$\mathscr{D}$-script User's Guide

B. Giffin

December 7, 2016

D-script is a scripting language written for the purposes of automating the generation of analysis-ready `imitor` input files (`.i` and `.m` files) through the creation of a `.d` script file. In comparison to the standard `.i` file, a `.d` file aims to provide a more "descriptive" and flexible specification of problem parameters, whilst also easing the prescription of intricately defined time steps, material regions, time-functions, boundary conditions, and contact interactions.

This document seeks to elaborate on the basic layout of a `.d` file, highlighting some of D-script's key features, and providing a handful of useful examples.

Contents

1 Overview

The basic layout of a `.d` file includes a sequence of "command blocks" which define and control independent options for the analysis. The syntax for specifying these structures is described herein.

1.1 The `.d` File Structure

A typical `.d` file may contain some (or all) of the following command blocks, shown below:

```
VERSION = <string>imitor_version_name

MESH = <string>mesh_file_name

PHYSICS {
  # PHYSICS commands
}

MATERIAL <string>material_name {
  # MATERIAL commands
}

SOLID {
  # SOLID commands
}

DIRECTION <string>direction_name {
  # DIRECTION commands
}

FUNCTION <string>function_name {
  # FUNCTION commands
}

DISPLACEMENT {
  # DISPLACEMENT commands
}

TRACTION/PRESSURE/FOLLOWER {
  # TRACTION/PRESSURE/FOLLOWER commands
}

CONTACT_INTERACTION {
  # CONTACT_INTERACTION commands
}
```

For the most part, the order in which the above command blocks appear is irrelevant/arbitrary. The only exception to this rule occurs if a given command block references a named `MATERIAL`, `DIRECTION`, or `FUNCTION` definition. In these circumstances, the command block for the corresponding `MATERIAL/DIRECTION/FUNCTION` must be defined *before* it can be referenced within another command block.

1.2 Command Block Structure

A command block consists of a (possibly named) scoped region – delimited by curly braces – which is identified by a leading keyword, and contains a series of commands which specify context-specific problem parameters. These commands can be listed in *arbitrary order* within the scope of the command block. For example, a traction boundary condition can be applied to a set of facets defined on the mesh through the invocation of a `TRACTION` command block:

```
TRACTION {
  TYPE = PIOLA|CAUCHY
  FACET_SET = <integer>facet_set_id_1 (<integer>facet_set_id_2 ...)
  DIRECTION = <string>direction_name
  MAGNITUDE = <string>function_name
  SCALE_FACTOR = <float>scale_factor_value
}
```

The user is expected to provide (problem-specific) arguments for all commands appearing in a command block.

1.3 Command Syntax

In general, the specification of problem parameters within the scope of a given command block adheres to the following syntax:

```
KEYWORD = <type>value
```

Keywords are case-insensitive string expressions that contain no blank spaces (e.g. `SCALE_FACTOR`, `scale_factor`, and `Scale_Factor` would all parse to the same keyword, but `SCALE FACTOR` would result in an error due to the included blank space.) A `value`'s data `<type>` can be `integer`, `float`, or `string`. In contrast with keywords, most `value`s of type `string` *are* case-sensitive, unless otherwise noted.

For certain commands, only a finite set of values are accepted, in which case, the options are explicitly enumerated, of which one must be chosen:

```
KEYWORD = OPTION_1|OPTION_2|...|OPTION_N
```

If the options are `string`-valued (e.g. `ON|OFF`), they will be read and interpreted as case-insensitive strings (i.e. specifying `on|off` or `On|Off` is acceptable.)

If a command or its arguments can be optionally specified (omitted), then said command/arguments are denoted in parentheses:

```
KEYWORD = <integer>value_1 (<integer>optional_value_2 ...)
```

All keyword-value pairs are delimited by the `=` sign, such that the inclusion of extra white space is unnecessary, e.g.

```
SCALE_FACTOR=1.0

SCALE_FACTOR= 1.0

SCALE_FACTOR =1.0

SCALE_FACTOR = 1.0
```

are all acceptable syntax.

