AD FALCON API Manual

Parametric Studies and Uncertainty Quantification

Javad Ghorbani

March 26, 2026

Contents

1 Parametric Studies and Uncertainty Quantification **4**
1.1 Overview . . . 4
1.1.1 Applications . . . 4
1.2 Workflow . . . 5
1.2.1 Execution Process . . . 5
1.3 Configuration . . . 5
1.3.1 Input File Section . . . 5
1.3.2 Basic Settings . . . 6
1.4 Parallel Execution . . . 7
1.4.1 Enabling Parallel Mode . . . 7
1.4.2 How It Works . . . 7
1.4.3 Configuration Options . . . 7
1.4.4 Performance Expectations . . . 8
1.4.5 Example: Large-Scale Monte Carlo . . . 8
1.4.6 Thread Safety Notes . . . 9
1.5 Parameter Definitions . . . 10
1.5.1 Syntax . . . 10
1.5.2 Target Types . . . 10
1.5.3 Statistical Distributions . . . 10
1.6 Parameter Formats . . . 12
1.6.1 Format 1: Name=Value (UMAT Style) . . . 13
1.6.2 Format 2: Name Value (Standard Properties) . . . 13
1.6.3 Format 3: Named Placeholders (Initial State Variables) . . . 14
1.6.4 Format 4: $-Tagged Numeric Values (Any Section / Any Number) . . . 15
1.7 Quantities of Interest (QoIs) . . . 15
1.7.1 Syntax . . . 16
1.7.2 QoI Types . . . 16
1.8 Output Dataset . . . 18
1.8.1 CSV Structure . . . 18
1.8.2 Example Output . . . 18
1.9 Complete Example: Consolidation UQ Study . . . 19
1.9.1 Post-Processing: Uncertainty Quantification . . . 22
1.10 Error Handling . . . 24
1.10.1 Missing Parameter Error . . . 24
1.10.2 Failed Simulation Runs . . . 24
1.11 Best Practices . . . 24
1.12 Advanced Applications . . . 27
1.12.1 Latin Hypercube Sampling . . . 27
1.12.2 Multi-Fidelity Surrogate Training . . . 27

1.12.3 Sobol Sensitivity Analysis 27
1.12.4 Bayesian Calibration 28
1.13 Summary Reference Table 29
1.14 Keywords for Search 30

1 Parametric Studies and Uncertainty Quantification

1.1 Overview

FALCON provides comprehensive capabilities for **parametric analysis**, **uncertainty quantification (UQ)**, and **surrogate model training** through automated ensemble simulations. This feature enables systematic exploration of parameter spaces by running multiple finite element analyses with varied input parameters while recording user-defined quantities of interest.

1.1.1 Applications

!!! success "Monte Carlo Simulation" Generate large ensembles with randomly sampled parameters to:

```
- Assess probabilistic response of geotechnical structures
- Quantify uncertainty propagation from material properties to system behavior
- Estimate failure probabilities and reliability indices
- Perform risk assessments for engineering design
```

!!! success "Surrogate Model Training" Create high-quality datasets for machine learning and metamodeling:

```
- Train neural networks, Gaussian processes, polynomial chaos expansions
- Build fast-running surrogate models for real-time applications
- Enable optimization and inverse analysis with expensive FE models
- Support digital twins and model-predictive control
```

!!! success "Sensitivity Analysis" Identify influential parameters and understand system behavior:

```
- Global sensitivity analysis (Sobol indices, Morris screening)
- Parameter importance ranking
- Interaction effects between parameters
- Variance-based decomposition
```

!!! success "Model Calibration" Generate synthetic data for parameter identification:

```
- Bayesian inference and Markov Chain Monte Carlo (MCMC)
- Inverse analysis and data assimilation
- Model updating with experimental observations
- Confidence interval estimation
```

!!! success "Stochastic Analysis" Explore parameter uncertainties in soil properties:

```
- Material heterogeneity effects
- Load uncertainty and environmental factors
- Initial condition variations
```

1.2 Workflow

```
flowchart LR
    A[Define Parameters] --> B[Select Distributions]
    B --> C[Specify QoIs]
    C --> D[Configure Runs]
    D --> E[Automated Sampling]
    E --> F[Parallel Simulations]
    F --> G[Extract Results]
    G --> H[CSV Dataset]
    H --> I[ML Training / UQ Analysis]
```

1.2.1 Execution Process

For each simulation in the ensemble:

1. **Parameter Sampling**: Draw values from specified statistical distributions (uniform, normal, log-normal, etc.)
2. **Input Generation**: Create a modified input file with substituted parameter values
3. **FE Analysis**: Execute full finite element simulation with varied parameters
4. **QoI Extraction**: Record quantities of interest (displacements, stresses, reactions, state variables)
5. **Data Aggregation**: Write parameters and responses to structured CSV output
6. **Error Handling**: Track successful/failed runs and continue or terminate based on user settings

The result is a structured dataset where each row represents one simulation with its input parameters and corresponding outputs - ideal for uncertainty quantification, machine learning, or statistical analysis.

