AD FALCON API Manual

Rigid Body Motion in FALCON

Javad Ghorbani

March 26, 2026

Contents

1 Rigid Body Motion in FALCON

This page documents FALCON's rigid body capabilities for modeling structures and components that move as single entities, including force-controlled motion with static equilibrium or dynamic integration.

1.1 Overview

FALCON's rigid body implementation provides:

- **Grouped node motion**: Multiple nodes move together as a rigid entity
- **Mass and damping properties**: Physical properties for dynamic analyses
- **Prescribed motion**: Translation and rotation via kinematic equations
- **Force-controlled motion**: Apply forces with automatic equilibrium enforcement
- **Static force regulation**: Iterative displacement adjustment to match prescribed forces
- **Dynamic force integration**: Direct ODE integration using mass/damping for transient analyses
- **Comprehensive monitoring**: Output reactions, velocities, accelerations, and prescribed forces via CSV

The rigid body system is designed for soil-structure interaction problems where structural components (e.g., foundations, retaining walls, piles) interact with deformable soil in coupled hydro-mechanical or multiphase analyses.

1.2 Rigid Body Definition

Rigid bodies are defined using the `% RigidBodies` section.

1.3 Syntax

1.3.1 Section header

FALCON treats section names as case-insensitive and whitespace-insensitive, so these headers are equivalent:

```
% RigidBodies
% Rigid Bodies
% rigid_bodies
```

1.3.2 Section format

```
% RigidBodies
@RigidBody <ID>
  @@NodeIDs: <n1> <n2> ...
  @@Mass: <value>
  [@@DampingLinear: <value>]
  [@@DampingAngular: <value>]
  [@@ReferenceNodeID: <nodeId>]
  [@@ReferenceDOFs: <dof1> <dof2> ...]
  [@@FollowerNodeIDs: <n1> <n2> ...]
  [@@InertiaDiag: <Ixx> <Iyy> <Izz>]
  [@@InertiaTensor: <Ixx> <Ixy> <Ixz> <Iyx> <Iyy> <Iyz> <Izx> <Izy> <Izz>]
@RigidBody <ID>
  ...
%%%
```

Notes:

- `@RigidBody` headers accept an optional trailing `:` on the ID (so `@RigidBody 1:` works).
- Directive names are case-insensitive and tolerate one or more leading `@` characters (recommended style: `@@Key: ...`).

1.3.3 Multiple rigid bodies

- Define multiple rigid bodies by repeating `@RigidBody <ID> ...` blocks inside the same `% RigidBodies` section.
- Each block ends when the next `@RigidBody ...` begins or when the section ends (`%%%` or the next `% ...` section).

1.3.4 Parameters

```
@@NodeIDs: <Node1> <Node2> ... <NodeN>
@@Mass: <Value>
@@DampingLinear: <Value>
```

Parameter	Type	Description	Validation
`@RigidBody`	Integer	Unique identifier for the rigid body	Must be unique
`@@NodeIDs`	List	Node IDs forming the rigid body	Must exist in mesh
`@@Mass`	Double	Mass of the rigid body (for dynamic integration)	> 0

Parameter	Type	Description	Validation
@@DampingLinear	Double	Linear viscous damping coefficient (optional)	≥ 0
@@DampingAngular	Double	Rotational viscous damping coefficient (optional)	≥ 0
@@ReferenceNodeID	Integer	Reference point for reporting/moments (optional)	Must exist in mesh
@@ReferenceDOFs	List	Informational DOF list for the reference node	Must be valid DOFs
@@FollowerNodeIDs	List	Informational follower node list	Must exist in mesh
@@InertiaDiag	3 Doubles	Diagonal inertia about reference point (optional)	Each > 0
@@InertiaTensor	9 Doubles	Full 3x3 inertia tensor (row-major, optional)	Positive definite

1.3.5 Example

```
% RigidBodies
@RigidBody 1
@@NodeIDs: 100 101 102 103 104 105
@@Mass: 5000.0
@@DampingLinear: 50.0
%%%
```

Interpretation: Rigid body 1 consists of 6 nodes that move as a rigid group, with mass 5000 and linear damping coefficient 50.