1.4 Comments

Blank lines, and comments (initiated with the `#` symbol) may be included anywhere in a `.d` file for the purposes of clarity and formatting. For example, the following two code excerpts would be handled identically by the D-script parser:
Excerpt 1:

```
# ============================================================ #
# Define a linear elastic material whose mechanical properties
# conform with 6061 aluminum alloy.
# ============================================================ #

MATERIAL aluminum {
  TYPE = ELASTIC # CM_1
  # Using SI units (kg,m,s)
  DENSITY        = 2.7E3  # kg/m^3
  YOUNGS_MODULUS = 6.9E10 # N/m^2
  POISSONS_RATIO = 0.25   # m/m
} # END MATERIAL aluminum

# ============================================================ #
```

Excerpt 2:

```
MATERIAL aluminum {
  TYPE = ELASTIC
  DENSITY = 2.7E3
  YOUNGS_MODULUS = 6.9E10
  POISSONS_RATIO = 0.25
}
```

1.5 Using `preppy` Variables

D-script supports the use of `preppy`: a Python-based text file preprocessor. Within any text file, `preppy` allows for the inclusion of preprocessing variable "definitions" (phrases containing the `=` symbol), and the evaluation of Python "expressions" which may contain defined variables. "Phrases" (definitions and expressions) which are to be interpreted by `preppy` are enclosed in square brackets, with each expression being replaced by its corresponding Python evaluation. Python functions from the math.py and sys.py libraries are currently supported.

As an example, the following code excerpt:

```
DIRECTION skew_surface_normal {
  [theta = radians(30.0)] # from degrees
  VECTOR = [cos(theta)]  [sin(theta)]  0.0
}
```

would evaluate as

```
DIRECTION skew_surface_normal {
  VECTOR = 0.86602540378  0.5  0.0
}
```

following a (built-in) call to `preppy`.

2 Version

The VERSION command (defined outside of any command block) specifies which release version of imitor (2014 or 2015) is to be used when running the resulting .i and .m files.

The VERSION command appears as:

```
VERSION = V14|V15
```

3 Mesh

The MESH command (defined outside of any command block) specifies which mesh file to reference when considering element/facet/node sets. Currently the mesh file in question must be of type .vtk, and must contain FieldData which specifies appropriate element/facet/node sets (if any). Mesh files which conform to this specific format can be generated with the vtkgen utility.

The MESH command appears as:

```
MESH = <string>mesh_file_name
```

where mesh_file_name must specify either a file name which is local to the .d file's directory, or a file name with its absolute or relative path defined with reference to the .d file.

4 Physics

The PHYSICS command block specifies the main problem parameters and solution options for the analysis, and defines the time discretization within one or more STEP command blocks. Only one PHYSICS command block should appear within the .d file.

The PHYSICS command block appears as:

```
PHYSICS {
  DIMENSION = 2|3
  DEFORMATION = SMALL_DEF|FINITE_DEF
  ANALYSIS = STATIC|DYNAMIC
 (CONTACT = ON|OFF)
  STRESS_TOLERANCE = <float>stress_tolerance_value
  MAX_NR_ITERATIONS = <integer>maximum_newton_raphson_iterations
  MAX_SOLUTION_PASSES = <integer>maximum_solution_passes
  STEP {
    ...
  }
}
```

DIMENSION indicates the spatial dimension of the problem being considered (2D or 3D). DEFORMATION specifies whether or not the problem should operate under the assumption of small strains (SMALL_DEF), or if finite deformations are accounted for (FINITE_DEF). ANALYSIS determines if the problem is to be run using a quasi-static analysis (STATIC), or if a dynamic analysis utilizing the HHT-α time integration scheme is to be used (DYNAMIC). Optionally, the user may explicitly state whether or not the simulation is to include CONTACT via an ON/OFF parameter (by default, CONTACT is set to OFF.) STRESS_TOLERANCE acts as a residual convergence tolerance parameter with the dimension of stress. The MAX_NR_ITERATIONS command specifies the maximum number of Newton-Raphson iterations to be performed for a given time step, whereas the MAX_SOLUTION_PASSES command indicates the maximum allowable number of solution passes to be made during the solution procedure for each time step.

4.1 Step

The STEP command block (which appears inside of the PHYSICS command block) defines the set of time steps to be used for a given duration of the analysis. Multiple STEP command blocks may be specified for the sake of having different sized time steps or different solution strategies during different portions of the analysis, provided that consecutive STEP command blocks share the same start/end times.

The STEP command block appears as:

```
STEP {
  START_TIME = <float>current_step_start_time
  END_TIME = <float>current_step_end_time
  TIME_STEP = <float>time_step_value
  SKIP = <integer>number_of_steps_to_skip_for_output
  SOLUTION_STRATEGY = SC|SU|AC
}
```

START_TIME and END_TIME correspond to the beginning and ending times which bound the time interval for a given STEP (not necessarily the absolute starting/ending times for the entire simulation, as there could be multiple STEP time intervals which together from the complete simulation time interval). TIME_STEP indicates the size of the time step to be used over the specified time interval (this need not be an integer multiple of the STEP time interval, but may result in a variable end-step size.) The SKIP command dictates the number of time steps to "skip over" (omit) for the purposes of output (a value of 0 will result in all steps being output from the analysis, a value of 1 will output every other step, etc.) The SOLUTION_STRATEGY for all time steps defined for a given STEP may be set to one of three options: a stepwise-coupled solution (SC), a stepwise-uncoupled solution (SU), or an alternating-coupled solution (AC). Presently, only the SC option has been tested in imitor, and is therefore recommended.