1.3 Configuration

1.3.1 Input File Section

All parametric study settings are defined in the `% SurrogateTraining` section of your FALCON input file:

```
% SurrogateTraining
@NumRuns: 100
@Seed: 42
@OutputFile: parametric_study.csv
@ContinueOnError: Yes
@SaveIntermediate: No
@Parallel: Yes
```

```
@MaxJobs: 8

# Define varying parameters
@Parameter: <Name> <Target> <Distribution> <Params>

# Define outputs to record
@QoI: <Name> <Type> <Arguments>
%%%
```

!!! tip "Tag Any Numeric Value With `$...`" You can tag *any* numeric value anywhere in the input file by inserting a `$name` token immediately before it. Tags are ignored by normal analysis parsing, but can be targeted in surrogate training using `@Parameter: $name ...`.

Supported tag forms (equivalent for analysis): - `$tag value` (recommended) - `$tag=value` - `$tag = value`

Examples: - Material/parameter line: `E $YM 2.0e8` - Node coordinates: `1 $x1 0.0 $y1 0.0`

1.3.2 Basic Settings

Setting	Description	Example	Default
`@NumRuns:`	Number of simulations in ensemble	`@NumRuns: 1000`	Required
`@Seed:`	Random seed for reproducible sampling	`@Seed: 42`	`0`
`@OutputFile:`	Path to output CSV dataset	`@OutputFile: uq_data.csv`	`surrogate_output.csv`
`@ContinueOnError:`	Continue if simulation fails	`@ContinueOnError: Yes`	`No`
`@SaveIntermediate:`	Save individual GiD results	`@SaveIntermediate: No`	`No`
`@Parallel:`	Enable parallel execution	`@Parallel: Yes`	`No`
`@MaxJobs:`	Maximum parallel workers	`@MaxJobs: 8`	`4`

!!! tip "Performance Tip" Set `@SaveIntermediate: No` for large parametric studies to avoid generating thousands of result files. The CSV output contains all essential data for post-processing.

!!! tip "Reproducibility" Always set `@Seed:` to ensure your Monte Carlo or Latin Hypercube sampling can be reproduced exactly. This is critical for scientific reproducibility and debugging.

1.4 Parallel Execution

FALCON supports **parallel execution** of parametric studies, dramatically reducing total computation time for large ensembles. This is especially valuable for:

- Monte Carlo simulations with thousands of samples
- Surrogate model training requiring large datasets
- Time-critical uncertainty quantification studies

1.4.1 Enabling Parallel Mode

Add these settings to your `% SurrogateTraining` section:

```
@Parallel: Yes
@MaxJobs: 8
```

1.4.2 How It Works

```
flowchart TB
    A[Parameter Sampling] --> B[Distribute Runs to Workers]
    B --> C1[Worker 1: Runs 1,5,9...]
    B --> C2[Worker 2: Runs 2,6,10...]
    B --> C3[Worker 3: Runs 3,7,11...]
    B --> C4[Worker 4: Runs 4,8,12...]
    C1 --> D[Thread-Safe CSV Append]
    C2 --> D
    C3 --> D
    C4 --> D
    D --> E[CSV Output (completion order)]
```

1. **Work Distribution**: Runs are distributed round-robin across available workers
2. **Independent Execution**: Each worker runs simulations in its own thread with isolated resources
3. **Thread-Safe Writes**: Results are appended with mutex-protected writes
4. **Crash-Safe Output**: Each completed run is appended to the CSV immediately (row order may follow completion order; use `run_id` to sort in post-processing)

1.4.3 Configuration Options

Setting	Description	Recommendation
`@Parallel: Yes`	Enable parallel execution	Enable for >10 runs
`@MaxJobs: N`	Maximum parallel workers	Typically = number of CPU cores

!!! tip "Optimal Worker Count" The optimal number of workers depends on:

- **CPU cores**: Start with `@MaxJobs: <num_cores>`
  - **Memory**: Each worker needs ~2-3× the memory of a single run
  - **Analysis complexity**: Memory-intensive analyses may need fewer workers

For a typical 8-core workstation: `@MaxJobs: 6  # Leave 2 cores for OS/overhead`
!!! warning "Memory Considerations" Parallel execution multiplies memory usage. For a model requiring 2 GB RAM:

- Sequential: ~2 GB total
  - `@MaxJobs: 4`: ~8 GB total
  - `@MaxJobs: 8`: ~16 GB total

Monitor memory usage and reduce `@MaxJobs` if swapping occurs.

