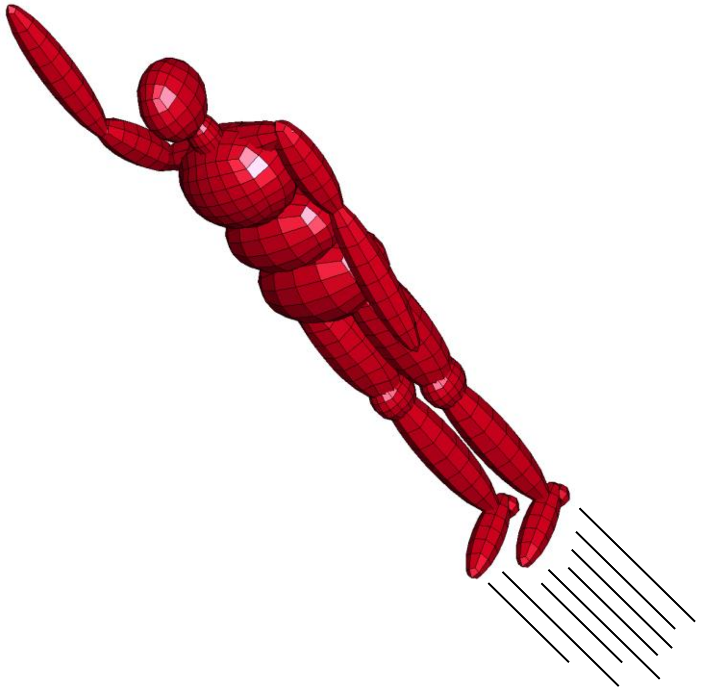
WEBinar: Rigid Bodies in LS-DYNA



Marcus Gustavsson, marcus.Gustavsson@dynamore.se
DYNAmore Nordic AB

Document information

Doc. no.: 1

Revision: 1

Prepared for: WEBinar on Rigid Bodies in LS-DYNA hosted by DYNAmore Nordic AB

Project no.:

Approved by:

Release date: 2019-11-19

Distribution: Approved for public release

DYNAmore Nordic AB
Brigadgatan 5
SE-587 58 LINKÖPING
Sweden

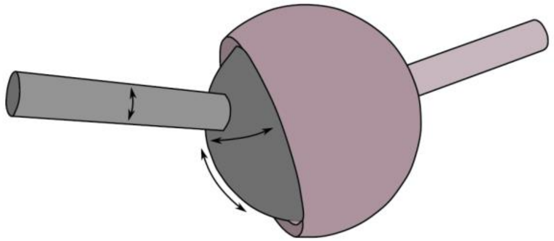
Org. no. 556819-8997
EC VAT: SE556819899701

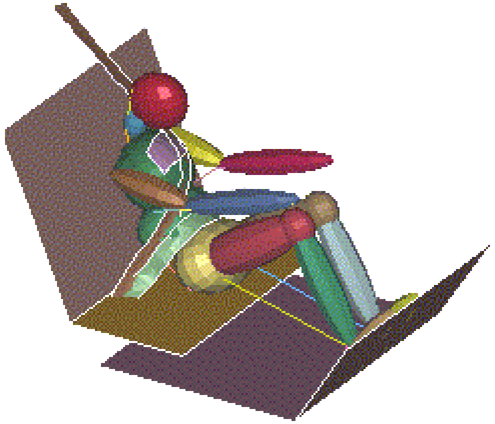
Phone: +46 (0)13 236680
Fax: +46 (0)13 214104
E-mail: info@dynamore.se
Web: www.dynamore.se

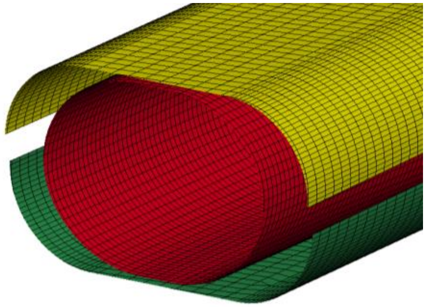
Rigid Bodies in LS-DYNA

It is common practice to make use of rigid bodies in mechanical simulation as a mean to reduce computational cost and complexity; this by not doing unnecessary evaluation of history variables for parts where the deformations/stresses are insignificant or not of interest. One example of this is sheet metal forming where the tooling often can be accurately be treated as rigid, another is automotive safety where the occupant can be treated as rigid in some cases. Rigid bodies are bypassed in the element processing and the mesh is only used to define inertial properties and geometry for contacts.

- Examples of use
  - Tooling
  - Simplifying parts of a large model
  - Vehicle crash
    - Restraint system
    - Roadside safety evaluation
  - Mechanical joints
    - Rigid bodies can also be used to define joints between deformable parts
  - Boundary condition application
    - Distribute and apply load
    - Define constraints
- Related to rigid bodies but not included in this webinar
  - Welds
  - Rigid/deformable switching
  - Rigidwall







*MAT_RIGID (020)

Material ID

Elastic material properties
(Used in contact definition)

3rd party coupling options, not dealt with in this webinar

```
*MAT_RIGID
$#     mid        ro         e        pr         n    couple         m     alias
         1   7.82e-9     2.1e5       0.3       0.0       0.0       0.0
$#     cmo      con1      con2
       0.0         0         0
$#lco or a1       a2        a3        v1        v2        v3
       0.0       0.0       0.0       0.0       0.0       0.0
```

Constraint options

Local coord. Sys. for motion and output

Constraint options

<u>cmo Centre of Mass constraint Option</u>

= 1 constraints in global system

= 0 no constraints

= -1 constraints in local system

<u>con1 Constraint 1</u>

= [translational constraint] if cmo = 1

= [local coord. sys. ID] if cmo = -1

<u>con2 Constraint 2</u>

= [rotational constraint] f cmo = 1

= [SPC constraint] if cmo = -1

Local coordinate system

<u>lco Local Coordinate system</u>

= [coord. sys. ID]

Alt.

<u>a1-v3 Local Coordinate system from vectors</u>

a1-a3 = [nodes defining vector **a**]

v1-v3 = [nodes defining vector **v**]

Defaults to the principal coordinate system of the rigid body.
Will follow the rigid body motion.

