

Real-ESSI Simulator

Domain Specific Language

José Abell, Yuan Feng, Han Yang, Hexiang Wang
and
Boris Jeremić

University of California, Davis, CA



Version: 9Jul2025, 13:38

<http://real-essi.us>

This document is an excerpt from: <http://sokocalo.engr.ucdavis.edu/~jeremic/LectureNotes/>

please use google-chrome to view this PDF so that hyperlinks work



Contents

1	Input, Domain Specific Language (DSL)	1991-2005-2010-2011-2012-2015-2016-2017-2020-2021-	3
1.1	Chapter Summary and Highlights		4
1.2	Introduction		4
1.3	Domain Specific Language (DSL), English Language Binding		5
1.3.1	Running Real-ESSI		5
1.3.2	Finishing Real-ESSI Program Run		7
1.3.3	Real-ESSI Variables, Basic Units and Flow Control		8
1.3.4	Modeling		12
	Modeling, Material Model: Adding a Material Model to the Finite Element Model . .		14
	Modeling, Material Model: Linear Elastic Isotropic Material Model		15
	Modeling, Material Model: Cross Anisotropic Linear Elastic Material Model		16
	Modeling, Material Model: von Mises Associated Material Model with Linear Isotropic and/or Kinematic Hardening		17
	Modeling, Material Model: von Mises Associated Material Model with Isotropic Hard- ening and/or Armstrong-Frederic Nonlinear Kinematic Hardening		18
	Modeling, Material Model: Drucker-Prager Associated Material Model with Linear Isotropic and/or Kinematic Hardening		19
	Modeling, Material Model: Drucker-Prager Associated Material Model with Isotropic Hardening and/or Armstrong-Frederick Nonlinear Kinematic Hardening . . .		20
	Modeling, Material Model: Drucker-Prager Associated Material Model with Isotropic Hardening and/or Armstrong-Frederick Nonlinear Kinematic Hardening and Nonlinear Duncan-Chang Elasticity		21
	Modeling, Material Model: Drucker-Prager Nonassociated Material Model with Linear Isotropic and/or Kinematic Hardening		22
	Modeling, Material Model: Drucker-Prager Nonassociated Material Model with Linear Isotropic and/or Armstrong-Frederick Nonlinear Kinematic Hardening		23

Modeling, Material Model: Hyperbolic Drucker-Prager Nonassociated Material Model with Linear Isotropic and/or Armstrong-Frederick Nonlinear Kinematic Hardening	25
Modeling, Material Model: Rounded Mohr-Coulomb Associated Linear Isotropic Hardening Material Model	27
Modeling, Material Model: Cam Clay Material Model	28
Modeling, Material Model: von Mises Associated Multiple Yield Surface Material Model	29
Modeling, Material Model: von Mises Associated Multiple Yield Surface Material Model that Matches G/G_{max} Curves	30
Modeling, Material Model: Drucker-Prager Nonassociated Multi-Yield Surface Material Model	31
Modeling, Material Model: Drucker-Prager Nonassociated Material Model that Matches G/G_{max} Curves	33
Modeling, Material Model: Rounder Mohr-Coulomb Nonassociated Multi-Yield Surface Material Model	35
Modeling, Material Model: Tsinghua Liquefaction Material Model	37
Modeling, Material Model: SANISand Material Model, version 2004	39
Modeling, Material Model: SANISand Material Model, version 2008	41
Modeling, Material Model: Cosserat Linear Elastic Material Model	44
Modeling, Material Model: von Mises Cosserat Material Model	45
Modeling, Material Model: Uniaxial Linear Elastic, Fiber Material Model	46
Modeling, Material Model: Stochastic Uniaxial Linear Elastic Model	47
Modeling, Material Model: Stochastic Uniaxial Nonlinear Armstrong Frederick Model .	48
Modeling, Material Model: Uniaxial Nonlinear Concrete, Fiber Material Model, version 02	50
Modeling, Material Model: Faria-Oliver-Cervera Concrete Material	51
Modeling, Material Model: Plane Stress Layered Material	52
Modeling, Material Model: Uniaxial Nonlinear Steel, Fiber Material Model, version 01	53
Modeling, Material Model: Uniaxial Nonlinear Steel, Fiber Material Model, version 02	54
Modeling, Material Model: Plane Stress Plastic Damage Concrete Material	55
Modeling, Material Model: Plane Stress Rebar Material	56
Modeling, Nodes: Adding Nodes	57
Modeling, Nodes: Adding Stochastic Nodes	58
Modeling, Nodes: Define Nodal Physical Group	59
Modeling, Nodes: Adding Nodes to Nodal Physical Group	60

Modeling, Nodes: Removing Nodal Physical Group	61
Modeling, Nodes: Print Nodal Physical Group	62
Modeling, Nodes: Removing Nodes	63
Modeling, Nodes: Adding Nodal Mass, for 3DOFs and/or 6DOFs	64
Modeling, Finite Element: Adding Finite Elements	65
Modeling, Finite Element: Define Finite Element Physical Group	66
Modeling, Finite Element: Adding Elements to Physical Element Group	67
Modeling, Finite Element: Remove Physical Finite Element Group	68
Modeling, Finite Element: Print Physical Finite Element Group	69
Modeling, Finite Element: Remove Finite Element	70
Modeling, Finite Element: Truss Element	71
Modeling, Finite Element: Kelvin-Voigt Element	72
Modeling, Finite Element: Inerter Element	73
Modeling, Finite Element: Shear Beam Element	74
Modeling, Finite Element: Stochastic Shear Beam Element	75
Modeling, Finite Element: Elastic Beam–Column Element	76
Modeling, Finite Element: Large Displacement Elastic Beam–Column Element, with Corotational Transformation	79
Modeling, Finite Element: Timoshenko Elastic Beam–Column Element	80
Modeling, Finite Element: Timoshenko Elastic Beam–Column Element with Directional Shear Correction Coefficients	82
Modeling, Finite Element: Adding 1D Fiber to a Beam Cross Section	84
Modeling, Finite Element: Adding Fiber Section to the Finite Element Model	85
Modeling, Finite Element: 3D Displacement Based Fiber Beam-Column Element	86
Modeling, Finite Element: 3D Displacement Based Fiber Beam-Column Element with Corotational Coordinate Transformation	87
Modeling, Finite Element: 3DOF+6DOF=9DOF Beam-Column Element	88
Modeling, Finite Element: 4 Node ANDES Shell with Drilling DOFs	90
Modeling, Finite Element: 3 Node ANDES Shell with Drilling DOFs	91
Modeling, Finite Element: 4 Node Shell NLDKGQ, or 4 Node Shell Xin-Zheng-Lu	92
Modeling, Finite Element: Inelastic Layered Shell Section	93
Modeling, Finite Element: ElasticMembranePlaneStress Element (to be removed!)	94
Modeling, Finite Element: InelasticMembranePlaneStress Element (to be removed!)	95
Modeling, Finite Element: SuperElementLinearElasticImport	96
Modeling, Finite Element: 8 Node Brick Element	98

Modeling, Finite Element: 20 Node Brick Element	100
Modeling, Finite Element: 27 Node Brick Element	102
Modeling, Finite Element: Variable 8-27 Node Brick Element	104
Modeling, Finite Element: 8 Node Brick u-p Element	106
Modeling, Finite Element: 20 Node Brick u-p Element	109
Modeling, Finite Element: 27 Node Brick u-p Element	112
Modeling, Finite Element: 8 Node Brick u-p-U Element	115
Modeling, Finite Element: 20 Node Brick u-p-U Element	118
Modeling, Finite Element: 27 Node Brick u-p-U Element	121
Modeling, Finite Element: 8 Node Cosserat Brick Element	124
Modeling, Finite Element: Bonded Contact/Interface/Joint Element	125
Modeling, Finite Element: Coupled Bonded Contact/Interface/Joint Element	126
Modeling, Finite Element: Force Based Dry Hard Contact/Interface/Joint Element	127
Modeling, Finite Element: Force Based Dry Soft Contact/Interface/Joint Element	128
Modeling, Finite Element: Force Based Coupled Hard Contact/Interface/Joint Element	130
Modeling, Finite Element: Force Based Coupled Soft Contact/Interface/Joint Element	132
Modeling, Finite Element: Stress Based Dry Hard Contact/Interface/Joint Element with Elastic Perfectly Plastic Shear Behavior	134
Modeling, Finite Element: Stress Based Dry Hard Contact/Interface/Joint Element with Nonlinear Hardening Shear Behavior	136
Modeling, Finite Element: Stress Based Dry Hard Contact/Interface/Joint Element with Nonlinear Hardening and Softening Shear Behavior	138
Modeling, Finite Element: Stress Based Dry Soft Contact/Interface/Joint Element with Elastic Perfectly Plastic Shear Behavior	141
Modeling, Finite Element: Stress Based Dry Soft Contact/Interface/Joint Element with Nonlinear Hardening Shear Behavior	143
Modeling, Finite Element: Stress Based Dry Soft Contact/Interface/Joint Element with Nonlinear Hardening and Softening Shear Behavior	145
Modeling, Finite Element: Stress Based Coupled Hard Contact/Interface/Joint Element with Elastic Perfectly Plastic Shear Behavior	148
Modeling, Finite Element: Stress Based Coupled Hard Contact/Interface/Joint Element with Nonlinear Hardening Shear Behavior	150
Modeling, Finite Element: Stress Based Coupled Hard Contact/Interface/Joint Element with Nonlinear Hardening and Softening Shear Behavior	152