1.4.4 Performance Expectations

Scenario	Sequential (100 runs × 30s)	Parallel (8 workers)	Speedup
Ideal	3,000 s (50 min)	375 s (6.25 min)	~8×
Typical	3,000 s (50 min)	500 s (8.3 min)	~6×
I/O bound	3,000 s (50 min)	750 s (12.5 min)	~4×

!!! note "Speedup Factors" Actual speedup depends on:

- **CPU-bound analyses**: Near-linear speedup (e.g., 8× with 8 workers)
  - **I/O-bound analyses**: Reduced speedup due to disk contention
  - **Memory-bound analyses**: May hit memory limits before CPU limits
  - **UMAT compilation**: First run per worker has compilation overhead

1.4.5 Example: Large-Scale Monte Carlo

```
% SurrogateTraining
@NumRuns: 10000
@Seed: 12345
@OutputFile: monte_carlo_10k.csv
@ContinueOnError: Yes
@SaveIntermediate: No

# Enable parallel execution for large study
@Parallel: Yes
@MaxJobs: 12  # On a 16-core machine

@Parameter: E Material LogNormal 11.5 0.3
@Parameter: nu Material TruncatedNormal 0.3 0.03 0.0 0.49
@Parameter: k Material LogNormal -16 2.0
```

```
@Parameter: c Material LogNormal 2.3 0.5

@QoI: Settlement DOF 100 DisY
@QoI: MaxStress MaxState StressYY
@QoI: FailureIndicator MinState MeanStress
%%%
```

Expected output:

```
[Surrogate] Running 10000 analyses (parallel: 12 workers)
[Surrogate] Output file: monte_carlo_10k.csv

[Progress] 100/10000 completed - Run 100: 2.34s
[Progress] 200/10000 completed - Run 205: 1.98s
...
[Progress] 10000/10000 completed - Run 9987: 2.12s

=== Surrogate Training Completed ===
Completed runs: 10000/10000
Successful: 9847, Failed: 153
Total training time: 4521.3 seconds
Average time per run: 0.45 seconds
Parallel speedup: 8.2x (estimated)
Results saved to: monte_carlo_10k.csv
```

1.4.6 Thread Safety Notes

FALCON ensures thread-safe execution by:

1. **Isolated meshes**: Each worker creates an independent mesh instance
2. **Unique temp files**: Each worker uses uniquely-named temporary files
3. **Mutex-protected writes**: CSV output is protected by mutexes
4. **Independent UMAT handlers**: Each worker compiles/loads its own UMAT instances

!!! info "Reproducibility with Parallel Execution" Results are **deterministic** regardless of execution order because:

- All parameter samples are pre-generated using the seed
  - Parameter sampling is deterministic, but CSV row order may differ in parallel mode (sort by `run_id` in post-processing if needed)
  - Each run ID always gets the same parameters

Sequential and parallel modes produce the same sampled parameters for each `run_id`, but parallel mode may write rows in a different order.

1.5 Parameter Definitions

Parameters are the **inputs** to your parametric study - the material properties, boundary conditions, loads, or initial conditions that you want to vary systematically or stochastically.

1.5.1 Syntax

```
@Parameter: <ParameterName> <Target> <Distribution> <DistributionParams>
```

1.5.2 Target Types

Specify what aspect of the model the parameter controls:

Target	Aliases	Controls	Example Use Case
`Material`	`Mat`, `Material Property`	Material properties (E, ν, c, ϕ, k, ...)	Uncertainty in Young's modulus
`StateVar`	`Custom`	Initial state variables (PW, e_0, σ_0)	Varying initial stress or pore pressure
`Boundary`	`BC`, `Boundary Condition`	Prescribed displacement values	Uncertain settlement magnitude
`Load`	`Force`	Applied force magnitudes	Variable surface load intensity

1.5.3 Statistical Distributions

Uniform Distribution Equally probable sampling across an interval - ideal for exploring parameter ranges without prior knowledge.

```
@Parameter: YoungsModulus Material Uniform 50000 200000
```

- **Aliases**: `uniform`
- **Parameters**: `min max`
- **Use Case**: Bounded uncertainty with no preferred central value, or systematic parameter space exploration
- **PDF**: Constant between min and max

  Example: Young's modulus between 50 MPa and 200 MPa with equal probability

---

Normal (Gaussian) Distribution Bell-shaped distribution for parameters with known mean and standard deviation.

```
@Parameter: PoissonRatio Material Normal 0.3 0.03
```

- **Aliases**: `normal`, `Gaussian`, `gaussian`
- **Parameters**: `mean stddev`
- **Use Case**: Well-characterized parameters with measurement uncertainty
- **PDF**: $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

Example: Poisson's ratio with mean 0.3 and 10% coefficient of variation (CoV = 0.03/0.3 = 10%)

!!! warning "Unbounded Distribution" Normal distribution extends to $\pm\infty$. For physically bounded parameters, use Truncated Normal instead.