*MAT_RIGID (020)

Material ID

Elastic material properties
(Used in contact definition)

3rd party coupling options, not dealt with in this webinar

```
*MAT_RIGID
$#     mid        ro         e        pr         n    couple         m     alias
         1   7.82e-9     2.1e5       0.3       0.0       0.0       0.0
$#     cmo      con1      con2
       0.0         0         0
$#lco or a1       a2        a3        v1        v2        v3
       0.0       0.0       0.0       0.0       0.0       0.0
```

Constraint options

Local coord. Sys. for motion and output

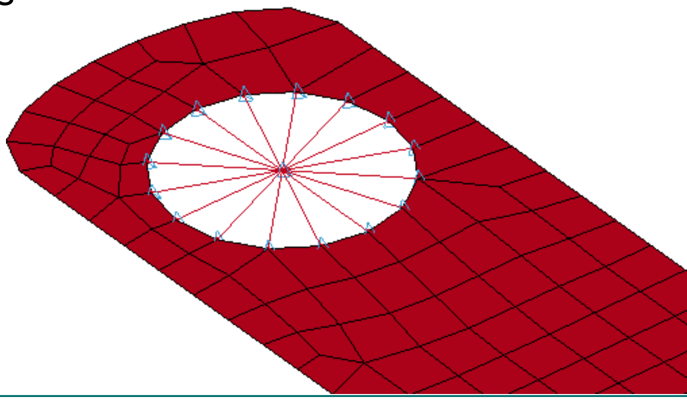
- Assigned to a part just like any other mechanical material, the part will then be bypassed in the element procedure.
  - Also, remember that the rigid body formulation in itself is a constraint
- Local coordinate systems gives orientation, origin is always the center of mass.
- Avoid using global constraints (`cmo = 1`) with `_LOCAL` prescribed motion, this is not permitted at all in implicit analysis.
- Take care when using local constraints (`cmo = -1`) and `_LOCAL` prescribed motion as conflicting motion and constraint can lead to unintended behavior; this will produce error termination in implicit analysis.
- For constraints where the orientation is to be updated with time as the rigid body rotates it's advised to use `*BOUNDARY_PRESCRIBED_MOTION_RIGID_LOCAL` instead, for a consistent behavior.

*CONSTRAINED_NODAL_RIGID_BODY[_SPC] (CNRB)

```
*CONSTRAINED_NODAL_RIGID_BODY_SPC
$#     pid       cid      nsid     pnode      iprt    drflag    rrflag
         1         0         1         0         0         0         0
$#     cmo      con1      con2
       0.0         0         0
```

- pid: Part* ID
- cid: Local coord. Sys. for motion and output
- nsid: Node set defining the rigid body
- pnode: Optional node used for post processing
- iprt: Write CNRB data to rbdout [0|1]
- drflag, rrflag: Disp. & rot. nodal release options (≠0 is generally not recommended)
- cmo, con1, con2: Constraint options (as for `*MAT_RIGID`)

- Define a nodal rigid body which is a rigid body that consists of defined nodes.
- * Sometimes if can be referred to as a part, sometimes not. Assigning unique PID for *PART and CNRB is good practice.
- Mass properties are determined from the nodal masses and coordinates unless the `_INERTIA` option is used.
- Lumped mass can be assigned to a node via `*ELEMENT_MASS`.
- Commonly used to apply constraints or loads to deformable bodies, making sure nodes moved together.
- `plotel = [1|2]` on `*CONTROL_RIGID` to generate parts for visual representation in d3plot.
- Don't let the name fool you, won't show up in `spcforc`.



General comments on Rigid Bodies

- If the model doesn't contain any deformable bodies a reasonable timestep must be set manually.
  - Unexpected rigid body motion could be the result of a too large time-step
  - Cancellation errors can occur for rotations if the time-step is too small
- A rigid part doesn't have to be continuous, all nodes in one part will still move as one continuous part; take care when copying parts.
- Two independent rigid bodies (PIDS) cannot share nodes unless merged.

*BOUNDARY_PRESCRIBED_MOTION_RIGID (BPM)

Part ID · Degree of freedom, 1-8 available for rigid bodies · Vel/acc/disp flag · Curve/function ID · Scale factor · Vector for arbitrary orientation (dof 4 & 8) · Birth and death of motion

```
*BOUNDARY_PRESCRIBED_MOTION_RIGID
$#     pid       dof       vad      lcid        sf       vid     death     birth
         1         1         2         1       1.0         0 1.0000E28       0.0
```

- Always act on the center of mass, if one wish for the BPM to act anywhere else than the geometrical center of mass use `*PART_INERTIA`.
- Vector (`vid`) is for motion in directions other than principle global direction and is fixed in time.
- `_LOCAL` will use `lco/cid` (MAT_020/CNRB) to define orientation instead of the global system, orientation is updated with time.
- It's advised not to use global constraint `cmo = 1` with the `_LOCAL` option.
- For `vad = 3` the abscissa in `lcid` is displacement instead of time.
- For `vad = 4` additional card 3 is needed to define displacement relation.
- Birth defines at which time the BPM is activated, but it also offsets the time in `lcid` equal to that of the time of birth.
- It is possible to use the `_NODE` option instead of `_RIGID` but in most cases a joint is a better option. If `_NODE` is used, only prescribe **one** node.

dof	Deegreee of freedom
= 1-3	x-z axis translation
= 4	translation along vector (`vid`)
= 5-7	x-z axis rotation
= 8	rotation around vector (`vid`)

vad	Velocity/acceleration/displacement flag
= 0	velocity
= 1	acceleration
= 2	displacement
= 3	velocity as a function of displacement
= 4	relative displacement

Example 1 – Translation and rotation using BPM

- Prescribing translational and rotational motion with and without the _LOCAL option
- Case was originally run in LS-DYNA R11.1

Summary on constraining Rigid Bodies

- Don’t use `*BOUNDARY_SPC_`*`OPTION`* on rigid bodies!
- CMO in MAT_020 acts on CoM, convenient but can’t output reaction forces ☹
- `_SPC` OPTION on CNRB acts on CoM, convenient but can’t output reaction forces ☹
- `*boundary_prescribed_motion_rigid` acts on CoM, can output reaction forces ☺
- Joints can be used as constraints and acts on nodes, can output reaction forces ☺