Modeling, Finite Element: Stress Based Coupled Soft Contact/Interface/Joint Element with Elastic Perfectly Plastic Shear Behavior	155
Modeling, Finite Element: Stress Based Coupled Soft Contact/Interface/Joint Element with Nonlinear Hardening Shear Behavior	157
Modeling, Finite Element: Stress Based Coupled Soft Contact/Interface/Joint Element with Nonlinear Hardening and Softening Shear Behavior	159
Modeling, Finite Element: Neoprene Isolator Finite Element	162
Modeling, Finite Element: Lead Core Rubber Isolator/Dissipator Element	163
Modeling, Finite Element: Frictional Pendulum Isolator/Dissipator Finite Element ver- sion01	164
Modeling, Finite Element: Frictional Pendulum Isolator/Dissipator Finite Element ver- sion03	165
Modeling, Damping: Adding Rayleigh Damping	166
Modeling, Damping: Adding 3rd Order Caughey Damping	167
Modeling, Damping: Adding 4th Caughey Damping	168
Modeling, Constraints and Supports: Adding Constraints or Supports	169
Modeling, Constraints and Supports: Adding Stochastic Constraints or Supports	170
Modeling, Constraints and Supports: Free Constraint or Support	171
Modeling, Constraints and Supports: Add Tied/Connected Main-Follower Nodes for the Same DOFs	172
Modeling, Constraints and Supports: Adding Tied/Connected, Main-Follower Nodes for Different DOFs	173
Modeling, Constraints and Supports: Remove Tied/Connected Main-Follower equal DOFs	174
Modeling, Constraints and Supports: Adding Single Point Constraint to Nodes	175
Modeling, Acceleration Field: Adding Acceleration/Inertia Field	176
Modeling, Loads: Nodal Loads	177
Modeling, Loads: Nodal Path Loads	179
Modeling, Loads: Nodal Loads From Reactions	180
Modeling, Loads: Selfweight Element Load	181
Modeling, Loads: Selfweight Nodal Load	182
Modeling, Loads: 8 Node Brick Surface Load with the Constant Pressure	183
Modeling, Loads: 8 Node Brick Surface Load with Variable Pressure	184
Modeling, Loads: 20Node Brick Surface Load with the Constant Pressure	185
Modeling, Loads: 20 Node Brick, Surface Load with Variable Pressure	186

Modeling, Loads: 27 Node Brick Surface Load with the Constant Pressure	187
Modeling, Loads: 27 Node Brick Surface Load with Variable Pressure	188
Modeling, Loads: Removing Loads	189
Modeling, Loads: Domain Reduction Method, DRM	190
Modeling, Wave Field for Creating DRM Loads: Add Wave Field	194
Modeling, Wave Field for Creating DRM Loads: Deconvolution	197
Modeling, Wave Field for Creating DRM Input: Motions	198
Modeling, Wave Field for Creating DRM Input: Forces	199
Modeling, Wave Field for Creating DRM Loads: Add Inclined Plane Wave Field from Incident SV Wave Potential Magnitude	200
Modeling, Wave Field for Creating DRM Loads: Add Inclined Plane Wave Field from Incident SV Wave Time Series Signal	203
Modeling, Wave Field for Creating DRM Loads: Add Inclined Plane Wave Field from Incident P Wave Potential Magnitude	206
Modeling, Wave Field for Creating DRM Loads: Add Inclined Plane Wave Field from Incident P Wave Time Series Signal	208
Modeling, Wave Field for Creating DRM Loads: DRM Inclined Motion	210
Modeling, Imposed Motions: through Loads, Motion Time History, Constant Time Step	211
Modeling, Imposed Motions: through Loads, Stochastic Motion Time History, Constant Time Step	212
Modeling, Imposed Motions: through Loads, Stochastic Random Process Motions, Constant Time Step	214
Modeling, Imposed Motions: through Loads, Motion Time History, Variable Time Step	215
Modeling, Imposed Motions: Adding Load for Uniform Acceleration Time History . . .	216
Modeling, Imposed Motions: Remove Imposed Motions	217
Modeling, Random Variable: Adding Gaussian Random Variables	218
Modeling, Random Variable: Adding Gaussian Random Variables with Location	219
Modeling, Random Variable: Adding Lognormal Random Variables	220
Modeling, Random Variable: Adding Lognormal Random Variables with Location . . .	221
Modeling, Random Variable: Adding Lognormal Random Variables using Logarithmic Input	222
Modeling, Random Variable: Adding Lognormal Random Variables using Logarithmic Input with Location	223
Modeling, Random Variable: Adding Gamma Random Variables using Shape and Scale Parameters	224

Modeling, Random Variable: Adding Gamma Random Variables using Shape and Scale Parameters with Location	225
Modeling, Random Variable: Adding Gamma Random Variables using Mean and Standard Deviation Parameters	226
Modeling, Random Variable: Adding Gamma Random Variables using Mean and Standard Deviation Parameters with Location	227
Modeling, Random Variable: Adding Weibull Random Variables using Shape and Scale Parameters	228
Modeling, Random Variable: Adding Weibull Random Variables using Shape and Scale Parameters with Location	229
Modeling, Random Variable: Remove Random Variables	230
Modeling, Random Variable: Hermite Polynomial Chaos Expansion	231
Modeling, Random Variable: Output Hermite Polynomial Chaos Expansion Result . .	232
Modeling, Random Variable: Hermite Polynomial Chaos Expansion & Output Results .	233
Modeling, Random Field: Adding Random Field with Dimension and Order	234
Modeling, Random Field: Define Global Dimension Index of Random Field	235
Modeling, Random Field: Define Global Dimension Index of Random Field from File Input	236
Modeling, Random Field: Set Number of Polynomial Chaos Terms of Random Field .	237
Modeling, Random Field: Adding Random Field with Zero Correlation	238
Modeling, Random Field: Adding Random Field with Exponential Correlation	239
Modeling, Random Field: Adding Random Field with Triangular Correlation	240
Modeling, Random Field: Adding Random Field with Exponentially Damped Cosine Correlation	241
Modeling, Random Field: Adding Random Field with Gaussian Correlation	242
Modeling, Random Field: Remove Random Fields	243
Modeling, Random Field: Adding Random Variable to Random Field	244
Modeling, Random Field: Remove Random Variable From Random Field	245
Modeling, Random Field: Hermite Polynomial Chaos Karhunen Loève Expansion . . .	246
Modeling, Random Field: Hermite Polynomial Chaos Karhunen Loève Expansion with Inverse Order	247
Modeling, Random Field: Hermite Polynomial Chaos Karhunen Loève Expansion with Number of FE Elements Larger than Dimension of Hermite Polynomials . . .	248
Modeling, Random Field: Hermite Polynomial Chaos Karhunen Loève Expansion Using HDF5 Input	250

Modeling, Random Field: Hermite Polynomial Chaos Karhunen Loève Expansion with Inverse Order Using HDF5 Input	252
Modeling, Random Field: Output Hermite Polynomial Chaos Karhunen Loève Expansion Result	254
Modeling, Random Field: Adding Random Field from Hermite Polynomial Chaos Karhunen Loève Expansion HDF5 File	256
Modeling, Random Field: Adding Random Field from Marginal Distribution and Correlation	258
Modeling, Random Field: Add Triple Product of Hermite Polynomial Chaos Basis . . .	259
Modeling, Random Field: Add Double Product of Hermite Polynomial Chaos Basis . .	260
Modeling, Random Field: Generate Triple Product of Hermite Polynomial Chaos Basis	261
Modeling, Random Field: Generate Double Product of Hermite Polynomial Chaos Basis	263
Modeling, Random Field: Add Triple Product of Hermite Polynomial Chaos Basis Using HDF5 Input	265
Modeling, Random Field: Add Double Product of Hermite Polynomial Chaos Basis Using HDF5 Input	267
Modeling, Solid-Fluid Interaction: Adding Solid-Fluid Interface	269
Modeling, Solid-Fluid Interaction: Defining Solid-Fluid Interface, ESSI Element Nodes	270
Modeling, Solid-Fluid Interaction: Defining Solid-Fluid Interface, ESSI Element Faces .	271
Modeling, Solid-Fluid Interaction: Defining Solid-Fluid Interface FOAM Nodes	273
Modeling, Solid-Fluid Interaction: Defining Solid-Fluid Interface FOAM Faces	274
1.3.5 Simulation	276
Simulation, Solvers: Sequential Solvers	277
Simulation, Solvers: Parallel Solvers	278
Simulation: Static Solution Advancement	281
Simulation: Dynamic Solution Advancement with the Constant Time Step	282
Simulation: Dynamic Solution Advancement with Variable Time Step	283
Simulation: Generalized Eigenvalue Analysis	284
Simulation: Displacement Control	285
Simulation: Load, Control, Factor Increment	286
Simulation: Dynamic Integrator, Newmark Method	287
Simulation: Dynamic Integrator, Hilber Hughes Taylor, HHT, α Method	288
Simulation: Absolute Convergence Criteria	289
Simulation: Average Convergence Criteria	290
Simulation: Relative Convergence Criteria	291

