True*Grid*®Output Manual For NIKE3D

A Guide and a Reference

by

Robert Rainsberger

Version 2.3.0

XYZ Scientific Applications, Inc.

October 12, 2006

Table of Contents

I. NIKE3D Output Guide

Introduction

NIKE3D is a nonlinear, implicit, three-dimensional finite element code for solid and structural mechanics developed at Lawrence Livermore National Laboratories. The focus in this manual will be on those features in **True*Grid*®** that are specific to creating a NIKE3D input file. The **True*Grid*®** User Manual covers the creation of a mesh and will not be covered in this manual. This manual is incomplete in another way because it cannot be used as a substitute for the NIKE3D manual. For a full understanding of the use of these features, the user must have a working knowledge of NIKE3D and be familiarity with the NIKE3D User Manual.

Font Conventions

Different fonts are used through out this manual to indicate their meaning. A literal is highlighted in bold. A symbol to be substituted with a literal or a number is *italicized*. A computer example uses the `Courier font`.

Supported Features

There are many features in **True*Grid*®** to create a model for NIKE3D. The table below shows the commands that are used for each feature. Sometimes there are several commands listed. For example, shells can be generated using both the **block** and **cylinder** commands. The **n** and **th** are used to set the properties of these shells. In another example, the **si** and **sii** commands are used to identify the faces of the mesh that form the sliding (or contact) surfaces. The associated **sid** command is used to assign properties to the sliding surface.

NIKE3D feature	**True*Grid*®** commands
analysis options	**nikeopts**
material	**nikemats**, **mate**, **mt**, **mti**
beam elements	**beam**, **bm**, **ibm**, **ibmi**, **jbm**, **jbmi**, **kbm**, **kmbi**
brick elements	**block**, **cylinder**
shell elements	**block**, **cylinder**
problem title	**title**
load curves	**lcd**, **flcd**
slide surface	**sid**, **si**, **sii**
beam integration rules	**bind**
shell integration rules	**sind**
symmetry planes	**plane**
stone walls	**plane**, **sw**, **swi**

beam cross section properties	**bsd**
springs and dampers	**spd**, **spdp**, **spring**
rebar	**ibm**, **ibmi**, **jbm**, **jbmi**, **kbm**, **kmbi**
orient shell elements	**n**
orient local material coordinate systems	**or**
nodal constraints	**b**, **bi**
nodal shared degrees of freedom	**jd**, **jt**, **mpc**
node print blocks	**npb**
element print blocks	**epb**
nodal forces	**fc**, **fci**, **fcc**, **fcci**, **fcs**, **fcsi**
prescribed velocities	**fv**, **fvi**, **fvc**, **fvci**, **fvs**, **fvsi**, **fvv**, **fvvi**, **fvvc**, **fvvci**, **fvvs**, **fvvsi**
distributed nodal loads	**ndl**, **ndli**
pressure loads on faces	**pr**, **pri**
pressure amplitude	**pramp**, **dom**
shock arrival calculations	**arri**, **dist**
nodal moments	**mom**, **momi**
fixed displacement boundary condition	**fd**, **fdi**, **fdc**, **fdci**, **fds**, **fdsi**
temperature profiles	**tepro**
rigid body motion	**rotation**, **velocity**
initial velocities	**ve**, **vei**
shell element thicknesses	**th**, **thi**
mass points	**pm**, **npm**
rigid node set	**nset**, **rigid**
quadratic brick elements	**quadratic**
linear brick elements	**linear**

You may want to view some of the properties graphically using the **condition** (**co**) command in the merge phase. The **tmm** command can be used to calculate the mass of each part. Be sure to merge the nodes using one of the merging commands such as **stp** and, finally, use:

nike3d for the latest version of NIKE3D
nnike3d for the old NN output format,
enike3d for the old EN output format, or
fnike3d (same as **nike3d**) for the new FN output format

to select NIKE3D as the output option and the **write** command to actually create the input deck for NIKE3D.

The file produced by **True*Grid*®** is an ASCII file that can be examined or modified using any text editor. Some experienced users always inspect the file and modify it instead to rerunning **True*Grid*®** when make easy changes to the model such as changing a material model parameter or the time step.

For this reason, the output file has helpful comments. However, this file can be very large and it might be easier to modify the **True*Grid*®** session file and rerun **True*Grid*®**. One of the comments that is automatically written has a time stamp for archiving purposes. The **title** is also helpful for archiving. You can insert your own comments with the use of the **comment** command.

Sliding (or Contact) Surfaces

To form a contact surface, use the **sid** command to define the surface type. The **sid** command also has optional parameters. This surface is assigned an identification number. Other commands that assign faces or nodes to this surface do so by referencing the surface number.

While in the part phase use the **si** or **sii** commands to select faces of that part for inclusion in the surface definition. If the face is from a shell element, be sure to use the **orpt** orientation command prior to issuing the **si** or **sii** command so that the orientation of the face is towards the opposing face in the sliding surface. If you are using part replication (**lrep**, **grep**, or **pslv**), then you may want to use the **lsii** or the **gsii** to increment the sliding interface command for each replication. You must use the **sid** command for each sliding surface that is referenced when the **lsii** or **gsii** commands are used with replication.

You can use sets in the merge phase to add faces or nodes to a sliding surface. These sets can be formed with the combined use of the **fset** (for faces) and the **nset** (for nodes) commands in the part and merge phase. Only use node sets when defining a sliding surface where nodes are on the slave side and otherwise only use face sets. The node density between the master and slave sides of the interface should be roughly equal. When forming the mesh in the part phase, it may be necessary to build into the mesh a small gap between the master and slave sides of the contact surfaces, depending on the mesh density and the curvature to avoid initial penetration of the slave side into the master side.

When you merge the nodes (in the merge phase), the nodes from the slave side will not be allowed to merge with the nodes on the master side. Use the **mns** command in the merge phase to override this condition. When you fist merge the nodes, a table will be printed to the text window and the tsave file listing the number of faces and nodes associated with each sliding surface. Check this table carefully. You can also see the faces and nodes of either side of the sliding surfaces using the **co** command. When using this in combination with the hide graphics option, you can see the orientation of the faces. Use **labels** command to show how the nodes have merged graphically.