*PART_INERTIA

```
*PART_INERTIA
$#                                                                     title

$#     pid     secid       mid     eosid      hgid      grav    adpopt      tmid
         1         1         1         0         0         0         0         0
$#      xc        yc        zc        tm      ircs    nodeid
       0.0       0.0       0.0       0.0         0         0
$#     ixx       ixy       ixz       iyy       iyz       izz
       0.0       0.0       0.0       0.0       0.0       0.0
$#     vtx       vty       vtz       vrx       vry       vrz
       0.0       0.0       0.0       0.0       0.0       0.0
```

Global coord. of center of mass

Translational mass

Flag for inertia orientation reference coord. system

Identical to regular *PART

Node ID defining center of mass

Inertia tensor

Initial velocities

- Rigid body mass and inertial properties can be modified through option `_INERTIA`
  - LS-DYNA will disregard from the computed mass properties
  - Centre of mass can be defined either via global coordinates or a node (that should be attached to the part in question)
  - It's possible to assign the inertial tensor orientation with respect to a local coordinate system, this requires also optional card 6 of the `*PART` keyword
  - Initial velocities can also be specified with the _INERTIA option; however, note that this may be overwritten by `*INITIAL_VELOCITY`
- Usage examples
  - Change part inertial properties, example: assign real inertial properties to a rigid vehicle made of shell elements
  - Change reference of prescribed motions and rigid body loads, only relevant with respect to rotation of rigid body
- Similar for CNRB:s by using the `_INERTIA` option on the CNRB card

Example 2 - BPM with *PART_INERTIA

- Changing reference of prescribed motions and rigid body loads, only relevant with respect to rotation of rigid body
- Case was originally run in LS-DYNA R11.1

*LOAD_RIGID_BODY

Curve/function ID

Local system

Degree of freedom

Nodes defining follower force direction

Scale factor

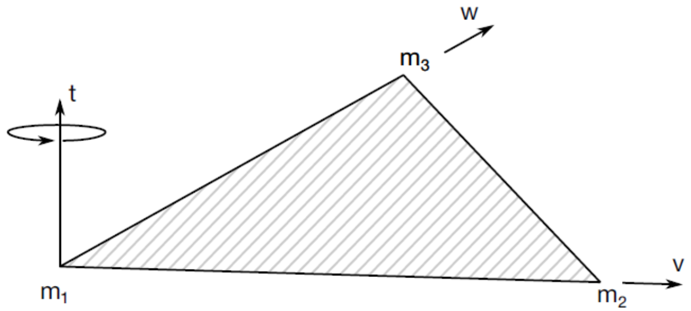
Part ID

```
*LOAD_RIGID_BODY
$#   pid       dof      lcid        sf       cid        m1        m2        m3
       0         1         0       1.0         0         0         0         0
```

- Force is applied at the center of mass or a moment around an axis in a global system.
- As an option a local system can be defined for force or moment directions using `cid`, these directions are fixed in space.
- Using `dof = 4|8` and `m1-m3` defines a force that follows the rigid body orientation in time (dir. t).

`dof`	Deegreee of freedom
`= 1-3`	x-z axis translation
`= 4`	follower force (`m1-m3`)
`= 5-7`	x-z axis rotation
`= 8`	follower moment (`m1-m3`)

`lcid`	Load curve/function ID
`> 0`	force as function of time
`< 0`	force as function of abs. disp. (curve only)



Other constraints for Rigid Bodies

- *CONSTRAINED_EXTRA_NODES_[NODE | SET], add nodes to a rigid body. Can be used to made part of deformable bodies rigid, or adding massless nodes for constraining purposes
- *CONSTRAINED_RIGID_BODIES, constrain slave rigid bodies to a master rigid body, i.e. merge (only applicable to MAT_020 rigid bodies)
- *CONSTRAINED_RIGID_BODIY_INSERT, constrain a slave rigid body to follow a master rigid body in all directions but one. Useful for example in metal forming with die inserts
- *CONSTRAINED_RIGID_BODY_STOPPERS … Next slide!

*CONSTRAINED_RIGID_BODY_STOPPERS

Curves for displacement limitation

Slave parts option

Curve for limitation of maximum absolute velocity

Vector for arbitrary direction

Direction of limitation

Part ID

```
*CONSTRAINED_RIGID_BODY_STOPPERS
$#     pid     lcmax     lcmin    psidmx    psidmn    lcvmnx       dir       vid
         1         0         0         0         0         0         1         0
$#      tb        td
       0.0 1.0000E21
```

Activation/deactivation time

- Limit displacement or absolute position as a function of time
- Limit slave parts’ position relative master part
- Limit absolute velocity as a function of time
- Limit can be applied in global or an arbitrary direction (fixed in time)

Example 3 – Deformable body & force controlled rigid punch

- Stabilizing a force-controlled problem
- Case was originally run in LS-DYNA R11.1

Short Comment on Rigid Bodies & Contacts

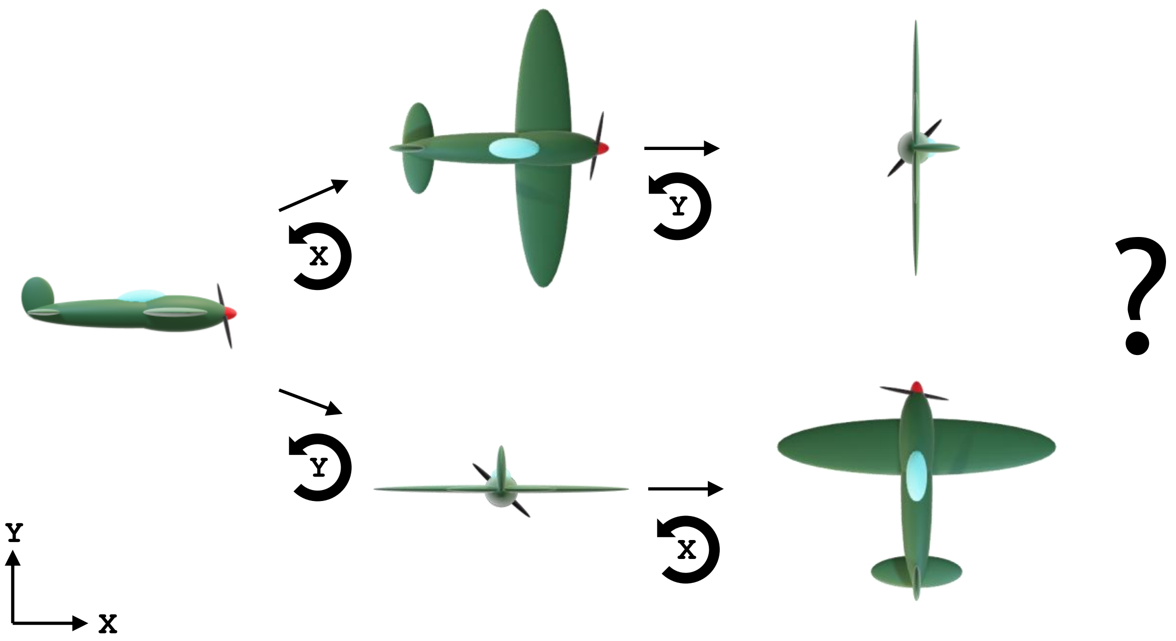
- Elastic material data for rigid bodies used in contact, reasonable values are required
- Contacts of type “Forming” will ignore master surface (rigid body) thickness
- Shell thickness offsets are always included in single surface, constraint-based, automatic surface-to-surface, and automatic nodes-to-surface contact types.
- Contact behavior can be more sensitive to the contact stiffness when all bodies are rigid

