanisand2004_ch= {} sanisand2004_nb= {} sanisand2004_A0= {} sanisand2004_nd= {} sanisand2004_z_max= {} sanisand2004_cz=

{} initial_confining_stress= {} algorithm= {} number_of_subincrements= {} maximum_number_of_iterations= {} tolerance_1= {} tolerance_2={}
Description :: *add material # < . > type sanisand2004 mass_density* = < M/L^3 > *e0* = < . > *sanisand2004_G0* = < . > *poisson_ratio* = < . > *sanisand2004_Pat* = < *stress* > *sanisand2004_p_cut* = < . > *sanisand2004_Mc* = < . > *sanisand2004_c* = < . > *sanisand2004_lambda_c* = < . > *sanisand2004_xi* = < . > *sanisand2004_ec_ref* = < . > *sanisand2004_m* = < . > *sanisand2004_h0* = < . > *sanisand2004_ch* = < . > *sanisand2004_nb* = < . > *sanisand2004_A0* = < . > *sanisand2004_nd* = < . > *sanisand2004_z_max* = < . > *sanisand2004_cz* = < . > *initial_confining_stress* = < *stress* > *algorithm* = < *explicit|implicit* > *number_of_subincrements* = < . > *maximum_number_of_iterations* = < . > *tolerance_1* = < . > *tolerance_2* = < . >

- gmESSI :: [Add_Sanisand2008{material_no, mass_density, e0, sanisand2008_G0, sanisand2008_K0, sanisand2008_Pat, sanisand2008_k_c, sanisand2008_alpha_cc, sanisand2008_c, sanisand2008_xi, sanisand2008_lambda, sanisand2008_ec_ref, sanisand2008_m, sanisand2008_h0, sanisand2008_ch, sanisand2008_nb, sanisand2008_A0, sanisand2008_nd, sanisand2008_p_r, sanisand2008_rho_c, sanisand2008_theta_c, sanisand2008_X, sanisand2008_z_max, sanisand2008_cz, sanisand2008_p0, sanisand2008_p_in, algorithm, number_of_subincrements, maximum_number_of_iterations, tolerance_1, tolerance_2}]
ESSI :: add material #{} type sanisand2008 mass_density= {} e0= {} sanisand2008_G0= {} sanisand2008_K0= {} sanisand2008_Pat= {} sanisand2008_k_c= {} sanisand2008_alpha_cc= {} sanisand2008_c= {} sanisand2008_xi={} sanisand2008_lambda= {} sanisand2008_ec_ref= {} sanisand2008_m= {} sanisand2008_h0= {} sanisand2008_ch= {} sanisand2008_nb= {} sanisand2008_A0= {} sanisand2008_nd= {} sanisand2008_p_r= {} sanisand2008_rho_c= {} sanisand2008_theta_c= {} sanisand2008_X= {} sanisand2008_z_max= {} sanisand2008_cz={} sanisand2008_p0= {} sanisand2008_p_in= {} algorithm= {} number_of_subincrements= {} maximum_number_of_iterations= {} tolerance_1= {} tolerance_2= {}
Description :: *add material # < . > type sanisand2008 mass_density* = < M/L^3 > *e0* = < . > *sanisand2008_G0* = < . > *sanisand2008_K0* = < . > *sanisand2008_Pat* = < *stress* > *sanisand2008_k_c* = < . > *sanisand2008_alpha_cc* = < . > *sanisand2008_c* = < . > *sanisand2008_xi* = < . > *sanisand2008_lambda* = < . > *sanisand2008_ec_ref* = < . > *sanisand2008_m* = < . > *sanisand2008_h0* = < . > *sanisand2008_ch* = < . > *sanisand2008_nb* = < . > *sanisand2008_A0* = < . > *sanisand2008_nd* = < . > *sanisand2008_p_r* = < . > *sanisand2008_rho_c* = < . > *sanisand2008_theta_c*

$= < \ . \ >$ *sanisand2008_X* $= < \ . \ >$ *sanisand2008_z_max* $= < \ . \ >$ *sanisand2008_cz* $= < . >$ *sanisand2008_p0* $= <$ *stress* $>$ *sanisand2008_p_in* $= < . >$ *algorithm* $= <$ *explicit|implicit* $>$ *number_of_subincrements* $= < . >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Add_Uniaxial_elastic{material_no, elastic_modulus, viscoelastic_modulus}]
  ESSI :: add material #{} type uniaxial_elastic elastic_modulus={} viscoelastic_modulus={}
  Description :: *add material* # $< . >$ *type uniaxial_elastic elastic_modulus* $= < F/L^2 >$ *viscoelastic_modulus* $= <$ *mass/length/time* $>$

- gmESSI :: [Add_Uniaxial_concrete02{material_no,comprESSIve_strength, strain_at_comprESSIve_strength, crushing_strength, strain_at_crushing_strength, lambda, tensile_strength, tension_softening_stiffness}]
  ESSI :: add material #{} type uniaxial_concrete02 comprESSIve_strength= {} strain_at_comprESSIve_strength= {} crushing_strength= {} strain_at_crushing_strength= {} lambda= {} tensile_strength= {} tension_softening_stiffness= {}
  Description :: *add material* # $< . >$ *type uniaxial_concrete02 comprESSIve_strength* $= < F/L^2 >$ *strain_at_comprESSIve_strength* $= < . >$ *crushing_strength* $= < F/L^2 >$ *strain_at_crushing_strength* $= < . >$ *lambda* $= < . >$ *tensile_strength* $= < F/L^2 >$ *tension_softening_stiffness* $= < F/L^2 >$

- gmESSI :: [Add_Uniaxial_steel01{material_no,yield_strength, elastic_modulus, strain_hardening_ratio, a1, a2, a3, a4}]
  ESSI :: add material #{} type uniaxial_steel01 yield_strength={} elastic_modulus={} strain_hardening_ratio={} a1={} a2={} a3={} a4={}
  Description :: *add material* # $< . >$ *type uniaxial_steel01 yield_strength* $= < F/L^2 >$ *elastic_modulus* $= < F/L^2 >$ *strain_hardening_ratio* $= < . >$ *a1* $= < . >$ *a2* $= < . >$ *a3* $= <>$ *a4* $= < . >$

- gmESSI :: [Add_Uniaxial_steel02{material_no,yield_strength, elastic_modulus, strain_hardening_ratio, R0, cR1, cR2, a1, a2, a3, a4}]
  ESSI :: add material #{} type uniaxial_steel02 yield_strength={} elastic_modulus={} strain_hardening_ratio={} R0={} cR1={} cR2={} a1={} a2={} a3={} a4={}
  Description :: *add material* # $< . >$ *type uniaxial_steel02 yield_strength* $= < F/L^2 >$ *elastic_modulus* $= < F/L^2 >$ *strain_hardening_ratio* $= < . >$ *R0* $= < . >$ *cR1* $= < . >$ *cR2* $= < . >$ *a1* $= < . >$ *a2* $= < . >$ *a3* $= <>$ *a4* $= < . >$

- gmESSI :: [Add_New_PisanoLT{Elemental_Command {without_material_no_argument},mass_density, elastic_modulus, poisson_ratio,

M_in, kd_in, xi_in, h_in, m_in, initial_confining_stress, n_in, a_in, eplcum_cr_in}]
ESSI :: add material #{} type New_PisanoLT mass_density={} elastic_modulus_1atm={} poisson_ratio={} M_in={} kd_in={} xi_in={} h_in={} m_in={} initial_confining_stress={} n_in ={} a_in ={} eplcum_cr_in ={}
Description :: *add material # < . > type New_PisanoLT mass_density =* $< M/L^2 >$ *elastic_modulus_1atm =* $< F/L^2 >$ *poisson_ratio = < . > M_in =* $< F/L^2 >$ *kd_in = < . > xi_in = < . > h_in = < . > m_in = < . > initial_confining_stress =* $< F/L^2 >$ *n_in = < . > a_in = < . > eplcum_cr_in = < . >*

- gmESSI :: [Add_Linear_elastic_isotropic_3d_LT{material_no, mass_density, elastic_modulus, poisson_ratio}]
  ESSI :: add material #{} type linear_elastic_isotropic_3d_LT mass_density={} elastic_modulus={} poisson_ratio={}
  Description :: *add material # < . > type linear_elastic_isotropic_3d_LT mass_density =* $< M/L^3 >$ *elastic_modulus =* $< F/L^2 >$ *poisson_ratio = < . >*

- gmESSI :: [Add_Section_Elastic_Membrane_Plate{section_no, elastic_modulus, poissons_ratio, thickness, mass_density}]
  ESSI :: add section #{} type Elastic_Membrane_Plate elastic_modulus={} poisson_ratio={} thickness={} mass_density={}
  Description :: *add section # < . > type Membrane_Plate_Fiber thickness = < length > use material # < . >;*

- gmESSI :: [Add_Section_Membrane_Plate_Fiber{section_no, thickness, material_no}]
  ESSI :: add section #{} type Membrane_Plate_Fiber thickness={} use material #{}
  Description :: *add section # < . > type Elastic_Membrane_Plate elastic_modulus =* $< F/L^2 >$ *poisson_ratio = < . > thickness = < length > mass_density =* $< M/L^3 >$*;*

6.5.4 Definig Simulation Options

- gmESSI :: [Define_Algorithm{With_no_convergence_check/Newton/Modified_Newton}]
  ESSI :: define algorithm {}
  Description :: *define algorithm* $< With\_no_convergence\_check|Newton|Modified\_Newton >$

- gmESSI :: [Define_Convergence_Test{convergence_test_method, tolerence, maximum_iterations, verbose_level}]
  ESSI :: define convergence test {} tolerance={} maximum_iterations={} verbose_level={}
  Description :: *define convergence test* $< Norm\_Displacement\_Increment|$

Energy_Increment| *Norm_Unbalance* > *tolerance* = < *length* > *maximum_iterations* = < . > *verbose_level* = < 0 > | < 1 > | < 2 >

- gmESSI :: [Define_Hibler_Hughes_Taylor_dynamic_integration{alpha}]
  ESSI :: define dynamic integrator Hilber_Hughes_Taylor with alpha={}
  Description :: *define dynamic integrator Hilber_Hughes_Taylor with alpha* = < . >

- gmESSI :: [Define_Newmark_dynamic_integration{gamma, beta}]
  ESSI :: define dynamic integrator Newmark with gamma={} beta={}
  Description :: *define dynamic integrator Newmark with gamma* = < . > *beta* = < . >

- gmESSI :: [Define_Load_Factor_increament{incr}]
  ESSI :: define load factor increment {}
  Description :: *define load factor increment* < . >

- gmESSI :: [Define_Solver_ProfileSPD{}]
  ESSI :: define solver ProfileSPD
  Description :: *define solver ProfileSPD*

- gmESSI :: [Define_Solver_UMFPack{}]
  ESSI :: define solver UMFPack
  Description :: *define solver UMFPack*

- gmESSI :: [Define_Static_Integrator_Displacement_Control{node#1, dof, incr}]
  ESSI :: define static integrator displacement_control using node #{} dof {} increment {}
  Description :: *define static integrator displacement_control using node* # < . > *dof DOFTYPE increment* < *length* >

- gmESSI :: [Simulate_Steps_Using_Static_Algorithm{nofsteps}]
  ESSI :: simulate {} steps using static algorithm
  Description :: *simulate* < . > *steps using static algorithm*

- gmESSI :: [Simulate_Steps_Using_Transient_Algorithm{nofsteps, time_step}]
  ESSI :: simulate {} steps using transient algorithm time_step={}
  Description :: *simulate* < . > *steps using transient algorithm time_step* = < *time* >

- gmESSI :: [Simulate_Steps_Using_Variable_Transient_Algorithm{nofsteps, time_step, min_time_step, max, nof_iter}]
  ESSI :: simulate {} steps using variable transient algorithm time_step={} minimum_time_step={} maximum_time_step={} number_of_iterations={}
  Description :: *simulate* < . > *steps using variable transient algorithm time_step*

= < time > minimum_time_step = < time > maximum_time_step = < time > number_of_iterations = < . >

- gmESSI :: [Simulate_Constitutive_Testing_Constant_Pressure_Triaxial_Strain_Control{material_
  strain_increment_size, maximun_strain, no_of_reaching_maximum_strain}]
  ESSI :: simulate constitutive testing constant mean pressure triaxial strain control use material #{} strain_increment_size={} maximum_strain={} number_of_times_reaching_maximum_strain={}
  Description :: *simulate constitutive testing constant mean pressure triaxial strain control use material # < . > strain_increment_size = < . > maximum_strain = < . > number_of_times_reaching_maximum_strain = < . >*

- gmESSI :: [Simulate_Constitutive_Testing_Drained_Triaxial_Strain_Control{material_no, strain_increment_size, maximun_strain, no_of_reaching_maximum_strain}]
  ESSI :: simulate constitutive testing drained triaxial strain control use material # {} strain_increment_size={} maximum_strain={} number_of_times_reaching_maximum_strain={}
  Description :: *simulate constitutive testing drained triaxial strain control use material # < . > strain_increment_size = < . > maximum_strain = < . > number_of_times_reaching_maximum_strain = < . >*

- gmESSI :: [Simulate_Constitutive_Testing_Undrained_Simple_Shear{material_no, strain_increment_size, maximun_strain, no_of_reaching_maximum_strain}]
  ESSI :: simulate constitutive testing undrained simple shear use material #{} strain_increment_size={} maximum_strain={} number_of_times_reaching_maximum_strain={}
  Description :: *simulate constitutive testing undrained simple shear use material # < . > strain_increment_size = < . > maximum_strain = < . > number_of_times_reaching_maximum_strain = < . >*

- gmESSI :: [Simulate_Constitutive_Testing_Undrained_Triaxial_Stress_Control{material_no, strain_increment_size, maximun_strain, no_of_reaching_maximum_strain}]
  ESSI :: simulate constitutive testing undrained triaxial stress control use material #{} strain_increment_size={} maximum_strain={} number_of_times_reaching_maximum_strain={}
  Description :: *simulate constitutive testing undrained triaxial stress control use material # < . > strain_increment_size = < . > maximum_strain = < . > number_of_times_reaching_maximum_strain = < . >*

- gmESSI :: [Simulate_Constitutive_Testing_Undrained_Triaxial{material_no, strain_increment_size, maximun_strain, no_of_reaching_maximum_strain}]
  ESSI :: simulate constitutive testing undrained triaxial use material #{} strain_increment_size = {} maximum_strain = {} number_of_times_reaching_maximum_strain = {}
  Description :: *simulate constitutive testing undrained triaxial use material #*

< . > strain_increment_size = < . > maximum_strain = < . > number_of_times_reaching_maximum_strain = < . >

- gmESSI :: [Simulate_Using_Eigen_Algorithm{no_of_modes}]
  ESSI :: simulate using eigen algorithm number_of_modes={}
  Description :: *simulate using eigen algorithm number_of_modes = < . >*

- gmESSI :: [Set_Output_ComprESSIon_level{ level}]
  ESSI :: set output comprESSIon level to {}
  Description :: *set output comprESSIon level to < . >*

- gmESSI :: [Disable_Output{}]
  ESSI :: disable output
  Description :: *disable output*

- gmESSI :: [Enable_Output{}]
  ESSI :: enable output
  Description :: *enable output*

- gmESSI :: [Enable_Element_Output{}]
  ESSI :: enable element output
  Description :: *enable element output*

- gmESSI :: [Print_Material{material_no}]
  ESSI :: print material #{}
  Description :: *print material # < . >*

- gmESSI :: [Print_Domain{}]
  ESSI :: print domain
  Description :: *print domain*

- gmESSI :: [Output_Step{step}]
  ESSI :: output every {} step
  Description :: *output every < . > steps*

- gmESSI :: [Add_New_Loading_Stage{name_string}]
  ESSI :: new loading stage {}
  Description :: *new loading stage "name_string"*

- gmESSI :: [Model_Name{name_in_string}]
  ESSI :: model name {}
  Description :: *model name "name_string"*

- gmESSI :: [Check_Mesh{filename}]
  ESSI :: check mesh {}
  Description :: *check mesh filename*

- gmESSI :: [Compute_Reaction_Forces{}]
  ESSI :: compute reaction forces
  Description :: *compute reaction forces*

6.5.5 Definig Single Node Loads

- gmESSI :: [Add_Domain_Reduction_Method{loading_no, hdf5_input_file}]
  ESSI :: add domain reduction method loading #{} hdf5_file={} acceleration_file={}
  Description :: *add domain reduction method loading* $\# < . >$ *hdf5_file* $= < string >$

- gmESSI :: [Add_Single_Node_mass{node#1,mx,my,mz}]
  ESSI :: add mass to node #{} mx ={} my ={} mz ={}
  Description :: *add mass to node* $\# < . >$ $mx = < mass >$ $my = < mass >$ $mz = < mass >$

- gmESSI :: [Add_Single_Node_mass{node#1,mx,my,mz,Imx,Imy,Imz}]
  ESSI :: add mass to node #{} mx ={} my ={} mz ={} Imx ={} Imy ={} Imz ={}
  Description :: *add mass to node* $\# < . >$ $mx = < mass >$ $my = < mass >$ $mz = < mass >$ $Imz = < mass * length^2 >$ $Imy = < mass * length^2 >$ $Imz = < mass * length^2 >$

- gmESSI :: [Add_Single_Node_Damping{damping#1,node#1}]
  ESSI :: add damping #{} to node #{}
  Description :: *add damping* $\# < . >$ *to node* $\# < . >$

- gmESSI :: [FixNode{node#1, dofs}]
  ESSI :: fix node #{} dofs {}
  Description :: *fix node* $\# < . >$ *dofs* $< . >$

- gmESSI :: [FreeNode{node#1, dofs}]
  ESSI :: free node #{} dofs {}
  Description :: *free node* $\# < . >$ *dofs* $< . >$