Boundary Conditions

There are several ways to constrain nodes. The **b** and **bi** commands in the part phase or the **b** command in the merge phase will constrain nodes in the global coordinate system. Use the **plane** command to specify symmetry plane constraints including symmetry planes with failure. Nodes in

the model will be assigned to these symmetry planes based on the tolerance you specify in the **plane** command.

Loads

There are numerous ways to assign loads. The list of commands that can be used to assign loads in the part phase includes:

fc	Cartesian concentrated nodal loads
fci	Cartesian concentrated nodal loads
fcc	cylindrical concentrated nodal loads
fcci	cylindrical concentrated nodal loads
fcs	spherical concentrated nodal loads
fcsi	spherical concentrated nodal loads
mom	nodal moment about one of the nodal axis in the global coordinate system
momi	nodal moment about one of the nodal axis in the global coordinate system
ndl	pressure converted to distributed nodal loads
ndli	pressure converted to distributed nodal loads
pr	pressure loads on element faces
pri	pressure loads on element faces
fv	Cartesian prescribed nodal velocities
fvi	Cartesian prescribed nodal velocities
fvc	cylindrical prescribed nodal velocities
fvci	cylindrical prescribed nodal velocities
fvs	spherical prescribed nodal velocities
fvsi	spherical prescribed nodal velocities
fvv	Cartesian variable prescribed nodal velocities
fvvi	Cartesian variable prescribed nodal velocities
fvvc	cylindrical variable prescribed nodal velocities
fvvci	cylindrical variable prescribed nodal velocities
fvvs	spherical variable prescribed nodal velocities
fvvsi	spherical variable prescribed nodal velocities
fd	Cartesian displacement
fdi	Cartesian displacement
fdc	cylindrical displacement
fdci	cylindrical displacement
fds	spherical displacement
fdsi	spherical displacement

The list of commands that can be used to assign loads in the merge phase includes:

fc	Cartesian concentrated nodal loads
mom	nodal moment about one of the nodal axis in the global coordinate system
ndl	pressure converted to distributed nodal loads
pr	pressure loads on element faces
fv	Cartesian prescribed nodal velocities
fvv	Cartesian variable prescribed nodal velocities
fd	Cartesian displacement

Load Curves

Load curves are 2D polygonal curves or tables of amplitude vs. time data that can be created using the **lcd** and **flcd** commands. They are used to specify the magnitude and/or time variation such as boundary conditions and material properties. It is best to define a load curve before it is referenced in a load or material model to avoid a warning message. When the output file is written, if a load curve is referenced but not defined, you will also receive a warning message. Then a simple load curve will be used in the output file so that a valid NIKE3D file is written. It is advised that you correct this by defining the appropriate load curve for the problem. Do not rely on the load curve that is automatically generated.

In some dialogue boxes you might be prompted for a load curve or a set id. Simply ignore the set id portion of the prompt and supply the load curve number.

Stone Walls

A stone wall is defined with two commands. Use the **plane** command to set the properties of the stone wall. Nodes to react to the stone wall will not be selected automatically based on the tolerance. Use the **sw** and **swi** commands in the part phase to assign faces of the model to react to the stone wall. You can use the **sw** command in the merge phase as well to assign faces from a face set to react to the stone wall.

Bricks

Brick elements refer to hexahedral, prism (wedge), and tetrahedral elements. The brick element is the default element type when defining a material model using the **nikemats**. The **mate**, **mt**, and **mti** commands are used to associate a material with the elements. It is required that the type of element (brick, shell, or beam) agree with the material element type. Only one element type can associated with a material definition. If you want two different element types with the same material properties, you must define two materials. No section properties are needed for bricks.

The element local coordinate system used in an orthotropic or anisotropic material is imposed by the order of the nodes that define the element. You can flip the nodal ordering to switch the orientation of this local coordinate system using the **or** command in the part phase.

The default order of the brick elements is linear. There is an experimental version of NIKE3D that uses quadratic bricks with 20 nodes. To generate these, use the **quadratic** command before creating a part. Be sure to return to linear order with the **linear** command before generating any shells because NIKE3D does not support quadratic shells.

Shells

Shell elements refer to both quadrilateral and triangular elements and sometimes referred to as structural elements. You must specify the shell element type when defining a material with the **nikemats** command. The **mate**, **mt**, and **mti** commands are used to associate a material with the elements. It is required that the type of element (brick, shell, or beam) agree with the material element type. Only one element type can associated with a material definition. If you want two different element types with the same material properties, you must define two materials. Cross sectional properties, such as the default thickness, can be included in the material model when the shell type is selected. These default thicknesses can be overridden with the use of the **thic** command in the part phase. Both can be overridden for a region of the part using the **th** and **thi** commands. If you have two surfaces that represent the inner and outer surfaces of a structure that is to be modeled using shell elements, than you can use the **ssf** and **ssfi** commands in the part phase to create shells with variable thickness.

The orientation of the positive normal direction to the shell is dictated by the nodal ordering of the nodes that define the shell. This positive direction is used, for example, to determine the direction of a positive pressure. This direction can be flipped using the **n** command in the part phase. The order of the nodes also dictates the local material coordinate system which can be important when using an orthotropic or anisotropic material. Use the **or** command to flip the coordinate system to the desired direction. When an angle is specified for the orientation of a composite material, it is with respect to this orientation.

You may need to specify the through thickness integration points when defining a composite material. This can be done by defining an integration rule with the **sind** command. Then identify this rule in the material definition.

Beams

Three nodes are required to form a beam element. The third node is needed to define the local coordinate system used to form the cross sectional properties. These element are sometimes referred to as structural elements. Use the **ibm**, **ibmi**, **jbm**, **jbmi**, **kbm**, and **kbmi** commands to form beam elements with shell or brick structures while in the part phase. If the material of the shell or brick structure is set to zero using the **mt**, **mti**, or **mate** command, then the shells or bricks will be ignored, but the embedded beams will not be ignored. This is a convenient way to build an array of beams using block structured methods. You can also use the **bm** command in the merge phase to build a

string of beams that can be made to follow a 3D curve. The **beam** command (this command has been denigrated) can also be used to form beam elements, but the command is not interactive.