All “fun and games” until you have rotations around multiple axis simultaneously…

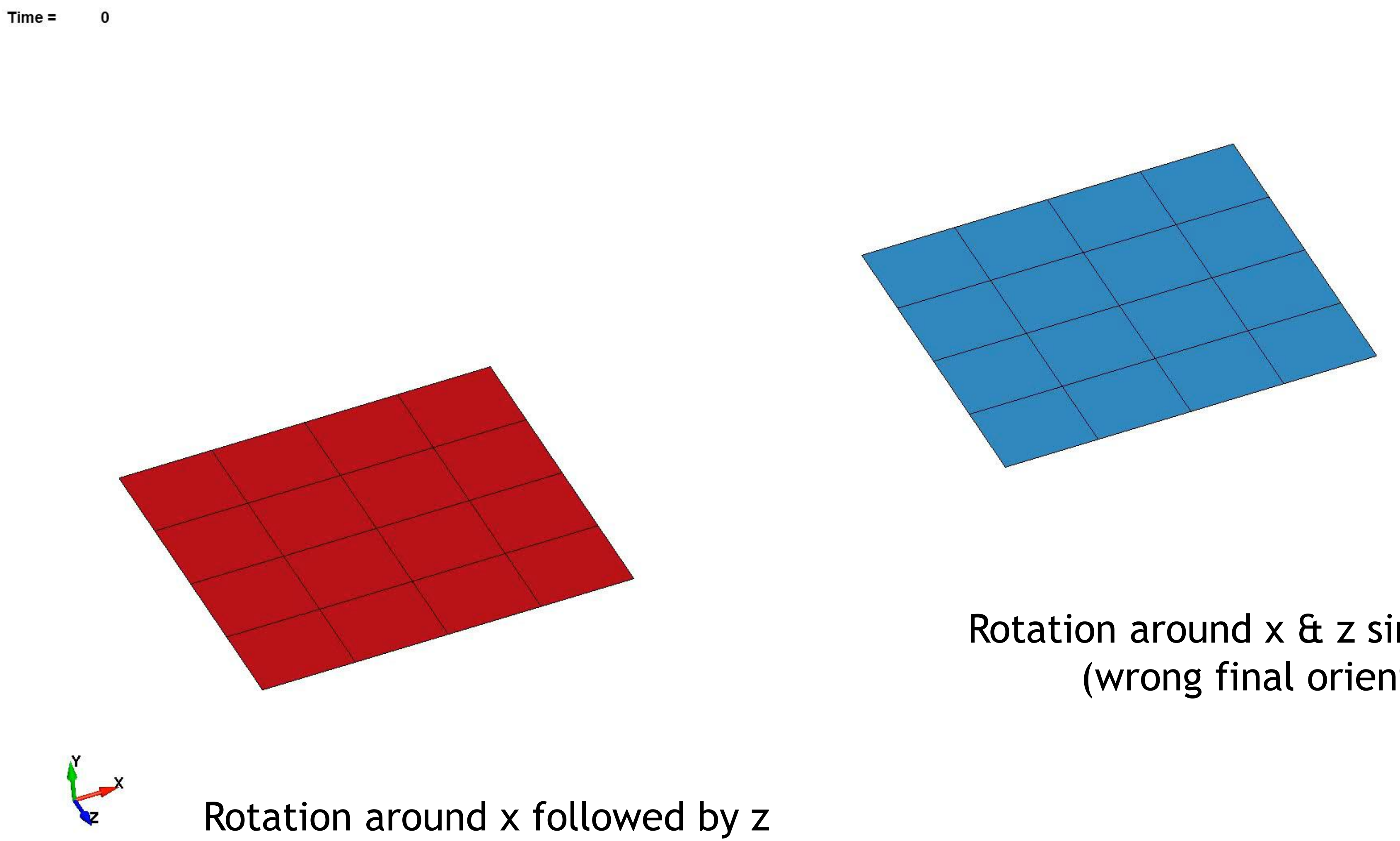
```
*BOUNDARY_PRESCRIBED_MOTION_RIGID
$#      pid       dof       vad      lcid        sf       vid     death     birth
          1         5         2         1       1.0         01.00000E28       0.0
*BOUNDARY_PRESCRIBED_MOTION_RIGID
$#      pid       dof       vad      lcid        sf       vid     death     birth
          1         6         2         1       1.0         01.00000E28       0.0
```

Rotations performed with *BOUNDARY_PRESCRBIED_MOTION_RIGID are done so using an incremental update. This method can produce unexpected results when rotations are prescribed simultaneously about multiple axes. It should work reliably if the rotations are prescribed about one axis at a time.