Notes: - @@ReferenceNodeID is used as the **reference point** for computing and reporting rigid-body moments. If omitted, FALCON uses the centroid of the rigid-body nodes. - If inertia is not provided, FALCON approximates the inertia tensor from @@NodeIDs and @@Mass assuming uniform mass distribution over the nodes (about the chosen reference point). This is intended for force/torque-controlled dynamics and may be a crude approximation for real structures.

1.4 Rigid Motion Constraints

Rigid motion constraints specify how rigid bodies move during simulation. Defined in the % RigidMotionConstraints section.

1.4.1 Understanding Motion Control Options

Rigid motion constraints provide two fundamentally different ways to control how a rigid body moves:

1. **Kinematic Motion (Prescribed Motion)**: You directly specify the motion of the rigid body as a function of time. This is useful when you know exactly how the body should move (e.g., a foundation settling at a constant rate, a retaining wall being pushed at a known velocity, or a prescribed displacement history from field measurements).

   - **Displacement equations** (`@@DispEqX/Y/Z`): Prescribe *absolute* displacement as a function of time (solver applies the increment $\Delta u = u(t) - u(t - \Delta t)$ internally)
   - **Velocity equations** (`@@VelEqX/Y/Z`): Prescribe velocity; displacement is computed by integration
   - **Acceleration equations** (`@@AccelEqX/Y/Z`): Prescribe acceleration; velocity and displacement are computed by integration
   - **Rotation equations** (about `@@RotationAxis` and `@@RotationCenter`):
     - `@@AngDispEq` (absolute angle, solver applies increment)
     - `@@AngVelEq` (angular velocity, solver integrates)
     - `@@AngAccelEq` (angular acceleration, solver integrates)

2. **Force-Controlled Motion**: You apply forces to the rigid body, and the solver determines the resulting motion. This is useful when you know the loading but not the exact motion (e.g., applying a known load to a foundation and letting it settle naturally, or applying a time-varying force and observing the response).

   - **Static mode**: The solver iteratively adjusts the rigid body position until the reaction forces match the prescribed forces (force equilibrium)
   - **Dynamic mode**: The solver integrates the equation of motion using mass and damping to compute velocity and displacement from the applied forces

3. **Torque-Controlled Rotation (Dynamic mode)**: You apply global torques (`@@TorqueX/Y/Z`) about the rigid-body reference point. In `SimulationMode: Dynamic`, FALCON integrates a rotational ODE using the rigid-body inertia and angular damping to generate incremental rigid rotations.

You can combine both approaches (e.g., prescribe motion in one direction while applying forces/torques in another). For best predictability, use **only one control type per DOF per step** (e.g., don't prescribe `@@DispEqY` and also use `@@ForceY` in the same step unless you intentionally want them to superpose).

1.4.2 Input Format

```
% RigidMotionConstraints
@RigidMotionConstraint <ID>
@@MotionType: <Translation|Rotation|Mixed>
@@RigidBodyID: <RigidBody_ID>
@@StepIds: <Step1> <Step2> ...

# --- Kinematic motion equations (optional) ---
@@VelEqX: a=<val> b=<val> c=<val> ...   (or @@DispEqX, @@AccelEqX)
@@VelEqY: a=<val> b=<val> c=<val> ...
@@VelEqZ: a=<val> b=<val> c=<val> ...
@@AngDispEq: a=<val> b=<val> ...        (or @@AngVelEq, @@AngAccelEq)
@@AngVelEq: a=<val> b=<val> ...
@@AngAccelEq: a=<val> b=<val> ...
@@RotationAxis: <x> <y> <z>
@@RotationCenter: <x> <y> <z>

# --- Force-controlled motion (optional) ---
@@ForceX: <baseline_force>
@@ForceY: <baseline_force>
@@ForceZ: <baseline_force>
@@ForceLoadX: LoadType <type> Step <step> ...
@@ForceLoadY: LoadType <type> Step <step> ...
@@ForceLoadZ: LoadType <type> Step <step> ...
@@ForcePropagateStepsX: <Step1> <Step2> ...
@@ForcePropagateStepsY: <Step1> <Step2> ...
@@ForcePropagateStepsZ: <Step1> <Step2> ...

# --- Force regulation (static equilibrium only) ---
@@ForceTolerance: <value>
@@ForceRegMaxIters: <value>
@@ForceRegGain: <value>

# --- Torque-controlled rotation (optional, Dynamic mode) ---
@@TorqueX: <baseline_torque>
@@TorqueY: <baseline_torque>
@@TorqueZ: <baseline_torque>
@@TorqueLoadX: LoadType <type> Step <step> ...
@@TorqueLoadY: LoadType <type> Step <step> ...
@@TorqueLoadZ: LoadType <type> Step <step> ...
@@TorquePropagateStepsX: <Step1> <Step2> ...
@@TorquePropagateStepsY: <Step1> <Step2> ...
@@TorquePropagateStepsZ: <Step1> <Step2> ...
%%%
```