- gmESSI :: [Add_Single_Master_Slave{slave_node,master_node,dof}]
  ESSI :: add constraint equal dof with master node #{} and slave node #{} dof to constrain {}
  Description :: *add constraint equal dof with master node* $\# < . >$ *and slave node* $\# < . >$ *dof to constrain* $< . >$

- gmESSI :: [Add_Single_Remove_Master_Slave{slave_node}]
  ESSI :: remove constraint equaldof node #{}
  Description :: *remove constraint equaldof node* $\# < . >$

- gmESSI :: [Add_Single_Node_Point_Constraint{node#1,dof,val}]
  ESSI :: add single point constraint to node #{} dof to constrain {} constraint value of {}]
  Description :: *add single point constraint to node # < . > dof to constrain < dof_type > constraint value of < correspondingunit >;*

- gmESSI :: [Add_Single_Node_Load_linear{node#1,dir,mag}]
  ESSI :: add load #{} to node #{} type linear {} = {}
  Description :: *add load # < . > to node # < . > type linear FORCETYPE = < forceormoment > //FORCETYPE = Fx Fy Fz Mx My Mz F_fluid_x F_fluid_y F_fluid_z*

- gmESSI :: [Add_Single_Node_Load_Path_Series{node#1,path_series,dir,mag,seriesfile}]
  ESSI :: add load #{} to node #{} type path_series {} = {} series_file ={}]
  Description :: *add load # < . > to node # < . > type path_series FORCETYPE = < forceormoment > time_step = < time > series_file = "STRING"*

- gmESSI :: [Add_Single_Node_Load_Path_Time_Series{node#1, path_time_series, dir, mag, time_step, seriesfile}]
  ESSI :: add load #{} to node #{} type path_time_series {} = {} time_step = {} series_file ={}
  Description :: *add load # < . > to node # < . > type path_time_series FORCETYPE = < forceormoment > series_file = "STRING"*

- gmESSI :: [Add_Single_Imposed_Motion_Time_Series{node#1, dof, time_step, dis_scl, dis_fl, vel_scl, vel_fl, acc_scl, acc_fl}]
  ESSI :: add imposed motion #{} to node #{} dof {} time_step = {} displacement_scale_unit = {} displacement_file = {} velocity_scale_unit = {} velocity_file = {} acceleration_scale_unit = {} acceleration_file ={}
  Description :: *add imposed motion # < . > to node # < . > dof DOFTYPE time_step = < t > displacement_scale_unit = < length > displacement_file = "filename" velocity_scale_unit = < velocity > velocity_file = "filename" acceleration_scale_unit = < acceleration > acceleration_file = "filename";*

- gmESSI :: [Add_Single_Imposed_Motion{node#1, dof, dis_scl, dis_fl, vel_scl, vel_fl, acc_scl, acc_fl}]
  ESSI :: add imposed motion #{} to node #{} dof {} time_step = {} displacement_scale_unit = {} displacement_file = {} velocity_scale_unit = {} velocity_file = {} acceleration_scale_unit = {} acceleration_file ={}
  Description :: *add imposed motion # < . > to node # < . > dof DOFTYPE displacement_scale_unit = < length > displacement_file = "filename" velocity_scale_unit = < velocity > velocity_file = "filename" acceleration_scale_unit = < acceleration > acceleration_file = "filename";*

- gmESSI :: [Remove_Single_Load{load#1}]
  ESSI :: remove load #{}
  Description :: *remove load # < . >*

- gmESSI :: [Remove_Imposed_Motion{motion#1}]
  ESSI :: remove imposed motion #{}
  Description :: *remove imposed motion # < . >*

- gmESSI :: [Remove_Single_Element{element#1}]
  ESSI :: remove element #{}]
  Description :: *remove element # < . >*

- gmESSI :: [Remove_Single_Node{node#1}]
  ESSI :: remove node #{}
  Description :: *remove node # < . >*

- gmESSI :: [Add_Single_Node_Self_Weight{node_no,field_no}]
  ESSI :: add load #{} to node #{} type self_weight use acceleration field #{}
  Description :: *add load # < . > to node # < . > type self_weight use acceleration field # < . >*

6.6 Special Commands

6.6.1 Connect Command

Connect Commands *creates/find layers of 2-noded elements* between any two parallel geometrical physical entities like two lines, two surface or two volumes and creates a physical group of those elements and updates this information in the **XYZ.msh** file. Since *Gmsh* does not include the feature of defining or creating 2-noded elements after the mesh creation, this command can be very usefull in that case. For example;- defining contacts, embedded piles, boundary conditions, connections etc.

- gmESSI :: [Connect{Physical_Group# or Entity_Group#tag_From , Physical_Group# or Entity_Group#tag_To, Physical_Group# or Entity_Group#tag_Between, dir_vect, mag, no_times, $algo\_(find|create)$, tolerence,New_Physical_Group_Name}]

  Physical_Group# or Entity_Group#tag_From :: It defines the starting nodes
  Physical_Group# or Entity_Group#tag_To :: It defines the set of end nodes
  Physical_Group# or Entity_Group#tag_Between:: It defines the set of nodes where the intermediatory nodes can be found, while searching. While creating nodes, it does not play any role.
  dir_vect :: It defines the direcion in which the user wants to create or find the

nodes. The direction vector argument is given as {x_comp \y_comp \z_comp}. Example:- {0 \0 \-1} , {1 \1 \0} .. etc.
mag :: It defines the length of each 2-noded line elements
no_times :: It defines number of layers of 2-noded elemnts, the user want to create/find
algo_(find/create) :: It defines the algo which is either 0 or 1 meaning whether to find or create the intermediatory nodes
tolerence :: It defines the tolerence is required to finding the nodes. It should be less than the minimum of the distance of neighbouring nodes.
New_Physical_Group_Name :: This argument enables the user to give a name to the 2-noded new-physical group formed

Figure 8 graphically describes arguments of connect command.

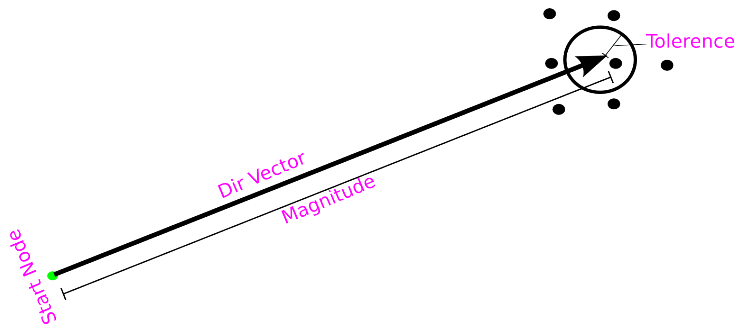


Figure 8: Pictorial representation of working of connect command

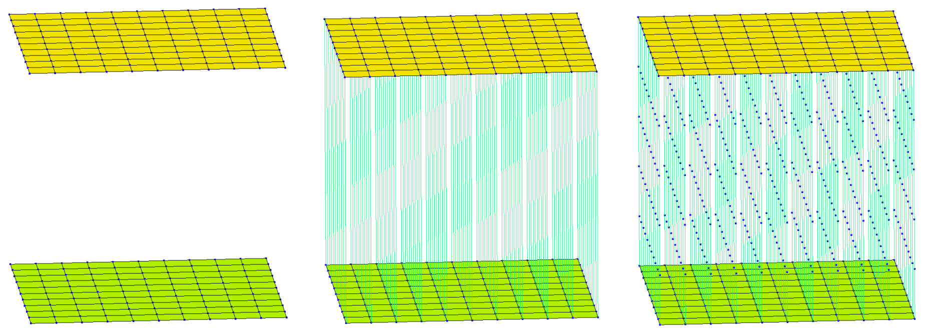
- This command updates creates **additional nodes and 2-noded elements** and **also assigns a physical group name "$New_Physical_Group_Name$"**. gmESSI automatically adds the next id available to the new physical group. The user can then manupulate this newly created physical group with any other gmESSI commands.

The working of this command would be more clear through an example. ***Example2*** and ***Example3*** describes two situtaion one where **new nodes are to be created** and the other **where allready present nodes needs to be found** respectively. In both the cases **2-noded line elements** are always created.

Example2 creates a contact elements between two prallell surfaces, described by physical group 1 and 2 respectively. Let us look at the Example2.geo file.

```
$ cat Example2.geo

  Point(1)= {0,0,0}; Point(2) ={0,0,1};
  Extrude{1,0,0} {Point{1,2};Layers{10};Recombine;}
  Extrude{0,1,0} {Line{1,2};Layers{10};Recombine;}
  Physical Surface ("$Surface1$ ") ={6};
  // Generating 1 layers of elements between Physical group 1 and 2
  Physical Surface ("$Surface2$ [Connect{Physical_Group#1, Physical_Group#2, Physical_Group#2,0\0\1,1 , 1, create,
        0.005, My_New_Physical_Group}]") ={10};
  // Generating 5 layers of elements between Physical group 1 and 2
// Physical Surface ("$Surface2$ [Connect{Physical_Group#1, Physical_Group#2, Physical_Group#2,0\0\1,0.2 , 5,
        create, 0.005, My_New_Physical_Group}]") ={10};
```

Running Example2.msh with gmESSI, produces additional nodes and elements as shown in FigureFigure 9 . **Please note "create" in the gmESSI command (b) and (c) refers to create new nodes**.



(a) Initial mesh file Example2.msh geneaert by gmsh. The dir vector is in Z axis 0,0,1

(b) 1 layer of contact elements [Connect{Physical_Group#1, Physical_Group#2, Physical_Group#2, 0\0\1, 1, 1, create, 0.005, My_New_Physical_Group}]

(c) 5 Layers of contact elements [Connect{ Physical_Group#1, Physical_Group#2, Physical_Group#2, 0\0\1 ,0.2, 5, create, 0.005, My_New_Physical_Group}]

Figure 9: Example 2 Contact Problem. (b) and (c) shows the nodes and elements generated by gmESSI Translator.

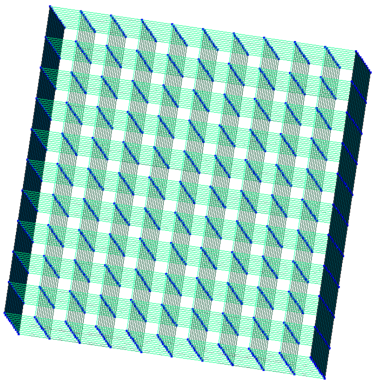
The termminal displays the information about number of elements and nodes created and also displays the information about the new physical group information i.e id and name. The new physical group creation can be seen in the Example2.msh file saves in Example2_ESSI_Simulation folder. The terminal message and mesh file is shown below. It also displays error message if more than one node is found in the tolerence provided.

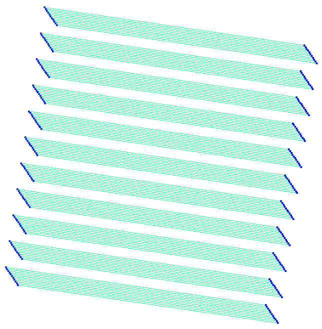
```
$ gmessi Example2.msh

  Connect{Physical_Group#1,Physical_Group#2, Physical_Group#3, 0\0\1, 1, 1, create, 0.005,
        My_New_Physical_Group} Found!! Sucessfully Converted
  New Physical Group "My_New_Physical_Group" having id 3 consisting of 242 Nodes and 121 , 2 noded elements
        created
```

```
$ cat Example2_ESSI_Simulation\Example2.msh
  ............
  $PhysicalNames
  3
  2 1 "$Surface1$ "
  2 2 "$Surface2$ [Connect{Physical_Group#1, Physical_Group#2, Physical_Group#2, 0\0\1, 1, 1, create, 0.005,
        My_New_Physical_Group}]"
  1 3 "$My_New_Physical_Group$"
  $EndPhysicalNames
  ...........
```

Example3 describes a solid cube where a physical group of all the 2-noded elements joining the nodes from physical surface 1 to 2 needs to be initialized. This physical group can then be manupulated by the user for any other gmESSI command like defining beam elements ... etc. Here the task is to find textbf(algo 'find') the allready present nodes and textbfcreate 2-noded elements. Let us look at the Example3.geo file.

```
$ cat Example3.geo

  Point(1)= {0,0,0};
  Extrude{1,0,0} {Point{1};Layers{10};Recombine;}
  Extrude{0,1,0} {Line{1};Layers{10};Recombine;}
  Extrude{0,0,1} {Surface{5};Layers{10};Recombine;}
  Physical Surface ("$Surface1$ ") ={5};
  // Generating 1 layers of elements between Physical group 1 and 2
  Physical Surface ("$Surface2$ [Connect{Physical_Group#1, Physical_Group#2, Physical_Group#3, 0\0\1, 1, 1, find,
        0.005}]") ={27};
  // Generating 5 layers of elements between Physical group 1 and 2
  // Physical Surface ("$Surface2$ [Connect{Physical_Group#1, Physical_Group#2, Physical_Group#3, 0\0\1, 0.2, 5, find,
        0.005}]") ={27};
  Physical Volume ("$Volume$") ={1};
```


(a) Initial mesh file Example3.msh genearet by gmsh. The dir vector is in Z axis 0,0,1


(b) New physical group 4 of 1 Layer of elements [Connect{Physical_Group#1, Physical_Group#2, Physical_Group#3, 0\0\1, 1, 1, 0, find, 005, My_New_Physical_Group}]

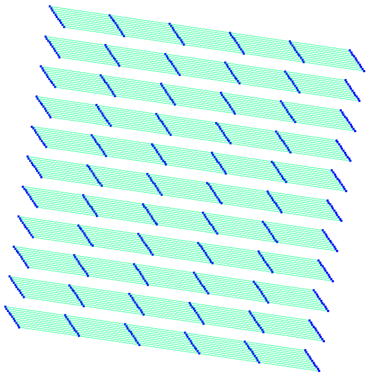

(c) New physical group 4 of 5 Layers of elements [Connect{ Physical_Group#1, Physical_Group#2, Physical_Group#3, 0\0\1 ,0.2, 5, find, 0.005, My_New_Physical_Group}]

Figure 10: Example 3 finding nodes problem. (b) and (c) shows the nodes and elements generated by gmESSI Translator.

Similarly the Example3.msh contains the new physical group and terminal shows the new physical group of 2-noded elements created. Figure 10 shows the new nodes found and creation of 2-noded elements.

```
$ gmESSI Example3.msh

  Connect{Physical_Group#1, Physical_Group#2, Physical_Group#3, 0\0\1, 1, 1, find, 0.005, My_New_Physical_Group}
        Found!!
  New Physical Group "My_New_Physical_Group" having id 4 consisting of 242 Nodes and 121 2-noded elements created
```

```
$ cat Example3_ESSI_Simulation\Example3.msh
  ............
  $PhysicalNames
  3
  2 1 "$Surface1$ "
  2 2 "$Surface2$ [Connect{Physical_Group#1, Physical_Group#2, Physical_Group#3,0\0\1, 1, 1, find, 0.005,
        My_New_Physical_Group}]"
  3 3 "$Volume$"
  1 4 "$My_New_Physical_Group$"
  $EndPhysicalNames
  ...........
```

Please note that since the algo is to find the nodes so no new nodes are created but only elements are created. The same message can be ssen on the terminal.

6.6.2 Material Variational command

Material Variational command is a special command to *define different element's material properties throughtout the physical/entity group* based on a function for each material parameter varying with the elements coordinate in (x,y,z). The corresponding new materials are written in the main **XYZ_analysis.fei** and the elements definition are written in **XYZ_geometry.fei**. This command is extremely useful if you want to assign different material properties to different layers let say as soil. This commands work on physical/entity group of elements. In this command each material property is like a **mathematical function of f(x,y,z)**. The function gets evaluated in octave, so one can write anything that octave can understand.

All the material properties are as a function and has two optional parameters **precision** and **unit** separate by separator '\' . A typical syntax of this type of command is explained below. An working example of this command is shown later

[Vary_Material{Elemental_Comm{a,b}, Par1_fun\Prec \Unit, Par1_fun\Prec \Unit,.......}]

Vary_Material :: It refers to the name of the command. The name of the command is *Vary_materialname*. For example: to vary a *Linear_elastic_isotropic_3d* the command name is Vary_Linear_elastic_isotropic_3d

Elemental_Command{a,b} :: It refers to the name of the elemental command defined in section 5.3. The elemental command refer to the type of element to define and apply material for the given physical group. The elemental command is the same as in section 5.3 but has no "mataerial_no" and the arguments are seprated "," as earlier. For example:- an elemental command

Add_Beam_Displacement_Based{ material_no, section_no, mass_density, integration_rule, xz_plane_vector_x, xz_plane_vector_y, xz_plane_vector_z, joint_1_offset_1, joint_1_offset_2, joint_1_offset_3, joint_2_offset_1, joint_2_offset_2, joint_2_offset_3}
would change to

Add_Beam_Displacement_Based(section_no; mass_density; integration_rule; xz_plane_vector_x; xz_plane_vector_y; xz_plane_vector_z; joint_1_offset_1;

joint_1_offset_2; joint_1_offset_3; joint_2_offset_1; joint_2_offset_2; joint_2_offset_3)
Note that material_no parameter has been removed from the command and then incorporated in material_variational command.

Par1_function :: All the properties/parameters of the material command is like a function in terms of variable x, y, z like $14100 * x + y$. If any parameter is a string it should be written withing double quotes "" like *algorithm type* in **ESSI** is a string and thus can be written as *"implicit"*.

Precision :: Precision is applied on the output of *parameter_function*. It is optional but if written, it should be separated from parameter_function by \. A precision value can be any integer i.e **+ve**, **-ve** or **0**.

- **+n -** It refers to rounding of the parameter_function output upto **n** significant digits in decimals. Example:- A precision of 3 and 2 applied to 163.23647 will produce 163.236 and 163.24 respectively.