Simulation: Solution Algorithms	292
Simulation: Constitutive Integration Algorithm	293
Simulation: Status Check	295
Simulation: Save State	296
Simulation: Restart Simulation	297
Simulation: Return Value for <code>simulate</code> Command	298
Simulation: New Elastic Loading Case	308
Simulation: Combine Elastic Load Cases	309
Simulation, Dynamic Solution Advancement for Solid-Fluid Interaction	310
Simulation, Dynamic Solution Advancement for Stochastic Finite Element Method	311
Simulation, Sobol Sensitivity Analysis	312
Simulation: 3D 3C Wave Field Inversion	316
1.3.6 Output Options	318
Output Options: Enable/Disable Output	319
Output Options: Enable/Disable Element Output	320
Output Options: Enable/Disable Displacement Output	321
Output Options: Enable/Disable Acceleration Output	322
Output Options: Enable/Disable Asynchronous Output	323
Output Options: Output Every n Steps	324
Output Options: Output Support Reactions	325
1.4 Checking the Model	326
1.5 Constitutive Testing	327
1.6 List of Available Commands (tentative, not up to date)	329
1.7 List of reserved keywords	339
1.8 Integrated Development Environment (IDE) for DSL	352
1.9 Mesh Generation using GiD	353
1.10 Model Development and Mesh Generation using gmesh	354
1.11 Model Input File Editing using Sublime	355

Chapter 1

Input, Domain Specific Language (DSL)

1991-2005-2010-2011-2012-2015-2016-2017-2020-2021-

1.1 Chapter Summary and Highlights

1.2 Introduction

This chapter presents the domain specific language developed for the Real-ESSI. The language was designed with a primary goal of developing FEA models and interfacing them with various Real-ESSI functionalities. In addition to that, syntax is used to self-document models, provide physical-unit safety, provide common flow control structures, provide modularity to scripting via user functions and “include” files, and provide an interactive environment within which models can be created, validated and verified.

The development of Real-ESSI Domain specific language (DSL) (the Finite Element Interpreter, $\text{\textcircled{F}}$) is based on LEX (Lesk and Schmidt, 1975) and YACC (Johnson, 1975).

Self-documenting ensures that the resulting model script is readable and understandable with little or no reference to the users manual. This is accomplished by providing a command grammar structure and wording similar to what would be used in a natural language description of the problem.

FEA analysis is unitless, that is, all calculations are carried out without referencing a particular unit system. This leaves the task of unit correctness up to the user of FEA analysis. This represents a recurring source of error in FEA analysis. Physical unit safety is enforced in Real-ESSI by implementing all base variables as physical quantities, that is, all variables have a unit associated with it. The adimensional unit is the base unit for those variables which have no relevant unit (like node numbers). Command calls are sensitive to units. For example, the node creation command call expects the node coordinates to be input with the corresponding units (length in this case). Additionally, the programming/command language naturally supports operation with units like arithmetic operations (quantities with different unit types will not add or subtract but may be multiplied). This approach to FEA with unit awareness provides an additional layer of security to FEA calculations, and forces the user to carefully think about units. This can help catch some common mistakes.

The Real-ESSI language provides modularity through the `include` directive/command, and user functions. This allows complex analysis cases to be parameterized into modules and functions which can be reused in other models.

Finally, an emphasis is placed on model verification and validation. To this end, Real-ESSI provides an interactive programming environment with all the ESSI syntax available. By using this environment. the user can develop tests to detect errors in the model that are not programming errors. For example, the user can query nodes and elements to see if they are set to appropriate states. Also, several standard tools are provided to check element validity (Jacobian, etc.).

The ESSI language provides reduced model development time by providing the aforementioned features along with meaningful error reporting (of syntax and grammatical errors), a help system, command completion and highlighting for several open source and commercial text editors.

Some additional ideas are given by [Dmitriev \(2004\)](#), [Stroustrup \(2005\)](#), [Niebler \(2005\)](#), [Mernik et al. \(2005\)](#), [Ward \(2003\)](#), etc.

1.3 Domain Specific Language (DSL), English Language Binding

Overview of the language syntax.

- Each command line has to end with a semicolon ";"
- Comment on a line begins with either "//" or "!" and last until the end of current line.
- Units are required (see more below) for all quantities and variables.
- Include statements allow splitting source into several files
- All variables are double precision (i.e. floats) with a unit attached.
- All standard arithmetic operations are implemented, and are unit sensitive.
- Internally, all units are represented in the base SI units ($m - s - kg$).
- The syntax ignores extra white spaces, tabulations and newlines. Wherever they appear, they are there for code readability only. (This is why all commands need to end with a semicolon).
- The user should be familiar with the list of the reserved keywords from Section 1.7 on page 339.

1.3.1 Running Real-ESSI

At the command line type "essi", to get to the ESSI prompt and start Real-ESSI in interactive mode.

Command line output

```
The Finite Element Interpreter Endeavor

The Real-ESSI Simulator
Modeling and Simulation of Earthquakes, and Soils, and Structures ←
and their Interaction

Sequential processing mode.

Version Name      : Real-ESSI Global Release, June2018. Release date: ←
Jun 13 2018 at 11:02:19. Tag: adc085ae70
Version Branch    : GLOBAL_RELEASE
Compile Date      : Jun 13 2018 at 14:36:56
Compile User      : jeremic
Compile Sysinfo   : sokocalo 4.13.0-43-generic x86_64 GNU/Linux
```

```

Runtime User      : jeremic
Runtime Sysinfo   : sokocalo 4.13.0-43-generic x86_64 GNU/Linux
Time Now          : Jun 13 2018 at 15:32:52
Days From Release : 0
PostProcessing Compatible Version: ParaView 5.1.2
PostProcessing Compatible Version: ESSI-pvESSI Date: Feb 15 2018 at ↵
11:00:28. Tag: 58fe430a19

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.

ESSI>

```

A number of useful information about Real-ESSI is printed on the screen. From here, commands can be input manually or a file may be included via the `include` command which is as follows.

```
1 include "foobar.fei";
```

to include the file `foobar.fei`.

A more efficient way to start Real-ESSI and analyze an example is to pass input file name to the command line. Real-ESSI command to execute an input file immediately is done by issuing the following command: `essi -f foobar.fei`. This will execute `essi` directly on input file `foobar.fei`. After executing the file, the `essi` interpreter will continue in interactive mode unless the command line flag `-n` or `--no-interactive` is set. A list of command line options is available by calling `essi` from the command line as `essi -h`.

Command line output

```

The Finite Element Interpreter Endeavor

The Real-ESSI Simulator
Modeling and Simulation of Earthquakes, and Soils, and Structures ↵
and their Interaction

Sequential processing mode.

Version Name      : Real-ESSI Global Release, June2018. Release date: ↵
Jun 13 2018 at 11:02:19. Tag: adc085ae70
Version Branch    : GLOBAL_RELEASE
Compile Date      : Jun 13 2018 at 14:36:56
Compile User      : jeremic
Compile Sysinfo   : sokocalo 4.13.0-43-generic x86_64 GNU/Linux
Runtime User      : jeremic
Runtime Sysinfo   : sokocalo 4.13.0-43-generic x86_64 GNU/Linux
Time Now          : Jun 13 2018 at 16:22:08
Days From Release : 0
PostProcessing Compatible Version: ParaView 5.1.2

```

```
PostProcessing Compatible Version: ESSI-pvESSI Date: Feb 15 2018 at ↵
11:00:28. Tag: 58fe430a19
```

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.

The Real-ESSI Simulator

Modeling and Simulation of Earthquakes, and Soils, and Structures and ↵
their Interaction

Usage: essi [-cfhnsmb FILENAME]

```
-c --cpp-output           : Output cpp version of the model.
-f --filename [FILENAME] : run ESSI on a FILENAME.
-h --help                 : Print this message.
-n --no-interactive       : Disable interactive mode.
-s --set-variable        : Set a variable from the command line.
-d --dry-run              : Do not execute ESSI API calls. ↵
  Just parse.
-m --model-name [NAME]   : Set the model name from the ↵
  command line.
-p --profile-report [FILENAME] : Set the filename for the profiler ↵
  report (and activate lightweight profiling)
```

Example to set a variable name from command line:

```
essi -s a=10,b=20,c=30
```

Runs ESSI with variables a, b, and c set to 10, 20 and 30 respectively.
At this time, only ESSIunits::unitless variables can be set.

