AD FALCON API Manual

Stress Boundary Conditions

Javad Ghorbani

March 26, 2026

Contents

1 Stress Boundary Conditions

This guide explains the **Stress Boundary** section of the input file, including syntax, supported types, required fields, validation rules, error messages, and examples.

1.1 Syntax

1.1.1 Section header

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

```
% StressBoundary
% Stress Boundary
% stress_boundary
```

1.1.2 Section format

Each stress boundary definition is a small block. The exact block layout depends on 2D vs 3D:

2D (edge-based):

```
@Pressure: ElemId <ElementID> Edgenodes <n1> <n2> ... <nK>
NormalPressures <p1> <p2> ... <pK>
TangentialPressures <t1> <t2> ... <tK>
LoadType <...> Step <StartStepId> ...
[TabularData <t1> <m1>; <t2> <m2>; ...]   # only for LoadType Tabular
Propagate: Yes | FinalStep <N>
```

3D (face-based):

```
@SurfacePressure: ElemId <ElementID> Facenodes <n1> <n2> ... <nK>
NormalPressures <p1> <p2> ... <pK>
[TangentialTractionX <tx1> ... <txK>
 TangentialTractionY <ty1> ... <tyK>
 TangentialTractionZ <tz1> ... <tzK]      # optional; if present, all 3 must
be provided
LoadType <...> Step <StartStepId> ...
[TabularData <t1> <m1>; <t2> <m2>; ...]   # only for LoadType Tabular
Propagate: Yes | FinalStep <N>
```

Notes:

- Marker keywords (`Pressure`, `Traction`, `SurfacePressure`, `SurfaceTraction`) are matched case-insensitively and tolerate extra leading `@` and a trailing `:` (recommended style: `@Pressure:`). For 3D, `@Pressure/@Traction` are accepted too; the solver decides 2D vs 3D from the element type.
- The labels `ElemId`, `Edgenodes`, `Facenodes` are case-insensitive (recommended style: `ElemId`, `Edgenodes`, `Facenodes`).
- Node ID lists support spaces/commas/semicolons and ranges like `10-20` or `10:20`.
- Pressure/traction values can be separated by spaces and/or commas; you must provide one value per edge/face node.
- For 3D, a **normal-only** boundary is allowed by omitting the tangential vector; in that case the `LoadType ...` line comes immediately after `NormalPressures ...`.

1.1.3 Multiple stress-boundary definitions

- Define multiple stress boundaries by repeating the boundary block format inside the same `% StressBoundary` section.
- Each boundary definition starts with a marker line (e.g., `@Pressure: ...` / `@Surface Pressure: ...`) and consumes the following required lines; the next marker line starts the next boundary definition.

1.2 Section Placement

Begin the section with:

```
% Stress Boundary
```

End with:

```
%%%
```

The section continues until a line `%%%` or the next line starting with `%` (the next section header) is encountered. In normal use you close the block with `%%%` and then start the next section.

1.3 Input Format

Each stress boundary definition starts with a **marker line** and is followed by related parameters. The syntax differs for 2D (edges) and 3D (faces):

1.3.1 2D Format (Element Edges)

```
@<Type>: ElemId <ElementID> Edgenodes <Node1> <Node2> ... <NodeN>
NormalPressures <P1> <P2> ... <PN>
TangentialPressures <T1> <T2> ... <TN>
LoadType <LoadTypeName> Step <StepId> [additional parameters]
[TabularData <time1> <val1>; <time2> <val2>; ...]   # only for Tabular
Propagate: <Option>
```

1.3.2 3D Format (Element Faces)

```
@Surface<Type>: ElemId <ElementID> Facenodes <Node1> <Node2> ... <NodeN>
NormalPressures <P1> <P2> ... <PN>
TangentialTractionX <TX1> <TX2> ... <TXN>
TangentialTractionY <TY1> <TY2> ... <TYN>
TangentialTractionZ <TZ1> <TZ2> ... <TZN>
LoadType <LoadTypeName> Step <StepId> [additional parameters]
[TabularData <time1> <val1>; <time2> <val2>; ...]   # only for Tabular
Propagate: <Option>
```

- **Type marker** (2D and 3D)
  FALCON accepts all of `Pressure`, `Traction`, `SurfacePressure`, `SurfaceTraction` as marker names. Currently these markers are **aliases** (same parsing and same force computation); the solver determines whether the boundary is 2D vs 3D from the element referenced by `ElemId`. Use `SurfacePressure/SurfaceTraction` in 3D for clarity.