- **0 -** It refers to rounding of the parameter_function output to an integer. Example:- A precision of 0 applied to 163.23647 will produce 163.

- **-n -** It refers to rounding of the parameter_function output upto $10^n th$ place in number system. Example:- A precision of -3 and -2 applied to 163.23647 will produce 200 and 170 respectively.

Unit :: It refers to unit of the material_parameter. It is optional but if written, it should be separated from Precision by \.

Note:- Every Material Variational command has the option of adding Physical_Group# or Entity_Group# argument in the front.

To illustrate an example of material variational command, Example3_geometry.fei is again used. Then Material variational command is applied on the **"Volume"** physical group to define layers of 8 noded bricks with varying material Linear_elastic_isotropic_3d property. Let say we increase the *Elastic modulus and density* of the 8noded-bricks down the depth in -z direction.

```
$ cat Example3.geo

  Point(1)= {0,0,0};
  Extrude{1,0,0} {Point{1};Layers{10};Recombine;}
  Extrude{0,1,0} {Line{1};Layers{10};Recombine;}
  Extrude{0,0,1} {Surface{5};Layers{10};Recombine;}
  Physical Surface ("$Surface1$ ") ={5};
  // Generating 1 layers of elements between Physical group 1 and 2
  Physical Surface ("$Surface2$ [Connect{Physical_Group#1,Physical_Group#2,Physical_Group#3,0\0\1,1,1,0,0.005,
        My_New_Physical_Group}]") ={27};
  // Generating 5 layers of elements between Physical group 1 and 2
  // Physical Surface ("$Surface2$ [Connect{Physical_Group#1,Physical_Group#2,Physical_Group#3,0\0\1,0.2,5,1,0.005,
        My_New_Physical_Group}]") ={27};
  Physical Volume ("$Volume$ [Vary_Linear_elastic_isotropic_3d{Add_8NodeBrick{},
    1600−100∗(1−z)\0\kg/m^3,2∗10^9−(1−z)∗10^8\0\Pa,0.35}]") ={1};
```

When Example3.msh is translated using gmESSI, it creates 8-noded bricks and assign different matrials layer-wise. The content of Example3_analysis.fei file contains all new declared materials as shown below

```
$ cat Example3_analysis.fei

  add material #1 type linear_elastic_isotropic_3d mass_density=1505.000000*kg/m^3
        elastic_modulus=1904999936.000000*Pa poisson_ratio=0.35;
  add material #2 type linear_elastic_isotropic_3d mass_density=1515.000000*kg/m^3
        elastic_modulus=1915000064.000000*Pa poisson_ratio=0.35;
  add material #3 type linear_elastic_isotropic_3d mass_density=1525.000000*kg/m^3
        elastic_modulus=1924999936.000000*Pa poisson_ratio=0.35;
  add material #4 type linear_elastic_isotropic_3d mass_density=1535.000000*kg/m^3
        elastic_modulus=1935000064.000000*Pa poisson_ratio=0.35;
  add material #5 type linear_elastic_isotropic_3d mass_density=1545.000000*kg/m^3
        elastic_modulus=1944999936.000000*Pa poisson_ratio=0.35;
  add material #6 type linear_elastic_isotropic_3d mass_density=1555.000000*kg/m^3
        elastic_modulus=1955000064.000000*Pa poisson_ratio=0.35;
  add material #7 type linear_elastic_isotropic_3d mass_density=1565.000000*kg/m^3
        elastic_modulus=1964999936.000000*Pa poisson_ratio=0.35;
  add material #8 type linear_elastic_isotropic_3d mass_density=1575.000000*kg/m^3
        elastic_modulus=1975000064.000000*Pa poisson_ratio=0.35;
  add material #9 type linear_elastic_isotropic_3d mass_density=1585.000000*kg/m^3
        elastic_modulus=1984999936.000000*Pa poisson_ratio=0.35;
  add material #10 type linear_elastic_isotropic_3d mass_density=1595.000000*kg/m^3
        elastic_modulus=1995000064.000000*Pa poisson_ratio=0.35;
```

```
$ cat Example3_geometry.fei
  .....................
  add element #201 type 8NodeBrick with nodes (1,9,117,27,81,198,603,441) use material #1;
  add element #202 type 8NodeBrick with nodes (81,198,603,441,82,199,604,442) use material #2;
  add element #203 type 8NodeBrick with nodes (82,199,604,442,83,200,605,443) use material #3;
  add element #204 type 8NodeBrick with nodes (83,200,605,443,84,201,606,444) use material #4;
  add element #205 type 8NodeBrick with nodes (84,201,606,444,85,202,607,445) use material #5;
  add element #206 type 8NodeBrick with nodes (85,202,607,445,86,203,608,446) use material #6;
  add element #207 type 8NodeBrick with nodes (86,203,608,446,87,204,609,447) use material #7;
  add element #208 type 8NodeBrick with nodes (87,204,609,447,88,205,610,448) use material #8;
  add element #209 type 8NodeBrick with nodes (88,205,610,448,89,206,611,449) use material #9;
  add element #210 type 8NodeBrick with nodes (89,206,611,449,5,45,522,80) use material #10;
  add element #211 type 8NodeBrick with nodes (27,117,118,28,441,603,612,450) use material #1;
  add element #212 type 8NodeBrick with nodes (441,603,612,450,442,604,613,451) use material #2;
  add element #213 type 8NodeBrick with nodes (442,604,613,451,443,605,614,452) use material #3;
  add element #214 type 8NodeBrick with nodes (443,605,614,452,444,606,615,453) use material #4;
  add element #215 type 8NodeBrick with nodes (444,606,615,453,445,607,616,454) use material #5;
  .................
```

In Example3_analysis.fei file, all the materials has been created for each layer and it can be seen that *mass_density* and *elastic_modulus* increases down the dept while poisson's ratio is constant. Similarly, Example3_geometry.fei shows how 8-nodeBricks are defined with different materials layer-wise. Different commands available in this category are listed below

- gmESSI :: [Vary_Linear_elastic_isotropic_3d{ Elemental_Command {without_material_no_argument}, mass_density, elastic_modulus, poisson_ratio}]
  ESSI :: add material #{} type linear_elastic_isotropic_3d mass_density={} elastic_modulus={} poisson_ratio={}
  Description :: *add material* $\# < . >$ *type linear_elastic_isotropic_3d mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$

- gmESSI :: [Vary_Linear_elastic_crossanisotropic{Elemental_Command {without_material_no_argument}, mass_density, elastic_modulus_horizontal, elastic_modulus_vertical, poisson_ratio_h_v, poisson_ratio_h_h, shear_modulus_h_v}]

ESSI :: add material #{} type linear_elastic_crossanisotropic mass_density={} elastic_modulus_horizontal={} elastic_modulus_vertical={} poisson_ratio_h_v={} poisson_ratio_h_h={} shear_modulus_h_v={}
Description :: *add material* $\#<.>$ *type linear_elastic_crossanisotropic mass_density* $= <$ *mass_density* $>$ *elastic_modulus_horizontal* $= < F/L^2 >$ *elastic_modulus_vertical* $= < F/L^2 >$ *poisson_ratio_h_v* $= < . >$ *poisson_ratio_h_h* $= < . >$ *shear_modulus_h_v* $= < F/L^2 >$

- gmESSI :: [Vary_Vonmises_perfectly_plastic{ Elemental_Command {without_material_no_argument}, mass_density, elastic_modulus, poisson_ratio, von_mises_radius, initial_confining_stress, initial_confining_stress, algorithm, number_of_subincrements, maximum_number_of_iterations, tolerance_1, tolerance_2}]
ESSI :: add material #{} type vonmises_perfectly_plastic mass_density={} elastic_modulus={} poisson_ratio={} von_mises_radius={} initial_confining_stress={} algorithm={} number_of_subincrements={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
Description :: *add material* $\#<.>$ *type vonmises_perfectly_plastic mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *von_mises_radius* $= < F/L^2 >$ *initial_confining_stress* $= < F/L^2 >$ *algorithm* $= <$ *explicit|implicit* $>$ *number_of_subincrements* $= < . >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Vary_Vonmises_perfectly_plastic_accelerated{ Elemental_Command {without_material_no_argument}, mass_density, elastic_modulus, poisson_ratio, von_mises_radius, initial_confining_stress, maximum_number_of_iterations, tolerance_1, tolerance_2}]
ESSI :: add material #{} type vonmises_perfectly_plastic_accelerated mass_density={} elastic_modulus={} poisson_ratio={} von_mises_radius={} initial_confining_stress={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
Description :: *add material* $\#<.>$ *type vonmises_perfectly_plastic_accelerated mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *von_mises_radius* $= < F/L^2 >$ *initial_confining_stress* $= < F/L^2 >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Vary_Vonmises_isotropic_hardening{ Elemental_Command {without_material_no_argument}, mass_density, elastic_modulus, poisson_ratio, von_mises_radius, isotropic_hardening_rate, initial_confining_stress, algorithm, number_of_subincrements, maximum_number_of_iterations, tolerance_1, tolerance_2}]

ESSI :: add material #{} type vonmises_isotropic_hardening mass_density={} elastic_modulus={} poisson_ratio={} von_mises_radius={} isotropic_hardening_rate={} initial_confining_stress={} algorithm={} number_of_subincrements={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
Description :: *add material* $\# < . >$ *type vonmises_isotropic_hardening mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *von_mises_radius* $= < F/L^2 >$ *isotropic_hardening_rate* $= < F/L^2 >$ *initial_confining_stress* $= < F/L^2 >$ *algorithm* $= < explicit|implicit >$ *number_of_subincrements* $= < . >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Vary_Vonmises_isotropic_hardening_accelerated{ Elemental_Command {without_material_no_argument}, mass_density, elastic_modulus, poisson_ratio, von_mises_radius, isotropic_hardening_rate, initial_confining_stress, maximum_number_of_iterations, tolerance_1, tolerance_2}]
  ESSI :: add material #{} type vonmises_isotropic_hardening_accelerated mass_density={} elastic_modulus={} poisson_ratio={} von_mises_radius={} isotropic_hardening_rate={} initial_confining_stress={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
  Description :: *add material* $\# < . >$ *type vonmises_isotropic_hardening_accelerated mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *von_mises_radius* $= < F/L^2 >$ *isotropic_hardening_rate* $= < F/L^2 >$ *initial_confining_stress* $= < F/L^2 >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Vary_Vonmises_kinematic_hardening{ Elemental_Command {without_material_no_argument}, mass_density, elastic_modulus, poisson_ratio, von_mises_radius, armstrong_frederick_ha, armstrong_frederick_cr, initial_confining_stress, algorithm, number_of_subincrements, maximum_number_of_iterations, tolerance_1, tolerance_2}]
  ESSI :: add material #{} type vonmises_kinematic_hardening mass_density={} elastic_modulus={} poisson_ratio={} von_mises_radius={} armstrong_frederick_ha={} armstrong_frederick_cr={} initial_confining_stress={} algorithm={} number_of_subincrements={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
  Description :: *add material* $\# < . >$ *type vonmises_kinematic_hardening mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *von_mises_radius* $= < F/L^2 >$ *armstrong_frederick_ha* $= < F/L^2 >$

armstrong_frederick_cr $= < F/L^2 >$ *initial_confining_stress* $= < F/L^2 >$ *algorithm* $= <$ *explicit*$|$*implicit* $>$ *number_of_subincrements* $= < . >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Vary_Vonmises_linear_kinematic_hardening{ Elemental_Command {without_material_no_argument}, mass_density, elastic_modulus, poisson_ratio, von_mises_radius, kinematic_hardening_rate, initial_confining_stress, algorithm, number_of_subincrements, maximum_number_of_iterations, tolerance_1, tolerance_2}]
  ESSI :: add material #{} type vonmises_linear_kinematic_hardening mass_density={} elastic_modulus={} poisson_ratio={} von_mises_radius={} kinematic_hardening_rate={} initial_confining_stress={} algorithm={} number_of_subincrements={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
  Description :: *add material* # $< . >$ *type vonmises_linear_kinematic_hardening mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *von_mises_radius* $= < F/L^2 >$ *kinematic_hardening_rate* $= < . >$ *initial_confining_stress* $= < F/L^2 >$ *algorithm* $= <$ *explicit*$|$*implicit* $>$ *number_of_subincrements* $= < . >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Vary_Vonmises_kinematic_hardening_accelerated{ Elemental_Command {without_material_no_argument}, mass_density, elastic_modulus, poisson_ratio, von_mises_radius, armstrong_frederick_ha, armstrong_frederick_cr, initial_confining_stress, initial_confining_strain, maximum_number_of_iterations, tolerance_1, tolerance_2}]
  ESSI :: add material #{} type vonmises_kinematic_hardening_accelerated mass_density={} elastic_modulus={} poisson_ratio={} von_mises_radius={} armstrong_frederick_ha={} armstrong_frederick_cr={} initial_confining_stress={} initial_confining_strain={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
  Description :: *add material* # $< . >$ *type vonmises_kinematic_hardening_accelerated mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *von_mises_radius* $= < F/L^2 >$ *armstrong_frederick_ha* $= < F/L^2 >$ *armstrong_frederick_cr* $= < F/L^2 >$ *initial_confining_stress* $= < F/L^2 >$ *initial_confining_strain* $= < . >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Vary_Vonmises_linear_kinematic_hardening_accelerated{ Elemental_Command {without_material_no_argument}, mass_density,

elastic_modulus, poisson_ratio, von_mises_radius, kinematic_hardening_rate, initial_confining_stress, initial_confining_strain, maximum_number_of_iterations, tolerance_1, tolerance_2}]
ESSI :: add material #{} type vonmises_linear_kinematic_hardening_accelerated mass_density={} elastic_modulus={} poisson_ratio={} von_mises_radius={} kinematic_hardening_rate={} initial_confining_stress={} initial_confining_strain={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
Description :: *add material* $\#<.>$ *type vonmises_linear_kinematic_hardening_accelerated mass_density* $= <M/L^3>$ *elastic_modulus* $= <F/L^2>$ *poisson_ratio* $= <.>$ *von_mises_radius* $= <F/L^2>$ *kinematic_hardening_rate* $= <.>$ *initial_confining_stress* $= <F/L^2>$ *initial_confining_strain* $= <.>$ *maximum_number_of_iterations* $= <.>$ *tolerance_1* $= <.>$ *tolerance_2* $= <.>$

- gmESSI :: [Vary_Vonmises_perfectly_plastic_LT{ Elemental_Command {without_material_no_argument}, mass_density, elastic_modulus, poisson_ratio, von_mises_radius,initial_confining_stress, algorithm, number_of_subincrements, maximum_number_of_iterations, tolerance_1, tolerance_2}]
  ESSI :: add material #{} type vonmises_perfectly_plastic_LT mass_density={} elastic_modulus={} poisson_ratio={} von_mises_radius={} initial_confining_stress= {} algorithm={} number_of_subincrements={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
  Description :: *add material* $\#<.>$ *type vonmises_perfectly_plastic_LT mass_density* $= <M/L^3>$ *elastic_modulus* $= <F/L^2>$ *poisson_ratio* $= <.>$ *von_mises_radius* $= <F/L^2>$ *initial_confining_stress* $= <F/L^2>$ *maximum_number_of_iterations* $= <.>$ *tolerance_1* $= <.>$ *tolerance_2* $= <.>$

- gmESSI :: [Vary_VonmisesLT{ Elemental_Command {without_material_no_argument}, mass_density, elastic_modulus, poisson_ratio, vonmises_radius, kinematic_hardening_rate, isotropic_hardening_rate}]
  ESSI :: add material #{} type VonMisesLT mass_density={} elastic_modulus={} poisson_ratio={} von_mises_radius={} kinematic_hardening_rate={} isotropic_hardening_rate={}
  Description :: *add material* $\#<.>$ *type VonMisesLT mass_density* $= <M/L^3>$ *elastic_modulus* $= <F/L^2>$ *poisson_ratio* $= <.>$ *von_mises_radius* $= <F/L^2>$ *kinematic_hardening_rate* $= <F/L^2>$ *isotropic_hardening_rate* $= <F/L^2>$

- gmESSI :: [Vary_Druckerprager_perfectly_plastic{ Elemental_Command {without_material_no_argument}, mass_density, elastic_modulus, pois-

son_ratio, druckerprager_angle, initial_confining_stress, algorithm, number_of_subincrements, maximum_number_of_iterations, tolerance_1, tolerance_2}]
ESSI :: add material #{} type druckerprager_perfectly_plastic mass_density={} elastic_modulus={} poisson_ratio={} druckerprager_angle={} initial_confining_stress={} algorithm={} number_of_subincrements={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
Description :: *add material # < . > type druckerprager_perfectly_plastic mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *druckerprager_angle* $= < F/L^2 >$ *initial_confining_stress* $= < F/L^2 >$ *algorithm* $= <$ *explicit|implicit* $>$ *number_of_subincrements* $= < . >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Vary_Druckerprager_perfectly_plastic_accelerated{ Elemental_Command {without_material_no_argument}, mass_density, elastic_modulus, poisson_ratio, druckerprager_angle, initial_confining_stress, maximum_number_of_iterations, tolerance_1, tolerance_2}]
ESSI :: add material #{} type druckerprager_perfectly_plastic_accelerated mass_density={} elastic_modulus={} poisson_ratio={} druckerprager_angle={} initial_confining_stress={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
Description :: *add material # < . > type druckerprager_perfectly_plastic_accelerated mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *druckerprager_angle* $= < F/L^2 >$ *initial_confining_stress* $= < F/L^2 >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Vary_Druckerprager_isotropic_hardening{ Elemental_Command {without_material_no_argument}, mass_density, elastic_modulus, poisson_ratio, druckerprager_angle, isotropic_hardening_rate, initial_confining_stress, algorithm, number_of_subincrements, maximum_number_of_iterations, tolerance_1, tolerance_2}]
ESSI :: add material #{} type druckerprager_isotropic_hardening mass_density={} elastic_modulus={} poisson_ratio={} druckerprager_angle={} isotropic_hardening_rate={} initial_confining_stress={} algorithm={} number_of_subincrements={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
Description :: *add material # < . > type druckerprager_isotropic_hardening mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *druckerprager_angle* $= < F/L^2 >$ *isotropic_hardening_rate* $= < F/L^2 >$

initial_confining_stress = < F/L^2 > *algorithm* = < *explicit|implicit* > *number_of_subincrements* = < . > *maximum_number_of_iterations* = < . > *tolerance_1* = < . > *tolerance_2* = < . >