1.4.3 Core Parameters

Parameter	Type	Description	Default
@@MotionType	String	Type of motion: Translation, Rotation, or Mixed	Required
@@RigidBody ID	Integer	ID of the rigid body to which this constraint applies	Required
@@StepIds	List	Simulation steps where this constraint is active	Required

1.4.4 Kinematic Motion Equations

For each axis (X, Y, Z), specify **one** of:

- @@DispEqX/Y/Z: **Absolute displacement** as function of time (the solver applies the *increment* each substep)
- @@VelEqX/Y/Z: Velocity as function of time (displacement = $\int v\,dt$)
- @@AccelEqX/Y/Z: Acceleration as function of time (displacement = $\int \int a\,dt^2$)

Equation format: a=<val> b=<val> c=<val> d=<val> f=<val> g=<val>
Evaluation: $\mathrm{eq}(t) = a + bt + d \cdot \exp(-ct) \cdot \sin(ft + g)$
For displacement-form equations, FALCON applies: - $\Delta u = u(t) - u(t_{\text{prev}})$ for @@DispEq* - $\Delta\theta = \theta(t) - \theta(t_{\text{prev}})$ for @@AngDispEq
Alternatively, use tabular data:

```
@@VelEqY: TabularData file=velocity_profile.txt
```

1.4.5 Rotation (Rigid Kinematics)

When @@MotionType is Rotation or Mixed, define the rotation geometry:

- @@RotationAxis: rx ry rz (global axis; does not need to be normalized)
- @@RotationCenter: cx cy cz (global point)

Then specify **one** of: - @@AngDispEq (absolute angle vs time) - @@AngVelEq (angular velocity vs time) - @@AngAccelEq (angular acceleration vs time)
All three use the same equation syntax (parametric or TabularData) as the translational @@*EqX/Y/Z keys.

1.4.6 Force-Controlled Motion

Apply forces to rigid bodies with automatic motion adjustment:

Parameter	Description
@@ForceX/Y/Z	**Baseline force magnitude** applied in the X, Y, or Z direction. This is the maximum or reference force value that will be scaled by the load type. For example, @@ForceY: -100.0 means apply a force of magnitude 100 downward (negative Y) in the specified direction. Units are consistent with your model's unit system.
@@ForceLoadX/Y/Z	**Time variation pattern** for the force. Specifies how the baseline force magnitude varies over time during the simulation. Options include: Ramp (gradually increase from 0 to full magnitude), Sinusoidal (oscillating force), DampedSinusoidal (oscillating with decay), Immediate (apply instantly), or Tabular (user-defined time history). Uses the same syntax as Load Application (see Stress Boundary for examples).
@@ForcePropagate StepsX/Y/Z	**Step IDs where the force is active** for each direction. This overrides the general @@StepIds if specified. For example, if @@ StepIds: 1 2 3 but @@ForcePropagateStepsY: 2 3, then the Y-direction force is only active during steps 2 and 3, while other directions may be active in all three steps. Leave empty or use "-" to use the general @@StepIds for that direction.

Example force specifications:

Example 1: Ramped force in single step

```
@@ForceY: -100.0
@@ForceLoadY: LoadType Ramp Step 2
@@ForcePropagateStepsY: 2
```

Interpretation: Apply a force of magnitude -100 in Y direction, ramping from 0 to full magnitude during step 2. The force is only active during step 2.