You must specify the beam element type when defining a material with the **nikemats** command. The beams elements are assigned a material number when they are formed or in the merge phase using the **mt** command. It is required that the type of element (brick, shell, or beam) agree with the material element type. Only one element type can associated with a material definition. The beam default cross section properties, such as thickness, are also defined within the material definition. You can also use the **bsd** command to define cross sectional properties to override the material default cross sectional properties. When you create a beam, refer to the **bsd** number to assign these cross sectional properties to the beam. Use the **bind** command to define a beam integration rule, if needed. Then refer to this integration rule when defining the material.

Springs and Dampers and Point Masses

Springs and dampers are treated the same in **True*Grid*®**. They are only distinguishable by the material properties assigned them. Use the **spd** command to define the properties of the spring or damper. Then use the **spring** command to assign nodes to a numbered spring. Alternatively, the **spdp** command can be used in the part phase to create an array of springs between two parts, analogous to a contact surface.

Point masses can be generated in the part or merge phase. There are two types of point masses. The **pm** command will assign a mass to an existing node. The **npm** will create a new node and assign it a mass. The latter must then be connected either to a spring or beam.

Shared Constraints

Use the **mpc** command to couple a set of nodes. This requires that you create a node set first. The **nset** or **nseti** command can be used in the part phase and the **nset** command in the merge phase to create a node set. Also, click on the pick button in the environment window during the merge phase. Then you can use the mouse to modify or create a node set. The nodes sharing a set of constraints will not be merged together. You can also use the **jd** with the **jt** command to form multiple point constraints for a small set of nodes.

The **rigid** command in the merge phase will form rigid node sets. Use the same methods for generating a node set as described above.

Post Processing

There are a number of options of the **nikeopts** command to control the data saved in the database by NIKE3D for post processing. You may wish to analyze in greater detail the evolution of certain

nodes or elements. Use the **npb** and **epb** commands (referred to as time history blocks), respectively, to identify areas of the mesh requiring a more detailed accumulation of data by NIKE3D.

II. Examples

III. NIKE3D Output Reference

The commands found here are provided to the user so that a complete input file can be generated by **True*Grid*®**. This manual does not try to explain the meaning of these parameters. For this, the user is referred to the NIKE3D User Manual.

Command Syntax Conventions

When an arbitrarily long list of arguments are required, a semi-colon terminates the list. Sometimes the abbreviation *#_things* is used to mean "number of things". Each command is described by an entry like the following:

command summary description

command *arguments* brief description of functionality
with brief descriptions of what the *arguments* should be.
indentation is used to indicate a list of options to the *arguments*

Some commands in the part phase require a region specification. The region selects a face of the mesh, among other things. Others may require a progression specification. The progression selects multiple faces, among other things. In the merge phase, such commands require an option. In all of these cases, a portion of the mesh is identified. For example, the **si**/**sii** command has this property.

Remarks

When present, the Remarks section describes the command in even greater detail. It may describe the context in which the command is normally used, and other commands used in association with this command. It may describe side effects. It may describe other, similar commands. In many cases, it includes a description of where to find the command in the menus.

Examples

When present, this shows the exact use of the command. If you use the dialogues, this command will be generated by simple selection options with the mouse and entering data where indicated. The command, as shown here, will appear in the session file for later reuse and possible modification. You can also enter the command into the text window or insert it into a command file to be run in batch mode.

bsd global beam cross section definition

bsd *option_list ;*

where an *option* can be:

sthi *thickness*	s-thickness at both ends
tthi *thickness*	t-thickness at both ends
sthi1 *thickness*	s-thickness at beginning
sthi2 *thickness*	s-thickness at ending
tthi1 *thickness*	t-thickness at beginning
tthi2 *thickness*	t-thickness at ending

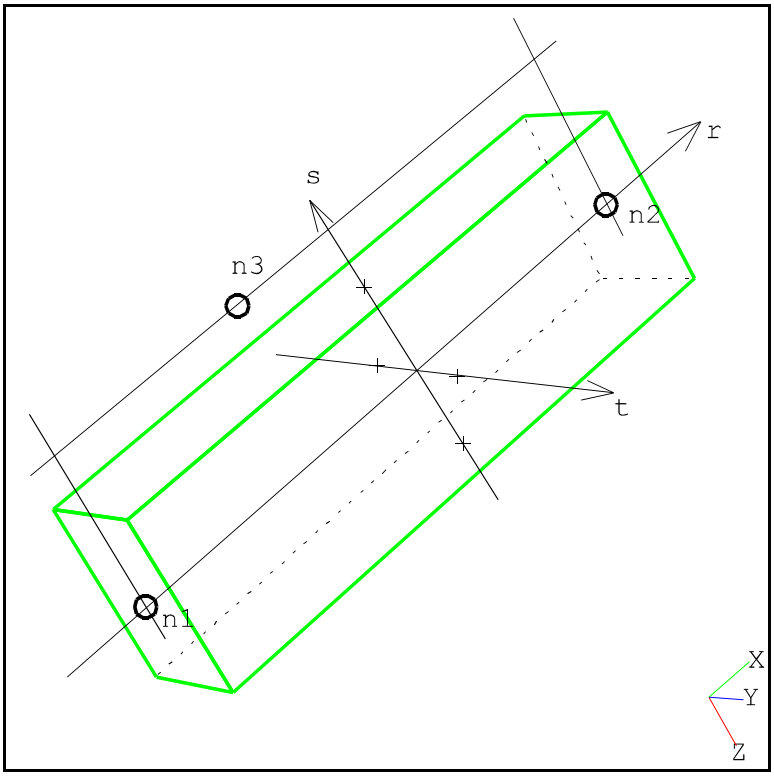

Figure 1 Beam Local Coordinate System for NIKE3D

Remarks

There are other options to this command, but they are not shown here because they do not apply to the NIKE3D output. For a complete description of the **bsd** command, see the **True*Grid*®** User Manual.