5 Material

The MATERIAL command block defines a named material model and its parameters. Multiple MATERIAL command blocks may appear within the same .d file, assigning different material properties to different regions of the mesh.

The generic MATERIAL command block appears as:

```
MATERIAL <string>material_name {
  DENSITY = <float>density_value
  TYPE = ELASTIC|HYPERELASTIC|ELASTOPLASTIC
  #
  # Type-dependent material property definitions
  #
}
```

A material_name must be provided following the MATERIAL keyword, such that the material definition can be referenced within a subsequent SOLID command block (refer to section 6.) The DENSITY command indicates the mass density per unit of reference configuration volume of the material, and the TYPE command specifies which of 3 available material models to use: ELASTIC, HYPERELASTIC, or ELASTOPLASTIC, elaborated upon further in the next section.

5.1 Types

Presently, D-script supports 3 material models: an isotropic linear elastic material (ELASTIC), a hyperelastic compressible Mooney-Rivlin solid (HYPERELASTIC), and an elastic-plastic material with a linear isotropic hardening rule (ELASTOPLASTIC). Each of these models possess a type-specific set of material parameters

that must be defined. The following sections describe the list of properties required for each material model.

5.1.1 Elastic

The `MATERIAL` command block for an `ELASTIC` material appears as:

```
MATERIAL <string>elastic_material_name {
  DENSITY = <float>density_value
  TYPE = ELASTIC
  YOUNGS_MODULUS = <float>youngs_modulus_value
  POISSONS_RATIO = <float>poissons_ratio_value
}
```

where the `YOUNGS_MODULUS` and `POISSONS_RATIO` of the material are the only required parameters for the `ELASTIC` model.

If a given analysis is expected to produce sufficiently large strains in the material, then it is recommended that the user select the `HYPERELASTIC` model, instead of the `ELASTIC` model.

5.1.2 Hyperelastic

The `MATERIAL` command block for a `HYPERELASTIC` material appears as:

```
MATERIAL <string>hyperelastic_material_name {
  DENSITY = <float>density_value
  TYPE = HYPERELASTIC
  D1  = <float>D1_value
  C1 = <float>C1_value
  C2 = <float>C2_value
}
```

where `C1` and `C2` are related to the distortional response of the material, and `D1` is related to its volumetric response.

To remain consistent with small-strain linear elasticity, `D1`, `C1`, and `C2` should be specified such that

$$\kappa = 2D_1, \qquad \mu = 2(C_1 + C_2),$$

where κ and μ are the bulk and shear moduli of the material, respectively.

Moreover, if we impose $C_2 = 0$, then we recover the special case of a neo-Hookean solid.

5.1.3 Elastoplastic

The `MATERIAL` command block for an `ELASTOPLASTIC` material appears as:

```
MATERIAL <string>elastoplastic_material_name {
  DENSITY = <float>density_value
  TYPE = ELASTOPLASTIC
  YOUNGS_MODULUS = <float>youngs_modulus_value
  POISSONS_RATIO = <float>poissons_ratio_value
  HARDENING_MODULUS = <float>hardening_modulus_value
  YIELD_STRESS = <float>initial_yield_stress_value
}
```

where, in addition to the `YOUNGS_MODULUS` and `POISSONS_RATIO`, a linear `HARDENING_MODULUS` and initial `YIELD_STRESS` must be specified.

6 Solid

The SOLID command block associates sets of elements defined on the mesh with specific element formulations and defined material models. Multiple SOLID command blocks may appear within the same .d file, affiliating different material properties or element formulations with different element sets.

The SOLID command block appears as:

```
SOLID {
  ELEMENT_SET = <integer>element_set_1 (<integer>element_set_2 ...)
  MATERIAL = <string>material_name
 (FORMULATION = <string>element_formulation)
}
```

The ELEMENT_SET command identifies one or more element set ids, corresponding to sets of continuum elements defined within the chosen MESH file. MATERIAL specifies the named material model (defined in a separate MATERIAL command block) that is to be associated with the aforementioned set(s) of elements.

Optionally, an element FORMULATION may be specified if an alternative method for constructing shape functions is to be used (i.e. PEM), and/or if a selective reduced quadrature scheme is desired (i.e. mean-dilatation). This option is currently considered a work-in-progress.

7 Direction

The DIRECTION command block defines a named direction vector, for the sake of later prescribing displacement or traction boundary conditions. Multiple DIRECTION command blocks may appear within the same .d file.