- gmESSI :: [Vary_Druckerprager_isotropic_hardening_accelerated{ Elemental_Command {without_material_no_argument}, mass_density, elastic_modulus, poisson_ratio, druckerprager_angle, isotropic_hardening_rate, initial_confining_stress, maximum_number_of_iterations, tolerance_1, tolerance_2}]
  ESSI :: add material #{} type druckerprager_isotropic_hardening_accelerated mass_density={} elastic_modulus={} poisson_ratio={} druckerprager_angle={} isotropic_hardening_rate={} initial_confining_stress={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
  Description :: *add material #* < . > *type druckerprager_isotropic_hardening_accelerated mass_density* = < M/L^3 > *elastic_modulus* = < F/L^2 > *poisson_ratio* = < . > *druckerprager_angle* = < F/L^2 > *isotropic_hardening_rate* = < F/L^2 > *initial_confining_stress* = < F/L^2 > *maximum_number_of_iterations* = < . > *tolerance_1* = < . > *tolerance_2* = < . >

- gmESSI :: [Vary_Druckerprager_kinematic_hardening{ Elemental_Command {without_material_no_argument}, mass_density, elastic_modulus, poisson_ratio, druckerprager_angle, armstrong_frederick_ha, armstrong_frederick_cr, initial_confining_stress, algorithm, number_of_subincrements, maximum_number_of_iterations, tolerance_1, tolerance_2}]
  ESSI :: add material #{} type druckerprager_kinematic_hardening mass_density={} elastic_modulus={} poisson_ratio={} druckerprager_angle={} armstrong_frederick_ha={} armstrong_frederick_cr={} initial_confining_stress={} algorithm= {} number_of_subincrements={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
  Description :: *add material #* < . > *type druckerprager_kinematic_hardening mass_density* = < M/L^3 > *elastic_modulus* = < F/L^2 > *poisson_ratio* = < . > *druckerprager_angle* = < F/L^2 > *armstrong_frederick_ha* = < F/L^2 > *armstrong_frederick_cr* = < F/L^2 > *initial_confining_stress* = < F/L^2 > *algorithm* = < *explicit|implicit* > *number_of_subincrements* = < . > *maximum_number_of_iterations* = < . > *tolerance_1* = < . > *tolerance_2* = < . >

- gmESSI :: [Vary_Druckerprager_kinematic_hardening_accelerated{ Elemental_Command {without_material_no_argument}, mass_density, elastic_modulus, poisson_ratio, druckerprager_angle, armstrong_frederick_ha, armstrong_frederick_cr, initial_confining_stress, maximum_number_of_iterations,

tolerance_1, tolerance_2}]
ESSI :: add material #{} type drucker-prager_kinematic_hardening_accelerated mass_density={} elastic_modulus={} poisson_ratio={} druckerprager_angle={} armstrong_frederick_ha={} armstrong_frederick_cr={} initial_confining_stress={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
Description :: *add material # < . > type drucker-prager_kinematic_hardening_accelerated mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *druckerprager_angle* $= < F/L^2 >$ *armstrong_frederick_ha* $= < F/L^2 >$ *armstrong_frederick_cr* $= < F/L^2 >$ *initial_confining_stress* $= < F/L^2 >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Vary_DruckerpragerLT{ Elemental_Command {without_material_no_argument}, mass_density, elastic_modulus, poisson_ratio, druckerprager_k, kinematic_hardening_rate, isotropic_hardening_rate}]
ESSI :: add material #{} type DruckerPragerLT mass_density={} elastic_modulus={} poisson_ratio={} druckerprager_k={} kinematic_hardening_rate={} isotropic_hardening_rate={}
Description :: *add material # < . > type DruckerPragerLT mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *druckerprager_k* $= < F/L^2 >$ *kinematic_hardening_rate* $= < F/L^2 >$ *isotropic_hardening_rate* $= < F/L^2 >$

- gmESSI :: [Vary_Camclay{ Elemental_Command {without_material_no_argument}, mass_density, reference_void_ratio, critical_stress_ratio_M, lambda, kappa, poisson_ratio, minimum_bulk_modulus, pressure_reference_p0, initial_confining_stress, algorithm, number_of_subincrements, maximum_number_of_iterations, tolerance_1, tolerance_2}]
ESSI :: add material #{} type camclay mass_density={} reference_void_ratio={} critical_stress_ratio_M={} lambda={} kappa={} poisson_ratio={} minimum_bulk_modulus={} pressure_reference_p0={} initial_confining_stress={} algorithm={} number_of_subincrements={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
Description :: *add material # < . > type camclay mass_density* $= < M/L^3 >$ *reference_void_ratio* $= < . >$ *critical_stress_ratio_M* $= < . >$ *lambda* $= < . >$ *kappa* $= < . >$ *poisson_ratio* $= < . >$ *minimum_bulk_modulus* $= < F/L^2 >$ *pressure_reference_p0* $= < F/L^2 >$ *initial_confining_stress* $= < F/L^2 >$ *algorithm* $= < explicit|implicit >$ *number_of_subincrements* $= < . >$ *maximum_number_of_iterations* $= < . >$ *tolerance_1* $= < . >$ *tolerance_2* $= < . >$

- gmESSI :: [Vary_Camclay_accelerated{ Elemental_Command {without_material_no_argument}, mass_density, reference_void_ratio, critical_stress_ratio_M, lambda, kappa, poisson_ratio, minimum_bulk_modulus, pressure_reference_p0, initial_confining_stress, maximum_number_of_iterations, tolerance_1, tolerance_2}]
ESSI :: add material #{} type camclay_accelerated mass_density={} reference_void_ratio={} critical_stress_ratio_M={} lambda={} kappa={} poisson_ratio={} minimum_bulk_modulus={} pressure_reference_p0={} initial_confining_stress={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
Description :: *add material # < . > type camclay_accelerated mass_density* = < M/L^3 > *reference_void_ratio* = < . > *critical_stress_ratio_M* = < . > *lambda* = < . > *kappa* = < . > *poisson_ratio* = < . > *minimum_bulk_modulus* = < F/L^2 > *pressure_reference_p0* = < F/L^2 > *initial_confining_stress* = < F/L^2 > *maximum_number_of_iterations* = < . > *tolerance_1* = < . > *tolerance_2* = < . >

- gmESSI :: [Vary_Sanisand2004{ Elemental_Command {without_material_no_argument}, mass_density, e0, sanisand2004_G0, poisson_ratio, sanisand2004_Pat, sanisand2004_p_cut, sanisand2004_Mc, sanisand2004_c, sanisand2004_lambda_c, sanisand2004_xi, sanisand2004_ec_ref, sanisand2004_m, sanisand2004_h0, sanisand2004_ch, sanisand2004_nb, sanisand2004_A0, sanisand2004_nd, sanisand2004_z_max, sanisand2004_cz, initial_confining_stress, algorithm, number_of_subincrements, maximum_number_of_iterations, tolerance_1, tolerance_2}]
ESSI :: add material #{} type sanisand2004 mass_density={} e0={} sanisand2004_G0={} poisson_ratio={} sanisand2004_Pat={} sanisand2004_p_cut={} sanisand2004_Mc={} sanisand2004_c={} sanisand2004_lambda_c={} sanisand2004_xi={} sanisand2004_ec_ref={} sanisand2004_m={} sanisand2004_h0={} sanisand2004_ch={} sanisand2004_nb={} sanisand2004_A0={} sanisand2004_nd={} sanisand2004_z_max={} sanisand2004_cz={} initial_confining_stress={} algorithm={} number_of_subincrements={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
Description :: *add material # < . > type sanisand2004 mass_density* = < M/L^3 > *e0* = < . > *sanisand2004_G0* = < . > *poisson_ratio* = < . > *sanisand2004_Pat* = < *stress* > *sanisand2004_p_cut* = < . > *sanisand2004_Mc* = < . > *sanisand2004_c* = < . > *sanisand2004_lambda_c* = < . > *sanisand2004_xi* = < . > *sanisand2004_ec_ref* = < . > *sanisand2004_m* = < . > *sanisand2004_h0* = < . > *sanisand2004_ch* = < . > *sanisand2004_nb* = < . > *sanisand2004_A0* = < . > *sanisand2004_nd* = < . > *sanisand2004_z_max* = < . > *sanisand2004_cz* = < . > *initial_confining_stress* = < *stress* > *algorithm* = < *explicit|implicit* > *number_of_subincrements* = < . > *max-*

imum_number_of_iterations = < . > tolerance_1 = < . > tolerance_2 = < . >

- gmESSI :: [Vary_Sanisand2008{ Elemental_Command {without_material_no_argument}, mass_density, e0, sanisand2008_G0, sanisand2008_K0, sanisand2008_Pat, sanisand2008_k_c, sanisand2008_alpha_cc, sanisand2008_c, sanisand2008_xi, sanisand2008_lambda, sanisand2008_ec_ref, sanisand2008_m, sanisand2008_h0, sanisand2008_ch, sanisand2008_nb, sanisand2008_A0, sanisand2008_nd, sanisand2008_p_r, sanisand2008_rho_c, sanisand2008_theta_c, sanisand2008_X, sanisand2008_z_max, sanisand2008_cz, sanisand2008_p0, sanisand2008_p_in, algorithm, number_of_subincrements, maximum_number_of_iterations, tolerance_1, tolerance_2}]
ESSI :: add material #{} type sanisand2008 mass_density={} e0={} sanisand2008_G0={} sanisand2008_K0={} sanisand2008_Pat={} sanisand2008_k_c={} sanisand2008_alpha_cc={} sanisand2008_c={} sanisand2008_xi={} sanisand2008_lambda={} sanisand2008_ec_ref={} sanisand2008_m={} sanisand2008_h0={} sanisand2008_ch={} sanisand2008_nb={} sanisand2008_A0={} sanisand2008_nd={} sanisand2008_p_r={} sanisand2008_rho_c={} sanisand2008_theta_c={} sanisand2008_X={} sanisand2008_z_max={} sanisand2008_cz={} sanisand2008_p0={} sanisand2008_p_in={} algorithm={} number_of_subincrements={} maximum_number_of_iterations={} tolerance_1={} tolerance_2={}
Description :: *add material # < . > type sanisand2008 mass_density = <* M/L^3 *> e0 = < . > sanisand2008_G0 = < . > sanisand2008_K0 = < . > sanisand2008_Pat = < stress > sanisand2008_k_c = < . > sanisand2008_alpha_cc = < . > sanisand2008_c = < . > sanisand2008_xi = < . > sanisand2008_lambda = < . > sanisand2008_ec_ref = < . > sanisand2008_m = < . > sanisand2008_h0 = < . > sanisand2008_ch = < . > sanisand2008_nb = < . > sanisand2008_A0 = < . > sanisand2008_nd = < . > sanisand2008_p_r = < . > sanisand2008_rho_c = < . > sanisand2008_theta_c = < . > sanisand2008_X = < . > sanisand2008_z_max = < . > sanisand2008_cz = < . > sanisand2008_p0 = < stress > sanisand2008_p_in = < . > algorithm = < explicit|implicit > number_of_subincrements = < . > maximum_number_of_iterations = < . > tolerance_1 = < . > tolerance_2 = < . >*

- gmESSI :: [Vary_Uniaxial_elastic{ Elemental_Command {without_material_no_argument}, elastic_modulus, viscoelastic_modulus}]
ESSI :: add material #{} type uniaxial_elastic elastic_modulus={} viscoelastic_modulus={}
Description :: *add material # < . > type uniaxial_elastic elastic_modulus =*

$< F/L^2 >$ *viscoelastic_modulus* $= <$ *mass/length/time* $>$

- gmESSI :: [Vary_Uniaxial_concrete02{ Elemental_Command {without_material_no_argument}, comprESSIve_strength, strain_at_comprESSIve_strength, crushing_strength, strain_at_crushing_strength, lambda, tensile_strength, tension_softening_stiffness}]
  ESSI :: add material #{} type uniaxial_concrete02 comprESSIve_strength={} strain_at_comprESSIve_strength= {} crushing_strength= {} strain_at_crushing_strength= {} lambda={} tensile_strength={} tension_softening_stiffness={}
  Description :: *add material # < . > type uniaxial_concrete02 comprESSIve_strength* $= < F/L^2 >$ *strain_at_comprESSIve_strength* $= < . >$ *crushing_strength* $= < F/L^2 >$ *strain_at_crushing_strength* $= < . >$ *lambda* $= < . >$ *tensile_strength* $= < F/L^2 >$ *tension_softening_stiffness* $= < F/L^2 >$

- gmESSI :: [Vary_Uniaxial_steel01{ Elemental_Command {without_material_no_argument},yield_strength, elastic_modulus, strain_hardening_ratio, a1, a2, a3, a4}]
  ESSI :: add material #{} type uniaxial_steel01 yield_strength={} elastic_modulus={} strain_hardening_ratio={} a1={} a2={} a3={} a4={}
  Description :: *add material # < . > type uniaxial_steel01 yield_strength* $= < F/L^2 >$ *elastic_modulus* $= < F/L^2 >$ *strain_hardening_ratio* $= < . >$ *a1* $= < . >$ *a2* $= < . >$ *a3* $= <>$ *a4* $= < . >$

- gmESSI :: [Vary_Uniaxial_steel02{ Elemental_Command {without_material_no_argument},yield_strength, elastic_modulus, strain_hardening_ratio, R0, cR1, cR2, a1, a2, a3, a4}]
  ESSI :: add material #{} type uniaxial_steel02 yield_strength={} elastic_modulus={} strain_hardening_ratio={} R0={} cR1={} cR2={} a1={} a2={} a3={} a4={}
  Description :: *add material # < . > type uniaxial_steel02 yield_strength* $= < F/L^2 >$ *elastic_modulus* $= < F/L^2 >$ *strain_hardening_ratio* $= < . >$ *R0* $= < . >$ *cR1* $= < . >$ *cR2* $= < . >$ *a1* $= < . >$ *a2* $= < . >$ *a3* $= <>$ *a4* $= < . >$

- gmESSI :: [Vary_New_PisanoLT{ Elemental_Command {without_material_no_argument},mass_density, elastic_modulus, poisson_ratio, M_in, kd_in, xi_in, h_in, m_in, initial_confining_stress, n_in, a_in, eplcum_cr_in}]
  ESSI :: add material #{} type New_PisanoLT mass_density={} elastic_modulus_1atm={} poisson_ratio={} M_in={} kd_in={} xi_in={} h_in={} m_in={} initial_confining_stress={} n_in ={} a_in ={} eplcum_cr_in ={}
  Description :: *add material # < . > type New_PisanoLT mass_density* =

$< M/L^2 >$ *elastic_modulus_1atm* $= < F/L^2 >$ *poisson_ratio* $= < . >$ *M_in* $= < F/L^2 >$ *kd_in* $= < . >$ *xi_in* $= < . >$ *h_in* $= < . >$ *m_in* $= < . >$ *initial_confining_stress* $= < F/L^2 >$ *n_in* $= < . >$ *a_in* $= < . >$ *eplcum_cr_in* $= < . >$

- gmESSI :: [Vary_Linear_elastic_isotropic_3d_LT{ Elemental_Command {without_material_no_argument}, mass_density, elastic_modulus, poisson_ratio}]
  ESSI :: add material #{} type linear_elastic_isotropic_3d_LT mass_density={} elastic_modulus={} poisson_ratio={}]
  Description :: *add material #* $< . >$ *type linear_elastic_isotropic_3d_LT mass_density* $= < M/L^3 >$ *elastic_modulus* $= < F/L^2 >$ *poisson_ratio* $= < . >$

6.6.3 Nodal Variational command

Nodal Variational commands has the similar function as Material variational command in which each parameter is like a mathematical function of coordinates i.e $f(x, y, z)$ which is evaluated in octave. Whatever is valid for octave is valid for all variational commands. Also similiar to Material variational command this command has two optional parameters **precision** and **unit** separate by separator '\'. Various commands under this category are
Note:- Every Nodal Variational command has the option of adding Physical_Group# or Entity_Group# argument in the front.