Truncated Normal Distribution Normal distribution constrained within physically realistic bounds.

```
@Parameter: FrictionAngle Material TruncatedNormal 30.0 5.0 15.0 45.0
```

- **Aliases**: `truncated_normal`, `tn`
- **Parameters**: `mean stddev min max`
- **Use Case**: Parameters with known uncertainty but physical constraints (angles, ratios, positive quantities)
- **PDF**: Normal distribution renormalized over [min, max]

Example: Friction angle with mean 30°, std 5°, but constrained between 15° and 45°

!!! note "Mean Location" The mean must lie within [min, max]. The distribution is renormalized so probabilities outside bounds are redistributed inside.

Log-Normal Distribution For strictly positive parameters with right-skewed variability (common in permeability, cohesion).

```
@Parameter: Permeability Material LogNormal -16.0 2.0
```

- **Aliases**: `lognormal`, `ln`
- **Parameters**: `mean_of_ln(X) stddev_of_ln(X)`
- **Use Case**: Hydraulic conductivity, cohesion, or other parameters spanning orders of magnitude
- **PDF**: $f(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln x-\mu)^2}{2\sigma^2}}$

Example: If permeability k has median 10^{-8} m/s, use `LogNormal -18.42 2.0` (since $\ln(10^{-8}) \approx -18.42$)

!!! tip "Parameter Selection" If you know the **median** value m and want high variability:

- Mean of ln(X): `ln(m)`

- Std of ln(X): Typical values are 0.5 (moderate variability) to 2.0 (high variability)

Example: Median k = 10^{-7} m/s → `LogNormal -16.12 1.0`

Fixed Value Hold parameter constant across all runs (useful for control parameters or one-at-a-time sensitivity).

```
@Parameter: DilationAngle Material Fixed 0.0
```

- **Aliases**: `fixed`, `constant`
- **Parameters**: `value`
- **Use Case**: Non-varying baseline parameters, or when testing influence of other parameters
- **PDF**: Delta function at specified value

Example: Fix dilation angle at 0° for associative plasticity

Interval (Deterministic Sampling) Deterministic, equally-spaced values across an interval - for sensitivity analysis or design of experiments.

```
@Parameter: LoadMagnitude Load Interval 10000 50000
```

- **Aliases**: `interval`, `linspace`
- **Parameters**: `min max`
- **Use Case**: One-factor-at-a-time (OFAT) sensitivity, response surface methodology
- **Sampling**: Creates `NumRuns` evenly-spaced points

Example: For `@NumRuns: 5`, generates [10000, 20000, 30000, 40000, 50000]

!!! note "Deterministic Nature" This is **not** random sampling - it produces a deterministic grid. For multi-parameter studies with `Interval`, results follow a structured grid (factorial design).

1.6 Parameter Formats

FALCON automatically detects and substitutes parameters in four different input file formats.

1.6.1 Format 1: Name=Value (UMAT Style)

For user-material (UMAT) definitions with inline parameter syntax:

Input file:

```
% Materials
Mat1
@UMAT:/path/to/MohrCoulomb.cpp /path/to/MohrCoulomb.hpp Mechanical E=100000.0
nu=0.3 Cohesion=10.0 FrictionAngle=30.0
%%%
```

Parametric configuration:

```
% SurrogateTraining
@Parameter: E Material Uniform 50000 200000
@Parameter: Cohesion Material LogNormal 2.3 0.5
@Parameter: FrictionAngle Material TruncatedNormal 30 5 20 40
%%%
```

FALCON finds `E=100000.0` and replaces the value after `=` with the sampled value.

!!! warning "Exact Name Matching" Parameter names are **case-sensitive** and must match exactly: `E` ≠ `e`, `Cohesion` ≠ `cohesion`

1.6.2 Format 2: Name Value (Standard Properties)

For built-in material properties with space-separated syntax:

Input file:

```
% Materials
Mat1
@Mech: Elastic YoungsModulus 210000 PoissonRatio 0.3
@Perm: Constant k_sat 1.0e-7
%%%
```

Parametric configuration:

```
% SurrogateTraining
@Parameter: YoungsModulus Material Uniform 100000 300000
@Parameter: PoissonRatio Material TruncatedNormal 0.3 0.02 0.0 0.49
@Parameter: k_sat Material LogNormal -16.12 1.5
%%%
```

FALCON finds `YoungsModulus 210000` and replaces the value immediately following the parameter name.

1.6.3 Format 3: Named Placeholders (Initial State Variables)

For initial conditions with pattern-based assignments (height-based profiles, spatial distributions):

Input file:

```
% InitialStepAssignment
Step 1
@Assign Gauss
@PW: H 0.0 values PW_surface H 10.0 values PW_bottom
@Void: H 0.0 values e0_top H 10.0 values e0_bottom
@Stress: H 0.0 values sig_xx_top sig_yy_top 0 H 10.0 values sig_xx_bot sig_yy_bot
0
%%%
```

Parametric configuration:

```
% SurrogateTraining
# Pore water pressure uncertainty at surface and depth
@Parameter: PW_surface StateVar Uniform -5000 0
@Parameter: PW_bottom StateVar Uniform -100000 -50000

# Void ratio variability with depth
@Parameter: e0_top StateVar Normal 0.85 0.03
@Parameter: e0_bottom StateVar Normal 0.75 0.03

# Initial stress state uncertainty
@Parameter: sig_xx_top StateVar Fixed -1000
@Parameter: sig_yy_top StateVar Fixed -1000
@Parameter: sig_xx_bot StateVar Uniform -50000 -30000
@Parameter: sig_yy_bot StateVar Uniform -80000 -50000
%%%
```

FALCON replaces `values <PlaceholderName>` with sampled numeric values.