Example 2: Immediate force applied in multiple steps

```
@@ForceY: -200.0
@@ForceLoadY: LoadType Immediate Step 1
@@ForcePropagateStepsY: 1 2 3
```

Interpretation: Apply a force of magnitude -200 in Y direction instantly at the start of step 1, and maintain this force throughout steps 1, 2, and 3.

Example 3: Sinusoidal oscillating force

```
@@ForceY: -500.0
@@ForceLoadY: LoadType Sinusoidal Frequency 2.0 PhaseLag 0.0 Step 3
@@ForcePropagateStepsY: 3
```

Interpretation: Apply a sinusoidal force in Y direction with baseline magnitude -500, oscillating at frequency 2.0 (cycles per unit time) with no phase lag. The force varies as $-500 \cdot \sin(2\pi \cdot 2.0 \cdot t)$ during step 3.

Example 4: Damped sinusoidal force

```
@@ForceY: -300.0
@@ForceLoadY: LoadType DampedSinusoidal Frequency 1.0 DampingFactor 0.1 Step
4
@@ForcePropagateStepsY: 4
```

Interpretation: Apply a damped sinusoidal force in Y direction with baseline magnitude -300, oscillating at frequency 1.0 with exponential decay controlled by damping factor 0.1. The force amplitude decays over time during step 4.

Example 5: Multi-directional forces with different load types

```
@@ForceX: 50.0
@@ForceY: -200.0
@@ForceLoadX: LoadType Ramp Step 1
@@ForceLoadY: LoadType Immediate Step 1
@@ForcePropagateStepsX: 1 2
@@ForcePropagateStepsY: 1
```

Interpretation: Apply a ramped force of magnitude 50 in X direction during steps 1 and 2, while simultaneously applying an immediate force of magnitude -200 in Y direction only during step 1. This creates a combined loading scenario with different time histories in each direction.

1.4.7 Force Regulation Parameters (Static and Consolidation Modes Only)

For `SimulationMode: Static` or `SimulationMode: Consolidation`, the solver iteratively adjusts rigid body displacement to match prescribed forces within tolerance:

Parameter	Type	Description	Default
`@@ForceTolerance`	Double	Convergence tolerance for force residual	1.0
`@@ForceRegMax Iters`	Integer	Maximum regulation iterations per step	3
`@@ForceRegGain`	Double	Displacement adjustment gain	10^{-6}

Parameter	Type	Description	Default

Regulation algorithm (static only):

1. Solve equilibrium with current rigid body position
2. Compute force residual: $r = F_{\text{prescribed}} - R_{\text{reaction}}$
3. If $|r| >$ tolerance, adjust displacement: $\Delta u \mathrel{+}= \text{gain} \cdot r$
4. Revert state and re-solve with adjusted displacement
5. Repeat until $|r| \leq$ tolerance or max iterations reached

---

1.5 Static vs Dynamic Force-Controlled Motion

1.5.1 Static and Consolidation Modes (`SimulationMode: Static` or `SimulationMode: Consolidation`)

Note: For rigid body force-controlled motion, Static and Consolidation modes behave identically. Both use iterative force regulation to achieve equilibrium.

- **Equilibrium enforcement**: Iterative adjustment ensures $F_{\text{prescribed}} = R_{\text{reaction}}$ within tolerance
- **No dynamics**: Mass and damping are not used for rigid body motion; quasi-static equilibrium is assumed
- **Force regulation**: Uses @@`ForceTolerance`, @@`ForceRegMaxIters`, and @@`ForceRegGain` to iteratively adjust rigid body position
- **Torque fields ignored**: @@`Torque*` is not used to drive rotation in Static/Consolidation. To prescribe rotation in these modes, use @@`AngDispEq` / @@`AngVelEq` / @@`AngAccelEq`.
- **Use case**: Foundation loading, retaining wall construction, static push tests, consolidation analyses

Workflow: 1. Apply prescribed force (time-scaled via LoadApplication) 2. Solve equilibrium 3. Check force residual per axis 4. If out of tolerance, adjust rigid body displacement and re-solve 5. Converge when all active axes satisfy $|F - R| \leq$ tolerance

Difference between Static and Consolidation: While both modes use the same force regulation for rigid bodies, Consolidation mode includes flow coupling terms (damping matrices) in the global system for coupled/fully coupled analyses. For rigid body motion specifically, they are equivalent.