- **ElemId**
  Integer ID of the element.

- **Edgenodes** (2D only)
  List of node IDs defining the element edge where the boundary is applied.
  - Must match an existing element's `edgeNodeCount`.
  - All nodes must belong to that element. - Node order matters: list nodes along the edge in the same **counterclockwise** direction as the element boundary so the solver's outward normal convention is consistent (negative pressure = compression). For 3-node edges, the midside node must be between the two corner nodes.

- **Facenodes** (3D only)
  List of node IDs defining the element face where the boundary is applied.
  - For the currently supported 3D element (10-node tetrahedron `T10` / `N10`), you must provide **exactly 6** face nodes (3 corner nodes + 3 midside nodes).
  - All nodes must belong to that element. - For **T10 tetrahedra**: the 6 nodes may be listed in any order. The solver identifies which element face they belong to (by set membership),

reorders them to the canonical 6-node triangular face order, and flips the orientation if needed so the computed normal points outward (away from the opposite tetrahedron corner node).

- **NormalPressures**
  One pressure value per edge/face node (sign convention is negative in compression).
- **TangentialPressures** (2D only)
  One tangential pressure value per edge node.
- **TangentialTractionX / TangentialTractionY / TangentialTractionZ** (3D only)
  Nodal values of a **global traction vector field** on the selected face (one value per face node). Internally, the solver: (1) interpolates the nodal vectors to face Gauss points, then (2) removes the normal component at each Gauss point to enforce tangential-only loading: `t_tan = t - (t·n)n`, where `n` is the outward unit normal. The applied shear traction is the projected vector `t_tan` (dimensions: force per area); nodal forces are obtained by integrating `t_tan * dA` over the face.

  Legacy labels `TangentialPressuresX/Y/Z` are accepted and mean exactly the same as `TangentialTractionX/Y/Z`: they are global X/Y/Z components of the traction vector, not tensor shear components like τ`_xy`. There are currently no `...XY/XZ/YZ` keyword aliases.

  How to use X/Y/Z: X, Y, Z are the **global model axes** (the same axes used for node coordinates and displacements). You are specifying the components of a traction vector `t = (Tx, Ty, Tz)` in that fixed global basis. FALCON then removes the component normal to the face, so only the in-surface part contributes.

  Examples (showing the effect of the tangential projection):

  - **Horizontal face (normal ≈ +Z), apply +X shear of 100:**
    Provide `TangentialTractionX 100 100 100 100 100 100` (and `TangentialTractionY/Z 0 ...`). Since +X is tangential to a Z-normal face, the applied shear traction is ≈ `(100, 0, 0)`.
  - **Vertical face (normal ≈ +X), same input:**
    If you provide `TangentialTractionX 100 ...` on an X-normal face, that vector is mostly normal to the face and is removed by the projection, so the applied shear traction is ≈ 0. To apply shear on this face you typically use `TangentialTractionY` and/or `TangentialTractionZ` instead.

- **LoadType**
  How the boundary load varies in time on the selected step (`Step <StepId>`): - `Immediate` — applied instantly at the start of the step and then held constant (subject to the propagation rules in loadtypes.md). - `Ramp` — linearly increases from 0 to full value over the duration of the step. - `Sinusoidal` — requires `Frequency <value>` and optionally `Phase Lag <degrees>`. - `DampedSinusoidal` — requires `Frequency <value>`; accepts `Damping`

Factor <value> (defaults to 0) and PhaseLag <degrees> (defaults to 0). - `Tabular` — time history is provided by a separate `TabularData` line (see below).

For all of these types the **amplitudes** come from the `NormalPressures` / `Tangential Pressures` values; there is no separate `Amplitude` keyword in the stress-boundary block.

- **TabularData** (only for `Tabular` LoadType)
  Semicolon-separated time and magnitude pairs: `plaintext TabularData 0 0; 1 50; 2 100;` Pairs may also be separated purely by whitespace (commas are not treated as separators).