The DIRECTION command block appears as:

```
DIRECTION <string>direction_name {
  VECTOR = <float>x_component <float>y_component (<float>z_component)
}
```

where VECTOR specifies the Cartesian components of a 2D (or 3D) direction vector, which need not have unit magnitude. A direction_name must be provided following the DIRECTION keyword, such that the defined direction can be referenced within subsequent DISPLACEMENT (9), TRACTION (10), or FOLLOWER (11) command blocks.

8 Function

The FUNCTION command block defines a (named) time-varying function through one of two possible methods: via a mathematical EXPRESSION written as a function of t, or through a TABLE of explicit abscissa-ordinate pairs. Multiple FUNCTION command blocks may appear within the same .d file.

Generically, the FUNCTION command block appears as:

```
FUNCTION <string>function_name {
  #
  # EXPRESSION or TABLE definition
  #
}
```

The user must provide a function_name following the FUNCTION keyword, such that the defined function can be referenced within subsequent DISPLACEMENT (9), TRACTION (10), FOLLOWER (11), or PRESSURE (12) command blocks.

8.1 Expression

The FUNCTION command block which defines a time-function through the use of an EXPRESSION appears as:

```
FUNCTION <string>function_name {
  EXPRESSION = <string>function_of_t
}
```

The argument to the EXPRESSION command must be a mathematical function of the variable t, and must appear in quotations, e.g.:

```
EXPRESSION = "5.0*sin(t*pi/2)"
```

Expressions are interpreted and evaluated in Python, and currently support the usage of Python functions found in the math.py library.

8.2 Table

The FUNCTION command block which defines a time-function through the use of a nested TABLE command block appears as:

```
FUNCTION <string>function_name {
  TABLE {
     <float>time_0  <float>value_0
     <float>time_1  <float>value_1
    (<float>time_2  <float>value_2)
     ...
  }
}
```

where each line in the TABLE command block lists an abscissa-ordinate pair for the evaluated value_k of the function when $t =$ time_k. An arbitrary number of abscissa-ordinate pairs may be provided. Function values at intermediate times are computed via linear interpolation.

9 Displacement

The DISPLACEMENT command block defines a (possibly time-varying) displacement boundary condition applied to a specified set of nodes in the mesh. Multiple DISPLACEMENT command blocks may appear within the same .d file, each constraining different components of the displacement for the same (or different) node sets.

The DISPLACEMENT command block appears as:

```
DISPLACEMENT {
  NODE_SET = <integer>node_set_id_1 (<integer>node_set_id_2 ...)
  DIRECTION = <string>direction_name
  MAGNITUDE = <string>function_name
  SCALE_FACTOR = <float>scale_factor_value
}
```

The NODE_SET command identifies one or more node set ids, corresponding to sets of nodes defined within the chosen MESH file. The DIRECTION command specifies the named direction (defined in a separate DIRECTION command block) of the displacement component that is to be constrained for the associated set(s) of nodes. The indicated displacement component is prescribed to have a value proportional in MAGNITUDE to the time-varying FUNCTION definition whose corresponding function_name is given, and scaled by the indicated SCALE_FACTOR.

10 Traction

The TRACTION command block defines a (possibly time-varying) traction boundary condition applied to a specified set of facets in the mesh. Multiple TRACTION command blocks may appear within the same .d file, and can be applied to the same (or different) facet sets.

The TRACTION command block appears as:

```
TRACTION {
  TYPE = PIOLA|CAUCHY
  FACET_SET = <integer>facet_set_id_1 (<integer>facet_set_id_2 ...)
  DIRECTION = <string>direction_name
  MAGNITUDE = <string>function_name
  SCALE_FACTOR = <float>scale_factor_value
}
```

TYPE indicates whether the applied natural boundary condition is a PIOLA traction (a force per unit of area in the *reference* configuration), or a CAUCHY traction (a force per unit of area in the *current* configuration). The FACET_SET command identifies one or more facet set ids, corresponding to sets of facets defined within the chosen MESH file. The DIRECTION command specifies the named direction (defined in a separate DIRECTION command block) of the traction vector that is to be applied the associated set(s) of facets. The indicated traction vector is prescribed to have a value proportional in MAGNITUDE to the time-varying FUNCTION definition whose corresponding function_name is given, and scaled by the indicated SCALE_FACTOR.

11 Follower

The FOLLOWER command block defines a (possibly time-varying) "follower" traction boundary condition applied to a specified set of facets in the mesh. In contrast with a standard TRACTION vector which preserves its *globally* defined direction throughout the analysis, a FOLLOWER traction vector maintains the same direction *relative* to the orientation (in the current configuration) of the facet(s) to which the traction is applied (hence the term "follower.") Multiple FOLLOWER command blocks may appear within the same .d file, and can be applied to the same (or different) facet sets.