1.5.2 Dynamic Mode (`SimulationMode: Dynamic`)

- **Direct integration**: No iterative regulation; rigid body motion follows ODE integration
- **Mass and damping**: Uses rigid body @@`Mass` and @@`DampingLinear`
- **Torque-controlled rotation (optional)**: Uses @@`TorqueX/Y/Z` with rigid-body inertia (@@`InertiaDiag` / @@`InertiaTensor`) and @@`DampingAngular`

- **Force regulation parameters ignored**: @@ForceTolerance, @@ForceRegMaxIters, and @@ForceRegGain are not used in dynamic mode
- **Use case**: Impact loading, seismic analysis, pile driving, transient soil-structure interaction

Governing equation (per axis):

$$m\ddot{u} + c\dot{u} = F_{\text{prescribed}}(t) - R_{\text{reaction}}$$

Integration scheme (semi-implicit Euler):

$$\begin{aligned} a^{n+1} &= \frac{r^{n+1} - cv^n}{m} \\ v^{n+1} &= v^n + a^{n+1}\Delta t \\ u^{n+1} &= u^n + v^{n+1}\Delta t \end{aligned}$$

where $r = F_{\text{prescribed}} - R$.

Rotational dynamics (optional):

If any @@TorqueX/Y/Z is active in the current step, FALCON integrates:

$$\mathbf{I}\,\dot{\boldsymbol{\omega}} + c_\theta\,\boldsymbol{\omega} = \mathbf{T}_{\text{prescribed}}(t) - \mathbf{M}_{\text{reaction}}$$

- $\mathbf{T}_{\text{prescribed}}(t)$ comes from @@Torque* scaled by @@TorqueLoad* (Ramp/Sinusoidal/DampedSinusoidal/Immediate/Tabular), using the same load syntax as forces.
- $\mathbf{M}_{\text{reaction}}$ is computed by summing moments of restrained-DOF reaction forces about a reference point: $\mathbf{M}_{\text{reaction}} = \sum_i (\mathbf{x}_i - \mathbf{x}_{ref}) \times \mathbf{f}_i$, where $\mathbf{f}_i$ is the reaction force vector at rigid-body node i and $\mathbf{x}_i$ is that node's **current** position.
- $\mathbf{x}_{ref}$ is @@ReferenceNodeID if provided, otherwise the rigid-body node centroid (current positions).
- $\mathbf{I}$ is the inertia tensor about $\mathbf{x}_{ref}$ (from @@InertiaDiag or @@InertiaTensor; if omitted, it is approximated from the rigid-body nodes and @@Mass).

Incremental rotation application:

FALCON integrates angular velocity using a semi-implicit Euler update and converts it into a small incremental rotation vector:

$$\begin{aligned} \boldsymbol{\alpha}^{n+1} &= \mathbf{I}^{-1}\left(\boldsymbol{\tau}^{n+1} - c_\theta \boldsymbol{\omega}^n\right), \quad \boldsymbol{\tau}^{n+1} = \mathbf{T}_{\text{prescribed}}(t^{n+1}) - \mathbf{M}_{\text{reaction}} \\ \boldsymbol{\omega}^{n+1} &= \boldsymbol{\omega}^n + \boldsymbol{\alpha}^{n+1}\Delta t \\ \Delta\boldsymbol{\theta} &= \boldsymbol{\omega}^{n+1}\Delta t \end{aligned}$$

That incremental rotation is applied as a rigid rotation about $\mathbf{x}_{ref}$ to the rigid-body nodes on the next increment:

$$\mathbf{x}^{n+1} = \mathbf{x}_{ref} + \mathbf{R}(\Delta\boldsymbol{\theta})\left(\mathbf{x}^n - \mathbf{x}_{ref}\right)$$

where $\mathbf{R}(\Delta\boldsymbol{\theta})$ is the axis-angle rotation associated with the rotation vector $\Delta\boldsymbol{\theta}$.