Since there are default cross section properties provided in the definition of the material using the **nikemats** command, not all of the parameters need to be assigned through the **bsd** command.

Each cross section definition is assigned a number by you so that you can reference it when defining a set of beams with the **ibm**, **ibmi**, **jbm**, **jbmi**, **kbm**, **kbmi**, **bm**, and **beam** commands.

Example

```
bsd 2 sthi .03 tthi .03 ; ;
```

sid sliding interface definition

sid *slide_# type options* ;

where *type* can be one of

tied	tied sliding surface
sl	sliding only
sv	sliding with voids
single	single sided slide surface
dummy	used to insure that nodes in this interface will not be merged

where an *option* can be

fric *factor*	static coefficient of friction
kfric *friction*	kinetic coefficient of friction
decay *decay*	exponential decay coefficient
bwmrad *#_facets*	bandwidth minimization radius
pnlt *factor*	sliding penalty
pnltm *factor*	penalty factor
penmax *distance*	small penetration search distance
iaug *flag*	augmentation flag
altoln *tolerance*	normal direction convergence tolerance
altolt *tolerance*	tangential direction convergence tolerance
tkmult *multiplier*	tangent stiffness multiplier
dtime *time*	interface death time
bury *time*	interface burial time

Remarks

Sliding interfaces or contact surfaces are constructed in 3 steps. These steps can be done in any order.

1. define the properties
2. select the slave side
3. select the master side, if applicable

The **sid** command is used to define the properties. The **si** and **sii** commands are used in the part phase or the merge phase to select the nodes or faces that form the master and slave sides of the interface.

When nodes are merged, nodes across a sliding interface will not be merged. When a merge command is first issued in the merge phase, a table is written listing the number of nodes and faces associated with each sliding interface.

The **dummy** type interface is actually used to avoid merging of nodes. A sliding interface of this type is not written to the output file.

The nodes and faces of a sliding interface or contact surface can be viewed in the merge phase using the **si** option of the **co** command.

If the output option has been selected prior to using the dialogue box to make a selection, only the options available to that output option will be displayed in the dialogue box.

si select nodes or faces for sliding interface in merge phase

si *option slide_# type*

where *option* can be one of:

n *node_numbe*r	select a single node
rt *x y z*	select a node close to a Cartesian point
cy *rho theta z*	select a node close to a cylindrical point
sp *rho theta phi*	select a node close to a spherical point
nset *name_of_set*	select an entire node set
fset *face_set*	select a face set

where *type* can be one of

m	master side of the interface
s	slave side of the interface

si assign sliding interface to region in part phase

si *region slide_# type*

where *type* can be one of

m	master side of the interface
s	slave side of the interface

Remarks

Care is needed to be sure that the outward normal of the facet off each side of the interface is pointing towards the opposite side. Use the **orpt** prior to issuing this commands for this purpose.

sii assign sliding interfaces to progression in part phase

sii *progression slide_# type*
where *type* can be one of

m	master side of the interface
s	slave side of the interface

Examples

A model was created by the use of the following command file. Some normals are displayed as circular arcs with arrows. This is caused by the normals pointing almost orthogonally from the screen. There is an angle parameter in the **co** command setting the range of such behavior. You can modify it, or totally disable it.

```
c Sliding Interface -> master side
c Part definition -> shells.
block 1 3 5 7 9;-1;1 3 5 7 9; 0 2 4 6 8; 9 ; 0 2 4 6 8;
c Definition of orientation point in Cartesian coord x,y, and z
orpt - 9 0 9
c Definition of the type of the sliding interface
sid 1 sv ;
c Assignment of region (;-1;;) and type(1 m)of slid. interf.
sii ;-1;; 1 m
c Definition of 3 global transformations around y-axis
gct 3 ry 90; ry 180; ry 270 ;  c for 90,180 and 270 degrees.

c Global replication 3 times by rotation for
grep 0 1 2 3;  90,180 and 270 degrees
c Cylinder part -> tube made from hexahedrons.
cylinder 1 6; 1 3 5 7 9 11 13 15 17 19 21 23 25; 1 10;
     2 4;0 30 60 90 120 150 180 210 240 270 300 330 360;0 20;
c Definition of the orientation point in the default coordinate
c system of the part (in Cylindrical coordinates r,eta,z).
orpt - 0 20 5
c Assignment of region (-2;;) and type(1 s) of slid. interf.
sii -2;;; 1 s
c Assignment of region (;;-1;) and type(1 s) of slid. interf.
sii ;;-1; 1 s
c Assignment of region (;;-1;) and type(1 s) of slid. interf.
sii ;;-2; 1 s
lct 1 my 20 ;  c Definition of the local transformation
lrep 1;        c Transformation 1 is applied.
merge
labels size 3       c Scale the size of arrows.
```

```
rx 20 ry 20 rz 20   c Rotate mesh in window.
center              c Center picture in window.
set tv disp         c Set hide display option.
co si 1 m;     c Display of master side of sliding interface 1
co si 1 s;     c Display of slave side of sliding interface 1
```