The FOLLOWER command block appears as:

```
FOLLOWER {
  TYPE = PIOLA|CAUCHY
  FACET_SET = <integer>facet_set_id_1 (<integer>facet_set_id_2 ...)
  DIRECTION = <string>direction_name
  MAGNITUDE = <string>function_name
  SCALE_FACTOR = <float>scale_factor_value
}
```

TYPE indicates whether the applied natural boundary condition is a PIOLA traction (a force per unit of area in the *reference* configuration), or a CAUCHY traction (a force per unit of area in the *current* configuration). The FACET_SET command identifies one or more facet set ids, corresponding to sets of facets defined within the chosen MESH file. The DIRECTION command specifies the named direction (defined in the global coordinate system) of the traction vector's *initial* orientation in the *reference* configuration. The indicated traction vector is prescribed to have a value proportional in MAGNITUDE to the time-varying FUNCTION definition whose corresponding function_name is given, and scaled by the indicated SCALE_FACTOR.

12 Pressure

The PRESSURE command block defines a (possibly time-varying) "pressure" traction boundary condition applied to a specified set of facets in the mesh. In contrast with a standard TRACTION vector which can act in any direction, a PRESSURE traction vector's direction is always *normal* to the facet(s) upon which the pressure is applied. Multiple PRESSURE command blocks may appear within the same .d file, and can be applied to the same (or different) facet sets.

The PRESSURE command block appears as:

```
PRESSURE {
  TYPE = PIOLA|CAUCHY
  FACET_SET = <integer>facet_set_id_1 (<integer>facet_set_id_2 ...)
  MAGNITUDE = <string>function_name
  SCALE_FACTOR = <float>scale_factor_value
}
```

TYPE indicates whether the applied natural boundary condition is a PIOLA traction (a force per unit of area in the *reference* configuration), or a CAUCHY traction (a force per unit of area in the *current* configuration). The FACET_SET command identifies one or more facet set ids, corresponding to the sets of facets (defined within the chosen MESH file) to which the pressure will be applied. The pressure is prescribed to have a value proportional in MAGNITUDE to the time-varying FUNCTION definition whose corresponding function_name is given, and scaled by the indicated SCALE_FACTOR.

13 Contact Interaction

The CONTACT_INTERACTION command block defines a frictionless contact constraint between two disjoint sets of facets. This constraint is only enforced if CONTACT is ON in the PHYSICS command block (4). Multiple CONTACT_INTERACTION command blocks may appear within the same .d file, and can be applied to the same (or different) facet sets.

The CONTACT_INTERACTION command block appears as:

```
CONTACT_INTERACTION {
  FACET_SET_1 = <integer>facet_set_id_1
  FACET_SET_2 = <integer>facet_set_id_2
}
```

where FACET_SET_1 and FACET_SET_2 identify facet set ids, corresponding to two disjoint sets of facets defined within the chosen MESH file.

14 Examples

In the following section, we provide a few practical examples for the sake of illustrating what a typical `.d` file looks like. Three examples are provided: a square pipe loaded in torsion, a thin plate subjected to a blast pressure load, and a rigid block contacting an elastic surface.