- **Propagate**
  - `Yes` — the boundary load is active from its start step through the final simulation step.
  - FinalStep <StepNumber> — the load is active only from its start step up to <Step Number>, after which it is ramped down to zero over the next step (see loadtypes.md for details). - For **Tabular** load type you must use `Propagate: Yes` (tabular loads always require propagation); `Propagate: FinalStep ...` is not allowed.

1.4 Validation & Errors

For a complete list of validation checks and error messages related to `% Stress Boundary`, see the error dictionary.

1.5 Examples

1.5.1 Pressure Boundary, Ramp Load

```
% Stress Boundary
@Pressure: ElemId 367 Edgenodes 815 814 813
NormalPressures -100 -100 -100
TangentialPressures 0 0 0
LoadType Ramp Step 1
Propagate: FinalStep 1
%%%
```

1.5.2 Pressure Boundary, Sinusoidal Load (Basic)

```
% Stress Boundary
@Pressure: ElemId 367 Edgenodes 815 814 813
NormalPressures -100 -100 -100
TangentialPressures 0 0 0
LoadType Sinusoidal Frequency 1e-5 PhaseLag 0 Step 1
Propagate: Yes
%%%
```

1.5.3 Pressure Boundary, Sinusoidal Load (Without Phase Lag)

When `PhaseLag` is omitted, it defaults to 0:

```
% Stress Boundary
@Pressure: ElemId 400 Edgenodes 900 901
NormalPressures -50 -50
TangentialPressures 0 0
LoadType Sinusoidal Frequency 0.5 Step 1
Propagate: Yes
%%%
```

1.5.4 Pressure Boundary, Sinusoidal Load (With Phase Lag)

```
% Stress Boundary
@Pressure: ElemId 450 Edgenodes 920 921 922
NormalPressures -75 -75 -75
TangentialPressures 0 0 0
LoadType Sinusoidal Frequency 2.0 PhaseLag 45.0 Step 1
Propagate: FinalStep 2
%%%
```

1.5.5 Traction Boundary, Sinusoidal Load (Variable Pressure)

When different nodes have different pressure values, the sinusoidal load applies proportionally:

```
% Stress Boundary
@Traction: ElemId 500 Edgenodes 1000 1001 1002
NormalPressures 50 75 50
TangentialPressures 10 15 10
```

```
LoadType Sinusoidal Frequency 1.0 PhaseLag 30.0 Step 2
Propagate: Yes
%%%
```

1.5.6 Pressure Boundary, Damped Sinusoidal Load

```
% Stress Boundary
@Pressure: ElemId 365 Edgenodes 813 812 811
NormalPressures -100 -100 -100
TangentialPressures 0 0 0
LoadType DampedSinusoidal Frequency 1e-5 PhaseLag 0 DampingFactor 1.e-4 Step
1
Propagate: Yes
%%%
```

1.5.7 Pressure Boundary, Damped Sinusoidal Load (With Phase Lag)

```
% Stress Boundary
@Pressure: ElemId 600 Edgenodes 1200 1201
NormalPressures -80 -80
TangentialPressures 0 0
LoadType DampedSinusoidal Frequency 0.1 PhaseLag 90.0 DampingFactor 0.05
Step 1
Propagate: FinalStep 3
%%%
```

1.5.8 Multiple Sinusoidal Boundaries

You can define multiple stress boundaries in the same section:

```
% Stress Boundary
@Pressure: ElemId 367 Edgenodes 815 814 813
NormalPressures -100 -100 -100
TangentialPressures 0 0 0
LoadType Sinusoidal Frequency 1e-5 PhaseLag 0 Step 1
Propagate: Yes
@Pressure: ElemId 365 Edgenodes 813 812 811
NormalPressures -100 -100 -100
TangentialPressures 0 0 0
LoadType Sinusoidal Frequency 1e-5 PhaseLag 0 Step 1
Propagate: Yes
```

```
%%%
```

1.5.9 Pressure Boundary, Tabular Load

```
% Stress Boundary
@Pressure: ElemId 450 Edgenodes 920 921 922
NormalPressures 20 20 20
TangentialPressures 0 0 0
LoadType Tabular Step 1
TabularData 0 0; 1 10; 2 20;
Propagate: Yes
%%%
```