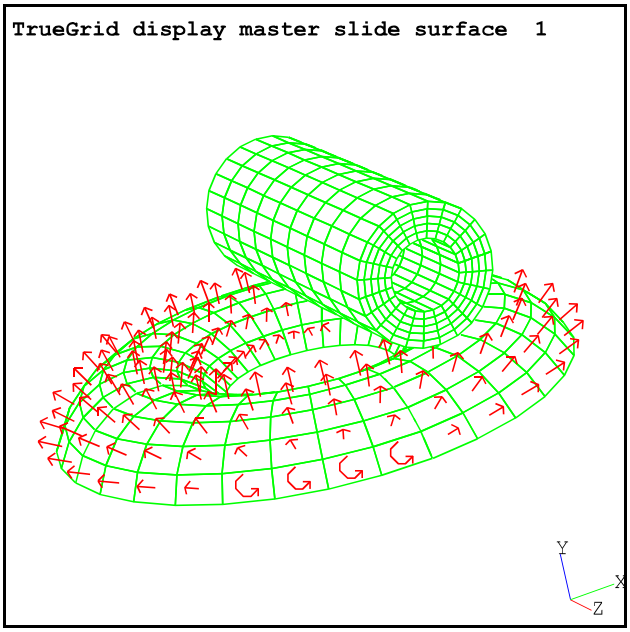

Figure 2 master side of interface

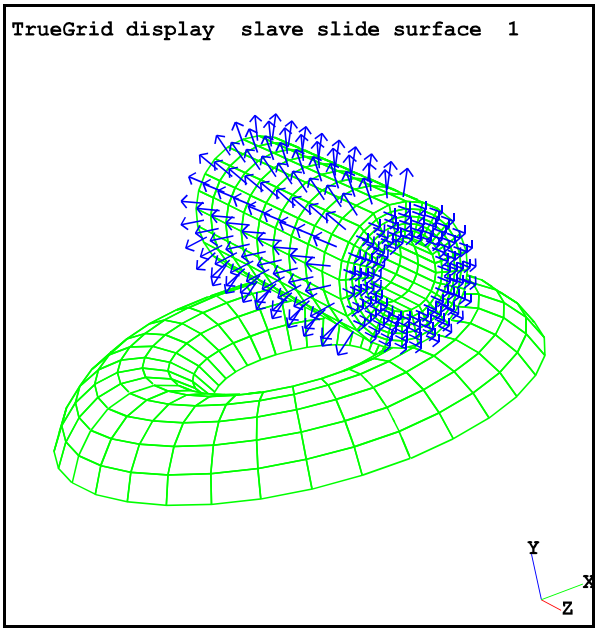

Figure 3 slave side of interface

Remarks

Care is needed to be sure that the outward normal of the facet off each side of the interface is pointing towards the opposite side. Use the **orpt** prior to issuing this commands for this purpose.

spd define the properties of a set of springs or dampers

spd *spring/damper_# type*
where *type* is the spring or damper's material model:

le *stiffness*	linear elastic
lv *damping*	linear viscous
iep *elastic tangent yield*	isotropic elastic
ne *ld_curve_#*	nonlinear elastic
nv *ld_curve_#*	nonlinear viscous

nesf *ld_curve_# force_curve_#* nonlinear elastic w/ force load curve
gn *loading_# unloading_# hardening tension compression* general nonlinear

Remarks

A spring or damper is defined using either the **spdp** command forming a set of springs/dampers between two surfaces, or using the **spring** command to create a single spring at a time. In each case, the definition of a spring includes a reference to a material definition **spd** number.

If the output option has been selected prior to using the dialogue box to make a selection, only the options available to that output option will be displayed in the dialogue box.

nikemats NIKE3D materials

nikemats *material_# material_type options parameter_list* ;
where the following *options* are available for all materials:

shell *features* shell element type
where a *feature* can be
shear *factor*
tsti *#_points*
propt *option*
where option can be
1 for element center
2 for plan integration points
3 for through thickness and plan integration points
quad *integration_rule_#*
where the *integration_rule_#* can be
n positive for the number of points using the trapezoidal rule
0 Gauss
-n negative of the user specified rule number (**sind**)
shth *thickness*
shth1 *thickness*
shth2 *thickness*
shth3 *thickness*
shth4 *thickness*
shloc *location*
where *location* can be
1 for top surface
0 for middle surface
-1 for bottom surface

beam *features* beam element type

where a feature can be

shear *factor*

quad *option*

where the *option* can be

1	a truss
2	2x2 Gauss quadrature
3	3x3 Gauss quadrature
4	3x3 Lobatto integration
5	4x4 Gauss quadrature

bmcross *shape*

where the shape can be

0	rectangular
1	tubular

sthi *thickness*

tthi *thickness*

sthi1 *thickness*

sthi2 *thickness*

tthi1 *thickness*

tthi2 *thickness*

sloc *location*

where *location* can be

1	meaning the side where s is 1
0	meaning centered
-1	meaning the side where s is -1

tloc *location*

where *location* can be

1	meaning the side where t is 1
0	meaning centered
-1	meaning the side where t is -1

rho *density*

rda *constant*

rdb *constant*

where the *parameter_list* is specific to the selected material type and are listed accordingly below:

Elastic - Material type 1

e *modulus*	Young's modulus
pr *ratio*	Poisson's ratio

Orthotropic Elastic - Material type 2

ea *ea*	constitutive matrix coefficient
eb *eb*	constitutive matrix coefficient

ec *ec* constitutive matrix coefficient
prba *vba* constitutive matrix coefficient
prca *vca* constitutive matrix coefficient
prcb *vcb* constitutive matrix coefficient
gab *gab* constitutive matrix coefficient
gbc *gbc* constitutive matrix coefficient
gca *gca* constitutive matrix coefficient
aopt *option parameters* for material orientation

where the *option* can be one of

0 by nodes
1 by point and element center
2 by normal vectors

where the *parameters* can be

xp *x-coordinate* used with **aopt 1**
yp *y-coordinate* used with **aopt 1**
zp *z-coordinate* used with **aopt 1**
ax *x-component* used with **aopt 2**
ay *y-component* used with **aopt 2**
az *z-component* used with **aopt 2**
dx *x-component* used with **aopt 2**
dy *y-component* used with **aopt 2**
dz *z-component* used with **aopt 2**

Elastic-Plastic - Material type 3

e *modulus* Young's modulus
pr *ratio* Poisson's ratio
sigy *stress* yield stress
etan *modulus* hardening modulus
beta *parameter* hardening parameter
es *strain_list* ; effective plastic strain
eps *stress_list* ; effective plastic stress

Thermo-Elastic-Plastic - Material type 4

temp *temperature_list* ; list of temperature
e *modulus_list* ; list of Young's modulus
pr *ratio_list* ; list of Poisson's ratio
alpha *secant_list* ; list of secant coefficient of thermal expansion
sigy *stress_list* ; list of yield stress
etan *modulus_list* ; list of plastic hardening modulus