14.1 Example 1: Tapered Square Pipe in Torsion

```
# ================================================================== #
#   EXAMPLE 1: TAPERED SQUARE PIPE IN TORSION
# ------------------------------------------------------------------ #
#   This example problem defines a square pipe geometry consisting of
#   PEM hex8 elements, and uses the "hyperelastic" material model.
#   One end of the pipe has all of its displacements degrees of freedom
#   constrained (fixed), whilst the other end of the pipe has follower
#   tractions applied to it which act in the "circumferential"-
#   direction, such that pipe deforms quasi-statically in torsion.
# ================================================================= #
#
# Define preppy variables:
#
# Time Control Parameters:
[T = 10.0]            # analysis end time
[N = 10]              # N time steps
#
# Material Parameters:
[E = 1.0E7]           # Elastic Modulus
[NU = 0.25]           # Poissons' Ratio
[G = E/(2*(1+NU))]    # (derived) Shear Modulus
[K = E/(3*(1-2*NU))] # (derived) Bulk Modulus
# Hyperelastic Parameter "beta": ranges from 0.0 (neo-Hookean) to 1.0
[BETA = 0.0]
#
# Boundary Condition Parameters:
[TAU = 2.0E6]         # magnitude of the applied torsional shear stresses
#
# ================================================================= #

VERSION = V15

MESH = ../../vtkgen/tests/example1.vtk

PHYSICS {
  DIMENSION = 3
  DEFORMATION = FINITE_DEF
  ANALYSIS = STATIC
  STRESS_TOLERANCE = 0.02
  MAX_NR_ITERATIONS = 50
  MAX_SOLUTION_PASSES = 100
```

```
  STEP {
    START_TIME = 0.0
    END_TIME = [T]
    TIME_STEP = [T/N]
    SKIP = 0
#   Use a stepwise-coupled (SC) solution strategy:
    SOLUTION_STRATEGY = SC
  }
}

MATERIAL ALUMINUM {
  TYPE = HYPERELASTIC
# SI Units (kg,m,s)
  DENSITY = 2.7E3
  D1 = [K/2]
  C1 = [BETA*G/2]
  C2 = [(1.0-BETA)*G/2]
}

SOLID {
  ELEMENT_SET = 1
  MATERIAL = ALUMINUM
  FORMULATION = pem_hex8q12
}

DIRECTION X_DIR {
         #  X    Y    Z
  VECTOR = 1.0  0.0  0.0
}

DIRECTION Y_DIR {
         #  X    Y    Z
  VECTOR = 0.0  1.0  0.0
}

DIRECTION Z_DIR {
         #  X    Y    Z
  VECTOR = 0.0  0.0  1.0
}

FUNCTION RAMP {
  TABLE {
  # abscissa ordinate
    0.0       0.0
    [T]       1.0
    [2*T]     1.0
  }
}
```

```
FUNCTION CONSTANT {
  TABLE {
  # abscissa ordinate
    0.0       1.0
    [T]       1.0
  }
}

# Fully constrain the fixed end of the square pipe (node set 1).
DISPLACEMENT {
  NODE_SET = 1
  SCALE_FACTOR = 0.0
  DIRECTION = X_DIR
  MAGNITUDE = CONSTANT
}

DISPLACEMENT {
  NODE_SET = 1
  SCALE_FACTOR = 0.0
  DIRECTION = Y_DIR
  MAGNITUDE = CONSTANT
}

DISPLACEMENT {
  NODE_SET = 1
  SCALE_FACTOR = 0.0
  DIRECTION = Z_DIR
  MAGNITUDE = CONSTANT
}

# Assign follower tractions to the faces at the free end
#   of the square pipe (facet sets 1-4).

# bottom-edge
FOLLOWER {
  FACET_SET = 1
  TYPE = PIOLA
  SCALE_FACTOR = [+TAU]
  DIRECTION = X_DIR
  MAGNITUDE = RAMP
}

# left-edge
FOLLOWER {
  FACET_SET = 2
  TYPE = PIOLA
  SCALE_FACTOR = [-TAU]
  DIRECTION = Y_DIR
  MAGNITUDE = RAMP
}
```

```
# top-edge
FOLLOWER {
  FACET_SET = 3
  TYPE = PIOLA
  SCALE_FACTOR = [-TAU]
  DIRECTION = X_DIR
  MAGNITUDE = RAMP
}

# right-edge
FOLLOWER {
  FACET_SET = 4
  TYPE = PIOLA
  SCALE_FACTOR = [+TAU]
  DIRECTION = Y_DIR
  MAGNITUDE = RAMP
}
```