!!! tip "Mixing Fixed and Variable Values" You can combine numeric values and placeholders in the same line: `@Stress: H 0.0 values -1000 sig_yy_top 0 H 10.0 values -5000 sig_yy_bot 0` Here, σ_{xx} is fixed while σ_{yy} varies stochastically.

!!! tip "Traditional Approach Still Works" If you don't need to vary initial conditions, use numeric values directly: `@Void: H 0.0 values 0.85 H 10.0 values 0.75` No `@Parameter` needed - values remain constant across all runs.

!!! warning "Placeholder Requirements" - Must be valid identifiers (letters, numbers, underscores; no spaces or special characters) - Must match `@Parameter` name exactly (case-sensitive) - Missing `@Parameter` for a placeholder causes error before simulations start

1.6.4 Format 4: $-Tagged Numeric Values (Any Section / Any Number)

This method lets you vary **any numeric value anywhere** in the input file, even when there is no convenient parameter name (e.g., a node coordinate, a geometric dimension, a repeated constant, etc.).

Rule: Insert a `$tag` token immediately before the numeric value you want to vary. The analysis parser ignores `$...` tags, but surrogate training can target the tag by name.

Supported syntaxes:

- `$tag value`
- `$tag = value`
- `$tag=value`

Example: tagging node coordinates
Input file:

```
% Nodes
1 $x1 0.0 $y1 0.0
2 $x2 1.0 $y2 0.0
3 $x3 1.0 $y3 1.0
%%%
```

Parametric configuration:

```
% SurrogateTraining
@Parameter: $x1 Material Uniform -0.05 0.05
@Parameter: $y1 Material Uniform -0.05 0.05
@Parameter: $x2 Material Uniform  0.95 1.05
@Parameter: $y3 Material Uniform  0.95 1.05
%%%
```

In non-surrogate runs, the same file still parses as if it were:

```
1 0.0 0.0
2 1.0 0.0
3 1.0 1.0
```

!!! note "Target Field for `$tag` Parameters" `$tag` replacement happens via the temporary input file substitution step, so the `<Target>` field is not used to locate the value. Use a non-`Custom` target (e.g., `Material` or `StateVar`) unless you intentionally want a Gauss-point custom variable named like the tag.

1.7 Quantities of Interest (QoIs)

QoIs are the **outputs** you want to extract from each simulation - the response variables for your uncertainty quantification or surrogate model training.

1.7.1 Syntax

```
@QoI: <QoIName> <Type> <Arguments>
```

1.7.2 QoI Types

DOF Value Extract degree-of-freedom value (displacement, pore pressure) at a specific node.

```
@QoI: TopSettlement DOF 7 DisY
@QoI: CenterPorePress DOF 5 PW
@QoI: LateralDisp DOF 12 DisX
```

- **Aliases**: `DOF`, `DOFValue`, `Displacement`
- **Arguments**: `NodeID DOFName`
- **Available DOFs**:
  - `DisX`, `DisY`, `DisZ` - Displacement components
  - `PW` - Pore water pressure
  - `PA` - Pore air pressure
- **Use Case**: Monitoring specific locations (crest settlement, wall deflection, etc.)

Reaction Force Sum of reaction forces at boundary nodes (useful for bearing capacity, passive resistance, etc.).

```
@QoI: BaseReaction Reaction 1,3,5,7 DisY
@QoI: WallForce Reaction 10,11,12 DisX
```

- **Aliases**: `Reaction`, `ReactionForce`, `RF`
- **Arguments**: `NodeList DOFName`
- **NodeList**: Comma-separated or space-separated node IDs
- **Output**: Total reaction force (sum over all specified nodes)
- **Use Case**: Foundation bearing capacity, retaining wall thrust, global equilibrium checks

---

State Variable at Point Interpolate state variable (stress, strain, pore pressure, void ratio) at arbitrary coordinates.

```
@QoI: CenterStressXX StateVar 1.0 1.0 StressXX
@QoI: MidPorePressure StateVar 0.5 0.5 PoreWaterPressure
@QoI: BottomVoidRatio StateVar 2.0 0.0 VoidRatio
```

- **Aliases**: `StateVar`, `StateVariableAtPoint`, `Point`

- **Arguments**: `X_coord Y_coord StateVariableName`
- **Supported Variables**:

Category | Variables | |----------|----------| | **Effective Stress** | `StressXX`, `StressYY`, `StressZZ`, `StressXY`, `StressZY`, `StressZX` | | **Strain** | `StrainXX`, `StrainYY`, `StrainZZ`, `StrainXY`, `StrainZY`, `StrainZX` | | **Pore Pressure** | `PoreWaterPressure` (or `PW`), `PoreAir Pressure` (or `PA`) | | **Void Ratio** | `VoidRatio` (or `e`), `InitialVoidRatio` | | **Saturation** | `DegreeOfSaturation` (or `Sr`) | | **Total Stress** | `TotalStressXX`, `TotalStressYY`, `Total StressZZ` |

- **Interpolation**: Uses element shape functions with nodal stress recovery
- **Use Case**: Extracting stress/strain at critical locations, monitoring consolidation state

---

Maximum State Variable Find maximum value of a state variable across the entire domain.

```
@QoI: MaxDevStress MaxState DevStress
@QoI: MaxPorePressure MaxState PoreWaterPressure
@QoI: MaxPlasticStrain MaxState EffectivePlasticStrain
```

- **Aliases**: `MaxState`, `MaxStateVariable`, `Max`
- **Arguments**: `StateVariableName`
- **Search Domain**: All nodes in the mesh
- **Use Case**: Peak stress/strain identification, localization detection, maximum pore pressure buildup

---

Minimum State Variable Find minimum value of a state variable across the entire domain.

```
@QoI: MinEffStress MinState StressYY
@QoI: MinVoidRatio MinState VoidRatio
@QoI: MaxSettlement MinState DisY
```

- **Aliases**: `MinState`, `MinStateVariable`, `Min`
- **Arguments**: `StateVariableName`
- **Search Domain**: All nodes in the mesh
- **Use Case**: Maximum settlement (min DisY), tensile stress check (min σ), densest region (min e)

!!! tip "Maximum Settlement = Minimum DisY" For downward settlement, use `MinState DisY` since negative displacements represent settlement.

1.8 Output Dataset

1.8.1 CSV Structure

The output CSV file is a structured dataset ready for machine learning or statistical analysis:

Column Type	Description	Example
`run_id`	Simulation number (0-indexed)	`0`, `1`, `2`, ...
`success`	Run completion status	`1` (success), `0` (failed)
`analysis_time`	Wall-clock computation time (seconds)	`3.45`
`final_step`	Last simulation step reached	`1`
`final_sim_time`	Final simulation time achieved	`1.0`
`substep_count`	Total adaptive substeps taken	`47`
`<ParamName>`	Input parameter columns (one per `@Parameter`)	`YoungsModulus`, `PoissonRatio`, ...
`<QoIName>`	Output QoI columns (one per `@QoI`)	`TopSettlement`, `MaxStress`, ...

1.8.2 Example Output

```
# FALCON Parametric Study Output
# Created: 2025-12-16
# NumRuns: 100
# Seed: 42
#
# Input Parameters (Uncertain):
#   - YoungsModulus (Uniform: 50000 to 200000)
#   - PoissonRatio (TruncatedNormal: mean=0.3, std=0.02)
#   - k_sat (LogNormal: ln_mean=-16.12, ln_std=1.5)
#
# Output Quantities of Interest:
#   - TopSettlement (DOF at node 7)
#   - BaseReaction (Reaction force at nodes 1,3,5)
#   - MaxStress (Max StressYY across domain)
#
run_id,success,analysis_time,final_step,final_sim_time,substep_count,
YoungsModulus,PoissonRatio,k_sat,TopSettlement,BaseReaction,MaxStress
0,1,2.45,1,1.0,32,156234.12,-0.31234,1.2e-07,-0.00234,365111.2,-45678.9
1,1,1.89,1,1.0,28,89456.78,0.28765,8.9e-08,-0.00412,358234.7,-52341.1
2,0,0.45,0,0.05,5,198234.56,0.47891,3.2e-06,NaN,NaN,NaN
3,1,2.12,1,1.0,35,178923.45,0.29877,1.5e-07,-0.00198,372456.8,-41234.5
```

!!! note "Failed Runs" - `success=0` indicates numerical failure (non-convergence, negative

Jacobian, etc.) - QoI values are `NaN` for failed runs - Parameter values are still recorded - If `@ContinueOnError: Yes`, execution continues; otherwise, stops immediately

!!! tip "CSV Comments" Lines starting with # contain metadata about the study configuration. Most CSV parsers (pandas, MATLAB, Excel) automatically skip these comment lines.