1.3.2 Finishing Real-ESSI Program Run

To properly finish Real-ESSI program run, and save and close all the output files, user has to use final, closure command:

```
1 bye;
```

Command `bye;` has to be included at the end of input file script, or at the end of each interactive/interpretative session. Command `bye;` ensures that Real-ESSI program gracefully exits simulation, and that all the output files are properly saved and closed. Proper finishing of simulation using Real-ESSI Simulator is very much necessary, while the choice of command `bye;` is done as an homage to Professor Knuth and his Literate Programming endeavor ([Knuth, 1984](#)), that is driving much of the Real-ESSI DSL development.

There are a number of alternative final commands, for example:

```
1 exit;
2 quit;
3 zdravo;
```

```

4  vozdra;
5  dvojka;
6  voljno;
7  zaijian;
8  tschuess;
9  geia-sou;
10 tchau;
11 sair;
12 khoda-hafez;
13 doi;
14 nasvidenje;
15 ajde-bok;
16 izhod;
17 konec;
18 czesc;
19 ciao;
20 hoscakal;

```

These additional, alternative final commands can all be written using original scripts:

zdravo ↔ здраво

vozdra ↔ воздра

dvojka ↔ двојка

voljno ↔ вољно

zaijian ↔ 再见

tschuess ↔ tschüss

geia-sou ↔ γεια σου

khoda-hafez ↔ خداحافظ

hoscakal ↔ hoşçakal

1.3.3 Real-ESSI Variables, Basic Units and Flow Control

Variables are defined using the assignment (=) operator. For example,

```

1  var_x    =  7;           //Results in the variable x be set to 7 (unitless)
2  var_y    =  3.972e+2;    //Scientific notation is available.

```

The language contains a list of reserved keywords. Throughout this documentation, reserved keywords are highlighted in [blue](#) or [red](#).

All standard arithmetic operations are available between variables. These operations can be combined arbitrarily and grouped together with parentheses.

```

1 var_a = var_x + var_y;      // Addition
2 var_b = var_x - var_y;      // Subtraction
3 var_c = var_x * var_y;      // Product
4 var_d = var_x / var_y;      // Quotient
5 var_e = var_y % var_x;      // Modulus (how many times x fits in y)

```

The 'print' command can be used to display the current value of a variable.

```

1 print var_x;
2 print var_y;
3 print var_a;
4 print var_b;
5 print var_c;
6 print var_d;
7 print var_e;

```

Command line output

```

var_x = 7 []
var_y = 397.2 []
var_a = 404.2 []
var_b = -390.2 []
var_c = 2780.4 []
var_d = 0.0176234 []
var_e = 5.2 []

```

Here the “unit” (sign) [] means that the quantities are unitless.

The command 'whos' is used to see all the currently defined variables and their values. After a fresh start of essi, needed to clear up all the previously defined variables, command whos;' produces a list of predefined variables:

Command line output

```

ESSI> whos;

Declared variables:
*      Day =          86400 [s]
*      GPa =           1 [GPa]
*      Hour =         3600 [s]
*      Hz =            1 [Hz]
*      MPa =           1 [MPa]
*      Minute =        60 [s]
*      N =             1 [N]
*      Pa =            1 [Pa]
*      Week =        604800 [s]
*      cm =            1 [cm]
*      feet =         0.3048 [m]
*      ft =            0.3048 [m]

```

```

*      g =          9.81 [m*s^-2]
*      inch =        0.0254 [m]
*      kN =          1 [kN]
*      kPa =          1 [kPa]
*      kg =          1 [kg]
*      kip =        4448.22 [N]
*      km =          1 [km]
*      ksi =        6.89476e+06 [Pa]
*      lbf =         4.44822 [N]
*      lbm =         0.453592 [kg]
*      m =          1 [m]
*      mile =        1609.35 [m]
*      mm =          1 [mm]
*      pi =          3.14159 []
*      psi =         6894.76 [Pa]
*      s =           1 [s]
*      yard =        0.9144 [m]

```

```

* = locked variable
ESSI>

```

Predefined variables shown above have a preceding asterisk to show they are locked variables which cannot be modified. The purpose of these locked variables are to provide names for units. Imperial units are also supported as shown above.

The units for variable are shown between the brackets. Note that unit variables have the same name as their unit, which is not the case for user defined variables. Variables preceded by a star (*) are locked variables which can't be modified.

For example, the variable 'm' defines 'meter'. So to define a new variable L1 which has meter units we do:

```

1 L1 = 1*m;      // Defines L1 to 1 m.
2 L2 = 40*mm;    // Defines L2 to be 40 millimeters.

```

Even though L2 was created with millimeter units, it is stored in base units.

```
print L2; displays
```

Command line output

```
L2 = 0.04 [m]
```

As additional examples, let us define few forces:

```

1 F1 = 10*kN;
2 F2 = 300*N;
3 F3 = 4*kg*g;

```

Here g is the predefined acceleration due to gravity.

Arithmetic operations do check (and enforce) for unit consistency. For example, `foo = L1 + F1;` produces an error because units are not compatible. However, `bar = L1 + L2;` is acceptable. On the other hand,

multiplication, division and modulus, always work because the result produces a quantity with new units (except when the adimensional quantity is involved).

```
1 A = L1*L2;
2 Stress_n = F1 / A;
```

Units for all variables are internally converted to SI units ($kg - m - s$) and stored in that unit system. Variables can be *displayed* using different units by using the [] operator. This does not change the variable, it just displays the value of variable with required unit. For example,

```
1 print Stress_n;           //Print in base SI units.
2 print Stress_n in Pa;     //Print in Pascal
3 print Stress_n in kPa;    //Print in kilo Pascal
```

Command line output

```
Stress_n = 250000 [kg*m^-1*s^-2]

Stress_n = 250000 [Pa]

Stress_n = 250 [kPa]
```

The DSL provides functions to test the physical units of variables. For example,

```
print isForce(F1);
```

Will print an adimensional, Boolean 1 because F1 has units of force. While,

```
print isPressure(F);
```

will print an adimensional, Boolean 0. The language also provides comparison of quantities with same units (remember all values are compared in SI Units).

```
print F1 > F2;
```

will print an adimensional, Boolean 1 since F1 is greater than F2.

The program flow can be controlled with `if` and `while` statements, i.e.:

```
1 if (isForce(F1))
2 {
3   print F1;    // This will be executed
4 };
5
6 if (isForce(L1))
7 {
8   print L1;    // This will not.
9 };
```

Note the necessary semicolon (;) at the closing brace. Unlike C/C++, the braces are always necessary. Closing colon is also always necessary.

The “else” statement is also available:

```

1  if (isForce(L1))
2  {
3    print L1;      // This will not execute
4  }
5  else
6  {
7    print L2;      // This will execute instead
8  };

```

While loops are also available:

```

1  i = 0;
2  while( i < 10)
3  {
4    print i;
5    i = i +1;
6  };

```

1.3.4 Modeling

This section details ESSI modeling commands. Angle brackets $\langle \rangle$ are used for quantity or variable placeholder, that is, they indicate where user input goes. Within the angle brackets, the expected unit type is given as well, i.e., $\langle L \rangle$ means the command expects an input with a value and a length unit. The symbol $\langle . \rangle$ represents the adimensional quantity.

In addition to that, the vertical bar $|$ (“OR” sign)) is used to separate two or more keyword options, i.e. $[a|b|c]$ is used indicate keyword options a or b or c. The symbol $|\dots|$ is used to denote where several long options exist and are explained elsewhere (an example of this is available below in a material model definitions).

All commands require unit consistency. Base units, SI or other can be used as indicated below:

- length, symbol L , units [m, inch, ft]
- mass, symbol M , units [kg, lbm],
- time, symbol T , units [s]

Derived units can also be used:

- angle, symbol rad (radian), unit [*dimensionless*, L/L]
- force, symbol N (Newton), units [N , kN , MN , $M * L/T^2$],
- stress, symbol Pa (Pascal), units [Pa , kPa , MPa , N/L^2 , $M/L/T^2$]

- strain, symbol (no symbol), units $[L/L]$
- mass density, symbol (no symbol), units $[M/L^3]$
- force density, symbol (no symbol), units $[M/L^2/T^2]$

All models have to be named: `model name "model_name_string";` This is important as output files are named based on model name.

Each loading stage has to be named as well. A new loading stage¹ is defined like this:

```
new loading stage "loading stage name string";
```

In addition to model name, loading stage name is used for output file name for given loading stage.

¹See more in section 101.4.5 on page 97 in Jeremić et al. (1989-2025).