Soil And Crushable Foam - Material type 5

g *modulus* shear modulus

ku *modulus* bulk unloading modulus
a0 *yield* yield function
a1 *yield* yield function
a2 *yield* yield function
pc *pressure* minimum pressure
ul *option* unloading
where *option* can be
0 volumetric crushing
1 no volumetric crushing
vs *strain_list* ; volumetric strain table
ps *pressure_list* ; pressure table

ViscoElastic - Material type 6
k *modulus* bulk modulus
g0 *modulus* short time shear modulus
gi *modulus* long time shear modulus
beta *decay* decay constant

Thermo-Orthotropic - Material type 7
ea *modulus* young's modulus in a-direction
eb *modulus* Young's modulus in b-direction
ec *modulus* Young's modulus in c-direction
prba *ratio* Poisson's ratio in ab-direction
prca *ratio* Poisson's ratio in ac-direction
prcb *ratio* Poisson's ratio in bc-direction
alpa *expansion* thermal expansion coefficient in a-direction
alpb *expansion* thermal expansion coefficient in b-direction
alpc *expansion* thermal expansion coefficient in c-direction
gab *modulus* shear modulus in ab-direction
gbc *modulus* shear modulus in bc-direction
gca *modulus* shear modulus in ca-direction
aopt *option parameters* material orientation
where the *option* can be one of
0 by nodes
1 by point and element center
2 by normal vectors
where the *parameters* can be
xp *x-coordinate* used with **aopt 1**
yp *y-coordinate* used with **aopt 1**
zp *z-coordinate* used with **aopt 1**
ax *x-component* used with **aopt 2**
ay *y-component* used with **aopt 2**

az *z-component*	used with **aopt 2**
dx *x-component*	used with **aopt 2**
dy *y-component*	used with **aopt 2**
dz *z-component*	used with **aopt 2**

Thermo-Elastic-Creep - Material type 8

temp *temperature_list* ;	list of temperatures
g *modulus_list* ;	list of shear modulus
k *modulus_list* ;	list of bulk modulus
alpha *secant_list* ;	list of secant coefficients of thermal expansion
a *creep_list* ;	list of first creep parameters
b *creep_list* ;	list of second creep parameters

Power Law Plasticity - Material type 9

e *modulus*	Young's modulus
pr *ratio*	Poisson's ratio
k *strength*	strength coefficient
n *hardening*	hardening exponent

Power Law Thermo-Elastic-Plastic - Material type 10

temp *temperature*	temperature
e *modulus*	Young's modulus
pr *ratio*	Poisson's ratio
alpha *expansion*	tangent coefficient of thermal expansion
k *strength*	strength coefficient
n *exponent*	hardening exponent

Transient Thermal Creep - Material type 11

pr *ratio*	Poisson's ratio
n *exponent*	stress exponent
temp *temperature*	temperature
e *modulus*	Young's modulus
a *stress*	stress coefficient
m *exponent*	time exponent
alpha *coefficient*	secant coefficient of thermal expansion

Ramberg-Osgood Elastoplastic - Material type 12

gammay *strain*	reference shear strain
tauy *stress*	reference shear stress
alpha *coefficient*	stress coefficient
r *exponent*	stress exponent
k *modulus*	bulk modulus

General Anisotropic Thermal-Elastic - Material type 13

matrix *c11 c12 c13 c14 c15 c16* compliance matrix
c22 c23 c24 c25 c26
c33 c34 c35 c36
c44 c45 c46
c55 c56
c66

alpha1 *expansion* thermal expansion coefficient in a-direction
alpha2 *expansion* thermal expansion coefficient in b-direction
alpha3 *expansion* thermal expansion coefficient in c-direction
aopt *option parameters* material orientation

where the *option* can be one of

0 by nodes
1 by point and element center
2 by normal vectors

where the *parameters* can be

xp *x-coordinate* used with **aopt 1**
yp *y-coordinate* used with **aopt 1**
zp *z-coordinate* used with **aopt 1**
ax *x-component* used with **aopt 2**
ay *y-component* used with **aopt 2**
az *z-component* used with **aopt 2**
dx *x-component* used with **aopt 2**
dy *y-component* used with **aopt 2**
dz *z-component* used with **aopt 2**

Oriented Brittle Damage - Material type 14

e *modulus* Young's modulus
pr *ratio* Poisson's ratio
ft *strength* tensile strength
fs *strength* cracked shear strength
sigy *strength* compressive yield strength
gc *toughness* fracture toughness
beta *factor* shear retention factor
eta *viscosity* viscosity

Mooney-Rivlin Rubber - Material type 15

ai *term* coefficient of first invariant term, a
bi *term* coefficient of second invariant term, b
pr *ratio* Poisson's ratio
aflg *option* augmented Lagrangian flag

where *option* can be

0 off
1 *tolerance* on with convergence tolerance

Thermo-Plastic Melt - Material type 16
lcyt *load_curve* Young's modulus load curve
lypt *load_curve* Poisson's ratio load curve
lyet *load_curve* thermal expansion load curve
fsm *option* flow strength model
where an *option* can be
0
1 *feature*
where a *feature* can be
bulk *modulus* bulk modulus
lcyst *load_curve* yield stress load curve
lcptt *load_curve* plastic tangent load curve
2 *feature*
where a *feature* can be
bulk *modulus* bulk modulus
lcist *load_curve* initial strength load curve
lcfst *load_curve* flow strength load curve
lcsrt *load_curve* strain rate load curve
lcscbt *load_curve* strengthening coef load curve
lcsent *load_curve* strengthening exp load curve
3 *feature*
where a *feature* can be
bulk *modulus* bulk modulus
lcist *load_curve* initial strength load curve
lcfst *load_curve* flow strength load curve
lcsrt *load_curve* strain rate load curve
lcsck1t *load_curve* strengthening coef load curve
lcsen1t *load_curve* strengthening exp load curve
lcrck2t *load_curve* recovery coef load curve
lcfren2t *load_curve* first recovery coef load curve
lcsren3t *load_curve* second recovery coef load curve
ffm *option* fluid flow model
where an *option* can be
0
1 *feature*
where a *feature* can be
lcfvt *load_curve* fluid viscosity load curve
vgm *option* void growth model
where an *option* can be