There are other ways of achieving this…



Showcase - Erroneous multi axis rotation using BPM



Rotation around x & z simultaneously
(wrong final orientation)

Rotation around x followed by z
(correct final orientation)

The solution: *BOUNDARY_PRESCRIBED_ORIONTATION_RIGID_*OPTION* (BPO)

Part whose motion is prescribed

Part that defines the orientation reference

Method for time history curve interpolation
(leave default)

Activation/deactivation time

Time offset flag

```
*BOUNDARY_PRESCRIBED_ORIENTATION_RIGID_OPTION
$#  pidb      pida      intrp     birth      death     toffset
       1         0         1        0.0 1.0000E20         0
...
```

This row is common for all orientation options

In contrast to BPM, *BOUNDARY_PRESCRIBED_ORIENTATION_RIGID implicitly prescribes an orientation as a function of time. Based on a reference state the angular offsets are computed exactly. This method produces correct results for any rotation sequence prescribed about any set of three unique axes.

- Options
  - _ANGLES - angles and sequence; specify in which sequence rotations shall be applied
  - _DIRCOS - direction cosine; path independent
  - _EULERP - Euler parameters (Quaternions); path independent
  - _VECTOR - vector direction and spin; prescribes the direction of one arbitrary axis of the body and spin about that axis
- Incompatible with *DEFINE_CURVE_FUNCTION
- `toffset` offsets the time in curve data equal to that of the time of birth
- A local coordinate system `lco` in `*MAT_RIGID` should be specified with `*DEFINE_COORDINATE_NODES` and `flag = 1`
- Double precision can make a difference if the orientation isn't as expected
- All load curves must contain the same number of points and the data must be uniformly spaced.

EXAMPLE - ORIENTATION

```
*BOUNDARY_PRESCRIBED_ORIENTATION_RIGID_VECTOR
$#    pidb      pida      intrp     birth     death   toffset
         1         0          1       0.0 1.0000E20         0
$#  lcidv1    lcidv2    lcidv3     lcids   valspin
         1         2          3         0       0.0
```

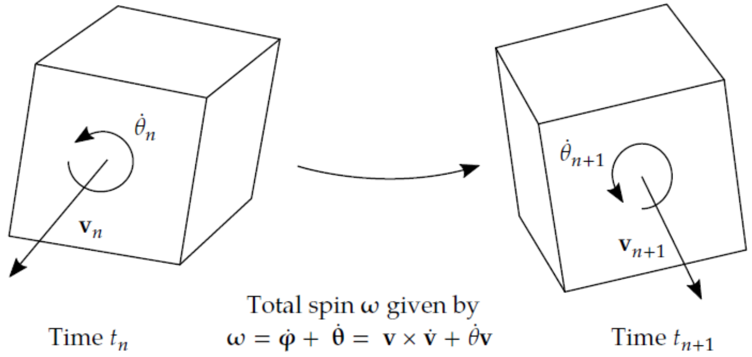
This row is common for all orientation options

Constant overlayed spin parallel to **v**, radians per unit time

Curves defining the x-z components of vector **v** as a function of time

Curve defining a time-varied overlayed spin parallel to **v**, radians per unit time

- The vector **v** is attached to the rigid body in its initial orientation, thereafter, the orientation of **v** affects the orientation of the rigid body.
- An overlayed spin(rad/s) of the rigid body, either constant or time-varied rotational velocity can be defined, this makes _VECTOR most suitable for cases where the orientation of only one axis is important.
- Sensitive to time-step size and double precision can improve result

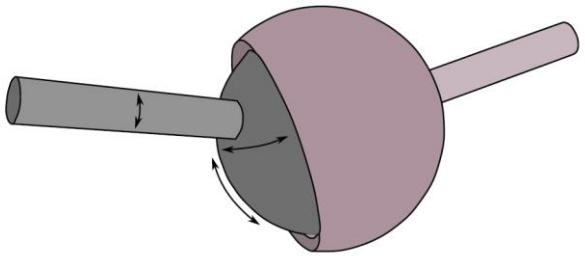


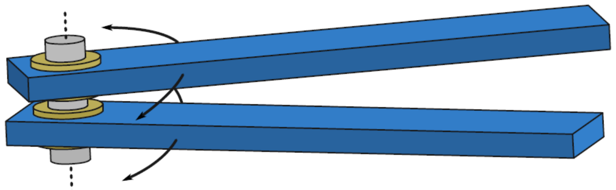
EXAMPLE 4 – Multiple rotations the correct way (BPO_VECTOR)

- One way of prescribing arbitrary orientation of rigid body
- Case was originally run in LS-DYNA R11.1

Joints

- Joints can be defined between two rigid bodies
- Often a convenient way to model movement in mechanisms but also to achieve desired motion of a single rigid body (alterative to *PART_INERTIA)
- *CONSTRAINED_EXTRA_NODES is often used to construct the joints
- There are also driven joints of type _MOTOR



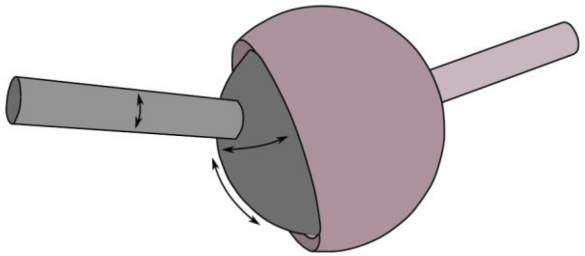


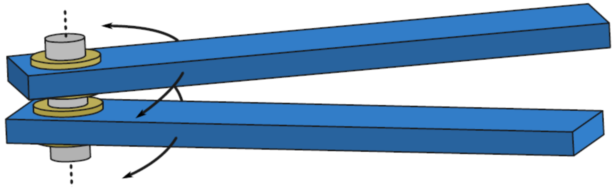
Joints

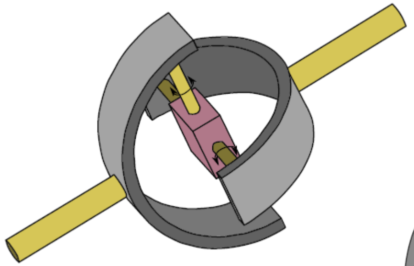
- Penalty or Constraint formulation (Lagrange Multiplier) available
- There is an option, `_COOR_`, that creates extra nodes automatically for all regular joints
- Joint forces can be output in `jntforc`
- Joint between deformable parts can in a way be created via CNRB:s
- Failure of joint is an option (penalty type only)
- Joint stiffness and damping can be defined via `*CONSTRAINED_JOINT_STIFFNESS_`*`OPTION`*