- gmESSI :: [Vary_Node_Mass{mx,my,mz}]
  ESSI :: Vary mass to node #{} mx ={} my ={} mz ={}
  Description :: *Vary mass to node #* $< . >$ *mx* $= <$ *mass* $>$ *my* $= <$ *mass* $>$ *mz* $= <$ *mass* $>$

- gmESSI :: [Vary_Node_Mass{mx,my,mz,Imx,Imy,Imz}]
  ESSI :: Vary mass to node #{} mx ={} my ={} mz ={} Imx ={} Imy ={} Imz ={}
  Description :: *Vary mass to node #* $< . >$ *mx* $= <$ *mass* $>$ *my* $= <$ *mass* $>$ *mz* $= <$ *mass* $>$ *Imz* $= < mass * length^2 >$ *Imy* $= < mass * length^2 >$ *Imz* $= < mass * length^2 >$

- gmESSI :: [Vary_Node_Damping{damping_no}]
  ESSI :: Vary damping #{} to node #{}
  Description :: *Vary damping #* $< . >$ *to node #* $< . >$

- gmESSI :: [Vary_Fix{dofs}]
  ESSI :: fix node #{} dofs {}
  Description :: *Vary damping #* $< . >$ *to node #* $< . >$

- gmESSI :: [Vary_Free{dofs}]
  ESSI :: free node #{} dofs {}
  Description :: *free node #* $< . >$ *dofs* $< . >$

- gmESSI :: [Vary_Master_Slave{master_node, dof}]
  ESSI :: Vary constraint equal dof with master node #{} and slave node #{} dof to constrain {}
  Description :: *Vary constraint equal dof with master node # < . > and slave node # < . > dof to constrain < . >*

- gmESSI :: [Vary_Single_Point_Constraint{constrain_dof, constrain_value}]
  ESSI :: Vary single point constraint to node #{} dof to constrain {} constraint value of {}
  Description :: *Vary single point constraint to node # < . > dof to constrain < dof_type > constraint value of < correspondingunit >;*

- gmESSI :: [Vary_Node_Load_Linear{direction, magnitude}]
  ESSI :: Vary load #{} to node #{} type linear {}= {}
  Description :: *Vary load # < . > to node # < . > type linear FORCETYPE = < force|moment > //FORCETYPE = Fx Fy Fz Mx My Mz F_fluid_x F_fluid_y F_fluid_z*

- gmESSI :: [Vary_Node_Load_Path_Series{direction, magnitude, series_file}]
  ESSI :: Vary load #{} to node #{} type path_series {} = {} series_file ={}
  Description :: *Vary load # < . > to node # < . > type path_series FORCETYPE = < forceormoment > time_step = < time > series_file = "STRING"*

- gmESSI :: [Vary_Node_Load_Path_Time_Series{direction, magnitude, time_step, series_file}]
  ESSI :: Vary load #{} to node #{} type path_time_series {} = {} time_step = {} series_file ={}
  Description :: *Vary load # < . > to node # < . > type path_time_series FORCETYPE = < forceormoment > series_file = "STRING"*

- gmESSI :: [Vary_Imposed_Motion_Time_Series{dof, time_step, disp_scale_unit, disp_file, vel_scale_unit, vel_file, acc_scale_unit, acc_file}]
  ESSI :: Vary imposed motion #{} to node #{} dof {} time_step = {} displacement_scale_unit = {} displacement_file = {} velocity_scale_unit = {} velocity_file = {} acceleration_scale_unit = {} acceleration_file ={}
  Description :: *Vary imposed motion # < . > to node # < . > dof DOFTYPE time_step = < t > displacement_scale_unit = < length > displacement_file = "filename" velocity_scale_unit = < velocity > velocity_file = "filename" acceleration_scale_unit = < acceleration > acceleration_file = "filename";*

- gmESSI :: [Vary_Imposed_Motion{dof, disp_scale_unit, disp_file, vel_scale_unit, vel_file, acc_scale_unit, acc_file}]
  ESSI :: Vary imposed motion #{} to node #{} dof {} displacement_scale_unit = {} displacement_file = {} velocity_scale_unit =

{} velocity_file = {} acceleration_scale_unit = {} acceleration_file ={}
Description :: *Vary imposed motion # < . > to node # < . > dof DOFTYPE displacement_scale_unit = < length > displacement_file = "filename" velocity_scale_unit = < velocity > velocity_file = "filename" acceleration_scale_unit = < acceleration > acceleration_file = "filename";*

- gmESSI :: [Vary_Node_Self_Weight{field_no}]
  ESSI :: Vary load #{} to node #{} type self_weight use acceleration field #{}
  Description :: *Vary load # < . > to node # < . > type self_weight use acceleration field # < . >*

6.6.4 Elemental Variational Command

Elemental Variational commands have the similar function as other variational commands in which each parameter is like a mathematical function of coordinates i.e $f(x, y, z)$ which is evaluated in octave. Whatever is valid for octave is valid for all variational commands. Also similiar to other variational commands this command has two optional parameters **precision** and **unit** separate by separator '\'. Various commands under this category are
Note:- Every Elemental Variational command has the option of adding Physical_Group# or Entity_Group# argument in the front.

- gmESSI :: [Vary_Beam_Elastic_Lumped_Mass{ cross_section, elastic_modulus, shear_modulus, torsion_Jx, bending_Iy, bending_Iz, mass_density, xz_plane_vector_x, xz_plane_vector_y, xz_plane_vector_z, joint_1_offset_1, joint_1_offset_2, joint_1_offset_3, joint_2_offset_1, joint_2_offset_2, joint_2_offset_3}]
  ESSI :: Vary element #{} type beam_elastic_lumped_mass with nodes ({},{}) cross_section={} elastic_modulus={} shear_modulus={} torsion_Jx={} bending_Iy={} bending_Iz={} mass_density={} xz_plane_vector=({},{},{}) joint_1_offset=({},{},{}) joint_2_offset=({},{},{})
  Description :: *Vary element # < . > type beam_elastic_lumped_mass with nodes (< . >, < . >) cross_section = < area > elastic_modulus =* $< F/L^2 >$ *shear_modulus =* $< F/L^2 >$ *torsion_Jx =* $< length^4 >$ *bending_Iy =* $< length^4 >$ *bending_Iz =* $< length^4 >$ *mass_density =* $< M/L^3 >$ *xz_plane_vector = (< . >, < . >, < . >) joint_1_offset = (< L >, < L >, < L >) joint_2_offset = (< L >, < L >, < L >)*

- gmESSI :: [Vary_Truss{material_no,cross_section, mass_density}]
  ESSI :: Vary element #{} type truss with nodes ({},{}) use material #{} cross_section ={} mass_density={}

Description :: *Vary element # $<\ .\ >$ type truss with nodes ($<\ .\ >$, $<\ .\ >$) use material # $<\ .\ >$ cross_section = $<length^2>$ mass_density = $<M/L^3>$*

- gmESSI :: [Vary_Frictional_Penalty_Contact{normal_stiffness, tangential_stiffness, normal_damping, tangential_damping, friction_ratio, contact_plane_vector_x, contact_plane_vector_y, contact_plane_vector_z}]
  ESSI :: Vary element #{} type FrictionalPenaltyContact with nodes ({},{}) normal_stiffness={} tangential_stiffness={} normal_damping={} tangential_damping={} friction_ratio={} contact_plane_vector= ({},{},{})
  Description :: *Vary element # $<\ .\ >$ type FrictionalPenaltyContact with nodes ($<\ .\ >$, $<\ .\ >$) normal_stiffness = $<F/L>$ tangential_stiffness = $<F/L>$ normal_damping = $<F/L>$ tangential_damping = $<F/L>$ friction_ratio = $<\ .\ >$ contact_plane_vector = ($<\ .\ >$, $<\ .\ >$, $<\ .\ >$)*

- gmESSI :: [Vary_Beam_Elastic{ cross_section,elastic_modulus, shear_modulus, torsion_Jx, bending_Iy, bending_Iz, mass_density, xz_plane_vector_x, xz_plane_vector_y, xz_plane_vector_z, joint_1_offset_1, joint_1_offset_2, joint_1_offset_3 , joint_2_offset_1, joint_2_offset_2, joint_2_offset_3}]
  ESSI :: Vary element #{} type beam_elastic with nodes ({},{}) cross_section={} elastic_modulus={} shear_modulus={} torsion_Jx={} bending_Iy={} bending_Iz={} mass_density={} xz_plane_vector= ({},{},{}) joint_1_offset= ({},{},{}) joint_2_offset= ({},{},{})
  Description :: *Vary element # $<\ .\ >$ type beam_elastic with nodes ($<\ .\ >$, $<\ .\ >$) cross_section = $<area>$ elastic_modulus = $<F/L^2>$ shear_modulus = $<F/L^2>$ torsion_Jx = $<length^4>$ bending_Iy = $<length^4>$ bending_Iz = $<length^4>$ mass_density = $<M/L^3>$ xz_plane_vector = ($<\ .\ >$, $<\ .\ >$, $<\ .\ >$) joint_1_offset = ($<L>$, $<L>$, $<L>$) joint_2_offset = ($<L>$, $<L>$, $<L>$)*

- gmESSI :: [Vary_Beam_Displacement_Based{ material_no, section_no, mass_density, integration_rule, xz_plane_vector_x, xz_plane_vector_y, xz_plane_vector_z, joint_1_offset_1, joint_1_offset_2, joint_1_offset_3, joint_2_offset_1, joint_2_offset_2, joint_2_offset_3}]
  ESSI :: Vary element #{} type beam_displacement_based with nodes ({},{}) with #{} integration_points use section #{} mass_density={} IntegrationRule={} xz_plane_vector=({},{},{}) joint_1_offset= ({},{},{}) joint_2_offset= ({},{},{})
  Description :: *Vary element # $<\ .\ >$ type beam_displacement_based with nodes ($<\ .\ >$, $<\ .\ >$) with # $<\ .\ >$ integration_points use section # $<\ .\ >$ mass_density = $<M/L^3>$ IntegrationRule = "" xz_plane_vector = ($<\ .\ >$, $<\ .\ >$, $<\ .\ >$) joint_1_offset = ($<L>$, $<L>$, $<L>$) joint_2_offset = ($<L>$, $<L>$, $<L>$)*

- gmESSI :: [Vary_Beam_9dof_Elastic{ cross_section, elastic_modulus, shear_modulus, torsion_Jx, bending_Iy, bending_Iz, mass_density, xz_plane_vector_x, xz_plane_vector_y, xz_plane_vector_z, joint_1_offset_1, joint_1_offset_2, joint_1_offset_3, joint_2_offset_1, joint_2_offset_2, joint_2_offset_3}]
  ESSI :: Vary element #{} type beam_9dof_elastic with nodes ({},{}) cross_section={} elastic_modulus={} shear_modulus={} torsion_Jx={} bending_Iy={} bending_Iz={} mass_density={} xz_plane_vector=({},{},{}) joint_1_offset= ({}, {}, {}) joint_2_offset= ({},{},{})
  Description :: *Vary element # < . > type beam_9dof_elastic with nodes (< . >, < . >) cross_section = < area > elastic_modulus =* $< F/L^2 >$ *shear_modulus =* $< F/L^2 >$ *torsion_Jx =* $< length^4 >$ *bending_Iy =* $< length^4 >$ *bending_Iz =* $< length^4 >$ *mass_density =* $< M/L^3 >$ *xz_plane_vector = (< . >, < . >, < . >) joint_1_offset = (< L >, < L >, < L >) joint_2_offset = (< L >, < L >, < L >)*

- gmESSI :: [Vary_ShearBeamLT{ cross_section, material_no}]
  ESSI :: Vary element #{} type ShearBeamLT with nodes ({},{}) cross_section={} use material #{}
  Description :: *Vary element # < . > type ShearBeamLT with nodes (< . >, < . >) cross_section =* $< l^2 >$ *use material # < . >*

- gmESSI :: [Vary_3NodeShell_ANDES{material_no, thickness}]
  ESSI :: Vary element #{} type 3NodeShell_ANDES with nodes ({},{},{}) use material #{} thickness= {}
  Description :: *Vary element # < . > type 3NodeShell_ANDES with nodes (< . >, < . >, < . >) use material # < . > thickness = < l >*

- gmESSI :: [Vary_4NodeShell_ANDES{material_no, thickness}]
  ESSI :: Vary element #{} type 4NodeShell_ANDES with nodes ({},{},{},{}) use material #{} thickness= {}
  Description :: *Vary element # < . > type 4NodeShell_ANDES with nodes (< . >, < . >, < . >, < . >) use material # < . > thickness = < l >*

- gmESSI :: [Vary_4NodeShell_MITC4{material_no, thickness}]
  ESSI :: Vary element #{} type 4NodeShell_MITC4 with nodes ({},{},{},{}) use material #{} thickness= {}
  Description :: *Vary element # < . > type 4NodeShell_MITC4 with nodes (< . >, < . >, < . >, < . >) use material # < . > thickness = < L >*

- gmESSI :: [Vary_4NodeShell_NewMITC4{material_no, thickness}]
  ESSI :: Vary element #{} type 4NodeShell_NewMITC4 with nodes ({},{},{},{}) use material #{} thickness= {}
  Description :: *Vary element # < . > type 4NodeShell_NewMITC4 with nodes (< . >, < . >, < . >, < . >) use material # < . > thickness = < L >*

- gmESSI :: [Vary_8NodeBrick_up{material_no}]
  ESSI :: Vary element #{} type 8NodeBrick_up with nodes ({}, {}, {}, {}, {}, {}, {}, {}) use material #{} porosity = {} alpha = {} rho_s = {} rho_f = {} k_x = {} k_y = {} k_z = {} K_s = {} K_f= {}
  Description :: *Vary element # < . > type 8NodeBrick_up with nodes (< . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >) use material # < . > porosity = < . > alpha = < . > rho_s =* $< M/L^3 >$ *rho_f =* $< M/L^3 >$ *k_x =* $< L^3T/M >$ *k_y =* $< L^3T/M >$ *k_z =* $< L^3T/M >$ *K_s = < stress > K_f = < stress >*

- gmESSI :: [Vary_8NodeBrick_upU{material_no}]
  ESSI :: Vary element #{} type 8NodeBrick_upU with nodes ({}, {}, {}, {}, {}, {}, {}, {}) use material #{} porosity = {} alpha = {} rho_s = {} rho_f = {} k_x = {} k_y = {} k_z = {} K_s = {} K_f= {}
  Description :: *Vary element # < . > type 8NodeBrick_upU with nodes (< . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >) use material # < . > porosity = < . > alpha = < . > rho_s =* $< M/L^3 >$ *rho_f =* $< M/L^3 >$ *k_x =* $< L^3T/M >$ *k_y =* $< L^3T/M >$ *k_z =* $< L^3T/M >$ *K_s = < stress > K_f = < stress >*

- gmESSI :: [Vary_20NodeBrick_upU{material_no}]
  ESSI :: Vary element #{} type 20NodeBrick_upU with nodes ({}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}) use material #{} porosity = {} alpha = {} rho_s = {} rho_f = {} k_x = {} k_y = {} k_z = {} K_s = {} K_f= {}
  Description :: *Vary element # < . > type 20NodeBrick_upU with nodes (< . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >, < . >) use material # < . > and porosity = < . > alpha = < . > rho_s =* $< M/L^3 >$ *rho_f =* $< M/L^3 >$ *k_x =* $< L^3T/M >$ *k_y =* $< L^3T/M >$ *k_z =* $< L^3T/M >$ *K_s = < stress > K_f = < stress >*

- gmESSI :: [Vary_8NodeBrick_SurfaceLoad{PhyEntSurfaceTag,m1}]
  ESSI :: Vary load #{} to element #{} type surface at nodes ({},{},{},{}) with magnitude ({})
  Description :: *Vary load # < . > to element # < . > type surface at nodes (< . > , < . > , < . > , < . >) with magnitudes (< . >)*

- gmESSI :: [Vary_8NodeBrick_SurfaceLoad{PhyEntSurfaceTag,m1,m2,m3,m4}]
  ESSI :: Vary load #{} to element #{} type surface at nodes ({},{},{},{}) with magnitudes ({},{},{},{})
  Description :: *Vary load # < . > to element # < . > type surface at nodes (< . > , < . > , < . > , < . >) with magnitudes (< . > , < . > , < . > , < . >)*

- gmESSI :: [Vary_20NodeBrick_SurfaceLoad{PhyEntSurfaceTag,magnitude}]
  ESSI :: Vary load #{} to element #{} type surface at nodes ({},{},{},{},{},{},{},{}) with magnitude ({})

Description :: *Vary load # < . > to element # < . > type surface at nodes (< . > , < . > , < . > , < . >, < . >, < . >, < . >, < . >) with magnitudes (< . >)*

- gmESSI :: [Vary_20NodeBrick_SurfaceLoad{PhyEntSurfaceTag,m1,m2,m3,m4,m5,m6,m7,m8}]
  ESSI :: Vary load #{} to element #{} type surface at nodes ({},{},{},{},{},{},{},{}) with magnitudes ({},{},{},{},{},{},{},{})
  Description :: *Vary load # < . > to element # < . > type surface at nodes (< . > , < . > , < . > , < . >, < . >, < . >, < . >, < . >) with magnitudes (< . > , < . > , < . > , < . >, < . >, < . >, < . >, < . >)*

- gmESSI :: [Vary_27NodeBrick_SurfaceLoad{PhyEntSurfaceTag,magnitude}]
  ESSI :: Vary load #{} to element #{} type surface at nodes ({},{},{},{},{},{},{},{},{}) with magnitude ({})
  Description :: *Vary load # < . > to element # < . > type surface at nodes (< . > , < . > , < . > , < . >, < . >, < . >, < . >, < . >, < . >) with magnitudes (< . >)*

- gmESSI :: [Vary_27NodeBrick_SurfaceLoad{PhyEntSurfaceTag,m1,m2,m3,m4,m5,m6,m7,m8,m9}]
  ESSI :: Vary load #{} to element #{} type surface at nodes ({},{},{},{},{},{},{},{},{}) with magnitudes ({},{},{},{},{},{},{},{},{})
  Description :: *Vary load # < . > to element # < . > type surface at nodes (< . > , < . > , < . > , < . >, < . >, < . >, < . >, < . >, < . >) with magnitudes (< . > , < . > , < . > , < . >, < . >, < . >, < . >, < . >, < . >)*

6.6.5 Write Command

Write command takes filename as an argument and writes the content of a physical/entity group in two separate files one containig all the nodes info and other containig all the elemnets info and places in the same XYZ_ESSI_Simulation folder. The command syntax is

- gmESSI:: [Write_Data{filename}] - *Creates files **XYZ_filename_Nodes.txt** and **XYZ_filename_Elements.txt***

- **XYZ_filename_Nodes.txt ::** Contains data for all nodes in a physical/entity group. Each node data is represented in one line as
  Node_no x_coord y_cord z_cord
  with meanings as usual.