Modeling, Material Model: Adding a Material Model to the Finite Element Model

Adding constitutive material model to the finite element model/domain is done using command:

```
1 add material # <.> type |...|  
2 mass_density = <M/L^3>  
3 (more model dependent parameters) ;
```

- Material number # (or alternatively No) is a distinct integer number used to uniquely identify this material.
- Mass density should be defined for each material (even if only static analysis is performed, for example if self weight is to be used as a loading stage).
- Depending on material model, there will be additional material parameters that are defined for each material model/type below:

Starting with version 03-NOV-2015 all elastic-plastic material models require explicit specification of the constitutive integration algorithm. More information on this can be found in [1.3.5](#). Only the material `linear_elastic_isotropic_3d_LT` ignores this option.

Choices for `material_type` are listed below.

1.4 List of Available Commands (tentative, not up to date)

```

1 add acceleration field # <.> ax = <accel> ay = <accel> az = <aaccel> ;
2 add constraint equal_dof with master node # <.> and slave node # <.> ←
  dof to constrain <.>;
3 add constraint equal_dof with node # <.> dof <.> master and node # ←
  <.> dof <.> slave;
4 add damping # <.> to element # <.>;
5 add damping # <.> to node # <.>;
6 add damping # <.> type Caughey3rd with a0 = <1/time> a1 = <time> a2 = ←
  <time^3> stiffness_to_use = ←
  <Initial_Stiffness|Current_Stiffness|Last_Committed_Stiffness>;
7 add damping # <.> type Caughey4th with a0 = <1/time> a1 = <time> a2 = ←
  <time^3> a3 = <time^5> stiffness_to_use = ←
  <Initial_Stiffness|Current_Stiffness|Last_Committed_Stiffness>;
8 add damping # <.> type Rayleigh with a0 = <1/time> a1 = <time> ←
  stiffness_to_use = ←
  <Initial_Stiffness|Current_Stiffness|Last_Committed_Stiffness>;
9 add domain reduction method loading # <.> hdf5_file = <string> ←
  scale_factor = <.>;
10 add domain reduction method loading # <.> hdf5_file = <string>;
11 add element # <.> type 20NodeBrick using <.> Gauss points each ←
  direction with nodes (<.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, ←
  <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>) use ←
  material # <.>;
12 add element # <.> type 20NodeBrick with nodes (<.>, <.>, <.>, <.>, ←
  <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, ←
  <.>, <.>, <.>) use material # <.>;
13 add element # <.> type 20NodeBrick_up using <.> Gauss points each ←
  direction 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>;
14 add element # <.> type 20NodeBrick_up 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>;
15 add element # <.> type 20NodeBrick_upU using <.> Gauss points each ←
  direction 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>;
16 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>;
17 add element # <. .> type 27NodeBrick using <. .> Gauss points each ↵
    direction with nodes (<. .>, <. .>, <. .>, <. .>, <. .>, <. .>, <. .>, <. .>, ↵
    <. .>, <. .>, <. .>, <. .>, <. .>, <. .>, <. .>, <. .>, <. .>, ↵
    <. .>, <. .>, <. .>, <. .>) use material # <. .>;
18 add element # <. .> type 27NodeBrick with nodes (<. .>, <. .>, <. .>, <. .>, ↵
    <. .>, <. .>, <. .>, <. .>, <. .>, <. .>, <. .>, <. .>, <. .>, <. .>, ↵
    <. .>, <. .>, <. .>, <. .>, <. .>, <. .>, <. .>, <. .>, <. .>) use material # <. .>;
19 add element # <. .> type 27NodeBrick_up using <. .> Gauss points each ↵
    direction 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>;
20 add element # <. .> type 27NodeBrick_up 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>;
21 add element # <. .> type 27NodeBrick_upU using <. .> Gauss points each ↵
    direction 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>;
22 add element # <. .> type 27NodeBrick_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>;
23 add element # <. .> type 3NodeShell_ANDES with nodes (<. .>, <. .>, <. .>) ↵
    use material # <. .> thickness = <l> ;
24 add element # <. .> type 4NodeShell_ANDES with nodes (<. .>, <. .>, <. .>, ↵
    <. .>) use material # <. .> thickness = <l> ;
25 add element # <. .> type 4NodeShell_MITC4 with nodes (<. .>, <. .>, <. .>, ↵
    <. .>) use material # <. .> thickness = <L>;
26 add element # <. .> type 4NodeShell_NewMITC4 with nodes (<. .>, <. .>, <. .>, ↵
    <. .>) use material # <. .> thickness = <L>;
27 add element # <. .> type 8_27_NodeBrick using <. .> Gauss points each ↵
    direction with nodes (<. .>, <. .>, <. .>, <. .>, <. .>, <. .>, <. .>, <. .>, ↵

```


[illegible]

```

    <stress> K_f = <stress>;
38 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>;
39 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>);
40 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>);
41 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>);
42 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>);
43 add element # <.> type BeamColumnDispFiber3d with nodes (<.>, <.>)
    number_of_integration_points = <.> section_number = <.> mass_density = <M/L^3> xz_plane_vector = (<.>, <.>, <.>)
    joint_1_offset = (<L>, <L>, <L>) joint_2_offset = (<L>, <L>, <L>);
44 add element # <.> type HardContact with nodes (<.>, <.>)
    axial_stiffness = <F/L> shear_stiffness = <F/L> normal_damping = <F/L> tangential_damping = <F/L> friction_ratio = <.>
    contact_plane_vector = (<.>, <.>, <.>);
45 add element # <.> type HardWetContact with nodes (<.>, <.>)
    axial_stiffness = <F/L> shear_stiffness = <F/L> normal_damping = <F/L> tangential_damping = <F/L> friction_ratio = <.>
    contact_plane_vector = (<.>, <.>, <.>);
46 add element # <.> type ShearBeam with nodes (<.>, <.>) cross_section = <l^2> use material # <.>;
47 add element # <.> type SoftContact with nodes (<.>, <.>)
    initial_axial_stiffness = <F/L> stiffening_rate = <m^-1> shear_stiffness = <F/L> normal_damping = <F/L> tangential_damping = <F/L>
    friction_ratio = <.> contact_plane_vector = (<.>, <.>, <.>);

```

```

    <.> );
48 add element # <.> type SoftWetContact with nodes (<.>, <.>) ←
    initial_axial_stiffness = <F/L> stiffening_rate = <m^-1> ←
    shear_stiffness = <F/L> normal_damping = <F/L> tangential_damping ←
    = <F/L> friction_ratio = <.> contact_plane_vector = (<.>, <.>, ←
    <.> );
49 add element # <.> type truss with nodes (<.>, <.>) use material # <.> ←
    cross_section = <length^2> mass_density = <M/L^3> ;
50 add element # <.> type variable_node_brick_8_to_27 using <.> Gauss ←
    points each direction with nodes (<.>, <.>, <.>, <.>, <.>, <.>, ←
    <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>, ←
    <.>, <.>, <.>, <.>, <.>, <.>, <.>, <.>) use material # <.>;
51 add elements (<.>) to physical_element_group "string";
52 add fiber # <.> using material # <.> to section # <.> ←
    fiber_cross_section = <area> fiber_location = (<L>,<L>);
53 add imposed motion # <.> to node # <.> dof DOFTYPE ←
    displacement_scale_unit = <displacement> displacement_file = ←
    "disp_filename" velocity_scale_unit = <velocity> velocity_file = ←
    "vel_filename" acceleration_scale_unit = <acceleration> ←
    acceleration_file = "acc_filename";
54 add imposed motion # <.> to node # <.> dof DOFTYPE time_step = <t> ←
    displacement_scale_unit = <length> displacement_file = ←
    "disp_filename" velocity_scale_unit = <velocity> velocity_file = ←
    "vel_filename" acceleration_scale_unit = <acceleration> ←
    acceleration_file = "acc_filename";
55 add load # <.> to all elements type self_weight use acceleration ←
    field # <.>;
56 add load # <.> to element # <.> type self_weight use acceleration ←
    field # <.>;
57 add load # <.> to element # <.> type surface at nodes (<.> , <.> , ←
    <.> , <.>) with magnitude <.>;
58 add load # <.> to element # <.> type surface at nodes (<.> , <.> , ←
    <.> , <.>) with magnitudes (<.> , <.> , <.> , <.>);
59 add load # <.> to element # <.> type surface at nodes (<.> , <.> , ←
    <.> , <.>, <.>, <.>, <.>, <.>) with magnitude <.>;
60 add load # <.> to element # <.> type surface at nodes (<.> , <.> , ←
    <.> , <.>, <.>, <.>, <.>, <.>) with magnitudes (<.> , <.> , <.> , ←
    <.>, <.>, <.>, <.>, <.>);
61 add load # <.> to element # <.> type surface at nodes (<.> , <.> , ←
    <.> , <.>, <.>, <.>, <.>, <.>, <.>) with magnitude <.>;
62 add load # <.> to element # <.> type surface at nodes (<.> , <.> , ←
    <.> , <.>, <.>, <.>, <.>, <.>, <.>) with magnitudes (<.> , <.> , ←
    <.> , <.>, <.>, <.>, <.>, <.>, <.>);
63 add load # <.> to node # <.> type from_reactions;