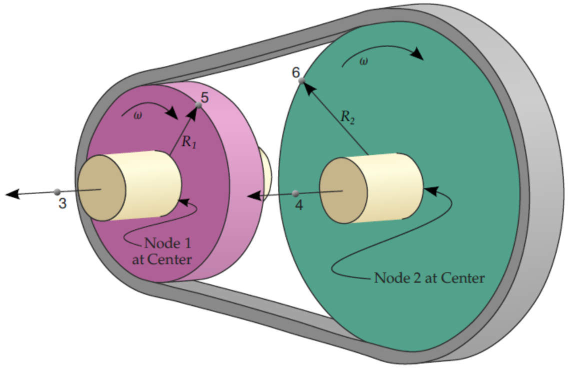
Joints types

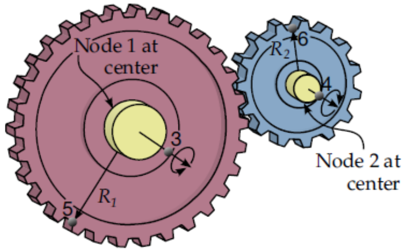
- *CONSTRAINED_JOINT_SPHERICAL
- *CONSTRAINED_JOINT_REVOLUTE
- *CONSTRAINED_JOINT_CYLINDRICAL
- *CONSTRAINED_JOINT_PLANAR
- *CONSTRAINED_JOINT_UNIVERSAL
- *CONSTRAINED_JOINT_TRANSLATIONAL
- *CONSTRAINED_JOINT_LOCKING
- *CONSTRAINED_JOINT_TRANSLATIONAL_MOTOR
- *CONSTRAINED_JOINT_ROTATIONAL_MOTOR
- *CONSTRAINED_JOINT_GEARS
- *CONSTRAINED_JOINT_RACK_AND_PINION
- *CONSTRAINED_JOINT_CONSTANT_VELOCITY
- *CONSTRAINED_JOINT_PULLEY
- *CONSTRAINED_JOINT_SCREW

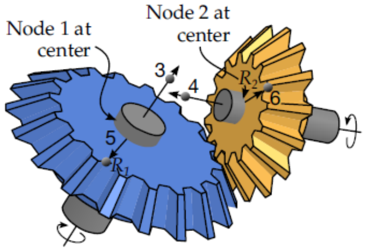












*CONSTRAINED_JOINT_REVOLUTE_ID

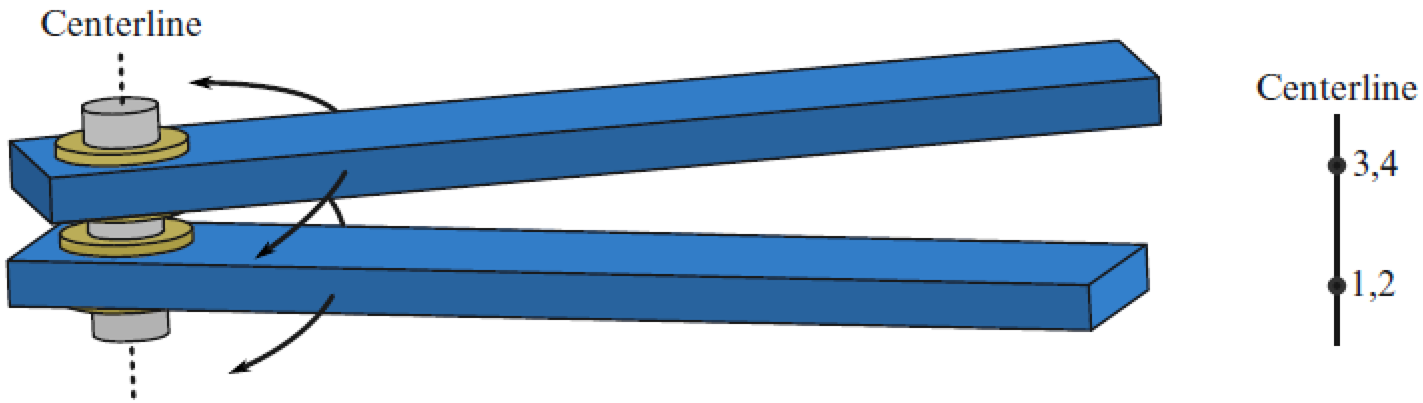
Joint ID

```
*CONSTRAINED_JOINT_REVOLUTE_ID
$#  jid                                                              title
      1
$#    n1        n2        n3        n4        n5        n6      rps      damp
       1         3         2         4         0         0      1.0       1.0
```

Nodes defining the joint, how depends on the joint type

Relative penalty stiffness

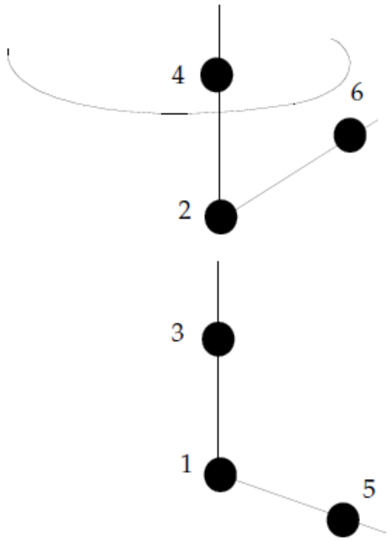
Damping scale factor
(revolute & spherical only)



Note that nodes must be coincident

*CONSTRAINED_JOINT_ROTATIONAL_MOTOR_ID

```
*CONSTRAINED_JOINT_ROTATIONAL_MOTOR_ID
$#     jid                                                                     title
         2
$#      n1        n2        n3        n4        n5        n6       rps      damp
         1         2         3         4         5         6       1.0       1.0
$#    parm      lcid      type        r1   h_angle
       0.0         1         0       0.0       0.0
```



`parm`	parameter for joint function (default for motor)
`lcid`	load curve for motor joints
`type`	vel/acc/disp for motor joints
`r1`	radius 1 for gear and pully types
`h_angle`	helix angle in degrees for ex. worm gears

Load curve for rotational motor defines rotation in radians per time unit,
1-3 and 2-4 are coaxial but nodes doesn't have to be coincident.