Workflow:

1. Apply prescribed force (time-scaled via LoadApplication)
2. Solve equilibrium with current rigid body configuration
3. Compute the rigid-body reaction R by summing restrained-DOF reaction forces over the rigid-body nodes
4. (If torques are active) compute $\mathbf{M}_{\text{reaction}}$ about $\mathbf{x}_{ref}$ and integrate $\omega \rightarrow \Delta\theta_{\text{pending}}$
5. Integrate rigid body translation: compute a, v, u_{pending}
6. Apply pending translation and rotation increments at the start of the next increment

1.5.3 Multiple Equilibrium Solves Within One Increment (What to Expect)

Depending on your setup, FALCON may perform multiple full equilibrium solves at the same step time (same increment) before accepting the increment:

- **Newton iterations** happen inside one equilibrium solve (these are not separate "re-solves"; they are the nonlinear iterations of the same solve).
- **Force regulation (Static/Consolidation)** can trigger **additional equilibrium re-solves**: after a converged solve, FALCON checks force residuals for active `@@Force*` axes and, if needed, perturbs the rigid-body command and re-solves up to `@@ForceRegMaxIters`.
- **Contact augmented Lagrangian (if enabled)** can also cause **outer restarts** that repeat the equilibrium solve at the same step time.

When both translation force-control and torque-control are active in **Dynamic** mode, the converged equilibrium state supplies **both** the reaction forces and reaction moments used by the rigid-body ODE update in that increment.

1.5.4 Directional Gating

To prevent spurious motion accumulation in non-forced directions:

- Motion is integrated **only on axes with an active, flagged force**
- Axes without `@@ForceX/Y/Z` specified, or with forces inactive at the current step, have their velocity, acceleration, and pending displacement zeroed each step
- Similarly, rotational dynamics are integrated only when an active `@@TorqueX/Y/Z` is present; otherwise angular state is reset.

1.6 Complete Examples

1.6.1 Example Index

2D (Plane Strain / Axisymmetric): - **2D-1**: Prescribed settlement with `@@DispEqY` - **2D-2**: Static force-controlled vertical load (regulation loop) - **2D-3**: Dynamic lateral harmonic force (mass/damping)

3D: - **3D-1**: Prescribed translation + prescribed rotation (`@@AngDispEq`) - **3D-2**: Dynamic torque-controlled twist (`@@TorqueZ` + inertia) - **3D-3**: Mixed translation + rotation with tabular

motion - **3D-4**: Dynamic heave (force-controlled) + spin (torque-controlled) together

Notes: - In 2D analyses, only in-plane translations (`DisX/DisY`) are active. Rotation is meaningful only about the out-of-plane axis (typically `@@RotationAxis: 0 0 1`). For torque-controlled rotation in 2D, use `@@TorqueZ` with `@@InertiaDiag` providing `Izz`. - Avoid prescribing multiple different rotation centers for the same rigid body in the same step/sub-step.

1.6.2 2D-1: Prescribed Settlement (Absolute Displacement)

```
% RigidBodies
@RigidBody 1
@@NodeIDs: 100 101 102 103 104 105
@@Mass: 5000.0
@@DampingLinear: 0.0
%%%

% RigidMotionConstraints
@RigidMotionConstraint 1
@@MotionType: Translation
@@RigidBodyID: 1
@@DispEqY: a=0 b=-0.01
@@StepIds: 1
%%%
```

Behavior:

- `@@DispEqY` is interpreted as an absolute target displacement $u_y(t)$. The solver applies the incremental change each substep: $\Delta u = u(t) - u(t_{prev})$.

1.6.3 2D-2: Static Force-Controlled Load (Regulation)

```
% SimulationStep
@Step 2
@@SimulationMode: Static
@@StepTime: 1.0
...
%%%

% RigidBodies
@RigidBody 1
@@NodeIDs: 100 101 102 103 104 105
@@Mass: 5000.0
@@DampingLinear: 0.0
%%%
```

```
% RigidMotionConstraints
@RigidMotionConstraint 2
@@MotionType: Translation
@@RigidBodyID: 1
@@ForceY: -100.0
@@ForceLoadY: LoadType Ramp Step 2
@@ForcePropagateStepsY: 2
@@ForceTolerance: 1.0
@@ForceRegMaxIters: 5
@@ForceRegGain: 1e-6
@@StepIds: 2
%%%

% MasterForceContact
@Id 1
@ContactID 1
@Steps 2
@OutputFile /path/to/static_forces.csv
@OutputFreq 1
%%%
```

Behavior:

- In static mode, the solver iteratively adjusts the rigid-body displacement until $|F_{prescribed} - R| \leq$ tolerance on each active axis.