```

```

64 add load # <.> to node # <.> type linear FORCETYPE = <force or ←
    moment>; //FORCETYPE = Fx Fy Fz Mx My Mz F_fluid_x F_fluid_y ←
    F_fluid_z
65 add load # <.> to node # <.> type path_series FORCETYPE = <force or ←
    moment> time_step = <time> series_file = "filename";
66 add load # <.> to node # <.> type path_time_series FORCETYPE = <force ←
    or moment> series_file = "filename";
67 add load # <.> to node # <.> type self_weight use acceleration field ←
    # <.>;
68 add mass to node # <.> mx = <mass> my = <mass> mz = <mass> Imx = ←
    <mass*length^2> Imy = <mass*length^2> Imz = <mass*length^2>;
69 add mass to node # <.> mx = <mass> my = <mass> mz = <mass>;
70 add material # <.> type CamClay mass_density = <M/L^3> M = <.> lambda ←
    = <.> kappa = <.> e0 = <.> p0 = <F/L^2> Poisson_ratio = <.> ←
    initial_confining_stress = <F/L^2>
71 add material # <.> type DruckerPrager mass_density = <M/L^3> ←
    elastic_modulus = <F/L^2> poisson_ratio = <.> druckerprager_k = <> ←
    kinematic_hardening_rate = <F/L^2> isotropic_hardening_rate = ←
    <F/L^2> initial_confining_stress = exp;
72 add material # <.> type DruckerPragerArmstrongFrederickLE ←
    mass_density = <M/L^3> elastic_modulus = <F/L^2> poisson_ratio = ←
    <.> druckerprager_k = <> armstrong_frederick_ha = <F/L^2> ←
    armstrong_frederick_cr = <F/L^2> isotropic_hardening_rate = ←
    <F/L^2> initial_confining_stress = <F/L^2>;
73 add material # <.> type DruckerPragerArmstrongFrederickNE ←
    mass_density = <M/L^3> DuncanChang_K = <.> DuncanChang_pa = ←
    <F/L^2> DuncanChang_n = <.> DuncanChang_sigma3_max = <F/L^2> ←
    DuncanChang_nu = <.> druckerprager_k = <> armstrong_frederick_ha = ←
    <F/L^2> armstrong_frederick_cr = <F/L^2> isotropic_hardening_rate ←
    = <F/L^2> initial_confining_stress = <F/L^2>;
74 add material # <.> type DruckerPragerNonAssociateArmstrongFrederick ←
    mass_density = <M/L^3> elastic_modulus = <F/L^2> poisson_ratio = ←
    <.> druckerprager_k = <> armstrong_frederick_ha = <F/L^2> ←
    armstrong_frederick_cr = <F/L^2> isotropic_hardening_rate = ←
    <F/L^2> initial_confining_stress = <F/L^2> plastic_flow_xi = <> ←
    plastic_flow_kd = <> ;
75 add material # <.> type DruckerPragerNonAssociateLinearHardening ←
    mass_density = <M/L^3> elastic_modulus = <F/L^2> poisson_ratio = ←
    <.> druckerprager_k = <> kinematic_hardening_rate = <F/L^2> ←
    isotropic_hardening_rate = <F/L^2> initial_confining_stress = ←
    <F/L^2> plastic_flow_xi = <> plastic_flow_kd = <> ;
76 add material # <.> type DruckerPragervonMises mass_density = <M/L^3> ←
    elastic_modulus = <F/L^2> poisson_ratio = <.> druckerprager_k = <> ←
    kinematic_hardening_rate = <F/L^2> isotropic_hardening_rate = ←
    <F/L^2> initial_confining_stress = exp;

```

```

77 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>;
78 add material # <.> type linear_elastic_isotropic_3d mass_density = ←
    <M/L^3> elastic_modulus = <F/L^2> poisson_ratio = <.>;
79 add material # <.> type linear_elastic_isotropic_3d_LT mass_density = ←
    <M/L^3> elastic_modulus = <F/L^2> poisson_ratio = <.>;
80 add material # <.> type roundedMohrCoulomb mass_density = <M/L^3> ←
    elastic_modulus = <F/L^2> poisson_ratio = <.> RMC_m = <.> RMC_qa = ←
    <F/L^2> RMC_pc = <F/L^2> RMC_e = <.> RMC_eta0 = <.> RMC_Heta = ←
    <F/L^2> initial_confining_stress = <F/L^2>
81 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> ;
82 add material # <.> type sanisand2004_legacy 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 ←
    = <.>;
83 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 ←
    = <.>;

```

```

84 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>;
85 add material # <.> type uniaxial_elastic elastic_modulus = <F/L^2> <←
    viscoelastic_modulus = <mass / length / time> ;
86 add material # <.> type uniaxial_steel01 yield_strength = <F/L^2> <←
    elastic_modulus = <F/L^2> strain_hardening_ratio = <.> a1 = <.> <←
    a2 = <.> a3 = <> a4 = <.> ;
87 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 = <.> ;
88 add material # <.> type vonMises 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> ;
89 add material # <.> type vonMisesArmstrongFrederick mass_density = <←
    <M/L^3> elastic_modulus = <F/L^2> poisson_ratio = <.> <←
    von_mises_radius = <> armstrong_frederick_ha = <F/L^2> <←
    armstrong_frederick_cr = <F/L^2> isotropic_hardening_rate = <←
    <F/L^2> ;
90 add node # <.> at (<length>,<length>,<length>) with <.> dofs;
91 add nodes (<.>) to physical_node_group "string";
92 add section # <.> type elastic3d elastic_modulus = <F/L^2> <←
    cross_section = <L^2> bending_Iz = <L^4> bending_Iy=<L^4> <←
    torsion_Jx=<L^4> ;
93 add section # <.> type Elastic_Membrane_Plate elastic_modulus = <←
    <F/L^2> poisson_ratio = <.> thickness = <length> mass_density = <←
    <M/L^3>;
94 add section # <.> type FiberSection TorsionConstant_GJ = <F*L^2>
95 add section # <.> type Membrane_Plate_Fiber thickness = <length> use <←
    material # <.>;
96 add single point constraint to node # <.> dof to constrain <dof_type> <←
    constraint value of <corresponding unit>;
97 add uniform acceleration # <.> to all nodes dof <.> time_step = <T> <←
    scale_factor = <.> initial_velocity = <L/S> acceleration_file = <←
    <string>;
98 check mesh filename;
99 compute reaction forces;
100 define algorithm With_no_convergence_check / Newton / Modified_Newton;
101 define convergence test Norm_Displacement_Increment / <←
    Energy_Increment / Norm_Unbalance / <←
    Relative_Norm_Displacement_Increment / Relative_Energy_Increment / <←
    Relative_Norm_Unbalance tolerance = <.> maximum_iterations = <.> <←
    verbose_level = <0>|<1>|<2>;

```

```

102 define dynamic integrator Hilber_Hughes_Taylor with alpha = <.>;
103 define dynamic integrator Newmark with gamma = <.> beta = <.>;
104 define load factor increment <.>;
105 define NDMaterial constitutive integration algorithm Forward_Euler;
106 define NDMaterial constitutive integration algorithm ↵
    Forward_Euler_Subincrement number_of_subincrements =<.>;
107 define NDMaterial constitutive integration algorithm ↵
    Forward_Euler|Forward_Euler_Subincrement|Backward_Euler|Backward_Euler_Subin
    yield_function_relative_tolerance = <.> stress_relative_tolerance ↵
    = <.> maximum_iterations = <.>;
108 define physical_element_group "string";
109 define physical_node_group "string";
110 define solver ProfileSPD / UMFPack;
111 define static integrator displacement_control using node # <.> dof ↵
    DOFTYPE increment <length>;
112 disable asynchronous output;
113 disable element output;
114 disable output;
115 enable asynchronous output;
116 enable element output;
117 enable output;
118 fix node # <.> dofs <.>;
119 fix node # <.> dofs all;
120 free node # <.> dofs <.>;
121 help;
122 if (.) { } else {};
123 if (.) { };
124 model name "name_string";
125 new loading stage "name_string";
126 output every <.> steps;
127 output non_converged_iterations;
128 output support reactions;
129 print <.>;
130 print element # <.>;
131 print node # <.>;
132 print physical_element_group "string";
133 print physical_node_group "string";
134 remove constraint equal_dof node # <.>;
135 remove displacement from node # <.>;
136 remove element # <.>;
137 remove imposed motion # <.>;
138 remove load # <.>;
139 remove node # <.>;
140 remove physical_node_group "string";
141 remove strain from element # <.>;