EXAMPLE 5 – Joint and Motor Joint

- Using joints to drive a small vehicle
- Case was originally run in LS-DYNA R11.1

CONSTRAINED_JOINT_STIFFNESS_*OPTION

```
*CONSTRAINED_JOINT_STIFFNESS_OPTION
$#    jsid      pida      pidb      cida      cidb       jid
         1         1         0         0         0         0
...
(Stiffness option parameters)
```

- `_GENERALIZED` - stiffness and damping based on relative angle
- `_FLEXION-TORSION` - similar to `_GENEREALIZED` but the definition and behavior of the flexion-torsion is slightly different
- `_TRANSLATIONAL` - stiffness and damping based on relative distance
- `_CYLINDRICAL` - models the stiffness and damping of a pin in a hole with a gap

- The energy that is dissipated with the joint stiffness option is written for each joint in `jntforc`
- Can also limit joint movement

A note on joints in implicit simulations

```
*CONTROL_IMPLICIT_JOINT
$#   ispher     irevol     icylin
          1          1          1
```

- Possible to change a penalty joint type in implicit simulation
  - Suitable if for instance joint failure is desired

*CONTROL_RIGID

```
*CONTROL_RIGID
$#       lmf      jntf    orthmd     partm    sparse    metalf    plotel     rbsms
           0         0         0         0         0         0         0         0
$#    norbic   gjadstf   gjadvsc   tjadstf   tjadvsc
           0       0.0       0.0       0.0       0.0
```

lmf	Joint treatment option
= 0	penalty formulation
= 1	constraint formulation with Lagrange multipliers

Only applies to explicit. If one has issues joint behavior the constraint formulation may solve the problem. However, penalty type is significantly faster. Also, functions such as joint failure require penalty formulation. The usage is also somewhat more convenient, for example constraint formulation doesn't allow for duplicate constraints.

*CONTROL_RIGID

```
*CONTROL_RIGID
$#     lmf      jntf    orthmd     partm    sparse    metalf    plotel     rbsms
         0         0         0         0         0         0         0         0
$#  norbic   gjadstf   gjadvsc   tjadstf   tjadvsc
         0       0.0       0.0       0.0       0.0
```

lmf	Joint treatment option
= 0	penalty formulation
= 1	constraint formulation with Lagrange multipliers
jntf	Generalized joint stiffness formulation
= 0	incremental update
= 1	total formulation
= 2	total formulation for implicit analysis

This affects mainly explicit (if IACC = 1 is jntf = 2 for implicit). The default joint behavior is for the relative angles between the two coordinate systems to be done incrementally. This is an approximation, in contrast to the total formulation where the angular offsets are computed exactly.

*CONTROL_RIGID

Has to do with linearized flexible bodies
to do, not dealt with in this webinar

```
*CONTROL_RIGID
$#     lmf      jntf    orthmd     partm    sparse    metalf    plotel     rbsms
         0         0         0         0         0         0         0         0
$#  norbic   gjadstf   gjadvsc   tjadstf   tjadvsc
         0       0.0       0.0       0.0       0.0
```

lmf	Joint treatment option
= 0	penalty formulation
= 1	constraint formulation with Lagrange multipliers
jntf	Generalized joint stiffness formulation
= 0	incremental update
= 1	total formulation
= 2	total formulation for implicit analysis
orthmd	Orthogonalize modes relative to each other
= 0|1	true | false, modes are already orthogonalized
partm	Global mass matrix defines part mass distribution
= 0|1	true | false

Default is that the global mass matrix determines part mass distribution, this means that mass from other parts that share nodes in included.

*CONTROL_RIGID

```
*CONTROL_RIGID
$#     lmf      jntf    orthmd     partm    sparse    metalf    plotel     rbsms
         0         0         0         0         0         0         0         0
$#  norbic   gjadstf   gjadvsc   tjadstf   tjadvsc
         0       0.0       0.0       0.0       0.0
```

lmf	Joint treatment option
= 0	penalty formulation
= 1	constraint formulation with Lagrange multipliers
jntf	Generalized joint stiffness formulation
= 0	incremental update
= 1	total formulation
= 2	total formulation for implicit analysis
orthmd	Orthogonalize modes relative to each other
= 0|1	true | false, modes are already orthogonalized
partm	Global mass matrix defines part mass distribution
= 0|1	true | false
sparse	Sparse matrix multiplication for modal and damping
= 0|1	false | true

Option for sparse matrix multiply subroutines for the modal stiffness and damping matrices. This save a substantial number of operations if the matrix is truly sparse. However, the overhead will slow the multipliers for densely populated matrices. Full matrix multiplication often faster

*CONTROL_RIGID

```
*CONTROL_RIGID
$#     lmf      jntf    orthmd     partm    sparse    metalf    plotel     rbsms
         0         0         0         0         0         0         0         0
$#  norbic   gjadstf   gjadvsc   tjadstf   tjadvsc
         0       0.0       0.0       0.0       0.0
```

lmf	Joint treatment option
= 0	penalty formulation
= 1	constraint formulation with Lagrange multipliers
jntf	Generalized joint stiffness formulation
= 0	incremental update
= 1	total formulation
= 2	total formulation for implicit analysis
orthmd	Orthogonalize modes relative to each other
= 0|1	true | false, modes are already orthogonalized
partm	Global mass matrix defines part mass distribution
= 0|1	true | false
sparse	Sparse matrix multiplication for modal and damping
= 0|1	false | true
metalf	Fast update of rigid body nodes (suppress rotation)
= 0	full treatment
= 1	fast update (applicable to metal forming using shell elements)

Fast update option of rigid body nodes, when active the rotational motion of all rigid bodies should be suppressed. Only applicable to sheet metal forming analysis using 2D and 3D shell elements.