14.2 Example 2: Blast Pressure Loaded Plate

```
# ================================================================== #
#   EXAMPLE 2: BLAST PRESSURE LOADED PLATE
# ------------------------------------------------------------------ #
#   This example problem defines a square plate geometry consisting of
#   isoparametric hex20 elements, and uses the "elastoplastic" material
#   model. The edges of the plate have all of their displacement
#   degrees of freedom held fixed, whilst the top surface of the plate
#   has a Cauchy pressure applied to it whose time history is
#   consistent with a fluid blast wave. Because inertial effects are
#   significant for this problem, the analysis setting has been
#   set to DYNAMIC.
# ================================================================== #
#
# Define preppy variables:
#
# Time Control Parameters:
[T = 0.2]       # analysis end time
[N = 40]        # N time steps
#
# Blast Pressure Parameters:
[PMAG = 3.0e5]  # magnitude of the initial blast pressure
[TP = 0.01]     # time period over which the blast pressure is positive
[B = 1.0]       # the blast pressure decay constant
#
# ================================================================== #

VERSION = V15

MESH = ../../vtkgen/tests/example2.vtk
```

```
PHYSICS {
  DIMENSION = 3
  DEFORMATION = FINITE_DEF
  ANALYSIS = DYNAMIC
  STRESS_TOLERANCE = 0.02
  MAX_NR_ITERATIONS = 50
  MAX_SOLUTION_PASSES = 100
  STEP {
    START_TIME = 0.0
    END_TIME = [T]
    TIME_STEP = [T/N]
    SKIP = 0
    # Use a stepwise-coupled (SC) solution strategy:
    SOLUTION_STRATEGY = SC
  }
}

MATERIAL ALUMINUM {
  TYPE = ELASTOPLASTIC
# SI Units (kg,m,s)
  DENSITY = 2.7E3
  YOUNGS_MODULUS = 1.0E7
  POISSONS_RATIO = 0.25
  HARDENING_MODULUS = 1.0E5
  YIELD_STRESS = 2.4E5
}

SOLID {
  ELEMENT_SET = 1
  MATERIAL = ALUMINUM
}

DIRECTION X_DIR {
         #  X    Y    Z
  VECTOR = 1.0  0.0  0.0
}

DIRECTION Y_DIR {
         #  X    Y    Z
  VECTOR = 0.0  1.0  0.0
}

DIRECTION Z_DIR {
         #  X    Y    Z
  VECTOR = 0.0  0.0  1.0
}
```

```
FUNCTION CONSTANT {
  TABLE {
  # abscissa ordinate
    0.0       1.0
    [T]       1.0
  }
}

# Define an exponentially decaying blast load with a pressure sign reversal
FUNCTION BLAST {
  EXPRESSION = "[PMAG]*(1.0-t/[TP])*exp(-[B]*t/[TP])"
}

# Constrain the edges of the elastic plate (node set 1)
DISPLACEMENT {
  NODE_SET = 1
  SCALE_FACTOR = 0.0
  DIRECTION = X_DIR
  MAGNITUDE = CONSTANT
}
DISPLACEMENT {
  NODE_SET = 1
  SCALE_FACTOR = 0.0
  DIRECTION = Y_DIR
  MAGNITUDE = CONSTANT
}
DISPLACEMENT {
  NODE_SET = 1
  SCALE_FACTOR = 0.0
  DIRECTION = Z_DIR
  MAGNITUDE = CONSTANT
}

# Assign the blast pressure to the top face of the plate (facet set 1)
PRESSURE {
  TYPE = CAUCHY
  FACET_SET = 1
  SCALE_FACTOR = 1.0
  MAGNITUDE = BLAST
}
```