```



```

142 remove physical_element_group "string";
143 runTest;
144 set output compression level to <.>;
145 simulate <.> steps using static algorithm;
146 simulate <.> steps using transient algorithm time_step = <time>;
147 simulate <.> steps using variable transient algorithm time_step = ←
    <time> minimum_time_step = <time> maximum_time_step = <time> ←
    number_of_iterations = <.>;
148 simulate constitutive testing BARDETMETHOD use material # <.> ←
    scale_factor = <.> series_file = <string> sigma0 = ( <F/L^2> , ←
    <F/L^2> , <F/L^2> , <F/L^2> , <F/L^2> ) verbose_output ←
    = <.>
149 simulate constitutive testing constant mean pressure triaxial strain ←
    control use material # <.> strain_increment_size = <.> ←
    maximum_strain = <.> number_of_times_reaching_maximum_strain = <.>;
150 simulate constitutive testing DIRECT_STRAIN use material # <.> ←
    scale_factor = <.> series_file = <string> sigma0 = ( <F/L^2> , ←
    <F/L^2> , <F/L^2> , <F/L^2> , <F/L^2> ) verbose_output = ←
    <.>
151 simulate constitutive testing drained triaxial strain control use ←
    material # <.> strain_increment_size = <.> maximum_strain = <.> ←
    number_of_times_reaching_maximum_strain = <.>;
152 simulate constitutive testing undrained simple shear use material # ←
    <.> strain_increment_size = <.> maximum_strain = <.> ←
    number_of_times_reaching_maximum_strain = <.>;
153 simulate constitutive testing undrained triaxial stress control use ←
    material # <.> strain_increment_size = <.> maximum_strain = <.> ←
    number_of_times_reaching_maximum_strain = <.>;
154 simulate constitutive testing undrained triaxial use material # <.> ←
    strain_increment_size = <.> maximum_strain = <.> ←
    number_of_times_reaching_maximum_strain = <.>;
155 simulate using eigen algorithm number_of_modes = <.>;
156 ux uy uz Ux Uy Uz rx ry rz;
157 while (.) { };
158 whos;

```


1.5 List of reserved keywords

The following keywords are reserved and cannot be used as variables in a script or interactive session. Doing so would result in a syntax error.

First Order (commands)

```
1 a0
2 a1
3 a2
4 a3
5 a4
6 acceleration
7 acceleration_depth
8 acceleration_file
9 acceleration_filename
10 acceleration_scale_unit
11 add
12 algorithm
13 algorithm
14 all
15 all
16 allowed_subincrement_strain
17 alpha
18 alpha1
19 alpha2
20 and
21 angle
22 armstrong_frederick_cr
23 armstrong_frederick_ha
24 asynchronous
25 at
26 ax
27 axial_penalty_stiffness
28 axial_stiffness
29 axial_viscous_damping
30 ay
31 az
32 bending_Iy
33 bending_Iz
34 beta
35 Beta
36 beta_min
37 case
38 cases
39 characteristic_strength
40 check
41 chi
42 cohesion
43 combine
44 compression
45 compressive_strength
```

```
46 compressive_yield_strength
47 compute
48 confinement
49 confinement_strain
50 constitutive
51 constrain
52 constraint
53 contact_plane_vector
54 control
55 convergence
56 cR1
57 cR2
58 cross_section
59 crushing_strength
60 Current_Stiffness
61 cyclic
62 damage_parameter_An
63 damage_parameter_Ap
64 damage_parameter_Bn
65 damping
66 define
67 depth
68 dilatancy_angle
69 dilation_angle_eta
70 dilation_scale
71 direction
72 disable
73 displacement
74 displacement_file
75 displacement_scale_unit
76 dof
77 dofs
78 dofs
79 domain
80 druckerprager_k
81 DuncanChang_K
82 DuncanChang_n
83 DuncanChang_nu
84 DuncanChang_pa
85 DuncanChang_sigma3_max
86 DYNAMIC_DOMAIN_PARTITION
87 e0
88 each
89 elastic
90 elastic_modulus
91 elastic_modulus_horizontal
92 elastic_modulus_vertical
93 element
94 elements
95 else
96 enable
97 every
```

```
98 factor
99 fiber
100 fiber_cross_section
101 fiber_location
102 field
103 file
104 fix
105 fluid
106 free
107 friction_angle
108 friction_ratio
109 from
110 gamma
111 Gamma
112 Gauss
113 generate
114 GoverGmax
115 h_in
116 hardening_parameters_of_yield_surfaces
117 hardening_parameters_scale_unit
118 hdf5_file
119 hdf5_filenames_list
120 help
121 if
122 imposed
123 Imx
124 Imy
125 Imz
126 in
127 inclined
128 increment
129 initial_axial_stiffness
130 initial_confining_stress
131 initial_elastic_modulus
132 initial_shear_modulus
133 initial_shear_stiffness
134 initial_velocity
135 integration
136 integration_points
137 IntegrationRule
138 integrator
139 interface
140 isotropic_hardening_rate
141 joint_1_offset
142 joint_2_offset
143 K_f
144 K_s
145 k_x
146 k_y
147 k_z
148 kappa
149 kd_in
```

```
150 kinematic_hardening_rate
151 lambda
152 level
153 line_search_beta
154 line_search_eta
155 line_search_max_iter
156 liquefaction_Alpha
157 liquefaction_c_h0
158 liquefaction_Dir
159 liquefaction_dre1
160 liquefaction_Dre2
161 liquefaction_EXP_N
162 liquefaction_gamar
163 liquefaction_mdc
164 liquefaction_mfc
165 liquefaction_pa
166 liquefaction_pmin
167 load
168 load_factors_list
169 loading
170 local_y_vector
171 local_z_vector
172 M
173 M_in
174 magnitude
175 magnitudes
176 mass
177 mass_density
178 master
179 material
180 max_axial_stiffness
181 maximum_iterations
182 maximum_number_of_iterations
183 maximum_strain
184 maximum_stress
185 maximum_time_step
186 method
187 minimal
188 minimum_time_step
189 model
190 model
191 moment_x_stiffness
192 moment_y_stiffness
193 monotonic
194 motion
195 mu
196 mx
197 my
198 mz
199 name
200 NDMaterial
201 new
```

```
202 newton_with_subincrement
203 node
204 nodes
205 number_of_cycles
206 number_of_files
207 number_of_increment
208 number_of_integration_points
209 number_of_iterations
210 number_of_layers
211 number_of_modes
212 number_of_subincrements
213 number_of_times_reaching_maximum_strain
214 of
215 output
216 output
217 output_filename
218 p0
219 parallel
220 peak_friction_coefficient_limit
221 peak_friction_coefficient_rate_of_decrease
222 penalty_stiffness
223 pi1
224 pi2
225 pi3
226 plastic_deformation_rate
227 plastic_flow_kd
228 plastic_flow_xi
229 plot
230 point
231 points
232 poisson_ratio
233 poisson_ratio_h_h
234 poisson_ratio_h_v
235 porosity
236 print
237 propagation
238 pure
239 R0
240 radiuses_of_yield_surface
241 radiuses_scale_unit
242 rate_of_softening
243 reduction
244 reference_pressure
245 remove
246 rempve
247 residual_friction_coefficient
248 restart
249 restart_files
250 results
251 rho_a
252 rho_f
253 rho_s
```

```
254 rho_w
255 RMC_e
256 RMC_eta0
257 RMC_Heta
258 RMC_m
259 RMC_pc
260 RMC_qa
261 RMC_shape_k
262 rounded_distance
263 runTest
264 sanisand2004_A0
265 sanisand2004_c
266 sanisand2004_ch
267 sanisand2004_cz
268 sanisand2004_ec_ref
269 sanisand2004_G0
270 sanisand2004_h0
271 sanisand2004_lambda_c
272 sanisand2004_m
273 sanisand2004_Mc
274 sanisand2004_nb
275 sanisand2004_nd
276 sanisand2004_p_cut
277 sanisand2004_Pat
278 sanisand2004_xi
279 sanisand2004_z_max
280 sanisand2008_A0
281 sanisand2008_alpha_cc
282 sanisand2008_c
283 sanisand2008_ch
284 sanisand2008_cz
285 sanisand2008_ec_ref
286 sanisand2008_G0
287 sanisand2008_h0
288 sanisand2008_K0
289 sanisand2008_k_c
290 sanisand2008_lambda
291 sanisand2008_m
292 sanisand2008_nb
293 sanisand2008_nd
294 sanisand2008_p0
295 sanisand2008_p_in
296 sanisand2008_p_r
297 sanisand2008_Pat
298 sanisand2008_rho_c
299 sanisand2008_theta_c
300 sanisand2008_X
301 sanisand2008_xi
302 sanisand2008_z_max
303 save
304 scale_factor
305 SCOTCHGRAPHPARTITIONER
```

```
306 section
307 section_number
308 sequential
309 series_file
310 set
311 shear
312 shear_length_ratio
313 shear_modulus
314 shear_modulus_h_v
315 shear_stiffness
316 shear_viscous_damping
317 shear_zone_thickness
318 ShearStrainGamma
319 sigma0
320 simulate
321 single
322 size_of_peak_plateau
323 sizes_of_yield_surfaces
324 slave
325 slave
326 soil
327 soil_profile_filename
328 soil_surface
329 solid
330 solver
331 stage
332 steps
333 steps
334 stiffening_rate
335 stiffness_to_use
336 strain
337 strain_at_compressive_strength
338 strain_at_crushing_strength
339 strain_hardening_ratio
340 strain_increment_size
341 stress
342 stress_increment_size
343 stress_relative_tolerance
344 sub-stepping
345 surface
346 surface_vector_relative_tolerance
347 tensile_strength
348 tensile_yield_strength
349 tension_softening_stiffness
350 test
351 test
352 testing
353 thickness
354 time_step
355 to
356 tolerance_1
357 tolerance_2
```

```

358 torsion_Jx
359 torsional_stiffness
360 TorsionConstant_GJ
361 total_number_of_shear_modulus
362 total_number_of_yield_surface
363 triaxial
364 type
365 uniaxial
366 uniaxial_material
367 uniform
368 unit_of_acceleration
369 unit_of_damping
370 unit_of_rho
371 unit_of_vs
372 use
373 using
374 value
375 velocity_file
376 velocity_scale_unit
377 verbose_output
378 viscoelastic_modulus
379 von_mises_radius
380 wave
381 wave1c
382 wave3c
383 while
384 whos
385 with
386 xi_in
387 xz_plane_vector
388 yield_function_relative_tolerance
389 yield_strength
390 yield_surface_scale_unit
391 x
392 y
393 z