*CONTROL_RIGID

```
*CONTROL_RIGID
$#       lmf      jntf    orthmd     partm    sparse    metalf    plotel     rbsms
           0         0         0         0         0         0         0         0
$#    norbic   gjadstf   gjadvsc   tjadstf   tjadvsc
           0       0.0       0.0       0.0       0.0
```

plotel	Generation of elements for visualization of CNRB:s
= 0	false
= 1	one part is generated
= 2	one part for each CNRB is generated

Automatic generation of *ELEMENT_PLOTEL for CNRB:s, used for visualization purposes in D3PLOT. WARNING! These get an automatic part ID, 1000000 alt. 1000000 + n for each CNRB. Be wary of part ID conflicts

*CONTROL_RIGID

```
*CONTROL_RIGID
$#       lmf      jntf    orthmd     partm    sparse    metalf    plotel    rbsms
           0         0         0         0         0         0         0        0
$#    norbic   gjadstf   gjadvsc   tjadstf   tjadvsc
           0       0.0       0.0       0.0       0.0
```

plotel Generation of elements for visualization of CNRB:s

= 0 false

= 1 one part is generated

= 2 one part for each CNRB is generated

rbsms Consistent mass scaling of rigid bodies

= 0|1 false | true

Generally not recommended. Imposes mass scaling on rigid bodies, but also calculates rotational inertia in a different way (default is a mean of shell element inertia)

Purpose: Rigid bodies connected to deformable elements can result in significant addition of inertia, such as in CNRB spotwelds in automotive. However, this option turns out to not be that effective and can cause more problems.

*CONTROL_RIGID

```
*CONTROL_RIGID
$#     lmf      jntf    orthmd     partm    sparse    metalf    plotel     rbsms
         0         0         0         0         0         0         0         0
$#  norbic   gjadstf   gjadvsc   tjadstf   tjadvsc
         0       0.0       0.0       0.0       0.0
```

plotel Generation of elements for visualization of CNRB:s

= 0 false
= 1 one part is generated
= 2 one part for each CNRB is generated

rbsms Consistent mass scaling of rigid bodies

= 0|1 false | true

norbic Circumvent rigid body inertia check

= 0|1 false | true

During initialization, the determinant of the rigid body inertia tensor is checked. If it falls below a tolerance value, LS-DYNA issues an error message and the calculation stops. In some rare cases (for example with an unfavorable system of units), such tiny values would still be valid.

*CONTROL_RIGID

```
*CONTROL_RIGID
$#     lmf      jntf    orthmd     partm    sparse    metalf    plotel     rbsms
         0         0         0         0         0         0         0         0
$#  norbic   gjadstf   gjadvsc   tjadstf   tjadvsc
         0       0.0       0.0       0.0       0.0
```

`plotel` Generation of elements for visualization of CNRB:s

= 0 false

= 1 one part is generated

= 2 one part for each CNRB is generated

`rbsms` Consistent mass scaling of rigid bodies

= 0|1 false | true

`norbic` Circumvent rigid body inertia check

= 0|1 false | true

`gjadstf`
`gjadvsc`
`tjadstf`
`tjadvsc` Add global joint stiffness|damping

= 0.0 stiffness or damping value

*CONSTRAINED_JOINT_STIFFNES_[GENERALIZED|TRANSLATIONAL] equivalent but globally for all joints. Can be useful to avoid zero energy modes in implicit analysis.

*BOUNDARY_PRESCRIBED_ACCELEROMETER_RIGID (BPA)

```
*BOUNDARY_PRESCRIBED_ACCELEROMETER_RIGID
$#     pid
         1
$#     nid       cid     lcidx     lcidy     lcidz
         1         1         1         2         3
         2         3         4         5         6
...
(Additional accelerometers)
```

Part ID

Node at origin of accelerometer

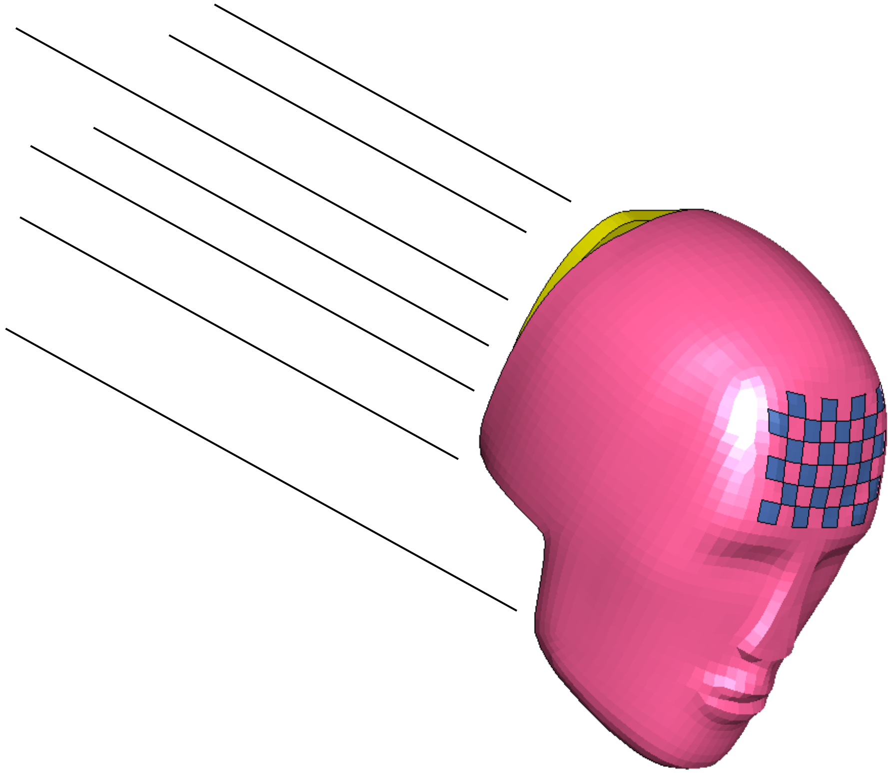
Accelerometer orientation coord. sys.

Load curves containing x, y, z local acceleration time histories
(1 row per accelerometer, minimum 3 rows)

- Prescribe motion from accelerometer data
  - A minimum of 3 accelerometers are needed (cannot all be collinear)
  - Curve data must contain the same number of points and must be uniformly spaced
  - Local axes of the accelerometers must be orthogonal
  - Coordinate system must be defined using `*DEFINE_COORDINARE_NODES`, with nodes attached to the rigid body whose motion is prescribed

EXAMPLE ACCELEROMETER

- Prescribing motion from “experiment” accelerometer data
- Case was originally run in LS-DYNA R11.1
- Original head model replaced with a ball



Thank you!

DYNA MORE

Your LS-DYNA distributor and more