- **XYZ_filename_Elements.txt ::** Contains data for all elements in a physical/entity group.Each element data is represented in one line as
  Element_no Element_type node1 node2 node3 ..
  with meanings as usual. Element_type refers to the same as in Gmsh Manual.

Note:- Write command also has the option of adding Physical_Group# or Entity_Group# argument in the front.

7 gmESSI Python Module

gmESSI also offers python module for advance translation and selection. The module can be imported in python with the name **gmESSI**. The module enables the user to difine their own conversions, make selection, create physical groups, mesh updates and lot more. The various functions are available in python from C++ through wrapper class *gmESSIPython, Element and Node.* Also, there are some maps and vectors of classes available as lists in python. The functions available for each class, their arguments and their description are listed down

7.0.6 gmESSIPython

It is the main class for gmESSI module where actual translation takes place.

- ___init___() ReturnType:: gmESSIPython
  Description :: It initializes an object of class **gmESSIPython**.

```
        >>> import gmessi
        >>> MyGmshConvertor = gmESSI.gmESSIPython()
```

- ___init___(string MeshFile) ReturnType:: gmESSIPython
  Description :: It initializes an object of class **gmESSIPython** and takes an *MeshFile* as argument. By default the **override** option for the XYZ_ESSI_Simulation folder is 1, which means it will overwrite the folder if present. By creating an object of gmESSIPython and adding a mshfile, it automatically translate all the gmESSI commands automatically and places inside XYZ_ESSI_Simulation.

```
      >>> import gmessi
      >>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
      Message::Files converted to
             /home/sumeet/sumeet.kumar507@gmail.com/git/gmESSI/Example1_ESSI_Simulation

     Add_Node_Load_Linear{Fx,10*kN} Found!! Sucessfully Converted
     Add_All_Node{m,3} Found!! Sucessfully Converted
     Fix{all} Found!! Sucessfully Converted
     Add_8NodeBrick{1} Found!! Sucessfully Converted
```

- ___init___(string MeshFile, int override) ReturnType:: gmESSIPython
  Description :: It initializes an object of class **gmESSIPython** and takes an *MeshFile* and **override** option as argument.

```
      >>> import gmessi
      >>> # 1 means you overwrite an earlier XYZ_ESSI_Simulation folder if any
      >>> # 0 means you create a new non-conflicying XYZ_ESSI_Simulation_n folder
      >>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh",0)
      Message::Files converted to
             /home/sumeet/sumeet.kumar507@gmail.com/git/gmESSI/Example1_ESSI_Simulation_1

     Add_Node_Load_Linear{Fx,10*kN} Found!! Sucessfully Converted
     Add_All_Node{m,3} Found!! Sucessfully Converted
     Fix{all} Found!! Sucessfully Converted
     Add_8NodeBrick{1} Found!! Sucessfully Converted
```

- pwd ReturnType:: string
  Description :: It is a public readonly string variable which contains the path to working directory of gmESSI Translator i.e location of XYZ_ESSI_Simulation folder.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> MyGmshConvertor.pwd
'/home/sumeet/sumeet.kumar507@gmail.com/git/gmESSI/Example1_ESSI_Simulation/'
```

- GmshFile ReturnType:: string
  Description :: It is a public readonly string variable which contains the path to user input GmshFile i.e location of XYZ.msh

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> MyGmshConvertor.GmshFile
'Example1.msh'
```

- LoadFile ReturnType:: string
  Description :: It is a public readonly string variable which contains the path of the load-file created gmESSI Translator i.e XYZ_load.fei .

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> MyGmshConvertor.LoadFile
'/home/sumeet/sumeet.kumar507@gmail.com/git/gmESSI/
Example1_ESSI_Simulation/Example1_load.fei'
```

- GeometryFile ReturnType:: string
  Description :: It is a public readonly string variable which contains the path of the geometry-file created by gmESSI Translator i.e XYZ_geometry.fei .

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> MyGmshConvertor.GeometryFile
'/home/sumeet/sumeet.kumar507@gmail.com/git/gmESSI/
Example1_ESSI_Simulation/Example1_geometry.fei'
```

- MainFile ReturnType:: string
  Description :: It is a public readonly string variable which contains the path of the main ESSI-file created gmESSI Translator i.e XYZ_analysis.fei .

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> MyGmshConvertor.MainFile
'/home/sumeet/sumeet.kumar507@gmail.com/git/gmESSI/
Example1_ESSI_Simulation/Example1_analysis.fei'
```

- setMainFile(string Filename) ReturnType:: void
  Description :: It changes the dafault mainfile XYZ_analysis.fei to a user-defined file. The user can use this option to write the conversion commands in different files of their own.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> # Changing to a new file "myNewMainfile.fei" in the same ESSI_Simulation_Folder
>>> MyGmshConvertor.setMainFile(MyGmshConvertor.pwd + "myNewMainfile.fei")
>>> MyGmshConvertor.MainFile
'/home/sumeet/sumeet.kumar507@gmail.com/git/gmESSI/
Example1_ESSI_Simulation/myNewMainfile.fei'
```

- setGeometryFile(string Filename) ReturnType:: void
  Description :: It changes the dafault mainfile XYZ_geometry.fei to a user-defined file. The user can use this option to write the conversion commands in different files of their own.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> # Changing to a new file "myNewGeometryfile.fei" in the same ESSI_Simulation_Folder
>>> MyGmshConvertor.setGeometryFile(MyGmshConvertor.pwd + "myNewGeometryfile.fei")
>>> MyGmshConvertor.GeometryFile
'/home/sumeet/sumeet.kumar507@gmail.com/git/gmESSI/
Example1_ESSI_Simulation/myNewGeometryfile.fei'
```

- setLoadFile(string Filename) ReturnType:: void
  Description :: It changes the dafault loadfile XYZ_load.fei to a user-defined file. The user can use this option to write the conversion commands in different files of their own.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> # Changing to a new file "myNewLoadfile.fei" in the same ESSI_Simulation_Folder
>>> MyGmshConvertor.setLoadFile(MyGmshConvertor.pwd + "myNewLoadfile.fei")
>>> MyGmshConvertor.LoadFile
'/home/sumeet/sumeet.kumar507@gmail.com/git/gmESSI/
Example1_ESSI_Simulation/myNewLoadfile.fei'
```

- Convert(string gmESSICommand) ReturnType:: void
  Description :: This function is to execute gmESSI command. The gmESSI command is given as string with or without [ ].

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> The gmESSI command can be given as "[Add_All_Node{m,3}]" or "Add_All_Node{m,3}"
>>> MyGmshConvertor.Convert("[Add_All_Node{m,3}]")
Add_All_Node{m,3} Found!! Sucessfully Converted
```

- getNewESSITag(string ESSITag) ReturnType:: int
  Description :: This command return the new tag number available for ESSI Tags. ESSITags variables are 'material' , 'load' , 'node' , 'damping' , 'displacement' , 'element' , 'field' and 'motion' and 'nodes'. The command take these variables as arguments.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
# new node no available for user
>>> MyGmshConvertor.getNewESSITag("node")
55
>>> # Suposse use want to add load of Fx = 12*kN to node 1 and write it in mainfile
>>> ESSICommand = "add load #" + str(MyGmshConvertor.getNewESSITag("load")) + " to node #1 type
      linear Fx= 10*kN; \n"
>>> f = open(MyGmshConvertor.MainFile, 'a') # opening the file in append mode
>>> f.write(ESSICommand) # Writing the ESSI Command in the Main XYZ_analysis.fei file
>>> f.close()
```

- getPhysicalGroupElements(int Physical_Group_No) ReturnType:: list[Element]
  Description :: This function return a list of all the Element in the specified physical group. The argument takes the physical group unique identificantion number. Here, the returned list contains object of class Element. Element Class and its available functions are discussed later in teh Manual.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> # In order to retrieve the all the elements in a list of Physical group of Id 3 "CantileverVolume"
>>> ElementList = MyGmshConvertor.getPhysicalGroupElements(3)
>>> len(ElementList) # getting the length of element list
20
```

- getPhysicalGroupElements(string Physical_Group_String_Tag) ReturnType:: list[Element]
  Description :: This function return a list of all the Element in the specified physical group. The argument takes the physical group string tag. This function is an overloading of the above function.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> # In order to retrieve the all the elements in a list of Physical group of Id 3 "CantileverVolume"
>>> ElementList = MyGmshConvertor.getPhysicalGroupElements("CantileverVolume")
>>> ElementList[1].gettype() # getting the type of Element stored in the list at index 1.
5 # 5 means it is a 8−noded Brick
```

- getPhysicalGroupNodes(int Physical_Group_No) ReturnType:: list[Node]
  Description :: This function return a list of all the Nodes in the specified physical group. The argument takes the physical group unique identificantion number. Here, the returned list contains object of class Node. Node Class and its available functions are discussed later in teh Manual.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> # In order to retrieve the all the nodes in a list of Physical group of Id 3 "CantileverVolume"
>>> NodeList = MyGmshConvertor.getPhysicalGroupNodes(3)
>>> len(NodeList)
54
```

- getPhysicalGroupNodes(string Physical_Group_String_Tag) ReturnType:: list[Node]
  Description :: This function return a list of all the Nodes in the specified physical group. The argument takes the physical group unique identificantion number.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> # In order to retrieve the all the nodes in a list of Physical group of Id 3 "CantileverVolume"
>>> NodeList = MyGmshConvertor.getPhysicalGroupNodes("CantileverVolume")
>>> NodeList[8].getXcord() # getting the coordinae of Node object in the list at index 8.
0.
```

- getEntityGroupElements(int Entity_Group_No) ReturnType:: list[Element]
  Description :: IThis function return a list of all the Elements in the specified entity group. The argument takes the entity group unique identificantion number. Please note that Entity groups do not have any name.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> # In order to retrieve the all the elements in a list of Entity group of Id 1
>>> ElementList = MyGmshConvertor.getEntityGroupElements(1)
>>> len(ElementList)
20
>> ElementList[1].getEntityTag() # retrieving the Entity tag
1 # obviously it would be 1 only
```

- getEntityGroupNodes(int Entity_Group_No) ReturnType:: list[Node]
  Description :: This function return a list of all the Nodes in the specified entity group. The argument takes the entity group unique identificantion number.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> # In order to retrieve the all the nodes in a list of Entity group of Id 1
>>> NodeList = MyGmshConvertor.getEntityGroupNodes(1)
>>> len(NodeList)
54
```

- getNodeMap() ReturnType:: dict[int,Node]
  Description :: It returns a dictionry of node_number mapped to Node objects. The Node objects contains all the data related to that node like coordinates and node_no. The node can be used to find out data for a node from node_no information.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> NodeMap = MyGmshConvertor.getNodeMap()
>>> NodeMap[10].getXcord() # getting x cord for node number 10.
1.6
```

- getGroupData(string Physical_Group# or Entity_Group#Tag) ReturnType:: SelectionData
  Description :: This function return an object of class SelectionData which conatains ElementList and NodeList of all the elements ond Nodes of the specified Physical/Entity group respectively. The argument is in the form of string and written in the same manner as Physical_Group#IdorStringName and Entity_Group#Id respectively. If the user writes "All" in the argument it would return all NodeList and ElementList of all the nodes and elements of the model respectively. SelectionData class is discussed later.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> SelectedData = MyGmshConvertor.getGroupData("All")
Group Data retrieved for All Model
>>> # Selecting the Data for CantileverVolume. Argument can be
>>> # "Physical_Group#CantileverVolume" or "Physical_Group#3"
>>> SelectedData = MyGmshConvertor.getGroupData("Physical_Group#CantileverVolume")
Group Data retrieved for Physical_Group#CantileverVolume
>>> ElementList = SelectedData.ElementList # retrieving elementlist from the SelectionData Class
```

- getSphereSelectionData(string Physical_Group# or Entity_Group#Tag,double radius, double center_x, double center_y, double center_z) ReturnType:: SelectionData
  Description :: This function allows Sphere selection over a Physical/Entity group or over all model. The argument is the same as described for getGroupData() function i.e. "All" refers to whole model and Physical_Group#IdorStringName and Entity_Group#Id refers to specific physical/entity group respectively. This function also return a object of class Selectiondata which basicaly has two public varaiables ElementList and NodeList containg the list of objects of class Element and Nodess respectively obtained from the applied sphere selection.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> # Applying sphere selection on whole model with centre at the cente of the model (2,0.5,0.5) and radius as
    2 units
>>> SelectionData = MyGmshConvertor.getSphereSelectionData("All",2,2,0.5,0.5)
Sphere_Selection_Initiated Sphere Selection Made over All Model with radius 2 and center at 2 0.5 0.5
>>> len(SelectionData.NodeList)
38 # 38 nodes got selected and if you do len(SelectionData.ElementList) it would show 20 elements got selected
>>> # Applying sphere selection over a Physical surface "FixedSurface" having id 1 can be done by writing
    "Physical_Group#1" or "Physical_Group#FixedSurface"
>>> SelectionData = MyGmshConvertor.getSphereSelectionData("Physical_Group#1",0.5,0,0.5,0.5)
phere_Selection_Initiated Sphere Selection Made over Physical_Group#1 with radius 0.5 and center at 0 0.5 0.5
```

- BoxSelectionData(string Physical_Group# or Entity_Group#Tag, double x1, double x2, double y1, double y2, double z1, double z2) ReturnType:: SelectionData
  Description :: Box selection is the similar kind of function that selectes nodes and elemnets enclosed within a box i.e $x1 < x < x2$, $y1 < y < y2$ and $z1 < z < z2$. The physical/entity group parameter is the same as Sphere selection where "all" refers to box selection over whole model and Physical_Group#IdorStringName and Entity_Group#Id refers to box selection over specific physical/entity group respectively.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> # selecting elements and nodes in the region $ 2 < x < 3$ , $0 < y < 1$ and $0 < z < 1$ on all model
>>> SelectionData = MyGmshConvertor.BoxSelectionData("All",2,3,0,1,0,1)
Box_Selection_Initiated Box Selection Made over All Model with 2 < x_cord < 3 , 0 < y_cord < 1 and 0<
    z_cord < 1
>>> len(SelectionData.NodeList)
9 # len(ElementData.NodeList) return 0 because there is no element bounded inside that region
```

- CreatePhysicalGroup(string Physical_Group_String, list[Node], list[Element]) ReturnType:: void
  Description :: This function takes a list of Element Class objects and a respective Node class objects and created a physical group with its name as specified by the user in its first argument. The unique number id of the new physical group is the next id available. That means if the model has physical group ids upto 15, the new created id of the physical griup would be 16. Please note that the user should give a unique physical group name which has beed not previously used. The newly created physical can be used in the program later and can be called by its name or id in other gmESSI commands.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> # Applying sphere selection on whole model with centre at the cente of the model (2,0.5,0.5) and radius as
    2 units
>>> SelectionData = MyGmshConvertor.getSphereSelectionData("All",2,2,0.5,0.5)
Sphere_Selection_Initiated Sphere Selection Made over All Model with radius 2 and center at 2 0.5 0.5
>>> # creating a physical group of this selection
>>> MyGmshConvertor.CreatePhysicalGroup ( "MyNewPhysicalGroup", SelectionData.NodeList,
    SelectionData.ElementList)
New Physical Group 4 with name "MyNewPhysicalGroup" created
>>> # the new physical group has got the next id 4 because Example1.msh had the last physical group with id
    3. The user can use the new created physical group.
>>> MyGmshConvertor.Convert("Add_Node_Load_Linear{Physical_Group#4,Fx,10*kN}")
Add_Node_Load_Linear{Physical_Group#4,Fx,10*kN} Found!! Sucessfully Converted
```

- UpdateGmshFile() ReturnType:: void

  Description :: This command updates the gmsh file in XYZ_ESSI_Simulation folder and add the newly created physical groups and its data to XYZ.msh file. The updated **XYZ.msh** file can be visulaised in gmsh to see the new physical groups.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> # Applying sphere selection on whole model with centre at the cente of the model (2,0.5,0.5) and radius as
    2 units
>>> SelectionData = MyGmshConvertor.getSphereSelectionData("All",2,2,0.5,0.5)
Sphere_Selection_Initiated Sphere Selection Made over All Model with radius 2 and center at 2 0.5 0.5
>>> # creating a physical group of this selection
>>> MyGmshConvertor.CreatePhysicalGroup ( "MyNewPhysicalGroup", SelectionData.NodeList,
    SelectionData.ElementList)
New Physical Group 4 with name "MyNewPhysicalGroup" created
>>> # Updating the gmsh file Example1.msh in Example1_ESSI_Simulation folder
>>> MyGmshConvertor.UpdateGmshFile()
>>> # the user can visualize the newly created physical groups if any in gmsh.
```

- DisplayNewTagNumbering() ReturnType:: void

  Description :: This function displays the new update Tag Number available for ESSI Tags on the terminal.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> # Applying sphere selection on whole model with centre at the cente of the model (2,0.5,0.5) and radius as
    2 units
>>> SelectionData = MyGmshConvertor.getSphereSelectionData("All",2,2,0.5,0.5)
Sphere_Selection_Initiated Sphere Selection Made over All Model with radius 2 and center at 2 0.5 0.5
>>> # creating a physical group of this selection
>>> MyGmshConvertor.CreatePhysicalGroup ( "MyNewPhysicalGroup", SelectionData.NodeList,
    SelectionData.ElementList)
New Physical Group 4 with name "MyNewPhysicalGroup" created
>>> # the new physical group has got the next id 4 because Example1.msh had the last physical group with id
    3. The user can use the new created physical group.
>>> MyGmshConvertor.Convert("Add_Node_Load_Linear{Physical_Group#4,Fx,10*kN}")
Add_Node_Load_Linear{Physical_Group#4,Fx,10*kN} Found!! Sucessfully Converted
>>> # To get the new update tag numbering as it would have changed by the execution of the above gmESSI
    command
>>> MyGmshConvertor.DisplayNewTagNumbering()
************************ Updated New Tag Numberring *********************
damping = 1
displacement = 1
element = 39
field = 1
load = 42
material = 2
motion = 1
node = 56
nodes = 56
```

7.0.7 Node

This class contains the data of Node objects. Each node object has attributes : Id, Xcord, Ycord and Zcord and theese data are retrieved using the functions datiled below.

- getId() ReturnType:: int
  Description :: This function return the Node_number of the Node object.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> NodeList = MyGmshConvertor.getGroupData("Physical_Group#ApplyForce").NodeList
>>> # geting the node number of Node object at index 1
>>> NodeList[1].getId()
7
```