1.6.4 2D-3: Dynamic Lateral Harmonic Force (Mass + Damping)

```
% SimulationStep
@Step 3
@@SimulationMode: Dynamic
@@StepTime: 2.0
...
%%%

% RigidBodies
@RigidBody 1
@@NodeIDs: 100 101 102 103 104 105
@@Mass: 10000.0
@@DampingLinear: 100.0
%%%

% RigidMotionConstraints
```

```
@RigidMotionConstraint 3
@@MotionType: Translation
@@RigidBodyID: 1
@@ForceX: 500.0
@@ForceLoadX: LoadType Sinusoidal Frequency 2.0 PhaseLag 0.0 Step 3
@@ForcePropagateStepsX: 3
@@StepIds: 3
%%%
```

1.6.5 3D-1: Prescribed Heave + Prescribed Rotation (Absolute Angle)

```
% RigidBodies
@RigidBody 2
@@NodeIDs: 200 201 202 203 204 205
@@Mass: 8000.0
%%%

% RigidMotionConstraints
@RigidMotionConstraint 10
@@MotionType: Mixed
@@RigidBodyID: 2
@@StepIds: 4

@@DispEqZ: a=0 b=-0.002

@@RotationAxis: 0 0 1
@@RotationCenter: 0 0 0
@@AngDispEq: a=0 b=0.05
%%%
```

1.6.6 3D-2: Dynamic Torque-Controlled Twist (Inertia + Angular Damping)

```
% SimulationStep
@Step 5
@@SimulationMode: Dynamic
@@StepTime: 1.0
...
%%%

% RigidBodies
@RigidBody 3
@@NodeIDs: 300 301 302 303 304 305
```

```
@@ReferenceNodeID: 300
@@Mass: 5000.0
@@DampingLinear: 0.0
@@DampingAngular: 10.0
@@InertiaDiag: 1e6 1e6 5e6
%%%

% RigidMotionConstraints
@RigidMotionConstraint 20
@@MotionType: Rotation
@@RigidBodyID: 3
@@StepIds: 5

@@TorqueZ: 2000.0
@@TorqueLoadZ: LoadType Ramp Step 5
@@TorquePropagateStepsZ: 5
%%%
```

1.6.7 3D-3: Mixed Motion with Tabular Translation

```
% RigidMotionConstraints
@RigidMotionConstraint 30
@@MotionType: Mixed
@@RigidBodyID: 2
@@StepIds: 6

@@DispEqX: TabularData file=rb_dx.tbl
@@DispEqY: a=0 b=0
@@DispEqZ: a=0 b=0

@@RotationAxis: 0 1 0
@@RotationCenter: 0 0 0
@@AngVelEq: a=0 b=0.2
%%%
```

1.6.8 3D-4: Dynamic Vertical Force-Control + Torque-Controlled Rotation (Simultaneous)

This pattern answers the common case “force-controlled vertical motion and rotational dynamics at the same time”: in `SimulationMode: Dynamic`, translation and rotation are integrated together using the same converged reaction state.

```
% SimulationStep
@Step 7
@@SimulationMode: Dynamic
@@StepTime: 0.2
...
%%%

% RigidBodies
@RigidBody 4
@@NodeIDs: 400 401 402 403 404 405
@@ReferenceNodeID: 400
@@Mass: 2000.0
@@DampingLinear: 20.0
@@DampingAngular: 5.0
@@InertiaDiag: 2e5 2e5 8e5
%%%

% RigidMotionConstraints
@RigidMotionConstraint 40
@@MotionType: Mixed
@@RigidBodyID: 4
@@StepIds: 7

@@ForceY: -1.0e5
@@ForceLoadY: LoadType Ramp Step 7
@@ForcePropagateStepsY: 7

@@RotationAxis: 0 0 1
@@TorqueZ: 5.0e3
@@TorqueLoadZ: LoadType Ramp Step 7
@@TorquePropagateStepsZ: 7
%%%
```