0
1 *features*
where a *feature* can be
ivs *strain* initial void strain
lctpt *load_curve* tensile pressure load curve
lccpt *load_curve* compressive pressure load curve
cemf *flag*
where *flag* can be
0 off
1 on
2 *features*
where a *feature* can be
ivs *strain* initial void strain
g1 *parameter* first gurson parameter
g2 *parameter* second gurson parameter
cemf *switch*
where *switch* can be
0 off
1 on

lsm *option* liquid-solid transition model
where an *option* can be
0
1 *feature*
where a *feature* can be
lcfsft *load_curve* fraction solid load curve

vardb *option* variable written to plot database
where *option* can be
0 effective plastic strain
1 void strain
2 flow strength
3 effective strain rate

Foundation Boundary Spring - Material type 17

matrix *k11 k12 k13 k14 k15 k16* full upper-triangular stiffness matrix
k22 k23 k24 k25 k26
k33 k34 k35 k36
k44 k45 k46
k55 k56
k66

Transversely Isotropic Hyperelasticity - Material type 18

c1 *constant* Mooney-rivlin coefficient

c2 *constant* Mooney-rivlin coefficient
c3 *stress* exponential stress coefficient
c4 *uncrimping* fiber uncrimping coefficient,
c5 *modulus* modulus of straightened fibers
k *modulus* bulk modulus
lambda *stretch* fiber stretch for straightened fibers
isf *option* initial stretch flag
where the *option* can be
0 off
1 on
lcis *load_curve* initial stretch load curve
aflg *option* augmented Lagrangian flag
where the *option* can be
0 off
1 on
altol *tolerance* tolerance for augmented Lagrangian iterations, only for aflg=1

Strain Rate Sensitive Power Law Plasticity - Material type 19
e *modulus* Young's modulus
pr *ratio* Poisson's ratio
sck *strength* strength coefficient
hen *exponent* hardening exponent
srsem *exponent* strain rate sensitivity exponent
isr *rate* initial strain rate

Rigid Body - Material type 20
e *modulus* Young's modulus
pr *ratio* Poisson's ratio
xtrans *x* x-translation. boundary condition code
ytrans *y* y-translation boundary condition code
ztrans *z* z-translation boundary condition code
xrot *x* x-rotation boundary condition code
yrot *y* y-rotation boundary condition code
zrot *z* z-rotation boundary condition code
comflg *option*
where option can be
0
1 *x y z* x,y,z coordinates of center of mass

Thermo-Orthotropic Elastic Laminate for material 23
ea *ea_list* ; list of orthotropic constants
eb *eb_list* ; list of orthotropic constants

ec *ec_list* ; list of orthotropic constants
vba *vba_list* ; list of orthotropic constants
vca *vca_list* ; list of orthotropic constants
vcb *vcb_list* ; list of orthotropic constants
aa *aa_list* ; list of orthotropic constants
ab *ab_list* ; list of orthotropic constants
ac *ac_list* ; list of orthotropic constants
gab *gab_list* ; list of orthotropic constants
gbc *gbc_list* ; list of orthotropic constants
gca *gca_list* ; list of orthotropic constants
t *temperature_list* ; list of temperatures
angles *angle_list* ; list of material angles at integration points
aopt *option parameters* for material orientation

where the *option* can be one of

0	for by nodes
1	for by point and element center
2	for by normal vectors

where the *parameters* can be

xp *x-coordinate*	for **aopt 1**
yp *y-coordinate*	for **aopt 1**
zp *z-coordinate*	for **aopt 1**
ax *x-component*	for **aopt 2**
ay *y-component*	for **aopt 2**
az *z-component*	for **aopt 2**
dx *x-component*	for **aopt 2**
dy *y-component*	for **aopt 2**
dz *z-component*	for **aopt 2**

Elastic-Plastic with Forming Limit Diagram for material 35

e *young's_modulus*	Young's Modulus
pr *poisson's_ratio*	Poisson's Ratio
sg0 *yield_stress*	yield Stress
lcxe *load_curve*	tangent modulus load curve
lclh *load_curve*	left side load curve
lcrh *load_curve*	right side load curve
lcrx *load_curve*	pressure load curve
lcedf *load_curve*	fld rate load curve
lcedm *load_curve*	yield stress load curve
eptr *strain*	transient strain
epf *strain*	effective strain
scldev *factor*	failure scale factor

Remarks

You must specify the element type when defining a material with the **nikemats** command. The **mate**, **mt**, and **mti** commands are used to associate a material with the elements. It is required that the type of element (brick, shell, or beam) agree with the material element type. Only one element type can associated with a material definition. If you want two different element types with the same material properties, you must define two materials.

Examples

```
nikemats 2 4
          temp 10 250 400 890;
          e 17e6 18e6 19e6 20e6;
          pr .31 .315 .317 .32;
          alpha 4.6e-6 5.3e-6 7.9e-6 10.4e-6;
          etan 8.73e5 8.74e5 8.91e5 9.03e5;
          sigy 4.9e4 2.6e4 2.1e4 1.1e4;;

nikemats 5 1
          shell
          e 2.e11
          pr .3
          shloc 0
          tsti 2 shth 0.025 ;

nikemats 1 1
          mhead elastic material
          beam shear 1.15
          quad 1
          bmcross 1
          rho 1.12e-4
          sthi .015 tthi .021
          e 1.71e7
          pr .311
          mrt 20 ;

nikemats 20 20
     rho 4.46e-03
     p31 114.0e+03
     p41 0.340e+00
     p51 -1.0
     p52 -1.0
     p53 -1.0
     p54 -1.0
     p55 -1.0
     p56 -1.0 ;
```

```
nikemats 7 1              c Titanium alloy 6Al4V
     rho 4.46e-03         c 0.00446 g/mm3
     e 114.0e+03          c 114 GPa
     pr 0.34a ;
```