14.3 Example 3: Rigidly Rotating Block Contacting an Elastic Surface

```
# ================================================================== #
#   EXAMPLE 3: RIGIDLY ROTATING BLOCK CONTACTING AN ELASTIC SURFACE
# ------------------------------------------------------------------ #
#   This example problem defines a rigid cube and an elastic surface
#   geometry consisting of isoparametric hex8 elements, and uses the
#   "elastic" material model. The bottom face of the elastic surface
#   has all of its displacement degrees of freedom held fixed, whilst
#   each vertical edge of the rigid block has its displacement
#   components independently controlled, such that the block rigidly
#   rotates and translates downward to come into contact with the
#   elastic surface. A pair of facet sets are defined on the top face
#   of the elastic surface and on the bottom face of the rigid block,
#   such that a contact interaction can be defined between them.
# ================================================================== #
#
# Define preppy variables:
#
# Time Control Parameters:
[T = 10.0]          # analysis end time
[N = 100]           # N time steps
[DP = 0.1]          # output (plotted) time step
#
# Block Displacement Parameters:
[D = 0.5]           # vertical block movement
[H = 1.1]           # block width
[R = sqrt(2)*H]     # block circumscribing radius
[OMEGA = 0.15621]   # angular velocity multiplier
#
# ================================================================== #

VERSION = V15

MESH = ../../vtkgen/tests/example3.vtk

PHYSICS {
  DIMENSION = 3
  DEFORMATION = FINITE_DEF
  ANALYSIS = STATIC
  CONTACT = ON
  STRESS_TOLERANCE = 0.02
  MAX_NR_ITERATIONS = 50
  MAX_SOLUTION_PASSES = 100
```

```
  STEP {
    START_TIME = 0.0
    END_TIME = [T]
    TIME_STEP = [T/N]
    SKIP = [DP*N/T-1]
    # Use a stepwise-coupled (SC) solution strategy:
    SOLUTION_STRATEGY = SC
  }
}

MATERIAL ALUMINUM {
  TYPE = ELASTIC
# SI Units (kg,m,s)
  DENSITY = 2.7E3
  YOUNGS_MODULUS = 1.0E7
  POISSONS_RATIO = 0.25
}

SOLID {
  ELEMENT_SET = 1
  MATERIAL = ALUMINUM
}

DIRECTION X_DIR {
         #  X    Y    Z
  VECTOR = 1.0  0.0  0.0
}

DIRECTION Y_DIR {
         #  X    Y    Z
  VECTOR = 0.0  1.0  0.0
}

DIRECTION Z_DIR {
         #  X    Y    Z
  VECTOR = 0.0  0.0  1.0
}

FUNCTION RAMP {
  TABLE {
  # abscissa ordinate
    0.0       0.0
    [T]       1.0
    [2*T]     1.0
  }
}
```

```
FUNCTION CONSTANT {
  TABLE {
  # abscissa ordinate
    0.0       1.0
    [T]       1.0
  }
}

# Constrain the bottom face of the elastic surface (node set 1)
DISPLACEMENT {
  NODE_SET = 1
  SCALE_FACTOR = 0.0
  DIRECTION = X_DIR
  MAGNITUDE = CONSTANT
}
DISPLACEMENT {
  NODE_SET = 1
  SCALE_FACTOR = 0.0
  DIRECTION = Y_DIR
  MAGNITUDE = CONSTANT
}
DISPLACEMENT {
  NODE_SET = 1
  SCALE_FACTOR = 0.0
  DIRECTION = Z_DIR
  MAGNITUDE = CONSTANT
}

# Constrain the edges of the rigid block (node sets 2-5)
# Edge 1:
FUNCTION UX1 {
  EXPRESSION = "[R]*(cos([OMEGA]*pi*t/[T]+5*pi/4)-cos(5*pi/4))"
}
FUNCTION UY1 {
  EXPRESSION = "[R]*(sin([OMEGA]*pi*t/[T]+5*pi/4)-sin(5*pi/4))"
}

DISPLACEMENT {
  NODE_SET = 2
  SCALE_FACTOR = 1.0
  DIRECTION = X_DIR
  MAGNITUDE = UX1
}
DISPLACEMENT {
  NODE_SET = 2
  SCALE_FACTOR = 1.0
  DIRECTION = Y_DIR
  MAGNITUDE = UY1
}
```

```
DISPLACEMENT {
  NODE_SET = 2
  SCALE_FACTOR = [-D]
  DIRECTION = Z_DIR
  MAGNITUDE = RAMP
}

# Edge 2:
FUNCTION UX2 {
  EXPRESSION = "[R]*(cos([OMEGA]*pi*t/[T]-pi/4)-cos(-pi/4))"
}
FUNCTION UY2 {
  EXPRESSION = "[R]*(sin([OMEGA]*pi*t/[T]-pi/4)-sin(-pi/4))"
}

DISPLACEMENT {
  NODE_SET = 3
  SCALE_FACTOR = 1.0
  DIRECTION = X_DIR
  MAGNITUDE = UX2
}
DISPLACEMENT {
  NODE_SET = 3
  SCALE_FACTOR = 1.0
  DIRECTION = Y_DIR
  MAGNITUDE = UY2
}
DISPLACEMENT {
  NODE_SET = 3
  SCALE_FACTOR = [-D]
  DIRECTION = Z_DIR
  MAGNITUDE = RAMP
}

# Edge 3:
FUNCTION UX3 {
  EXPRESSION = "[R]*(cos([OMEGA]*pi*t/[T]+pi/4)-cos(pi/4))"
}
FUNCTION UY3 {
  EXPRESSION = "[R]*(sin([OMEGA]*pi*t/[T]+pi/4)-sin(pi/4))"
}

DISPLACEMENT {
  NODE_SET = 4
  SCALE_FACTOR = 1.0
  DIRECTION = X_DIR
  MAGNITUDE = UX3
}
```

```
DISPLACEMENT {
  NODE_SET = 4
  SCALE_FACTOR = 1.0
  DIRECTION = Y_DIR
  MAGNITUDE = UY3
}
DISPLACEMENT {
  NODE_SET = 4
  SCALE_FACTOR = [-D]
  DIRECTION = Z_DIR
  MAGNITUDE = RAMP
}

# Edge 4:
FUNCTION UX4 {
  EXPRESSION = "[R]*(cos([OMEGA]*pi*t/[T]+3*pi/4)-cos(3*pi/4))"
}
FUNCTION UY4 {
  EXPRESSION = "[R]*(sin([OMEGA]*pi*t/[T]+3*pi/4)-sin(3*pi/4))"
}

DISPLACEMENT {
  NODE_SET = 5
  SCALE_FACTOR = 1.0
  DIRECTION = X_DIR
  MAGNITUDE = UX4
}
DISPLACEMENT {
  NODE_SET = 5
  SCALE_FACTOR = 1.0
  DIRECTION = Y_DIR
  MAGNITUDE = UY4
}
DISPLACEMENT {
  NODE_SET = 5
  SCALE_FACTOR = [-D]
  DIRECTION = Z_DIR
  MAGNITUDE = RAMP
}

# Define the contact interaction pair of facet sets (1-2)
CONTACT_INTERACTION {
  FACET_SET_1 = 1
  FACET_SET_2 = 2
}
```