- getXcord() ReturnType:: double
  Description :: This function return the X-coordinate of the Node.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> NodeList = MyGmshConvertor.getGroupData("Physical_Group#ApplyForce").NodeList
>>> # geting the X-coordinate of Node object at index 1
>>> NodeList[1].getXcord()
4.0
```

- getYcord() ReturnType:: double
  Description :: This function return the Y-coordinate of the Node.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> NodeList = MyGmshConvertor.getGroupData("Physical_Group#ApplyForce").NodeList
>>> # geting the Y-coordinate of Node object at index 1
>>> NodeList[1].getYcord()
1.0
```

- getZcord() ReturnType:: double
  Description :: This function return the Z-coordinate of the Node.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> NodeList = MyGmshConvertor.getGroupData("Physical_Group#ApplyForce").NodeList
>>> # geting the Z-coordinate of Node object at index 1
>>> NodeList[1].getZcord()
1.0
```

7.0.8 Element

This class contains the data of Element objects. Each element object has attributes : Id, Type, EntityTag, PhysicalTag and Nodelit (containg node numbers of all the nodes in that element) and theese data are retrieved using the functions datiled below.

- getId() ReturnType:: int
  Description :: This function return the element_no i.e the element_id of the Element.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> ElementList = MyGmshConvertor.getGroupData("Physical_Group#ApplyForce").ElementList
>>> # geting the element number of Element object at index 1
>>> ElementList[1].getId()
2
```

- getType() ReturnType:: int

  Description :: This function return the type_noi.e the type of the Element. Element_Type is an unique id for describing whether an element is 2-noded, 3-noded, a 8-nodedBrick ... etc. The type numbering is the same as what **gmsh** uses.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> ElementList = MyGmshConvertor.getGroupData("Physical_Group#ApplyForce").ElementList
>>> # geting the element type of Element object at index 1
>>> ElementList[1].getType()
1 # Type no 1 means it is a 2-noded element
```

- getEntityTag() ReturnType:: int

  Description :: This function return the entity_tag associated with that Element. The element belongs to that entity_group which has id same as entity_tag to that element.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> ElementList = MyGmshConvertor.getGroupData("Physical_Group#ApplyForce").ElementList
>>> # geting the element entity_tag of Element object at index 1
>>> ElementList[1].getEntityTag()
8 # This means that ElementList[1] belongs to entity group having id 8
```

- getPhysicalTag() ReturnType:: int

  Description :: This function return the physical_tag associated with that Element. The element belongs to that physical_group which has id same as physical_tag to that element.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> ElementList = MyGmshConvertor.getGroupData("Physical_Group#ApplyForce").ElementList
>>> # geting the element physical_tag of Element object at index 1
>>> ElementList[1].getPhysicalTag()
2 # This means that ElementList[1] belongs to physical group having id 2
```

- getNodeList() ReturnType:: list[int]

  Description :: This function return a list of all node_numbers belonging to that Element.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> ElementList = MyGmshConvertor.getGroupData("Physical_Group#ApplyForce").ElementList
>>> # geting the nodelist of Element object at index 1
>>> Nodelist = ElementList[1].getNodeList() # List of all node_no belonging to Element object at index 1
>>> len(Nodelist)
2 # Obviously this would be 2 beacuse it is a 2-noded element so, each element has exactly 2 nodes.
```

7.0.9 SelectionData

This class is like a container whch contains ElementList and NodeList of objects of class Element and Node respectively.

- NodeList ReturnType:: list[Node]
  Description :: This variable contains and returns the list of stored Node objects.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> # Applying sphere selection on whole model with centre at the cente of the model (2,0.5,0.5) and radius as
    2 units
>>> SelectionData = MyGmshConvertor.getSphereSelectionData("All",2,2,0.5,0.5)
Sphere_Selection_Initiated Sphere Selection Made over All Model with radius 2 and center at 2 0.5 0.5
>>> The NodeList of the selection made can be retrived by this function
>>> NodeList = SelectionData.NodeList
```

- ElementList ReturnType:: list[Element]
  Description :: This variable contains and returns the list of stored Element objects.

```
>>> import gmessi
>>> MyGmshConvertor = gmESSI.gmESSIPython("Example1.msh")
>>> # Applying sphere selection on whole model with centre at the cente of the model (2,0.5,0.5) and radius as
    2 units
>>> SelectionData = MyGmshConvertor.getSphereSelectionData("All",2,2,0.5,0.5)
Sphere_Selection_Initiated Sphere Selection Made over All Model with radius 2 and center at 2 0.5 0.5
>>> The ElementList of the selection made can be retrived by this function
>>> NodeList = SelectionData.ElementList
```

8 Using gmESSI Python Module

With gmESSI python module user gets more power and advanced features that gmsh does not support like selection of elements and creation of new physical groups, visualization etc. Using gmESSI it is very easy to convert a .msh file to ESSI (.fei) file. The next section guides the user through a simple Example4, the important steps for generating essi files directly from .msh file through gmESSI. Lets define a problem as shown in Figure 11.

It is a block of dimension $10mx10mx10m$ of soil mass whose all 4 lateral faces and the bottom face are fixed and a surface load of uniform pressure of 10 Pa is applied. The density and elastic modulus of the soil increases from $1600 * kg/m^3$ to $1700 * kg/m^3$ and from x to y down the depth as shown in Figure 11.

8.1 Making .geo and .msh file in Gmsh

The first step is to make the geometry file in **Gmsh**. While creating the geometry the user should also define all the physical groups on which they intend to either apply boundary condition, define elements, loads etc. In Example5, 3 physical groups are needed one for applying surface load, one for fixities, and one for defining the soil volume and assigning material. The content of Example4.geo file is shown below

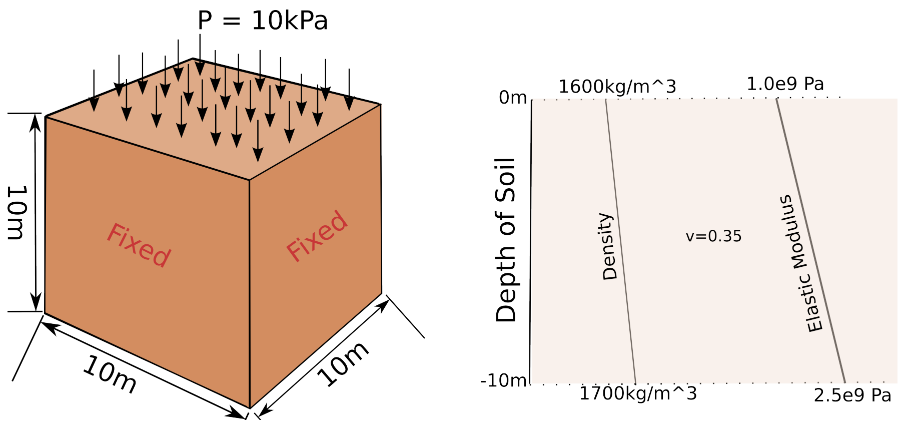


Figure 11: Description of a Example 4

```
$cat Example4.geo

Point(1) = {0,0,0};

// Extruding and subdividing each dimension in 10 layers
Extrude{10,0,0}{Point{1}; Layers{10};}
Extrude{0,10,0}{Line{1}; Layers{10};Recombine;}
Extrude{0,0,10}{Surface{5}; Layers{10};Recombine;}

//Creating Physical Groups
Physical Volume("$SoilVolume$")= {1};
Physical Surface("$Fixities$") = {5,26,18,14,22};
Physical Surface("$SurfaceLoad$") = {27};
```

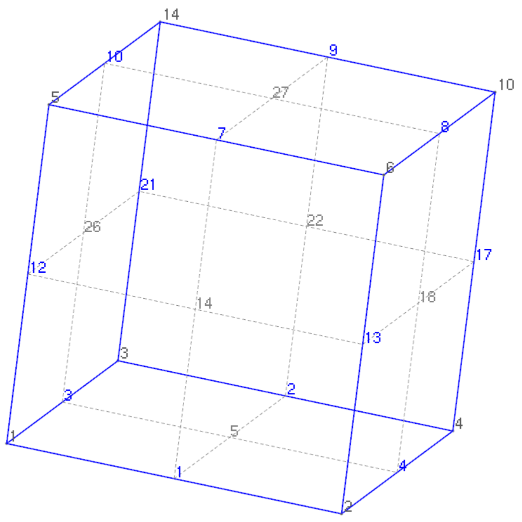
Once .geo file is ready with all the physical groups, next step is to mesh the model. The model can be messed from the terminal directly by running gmsh Example4.geo -3. A quick look at the Example4.msh file conatining physical groups is shown below.

```
$cat Example4.geo
..............
$PhysicalNames
3
2 2 "$Fixities$"
2 3 "$SurfaceLoad$"
3 1 "$SoilVolume$"
$EndPhysicalNames
................
```

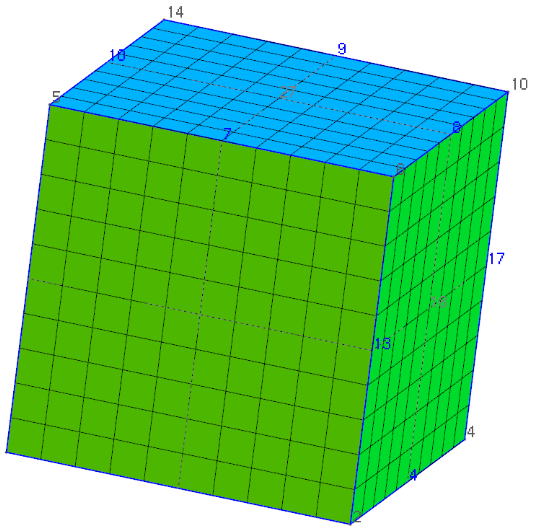
Figure 12 shows the geometry and mesh visualization in gmsh.

8.2 Translataing .msh file using gmESSI to ESSI files

Using gmESSI for mesh conversion is very easy. To achieve this, the step is further subdivided into two simpler substeps i.e. creating a Example4.gmessi file containing all the required gmESSI commands to be executed sequentially and Example4.py file for importing gmESSI python module and executing each gmESSI command sequentially from file Exampl4.gmessi. Let us look at each of them one by one.



(a) Geometry File



(b) Mesh File

Figure 12: Description of a Example 4

8.2.1 Writting all gmESSI Commands for the model

gmESSI commands for this particular problem is written in Example4.gmessi file. Since physical group names and ids are required for referring the gmESSI commands, its always best to copy all the physical group data from the .msh file (in this case Example4.msh file) in the header of .gmessi file, so that its easier for the user to refer to the physical groups while writing commands in .gmessi file. The contents of the Example4.gmessi are shown below.

```
$cat Example4.gmessi
////////////////////////////////////////////////////////
//2 2 "$Fixities$"
//2 3 "$SurfaceLoad$"
//3 1 "$SoilVolume$"
////////////////////////////////////////////////////////

//Adding Model Name
[Model_Name{"Example4"}]
[New_Line{}]

//Adding all nodes of the model
[Add_All_Node{m,3}]

//Defining and creating 8−noded brick elements with different material properties
[Vary_Linear_elastic_isotropic_3d_LT{ Physical_Group#1, Add_8NodeBrickLT{},
  1600+10∗(10−z)\0\kg/m^3, 1.0e9+1.5e8∗(10−z)\−8\Pa, 0.35}]
[New_Line{}]

//Applying Fixities to the model
[Fix{Physical_Group#2,all}]

//Including the geometry file generated
[Include{"Example4_geometry.fei"}]

[New_Line{}]
[New_Line{}]
```

```

//Adding new loading stage
[Add_New_Loading_Stage{"Stage1_Surface_Loading"}]
[New_Line{}]

// Adding surface load on the soil mass
[Add_8NodeBrick_SurfaceLoad{Physical_Group#1,Physical_Group#3,10*Pa]

//Including the load file generated
[Include{"Example4_load.fei"}]
[New_Line{}]
[New_Line{}]

// Just a comment inside the _analysis file.
[Comment{Starting the simulation}]
[New_Line{}]

// Defining algorithms for the model
[Define_Algorithm{With_no_convergence_check}]
[Define_Solver_UMFPack{}]
[New_Line{}]
[Define_Load_Factor_increament{1}]
[Simulate_Steps_Using_Static_Algorithm{10}]

// A nice good bye
[Bye{}]
```

Please Note - The gmESSI commands are executed and written to the file sequentially, so the user should be carefull with the order of translation.

8.2.2 Import gmESSI in python and carrying out Translation

Once .gmessi is ready, the next task is to make a small python script to import gmESSI module and read the commands from .gmessi file and carry out the convesion. The python script for this problem is written to file Example4.py

```
$cat Example4.py
import gmessi

Example4 = gmessi.gmESSIPython()

Example4.loadMshFile("Example4.msh")

Inputfile = open('Example4.gmessi', 'r')

# Skipping the blank line and line with comments (//)
for line in Inputfile:
    line = line.partition('//')[0]
    line = line.rstrip()
    if (line):
        Example4.Convert(line)
```

The next step is to run the python script. Running the python script would carryout the translation to all gmESSI commands in .gmessi file. The log of translation gets displayed on the terminal

```
$python Example4.py
Message::Files converted to
/home/sumeet/sumeet.kumar507@gmail.com/git/gmESSI/Example4_ESSI_Simulation

Model_Name{"Example4"} Found!! Sucessfully Converted
New_Line{} Found!! Sucessfully Converted
Add_All_Node{m,3} Found!! Sucessfully Converted
Vary_Linear_elastic_isotropic_3d_LT{ Physical_Group#1, Add_8NodeBrickLT{},
  1600+10*(10−z)\0\kg/m^3, 1.0e9+1.5e8*(10−z)\−8\Pa, 0.35} Found!! Sucessfully
 Converted
New_Line{} Found!! Sucessfully Converted
Fix{Physical_Group#2,all} Found!! Sucessfully Converted
Include{"Example4_geometry.fei"} Found!! Sucessfully Converted
New_Line{} Found!! Sucessfully Converted
```

```
New_Line{} Found!! Sucessfully Converted
Add_New_Loading_Stage{"Stage1_Surface_Loading"} Found!! Sucessfully
Converted
New_Line{} Found!! Sucessfully Converted
Add_8NodeBrick_SurfaceLoad{Physical_Group#1,Physical_Group#3,10*Pa Found!!
 Sucessfully Converted
Include{"Example4_load.fei"} Found!! Sucessfully Converted
New_Line{} Found!! Sucessfully Converted
New_Line{} Found!! Sucessfully Converted
Comment{Starting the simulation} Found!! Sucessfully Converted
New_Line{} Found!! Sucessfully Converted
Define_Algorithm{With_no_convergence_check} Found!! Sucessfully Converted
Define_SolveFound!! Sucessfully Converted
New_Line{} Found!! Sucessfully Converted
Define_Load_Factor_increament{1} Found!! Sucessfully Converted
Simulate_Steps_Using_Static_Algorithm{10}Found!! Sucessfully Converted
Bye{} Found!! Sucessfully Converted


************************ Updated New Tag Numberring ********************
damping = 1
displacement = 1
element = 701
field = 1
load = 101
material = 11
motion = 1
node = 1332
nodes = 1332
```

Running gmESSI creates a folder Example4_ESSI_Simulation and places Example4_load.fei file, Example4_geometry.fei file and Example4_analysis.fei i.e. ESSI main file. The user at this point do not need to write anything in the Example4_analysis.fei file as every command was sequentially written down in .gmessi file and is converted.

Please note that material variational commands has created exactly 10 different material layer-wise with increasing densitty and elastic modulus. The content of the main analysis.fei file is shown below

```
$cat Example4_analysis.fei
model name "Example4";
;
add material #1 type linear_elastic_isotropic_3d_LT mass_density=1695.000000*kg/m^3
        elastic_modulus=2420000000.000000*Pa poisson_ratio=0.35;
add material #2 type linear_elastic_isotropic_3d_LT mass_density=1685.000000*kg/m^3
        elastic_modulus=2280000000.000000*Pa poisson_ratio=0.35;
add material #3 type linear_elastic_isotropic_3d_LT mass_density=1675.000000*kg/m^3
        elastic_modulus=2120000000.000000*Pa poisson_ratio=0.35;
add material #4 type linear_elastic_isotropic_3d_LT mass_density=1665.000000*kg/m^3
        elastic_modulus=1980000000.000000*Pa poisson_ratio=0.35;
add material #5 type linear_elastic_isotropic_3d_LT mass_density=1655.000000*kg/m^3
        elastic_modulus=1820000000.000000*Pa poisson_ratio=0.35;
add material #6 type linear_elastic_isotropic_3d_LT mass_density=1645.000000*kg/m^3
        elastic_modulus=1680000000.000000*Pa poisson_ratio=0.35;
add material #7 type linear_elastic_isotropic_3d_LT mass_density=1635.000000*kg/m^3
        elastic_modulus=1520000000.000000*Pa poisson_ratio=0.35;
add material #8 type linear_elastic_isotropic_3d_LT mass_density=1625.000000*kg/m^3
        elastic_modulus=1380000000.000000*Pa poisson_ratio=0.35;
add material #9 type linear_elastic_isotropic_3d_LT mass_density=1615.000000*kg/m^3
        elastic_modulus=1220000000.000000*Pa poisson_ratio=0.35;
add material #10 type linear_elastic_isotropic_3d_LT mass_density=1605.000000*kg/m^3
        elastic_modulus=1080000000.000000*Pa poisson_ratio=0.35;
;
include "Example4_geometry.fei";
;
;
new loading stage "Stage1_Surface_Loading";
;
include "Example4_load.fei";
;
;
 //Starting the simulation;
;
define algorithm With_no_convergence_check;
define solver UMFPack;
```

```
;
define load factor increment 1;
simulate 10 steps using static algorithm;
bye;
```

8.3 Running ESSI and visualization in visit

With all files ready in their place, the next step is to run the Example4_analysis.fei file directly in ESSI.