1.9 Complete Example: Consolidation UQ Study

Here's a comprehensive example demonstrating uncertainty quantification for a coupled consolidation analysis.

```
% AnalysisType
PLCoupled
%%%

% Nodes
1  0.0 0.0
2  1.0 0.0
3  2.0 0.0
4  0.0 1.0
5  1.0 1.0
6  2.0 1.0
7  0.0 2.0
8  1.0 2.0
9  2.0 2.0
%%%

% Materials
Mat1
@Mech: Elastic YoungsModulus 100000 PoissonRatio 0.3
@Perm: Constant k_sat 1.0e-7
@PhaseChar: Solid rhos 2.7
@PhaseChar: Liquid rhow 1.0 K_l 2.25e9 l_vicosity 1.0e-6
%%%

% Elements
1 N6P6C 1 2 3 6 9 5 Mat1
2 N6P6C 9 8 7 4 1 5 Mat1
%%%

% Boundary Conditions
1 DisX 0 DisY 0
3 DisX 0 DisY 0
7 DisX 0
9 DisX 0
%%%
```

```
% Stress Boundaries
Step 1
@Elem 1 2
@EdgeNodes 7 8 9
@Type Pressure
@Magnitudes -100000
@LoadApp instant
%%%

% InitialStepAssignment
Step 1
@Assign Gauss
# Use named placeholders for stochastic initial conditions
@PW: H 0.0 values PW_top H 2.0 values PW_bottom
@Void: H 0.0 values e0_surface H 2.0 values e0_depth
%%%

% Steps
Step 1
  SolverType: Coupled_U_UP
  StepTime: 1.0
  NumberSteps: 10
  OutputInterval: 1
  OutputTypes: Dis PW Stress Strain
  ModernAutoInc: Yes
  DeltaTMin: 0.001
  DeltaTMax: 0.1
  MaxSubStep: 100
EndStep
%%%

% SurrogateTraining
# ========================================
# Uncertainty Quantification Configuration
# ========================================

@NumRuns: 500
@Seed: 123456
@OutputFile: consolidation_uq_dataset.csv
@ContinueOnError: Yes
@SaveIntermediate: No

# ----------------------------------------
# Material Property Uncertainties
# ----------------------------------------
```

```
# Young's modulus: moderate uncertainty (CoV ≈ 20%)
@Parameter: YoungsModulus Material TruncatedNormal 100000 20000 50000 200000

# Poisson's ratio: well-constrained, low uncertainty
@Parameter: PoissonRatio Material TruncatedNormal 0.3 0.02 0.0 0.49

# Hydraulic conductivity: high uncertainty spanning orders of magnitude
# LogNormal with ln(median)=ln(1e-7)=-16.12, high variability
@Parameter: k_sat Material LogNormal -16.12 1.5

# ----------------------------------------
# Initial Condition Uncertainties
# ----------------------------------------

# Surface pore pressure: may vary due to seasonal water table
@Parameter: PW_top StateVar Uniform -5000 0

# Pore pressure at depth: confined aquifer uncertainty
@Parameter: PW_bottom StateVar Uniform -25000 -10000

# Initial void ratio at surface: loose to dense
@Parameter: e0_surface StateVar TruncatedNormal 0.85 0.05 0.6 1.1

# Void ratio at depth: typically denser
@Parameter: e0_depth StateVar TruncatedNormal 0.75 0.04 0.5 1.0

# ----------------------------------------
# Quantities of Interest (Responses)
# ----------------------------------------

# Surface settlement at key locations
@QoI: TopSettlement DOF 7 DisY
@QoI: CenterSettlement DOF 5 DisY
@QoI: EdgeSettlement DOF 9 DisY

# Reaction forces for global equilibrium check
@QoI: TotalBaseReaction Reaction 1,3 DisY

# Pore pressure dissipation monitoring
@QoI: CenterPW StateVar 1.0 1.0 PoreWaterPressure
@QoI: MaxPW MaxState PW
@QoI: MinPW MinState PW

# Stress state for yielding assessment
@QoI: CenterStressXX StateVar 1.0 1.0 StressXX
@QoI: CenterStressYY StateVar 1.0 1.0 StressYY
@QoI: MaxEffStress MaxState StressYY
```

```
@QoI: MinEffStress MinState StressYY

# Global settlement metrics
@QoI: MaximumSettlement MinState DisY
%%%
```

1.9.1 Post-Processing: Uncertainty Quantification

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats

# Load dataset
df = pd.read_csv('consolidation_uq_dataset.csv', comment='#')
successful = df[df['success'] == 1].copy()

print(f"Success rate: {len(successful)/len(df)*100:.1f}%")

# ========================================
# Descriptive Statistics
# ========================================

response = 'TopSettlement'
print(f"\n{response} Statistics:")
print(f"  Mean:   {successful[response].mean():.6f}")
print(f"  Median: {successful[response].median():.6f}")
print(f"  Std:    {successful[response].std():.6f}")
print(f"  CoV:
{successful[response].std()/abs(successful[response].mean())*100:.1f}%")
print(f"  5th percentile:  {successful[response].quantile(0.05):.6f}")
print(f"  95th percentile: {successful[response].quantile(0.95):.6f}")