1.7 CSV Output Integration

When using `% MasterForceContact` with rigid bodies having force-controlled motion, the CSV output provides comprehensive diagnostics:

Header: `StepID,SimulationTime,RbRx,RbRy,RbRz,RbVx,RbVy,RbVz,RbAx,RbAy,RbAz,RbMx,RbMy,RbMz,RbOmegaX,RbOmegaY,RbOmegaZ,RbAlphaX,RbAlphaY,RbAlphaZ,PrescribedFx,PrescribedFy,PrescribedFz,PrescribedMx,PrescribedMy,PrescribedMz,RbUx,RbUy,RbUz`

Column	Source
StepID	Current simulation step ID
Simulation Time	Global time at the output point
RbRx/RbRy/RbRz	Reaction forces on the rigid body (summed over restrained-DOF reactions on rigid-body nodes)
RbVx/RbVy/RbVz	Translational velocity (dynamic integration in SimulationMode: Dynamic; otherwise typically 0 unless prescribed velocity equations are used)
RbAx/RbAy/RbAz	Translational acceleration (dynamic integration in SimulationMode: Dynamic; otherwise typically 0 unless prescribed acceleration equations are used)
RbMx/RbMy/RbMz	Reaction moments about the rigid-body reference point (see note below)
RbOmegaX/RbOmegaY/RbOmegaZ	Angular velocity vector (torque-driven dynamics; may also reflect prescribed angular-rate equations)
RbAlphaX/RbAlphaY/RbAlphaZ	Angular acceleration vector (torque-driven dynamics; may also reflect prescribed angular-acceleration equations)
PrescribedFx/Fy/Fz	Time-scaled prescribed forces from @@Force* and @@ForceLoad*
PrescribedMx/My/Mz	Time-scaled prescribed torques from @@Torque* and @@TorqueLoad*
RbUx/RbUy/RbUz	Representative rigid-body translation from the rigid-body reference node (or first rigid-body node if no reference node is set)

This enables direct comparison of prescribed vs reaction forces for validation and debugging.

Notes: - Moments RbMx/RbMy/RbMz (and prescribed torques) are taken about @@ ReferenceNodeID if provided; otherwise about the current centroid of the rigid-body nodes. - If you are not using torque-controlled rotation, the torque/moment and angular-rate columns will typically remain zero.

1.8 Validation Rules

1.8.1 Rigid Body Definition

1. **Node IDs**: All nodes in @@NodeIDs must exist in the mesh.
2. **Mass**: Must be positive (> 0).

3. **Damping**: Must be non-negative (≥ 0).
4. **Unique ID**: Each rigid body must have a unique ID.

1.8.2 Rigid Motion Constraint

1. **RigidBodyID**: Must reference an existing rigid body.
2. **StepIds**: All step IDs must exist in simulation steps.
3. **Force propagate steps**: If specified, must be valid step IDs.
4. **Equation conflict**: Cannot specify multiple equation types (Disp/Vel/Accel) for the same axis.
5. **Force regulation parameters**: Only meaningful in static and consolidation modes; ignored in dynamic mode.

1.8.3 Force-Controlled Motion

1. **Directional consistency**: Motion is integrated/regulated only on axes where `@@ForceX/Y/Z` is specified and active for the current step (via `@@StepIds` / `@@ForcePropagateSteps*`).
2. **Load application**: `@@ForceLoadX/Y/Z` syntax must match standard `LoadApplication` format (Ramp, Sinusoidal, DampedSinusoidal, Tabular).
3. **Static/Consolidation regulation**: `@@ForceTolerance`, `@@ForceRegMaxIters`, `@@ForceRegGain` are used only when `SimulationMode: Static` or `SimulationMode: Consolidation`. These parameters are ignored in dynamic mode.
4. **Multiple constraints**: If multiple `@RigidMotionConstraint` blocks target the same rigid body and prescribe forces/torques on the same axis in the same step, their contributions add. Avoid duplicate prescriptions unless intentional.

1.9 Troubleshooting

1.9.1 Static regulation not converging

- **Cause**: Gain too large or too small; max iterations insufficient; stiff system.
- **Solution**: Adjust `@@ForceRegGain` (try 10^{-7} to 10^{-5}); increase `@@ForceRegMaxIters`; check penalty coefficients in contact.