```
$essi −f Example4_analysis.fei


  The Finite Element Interpreter

  Real ESSI
  Earthquake Soil Structure Interaction Simulator

  Sequential processing mode.

Compiled: Jun 15 2015 at 15:28:24
Time Now: Aug 28 2015 at 19:09:57

Static startup tips:
 ∗ Remember: Every command ends with a semicolon ';'.
 ∗ Type 'quit;' or 'exit;' to finish.
 ∗ Run 'essi −h to see available command line options.

Input: STDIN

Including: "Example4_analysis.fei"


Model name is being set to "Example4"

Including: "Example4_geometry.fei"

Done including: "Example4_geometry.fei" (2352 lines included).
Continuing with "Example4_analysis.fei" at line 16.



Starting new stage: Stage1_Surface_Loading

Including: "Example4_load.fei"

Done including: "Example4_load.fei" (642 lines included).
Continuing with "Example4_analysis.fei" at line 22.


Starting sequential static multistep analysis
========================================================================
Creating analysis model...................................................Pass!
Checking constraint handler..............................................Pass!
Checking numberer........................................................Pass!
Checking analysis algorithm..............................................Pass!
Checking system of equation handler......................................Pass!
Checking transient integration handler...................................Pass!
setNumberOfOutputSteps −> nsteps / output_every_nsteps = 10
Static Analysis: Step Number is : 1 out of 10Domain::update( void ) −− Constitutive integration happening!
Static Analysis: Step Number is : 2 out of 10Domain::update( void ) −− Constitutive integration happening!
Static Analysis: Step Number is : 3 out of 10Domain::update( void ) −− Constitutive integration happening!
```

Running ESSI creates .feioutput file which can be visualized in visit software. Figure 13 shows the visualization of Example4_Stage1_Surface_Loading.h5.feioutput produced in visit.

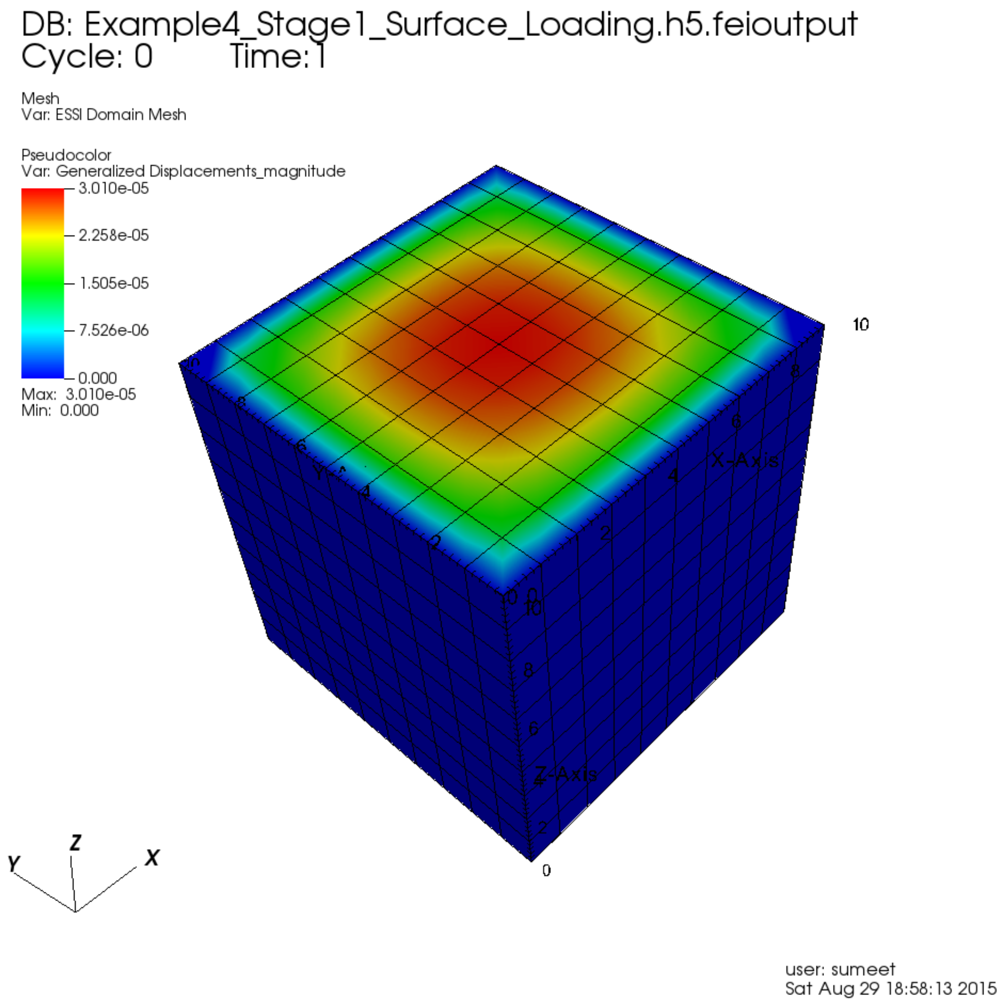


Figure 13: Visualizing output in visit

[Koiter(1960)] W. T. Koiter. General theorems for elastic-plastic solids. 1: 165–221, 1960. `file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Articles/W.T.Koiter(1960).pdf`.

[Baltov et al.(1964)Baltov, Sofia, and A. Sawczuk] A. Baltov, Sofia, and Warsaw A. Sawczuk. A rule of anisotropic hardening. 1/2:81–92, September 1964. `file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Articles/A.Baltov(1964).pdf`.

[Bardet and Choucair(1991)] J. P. Bardet and W. Choucair. A linearized integration technique for incremental constitutive equations. International Journal for Numerical and Analytical Methods in geomechanics, 15:1–19, 1991. `file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Articles/J.P.Bardet(1991).pdf`.

[Simo and Taylor(1985)] J. C. Simo and R. L. Taylor. Consistent tangent operators for rate-independent elastoplasticity. Computer Methods in Applied Mechanics and Engineering, 48:101–118, 1985. `file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Articles/J.C.Simo(1985).pdf`.

[Mindlin and Deresiewicz(1953)] R. D. Mindlin and H. Deresiewicz. Elastic spheres in contact under varying oblique forces. ASME Journal of Applied Mechanics, 53 (APM-14):327–344, September 1953. `file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Articles/R.D.Mindlin(1953).pdf`.

[Ortiz and Popov(1985)] M. Ortiz and E. P. Popov. Accuracy and stability of integration algorithms for elastoplastic constitutive relations. International Journal for Numerical Methods in Engineering, 21:1561–1576, 1985. `file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Articles/M.Ortiz(1985).pdf`.

[Etsion(2010)] Izhak Etsion. Revisiting the cattaneo–mindlin concept of interfacial slip in tangentially loaded compliant bodies. Journal of Tribology, 132(2):020801.1–020801.9, 2010. `file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Articles/I.Etsion(2010).pdf`.

[Rowe(1962)] P. W. Rowe. The stress–dilatancy relation for static equilibrium of an assembly of particles in contact. Proceedings of The Royal Society, 269(1339): 500–527, 9 October 1962. `file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Articles/P.W.Rowe(1962).pdf`.

[Oberkampf et al.(2002)Oberkampf, Trucano, and Hirsch] William L. Oberkampf, Timothy G. Trucano, and Charles Hirsch. Verification, validation and predictive capability in computational engineering and physics. Proceedings of the Foundations for Verification and Validation on the 21st Century Workshop,, pages 1–74, 22-23 October 2002. `file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Articles/W.L.Oberkampf_et_al(2002).pdf`.

[Duxbury et al.(1989)Duxbury, Crisfield, and Hunt] P. G. Duxbury, M. A. Crisfield, and G. W. Hunt. Benchtests for geometric nonlinearities. Computers and Structures, 33 (1):21–29, 1989. file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Articles/P.G.Duxbury_et_al(1989).pdf.

[Crisfield(1983)] M. A. Crisfield. An arc-length method including line searches and accelerations. International Journal for Numerical Methods in Engineering, 19:1296–1289, 1983. file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Articles/M.A.Crisfield(1983).pdf.

[Bryan and Leise(2006)] Kurt Bryan and Tanya Leise. The $25,000,000,000 eigenvector: The linear algebra behind google*. Society for Industrial and Applied Mathematics, 48(3):569–581, 2006. file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Articles/K.Bryan(2006).pdf.

[Bathe et al.(1975)Bathe, Ramm, and Wilson] Klaus Jurgen Bathe, Ekkehard Ramm, and Edward L. Wilson. Finite element formulation for large deformation analysis. International Journal For Numerical Methods In Engineering, 9:353–386, February 1975. file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Articles/K.J.Bathe_et_al(1975).pdf.

[Sinha et al.(2014)Sinha, Biswas, and Manna] Sumeet Kumar Sinha, Sanjit Biswas, and Bappaditya Manna. Nonlinear dynamic response of floating piles under vertical vibration. Computer Methods and Recent Advances in Geomechanics, pages 951–956, 2014. file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Articles/S.K.Sinha_et_al(2014).pdf.

[Sinha et al.(2015)Sinha, Biswas, and Manna] Sumeet Kumar Sinha, Sanjit Biswas, and Bappaditya Manna. Nonlinear characteristics of floating piles under rotating machine induced vertical vibration. Geotechnical And Geological Engineering, 33 (2):1–18, 2015. file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Articles/S.K.Sinha_et_al(2015).pdf.

[Novak(1977)] Milos Novak. Vertical vibrartion of floating piles. Journal of the Engineering Mechanics Division, 103:153–168, 1977. file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Articles/Novak(1977).pdf.

[Novak and Han(1990)] Milos Novak and Y C Han. Impedance of soil layer with boundary zone. Journal of Geotechnical Engineering, 116(6):1008–1014, 1990. file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Articles/Novak_and_Han(1990).pdf.

[Novak and Aboul-Ella(1978)] Milos Novak and Frakhry Aboul-Ella. Impedance of piles in layered media. Journal of the Engineering Mechanics Division, 117: 643–661, 1978. file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Articles/Novak_and_Aboul-Ella(1978).pdf.

[Novak et al.(1978)Novak, Nogami, and Aboul-Ella] Milos Novak, Toyoaki Nogami, and Frakhry Aboul-Ella. Dynamic soil reactions for plain strain case. Journal of the Engineering Mechanics Division, 104:953–959, 1978. file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Articles/Novak_et_al(1978).pdf.

[Vaziri and Aboul-Ella(1993)] Hans Vaziri and Frakhry Aboul-Ella. Impedance of piles in inhomogeneous media. Journal of Geotechnical Engineering, 119:1414–1430, 1993. file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Articles/Vaziri_and_Han(1993).pdf.

[Dafalias and Taiebat(2013)] Y F Dafalias and M Taiebat. Anatomy of rotational hardening in clay plasticity. Geotechnique, 63(16):1406–1418, 2013. file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Articles/Dafalias_and_Taiebat(2013).pdf.

[Dafalias and Taiebat(2011)] Y F Dafalias and M Taiebat. Sanisteel: Simple anisotropic steel plasticity model. Journal of Structural Engineering, 137(2): 185–194, 2011. file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Articles/SaniSteel(2011).pdf.

[Ibrahimbegovic(2009)] Adnan Ibrahimbegovic. Nonlinear Solid Mechanics, volume 160 of Solid Mechanics And Its Applications. Springer, 2009. file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Books/[AdnanIbrahimbegovinic]SolidMechanicsandItsApplications.pdf.

[Laub(2005)] Alan J. Laub. Matrix Analysis for Scientists and Engineers. Siam, 2005. file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Books/[Alan_J._Laub]MatrixAnalysisforScientistsandEngineers.pdf.

[Bathe(1996)] Klaus-Jurgen Bathe. Finite Element Procedures. Prentice Hall, 1996. file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Books/[Bathe,K.J.]FiniteElementProcedures.pdf.

[Kreyszig et al.(2011)Kreyszig, Kreyszig, and Norminton] Erwin Kreyszig, Herbert Kreyszig, and Edward J. Norminton. Advance Engineering Mathematics. John Wiley and Sons, tenth edition, 2011. file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Books/[Erwin_Kreyszig]AdvanceEngineeringMathematics.pdf.

[Yu(2006)] Hai Sui Yu. Plasticity and Geomechanics, volume 13 of Advances in Mechanics and Mathematics. Springer, 2006. file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Books/[HaiSuiYu]PlasticityandGeomechanics.pdf.

[Brenner and Scott(2008)] Susanne C. Brenner and L. Ridgway Scott. The Mathematical Theory of Finite Element Methods, volume 15 of Texts in Applied

Mathematics. Springer, third edition, 2008. file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Books/[SusanneC.Brenner,L.RidgwayScott]TheMathematicalTheoryofFiniteElements.pdf.

[Crisfield(1991a)] M. A. Crisfield. Non-Linear Finite Element Analysis, volume 1. John Wiley and Sons, 1991a. file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Books/[M.A.Crisfield]Non-LinearFiniteElementAnalysis.pdf.

[Hughes(1987)] Thomas J. R. Hughes. The Finite Element Method. Prentice Hall, 1987. file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Books/[ThomasJ.R.Hughes]Thefiniteelementmethod.pdf.

[Jeremic et al.(2016)Jeremic, Yang, Cheng, Jie, Sett, Taiebat, Preisig, Tafazzoli, Tasiopoulou, Pisano, Mer
Boris Jeremic, Zhaohui Yang, Zhao Cheng, Guanzhou Jie, Kallol Sett, Mahdi Taiebat, Matthias Preisig, Nima Tafazzoli, Panagiota Tasiopoulou, Federico Pisano, Jose Antonio Abell Mena, Kohei Watanabe, and Konstantinos Karapiperis. Inelastic Finite Element Method For Pressure Sensitive Materials. 2016. file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Books/[Jeremicetal]CompGeoMechLectureNotes.pdf.

[Slaughter(2002)] William S. Slaughter. The linearized theory of elasticity. Springer Science+Business Media, LLC, first edition, 2002. file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Books/[WilliamS.Slaughter]Thelinearizedtheoryofelasticity.pdf.

[Powrie(2014)] William Powrie. Soil Mechanics concepts and applications. CRC Press, third edition, 2014. file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Books/[WilliamPowrie]SoilMechanicsconceptsandapplications.pdf.

[McGuire et al.(2000)McGuire, Gallagher, and Ziemian] William McGuire, Richard H. Gallagher, and Ronal D. Ziemian. Matrix Structural Analysis. Johm Wiley and Sons, second edition, 2000. file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Books/[WilliamMcGuireetal]MatrixStructuralAnalysis.pdf.

[Hibbeler(2012)] R C Hibbeler. Structural Analysis. Prentice Hall, eighth edition, 2012. file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Books/[RCHibbeler]StructuralAnalysis.pdf.

[Kramer(1996)] Steven L Kramer. Geotechnical Earthquake Engineering. Prentice Hall, first edition, 1996. file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Books/[StevenLKramer]GeotechnicalEarthquakeEngineering.pdf.

[de Borst et al.(2012)de Borst, Crisfield, Remmers, and Verhoosel] Rene de Borst, Mike A. Crisfield, Joris J. C. Remmers, and Clemens V. Verhoosel. Non-Linear Finite Element Analysis of Solids and Structures. Wiley, second edition, 2012. `file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Books/[MCrisfieldetal]Non-LinearFiniteElementAnalysisofSolidsandStructures.pdf`.

[Saran(1999)] Swami Saran. Soil Dynamics and Machine Foundations. Galgotia Publications pvt ltd., first edition, 1999. `file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Books/[SwamiSaran]SoilDynamicsandMachineFoundations.pdf`.

[Fish and ted Belytschko(2007)] Jacob Fish and ted Belytschko. A First Course In Finite Elements. John Wiley and Sons Ltd, first edition, 2007. `file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Books/[FishandBelytschko]AFirstCourseInFEM.pdf`.

[Crisfield(1991b)] M. A. Crisfield. Non-Linear Finite Element Analysis of Soil and Structures, volume 1. John Wiley and Sons, 1991b. `file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Books/[Crisfield]NonlinearfiniteelementanalysisofsolidsandstructuresVolume1.pdf`.

[Crisfield(1997)] M. A. Crisfield. Non-Linear Finite Element Analysis of Soil and Structures - Advance Topics, volume 2. John Wiley and Sons, first edition, 1997. `file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Books/[Crisfield]NonlinearFiniteElementAnalysisofSolidsandStructuresVolume2AdvancedTopics.pdf`.

[Wood(2004)] David Muir Wood. Geotechnical Modelling. E and FN Spon, 2.2 edition, 2004. `file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Books/[DavidMuirWood]GeotechnicalModelling.pdf`.

[Wood(1990)] David Muir Wood. Soil behaviour and critical state soil mechanics. E and FN Spon, first edition, 1990. `file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Books/[DavidMuirWood]Soilbehaviourandcriticalstatesoilmechanics.pdf`.

[Wriggers(2002)] Peter Wriggers. Computational Contact Mechanics. John Wiley and Sons Ltd, first edition, 2002. `file:///home/sumeet/sumeet.kumar507@gmail.com/Academia_Repository/Books/[PeterWriggers]ComputationalContactMechanics.pdf`.