AD FALCON API Manual

MPCConstraints (Multi-Point Constraints)

Javad Ghorbani

March 26, 2026

Contents

1 MPCConstraints (Multi-Point Constraints)

`% MPCConstraints` lets you constrain nodal motion along an arbitrary **line direction** without defining contact surfaces. This is useful for "slider" boundary conditions such as *move only along an inclined line (2D)* or *move only along a given direction (3D)*.

FALCON enforces these constraints using an **Augmented Lagrangian (AL)** formulation (penalty + multipliers). The constraints can be activated/deactivated during the run using the existing staged boundary sections `% RestrainDOFs` / `% ReleaseBoundary`.

1.1 When to use

- You want a node (or a set of nodes) to **slide along** a prescribed direction.
- You want to avoid contact definitions but still restrict motion in a smooth, solver-friendly way.
- You need a constraint that works in **Implicit**, **Explicit**, and **IMEX** time integration.

1.2 Constraint meaning (SlideAlongDirection)

1.2.1 2D (plane strain / axisymmetric in the XY plane)

Given the unit direction vector `t = (tx, ty, 0)`, FALCON builds the in-plane unit normal:

- `n1 = (-ty, tx, 0)`

  Then it enforces (per node):

- `n1 · u = RHS1`

  This removes motion normal to the direction, leaving the node free to slide along `t`.

1.2.2 3D

Given the unit direction vector `t`, FALCON builds two unit normals `n1, n2` such that:

- `n1 ⊥ t`, `n2 ⊥ t`, and `n1 ⊥ n2`

  Then it enforces (per node):

- `n1 · u = RHS1`
- `n2 · u = RHS2`

  This removes the two transverse components, leaving only motion along `t`.
  Notes:

- If you provide only @@RHS (single value) in 3D, FALCON sets `RHS1 = RHS2 = RHS`.
- In 2D, @@RHS2 is ignored (a warning is printed).

1.3 Syntax

```
% MPCConstraints
@MPCConstraint <id>
@@Kind SlideAlongDirection
@@Method AL                          # optional (default:
AugmentedLagrangian)
@@NodeIDs: <list>                    # required
@@Direction: <dx> <dy> [dz]          # required
@@RHS <val>                          # optional (default: 0; in 3D applies to
both unless RHS1/RHS2 given)
@@RHS1 <val>                         # optional
@@RHS2 <val>                         # optional (3D only)
@@Penalty <rho>                      # optional (default: 1e8)
@@ALMaxIterations <n>                # optional (default: 0)
@@ALUpdateTolerance <tol>            # optional (default: 0)
@@ActiveSteps <All|list>             # optional (default: All)
@@Enabled <Yes|No>                   # optional (default: Yes)
%%%
```

Multiple `@MPCConstraint ...` blocks can appear inside the same `% MPCConstraints` section.

1.4 Settings

Setting	Required	Type	Default	Description
`@MPC Constraint <id>`	Yes	int	—	Unique constraint identifier. Used for activation/deactivation during analysis.
`@@Kind`	Yes	string	—	Currently supported: `SlideAlong Direction`.
`@@Method`	No	string	`Augmented Lagrangian`	Accepted values: `Augmented Lagrangian`, `Augmented_ Lagrangian`, `AL`.
`@@NodeIDs`	Yes	int list	—	Node IDs that the constraint applies to. Ranges are allowed (`1-10`).

Setting	Required	Type	Default	Description
@@Direction	Yes	2 or 3 floats	—	Direction vector. In 2D, only (dx,dy) is used; dz is ignored.
@@RHS	No	float	0.0	Convenience setter. Sets RHS1 (and RHS2 in 3D).
@@RHS1	No	float	0.0	Right-hand side for equation 1 (n1·u = RHS1).
@@RHS2	No	float	0.0	Right-hand side for equation 2 (n2·u = RHS2) in 3D.
@@Penalty	No	float	1e8	Augmented Lagrangian penalty parameter (rho). Larger values enforce more strongly but can make Explicit/IMEX time steps more restrictive.
@@ALMax Iterations	No	int	0	For Implicit time integration, enables AL "outer" updates when > 0. In Explicit/IMEX, multipliers update once per accepted substep (no outer iterations).
@@ALUpdate Tolerance	No	float	0.0	Threshold used to decide whether AL updates "changed" enough to trigger outer restarts (Implicit).
@@Active Steps	No	All or int list	All	Apply only on selected step IDs.
@@Enabled / @@Initially Active	No	bool	Yes	Whether the constraint starts active at the beginning of the run.

1.5 Examples

1.5.1 Example 1 — 2D inclined slider (move only along a line)

```
% MPCConstraints
@MPCConstraint 10
@@Kind SlideAlongDirection
@@NodeIDs: 100 101 102
@@Direction: 0.70710678 0.70710678   # 45° in XY
@@RHS 0.0
@@Penalty 1e8
%%%
```

1.5.2 Example 2 — 3D slider along an arbitrary direction

```
% MPCConstraints
@MPCConstraint 20
@@Kind SlideAlongDirection
@@NodeIDs: 500-520
@@Direction: 1.0 1.0 0.0
@@RHS 0.0
@@Penalty 5e7
%%%
```

1.5.3 Example 3 — Start disabled, then activate at a given step

```
% MPCConstraints
@MPCConstraint 30
@@Kind SlideAlongDirection
@@NodeIDs: 200
@@Direction: 1.0 0.0
@@Enabled No
@@Penalty 1e8
%%%

% RestrainDOFs
Step: 5
Constraint: 30
%%%
```

1.5.4 Example 4 — Deactivate during the run

```
% ReleaseBoundary
Step: 10
Constraint: 30
%%%
```

1.6 Restart & checkpoint notes

- MPCConstraints are serialized in checkpoint files starting from **checkpoint format v10**.
- If you restart from an older checkpoint, MPCConstraints will use the input-file definition, and AL multipliers/state will start from zero.