```

Second Order (inside commands)

```

1 20NodeBrick
2 20NodeBrick_up
3 20NodeBrick_upU
4 27NodeBrick
5 27NodeBrick_up
6 27NodeBrick_upU
7 3NodeShell_ANDES
8 4NodeShell_ANDES
9 4NodeShell_MITC4
10 4NodeShell_NewMITC4
11 8_27_NodeBrick
12 8_27_NodeBrick_up
13 8_27_NodeBrick_upU

```



```
14 8NodeBrick
15 8NodeBrick_fluid_incompressible_up
16 8NodeBrick_up
17 8NodeBrick_upU
18 Absolute_Norm_Displacement_Increment
19 Absolute_Norm_Unbalanced_Force
20 arclength_control
21 Average_Norm_Displacement_Increment
22 Average_Norm_Unbalanced_Force
23 Backward_Euler
24 BARDETMETHOD
25 beam_9dof_elastic
26 beam_displacement_based
27 beam_elastic
28 beam_elastic_lumped_mass
29 BeamColumnDispFiber3d
30 BearingElastomericPlasticity3d
31 BFGS
32 BondedContact
33 CamClay
34 Caughey3rd
35 Caughey4th
36 constant mean pressure triaxial strain control
37 Cosserat8NodeBrick
38 Cosserat_linear_elastic_isotropic_3d
39 Cosserat_von_Mises
40 DIRECT_STRAIN
41 displacement_control
42 DOFTYPE
43 domain reduction method
44 drained triaxial strain control
45 DruckerPrager
46 DruckerPragerArmstrongFrederickLE
47 DruckerPragerArmstrongFrederickNE
48 DruckerPragerMultipleYieldSurface
49 DruckerPragerMultipleYieldSurfaceGoverGmax
50 DruckerPragerNonAssociateArmstrongFrederick
51 DruckerPragerNonAssociateLinearHardening
52 DruckerPragervonMises
53 dynamic
54 eigen
55 elastic3d
56 Elastic_Membrane_Plate
57 ElasticFourNodeQuad
58 Energy_Increment
59 equal_dof
60 F_fluid_x
61 F_fluid_y
62 F_fluid_z
63 FiberSection
64 ForceBasedCoupledHardContact
65 ForceBasedCoupledSoftContact
```

```
66 ForceBasedElasticContact
67 ForceBasedHardContact
68 ForceBasedSoftContact
69 FORCETYPE
70 Forward_Euler
71 Forward_Euler_Subincrement
72 from_reactions
73 Fx
74 Fy
75 Fz
76 Hilber_Hughes_Taylor
77 HyperbolicDruckerPragerArmstrongFrederick
78 HyperbolicDruckerPragerLinearHardening
79 HyperbolicDruckerPragerNonAssociateArmstrongFrederick
80 HyperbolicDruckerPragerNonAssociateLinearHardening
81 linear
82 linear_elastic_crossanisotropic
83 linear_elastic_isotropic_3d
84 linear_elastic_isotropic_3d_LT
85 Membrane_Plate_Fiber
86 Modified_Newton
87 Mx
88 My
89 Mz
90 Newmark
91 Newton
92 non_converged_iterations
93 NonlinearFourNodeQuad
94 Norm_Displacement_Increment
95 Norm_Unbalance
96 Parallel
97 path_series
98 path_time_series
99 petsc
100 petsc_options_string
101 physical_element_group
102 physical_node_group
103 Pisano
104 PlaneStressLayeredMaterial
105 PlaneStressRebarMaterial
106 PlasticDamageConcretePlaneStress
107 pressure
108 ProfileSPD
109 Rayleigh
110 reaction_forces
111 reactions
112 Relative_Energy_Increment
113 Relative_Norm_Displacement_Increment
114 Relative_Norm_Unbalance
115 Relative_Norm_Unbalanced_Force
116 roundedMohrCoulomb
117 RoundedMohrCoulombMultipleYieldSurface
```

```
118 sanisand2004
119 sanisand2004_legacy
120 sanisand2008
121 self_weight
122 ShearBeam
123 solid fluid interaction transient
124 static
125 StressBasedCoupledHardContact_ElPP1Shear
126 StressBasedCoupledHardContact_NonLinHardShear
127 StressBasedCoupledHardContact_NonLinHardSoftShear
128 StressBasedCoupledSoftContact
129 StressBasedCoupledSoftContact_ElPP1Shear
130 StressBasedCoupledSoftContact_NonLinHardShear
131 StressBasedCoupledSoftContact_NonLinHardSoftShear
132 StressBasedHardContact_ElPP1Shear
133 StressBasedHardContact_NonLinHardShear
134 StressBasedHardContact_NonLinHardSoftShear
135 StressBasedSoftContact_ElPP1Shear
136 StressBasedSoftContact_NonLinHardShear
137 StressBasedSoftContact_NonLinHardSoftShear
138 SuperElementLinearElasticImport
139 support
140 surface
141 transient
142 truss
143 TsinghuaLiquefactionModelCirclePiPlane
144 TsinghuaLiquefactionModelNonCirclePiPlane
145 UMFPack
146 undrained simple shear
147 undrained triaxial
148 undrained triaxial stress control
149 uniaxial_concrete02
150 uniaxial_elastic
151 uniaxial_steel01
152 uniaxial_steel02
153 variable transient
154 variable_node_brick_8_to_27
155 vonMises
156 vonMisesArmstrongFrederick
157 vonMisesMultipleYieldSurface
158 vonMisesMultipleYieldSurfaceGoverGmax
159 With_no_convergence_check
160
161 beta
162 gamma
163 delta
164
165 ux
166 uy
167 uz
168 rx
169 ry
```

```
170 rz
171 Ux
172 Uy
173 Uz
174 p
175 M
176 m
177 kg
178 s
179 cm
180 mm
181 km
182 Hz
183 Minute
184 Hour
185 Day
186 Week
187 ms
188 ns
189 N
190 kN
191 Pa
192 kPa
193 MPa
194 GPa
195 pound
196 lbm
197 lbf
198 inch
199 in
200 feet
201 ft
202 yard
203 mile
204 psi
205 ksi
206 kip
207 g
208 pi
209
210 NUMBER_OF_NODES
211 NUMBER_OF_ELEMENTS
212 CURRENT_TIME
213 NUMBER_OF_SP_CONSTRAINTS
214 NUMBER_OF_MP_CONSTRAINTS
215 NUMBER_OF_LOADS
216 IS_PARALLEL
217 SIMULATE_EXIT_FLAG
218 then
219 while
220 do
221 let
```

```
222 vector
223
224 cos
225 sin
226 tan
227 cosh
228 sinh
229 tanh
230 acos
231 asin
232 atan
233 atan2
234 sqrt
235 exp
236 log10
237 ceil
238 fabs
239 floor
240 log
```

1.6 Integrated Development Environment (IDE) for DSL

1.7 Mesh Generation using GiD

1. Download the latest version of GiD from <http://www.gidhome.com/>, and also get a temporary license (or purchase it...).
2. Download [essi.gid.tar.gz](#), unpack it (`tar -xvzf essi.gid.tar.gz`) in `problemtypes` directory that is located in GiD's root directory.
3. When you run GiD, you will see `essi` in "Data > Problem types", and can start using it...
4. A simple movie with instructions for mesh generation is available: ([Link to a movie, 11MB](#)).

1.8 Model Development and Mesh Generation using gmesh

1.9 Model Input File Editing using Sublime

<http://www.sublimetext.com/>