nikeopts NIKE3D analysis options

nikeopts *options*

where an *option* can be

accflg *option*	select the acceleration data dump option
where the *option* can be	
0	no acceleration in the plot file
1	include relative acceleration data
2	include absolute acceleration data
altol *tolerance*	set the convergence tolerance on augmented Lagrangian
anal *type*	type of analysis
where *type* can be	
stat	static
dyn,	dynamic analysis
dyns	dynamic analysis with stresses initialized statically
arcl *arc_length*	arc length damping
arclcm *method*	arc length constraint method
where *method* can be	
crisfield	
ramm	
arcldamp	arc length damping
auto	automatic time step control invoked
bef *flag*	select the beam element formulation
where *flag* can be	
1	Hughes-Liu out of core
11	Hughes-Liu in core
begs 2	set the beam element geometric stiffness flag
bfgscore	BFGS update vectors storage option
bfor *formulation*	brick element formulation
where *formulation* can be	
0	B-bar
1	B-bar with incompatible modes
10	B-bar in core
11	B-bar with incompatible modes in core
brstif	brick element geometric stiffness included

bwmo *toggle* bandwidth minimization
where *toggle* can be
on
off
cost alternative automatic step control
dctol *tolerance* displacement convergence tolerance
delt *time* time step
dispnode *node_#* node number for displacement controlled arc length method
dispdir *direction* direction of displacement at node for arc length control
where *direction* can be
1 global x-direction
2 global y-direction
3 global z-direction
ectol *tolerance* energy convergence tolerance
fixed fixed time step size
ictol *tolerance* iteration convergence tolerance
igapfg *flag* interface gap plot file
where *flag* can be
0 do not overwrite y-velocity data
1 overwrite y-velocity data with nodal gap data
ilsbuf *buffer_size* out of core linear solver buffer size
iobuf *buffer_size* buffer size (words) for element data I/O
iplt *dump_interval* TUARUS dump interval
iprt *dump_interval* print dump interval
itewin *size* size of the iteration window
islvdt 2 iterative solver data storage on disk
lsolver *method* linear equation solver
where *method* can be
fissle direct solution with FISSLE
scaling iterative solution with diagonal scaling
crout iterative solution with Crout Element-By-Element
gs iterative solution with Gauss-Seidel EBE
cholesky iterative solution with Cholesky EBE
lstol *tolerance* line search convergence tolerance
maxaug #*_iterations* maximum augmented Lagrangian iterations
mem *percent* maximum memory
mnss *min_step* minimum allowable step size
msrf *max_#_step* maximum number of reform/time steps
munload *method* arc length unloading method which can be
where *method* can be
bfgs BFGS
broy Broyden

dfp Davidon-Fletcher-Powell
dav Davidon
mnewt modified Newton

mxitls *max_#_iterations* iteration limit for linear solver
mxnre *max_#_retries* maximum number of retries allowable per step
mxss *max_step_size* maximum allowable step size
nbei *#_steps* number of steps between equilibrium iterations
nbsr *#_steps* number of steps between matrix reformations
neig *#_eigenvectors* number of eigenvectors
nibsr *max_#_iterations* maximum number of equilibrium iter./matrix reform
nip1 *coefficient* first Newmark integration parameter
nip2 *coefficient* second Newmark integration parameter
ngrav *x_acceleration load_curve* gravity
y_acceleration load_curve
z_acceleration load_curve
noarclda no arc length damping
nrest *#_steps* number of time steps between restart file generation
nsbrr *#_steps* number of time steps between running restarts
nsmd *method* nonlinear solution method which can be
where *method* can be
bfgs, BFGS (default)
broy Broyden
dfp Davidon-Fletcher-Powell
mdav modified Davidon
mnewt modified Newton
marc modified constant arc length
mcls modified constant arc length with line search
mabfgs, modified constant arc length with BFGS
mabroy modified constant arc length with Broyden
madfp modified constant arc length with DFP
mambfgs modified constant arc length with modified BFGS
madav modified constant arc length with Davidon
newt full Newton
newtls full Newton with line search
nsteps *#_steps* number of time steps
nunload *#_steps* number of unloading steps in modified arc length method
opnit *#_iterations* optimal number of iterations per step
prlis *flag* linear iterative solver print-out flag with values
where *flag* is
0 no inner loop information
1 time step convergence information
2 iteration norm information

3	residual input, solution output
4	residual/solution each iteration

rail alternative automatic step control

retol *tolerance* set the convergence tolerance on residual norm

segs *flag*

where *flag* can be

0	neglect
1	include

sfor *formulation*

where *formulation* can be

1	Hughes-Liu
2	Yase
3	Belytschko-Tsay
4	Hughes-Liu with membrane thinning
5	hyperelastic Hughes-Liu
11	Hughes-Liu in core
12	Yase in corre
13	Belytschko-Tsay in core
14	Hughes-Liu with membrane thinning in core
15	hyperelastic Hughes-Liu in core

shift *freqency* frequency shift

ssdm shell surface strain data dumps

stifcore stiffness matrix storage option

sw3 toggle the default sense switch number 3

sw6 toggle the default sense switch number 6

sw7 toggle the default sense switch number 7

teo *value* thermal effects option

xvel *load* load due to x-angular velocity

yvel *load* load due to y-angular velocity

zvel *load* load due to z-angular velocity

Remarks

Examples

```
nikeopts
     auto
     nsteps 20
     delt 0.05
     mxss 0.05
     iprt 999
     iplt 1
     msrf 15
```

```
     dctol 0.0015
     ectol 0.0015
     anal stat
     nsmd bfgs
     bfgscore
     bfor 10
     bwmo on
     sw3
     sw6
     nbsr 1
     nbei 1
     nibsr 10
     stifcore 1
     maxmem 0
     lsolver fissle
     nrest 999
     nsbrr 0;

nikeopts
     nsteps 10
     delt .2
     teo 1
     mxnre 25
     opnit 11
     mnss .001
     mxss .2
     iprt 999
     sw6
     bwmo on
     anal stat;

nikeopts
     nsteps 1
     delt 1
     bwmo on
     anal stat
     maxmem 100.
     stifcore 1
     bfgscore sfor 0
     segs 1 ;
```

IV. Frequently Asked Questions

1. How do I create an output deck for NIKE3D?

Go to the merge phase, issue a merge command, such as stp, issue the nike3d command followed by the write command.

V. INDEX