# ========================================
# Sensitivity Analysis (Correlation-Based)
# ========================================

params = ['YoungsModulus', 'PoissonRatio', 'k_sat', 'PW_top', 'PW_bottom',
          'e0_surface', 'e0_depth']
qoi = 'TopSettlement'

correlations = successful[params +
[qoi]].corr()[qoi].drop(qoi).sort_values(ascending=False)
print(f"\nPearson Correlations with {qoi}:")
print(correlations)
```

```python
# Visualization
fig, axes = plt.subplots(2, 4, figsize=(16, 8))
axes = axes.flatten()

for i, param in enumerate(params):
    axes[i].scatter(successful[param], successful[qoi], alpha=0.3, s=10)
    axes[i].set_xlabel(param)
    axes[i].set_ylabel(qoi)
    axes[i].set_title(f"Corr: {correlations[param]:.3f}")

plt.tight_layout()
plt.savefig('uq_scatter_plots.png', dpi=300)

# ========================================
# Histogram and PDF
# ========================================

fig, ax = plt.subplots(1, 1, figsize=(8, 5))
ax.hist(successful[qoi], bins=50, density=True, alpha=0.7,
edgecolor='black')

# Fit normal distribution
mu, sigma = stats.norm.fit(successful[qoi])
x = np.linspace(successful[qoi].min(), successful[qoi].max(), 100)
ax.plot(x, stats.norm.pdf(x, mu, sigma), 'r-', lw=2, label=f'Normal fit
(μ={mu:.6f}, σ={sigma:.6f})')

ax.set_xlabel(qoi)
ax.set_ylabel('Probability Density')
ax.set_title('Response Distribution')
ax.legend()
plt.savefig('uq_histogram.png', dpi=300)

# ========================================
# Reliability Analysis (Example)
# ========================================

# Define failure criterion: settlement exceeds 50 mm
threshold = -0.050  # 50 mm downward
failures = successful[qoi] < threshold
p_failure = failures.sum() / len(successful)
print(f"\nReliability Analysis:")
print(f"  Failure criterion: Settlement > 50 mm")
print(f"  Probability of failure: {p_failure:.4f} ({p_failure*100:.2f}%)")
```

```
print(f"  Reliability index (β): {-stats.norm.ppf(p_failure):.2f}")
```

1.10 Error Handling

1.10.1 Missing Parameter Error

If a `@Parameter` name doesn't match anything in the input file, FALCON halts before starting simulations:

```
[ERROR] The following parameters specified in SurrogateTraining were NOT found:
  - 'InvalidParam'

Ensure parameter names match exactly (case-sensitive).

Supported formats:
  1. ParamName=Value  →  E=100000 in @UMAT line
  2. ParamName Value  →  YoungsModulus 210000 in @Mech line
  3. values ParamName →  values PW_surface in InitialStepAssignment

For initial state variables, use named placeholders:
  @PW: H 0.0 values PW_surface H 10.0 values PW_bottom
  @Parameter: PW_surface StateVar Uniform -5000 0
```

1.10.2 Failed Simulation Runs

Behavior with `@ContinueOnError: Yes`:

- Run marked with `success=0`
- Error logged to console
- QoI values set to `NaN`
- Next simulation starts automatically

Behavior with `@ContinueOnError: No` (default):

- Execution stops immediately
- CSV contains only successful runs up to failure point
- Error message displayed

1.11 Best Practices

!!! success "Design of Experiments"

1. Start small, scale up

Begin with 10-20 runs to verify configuration, then scale to hundreds or thousands: `@Num Runs: 10  # Validation phase     # @NumRuns: 1000  # Production phase`

2. Set random seed for reproducibility

Critical for scientific reproducibility and debugging: `@Seed: 42  # Any integer, documented in your publication`

3. Choose appropriate sample sizes

- **Correlation analysis**: 50-100 runs
  - **Surrogate training (neural net)**: 500-5000 runs depending on dimensionality
  - **Gaussian process**: 50-500 runs
  - **Monte Carlo convergence**: 1000-10000 runs
  - **Sensitivity analysis (Sobol)**: 1000*(d+2) where d = number of parameters

!!! success "Distribution Selection"

4. Match distributions to physical reality

Parameter Type | Recommended Distribution | |----------------|-------------------------| | Permeability, hydraulic conductivity | `LogNormal` (spans orders of magnitude) | | Friction angle, cohesion (bounded) | `TruncatedNormal` | | Young's modulus (bounded, symmetric) | `TruncatedNormal` or `Uniform` | | Poisson's ratio (strictly <0.5) | `TruncatedNormal` with max=0.49 | | Load factors (relative uncertainty) | `Normal` or `Uniform` | | Initial void ratio (bounded) | `TruncatedNormal` |

5. Validate bounds

Ensure sampled values don't cause numerical issues: "' # Bad: ν can exceed 0.5 (incompressible/unstable) @Parameter: PoissonRatio Material Normal 0.3 0.3

```
# Good: Truncated to physically valid range
@Parameter: PoissonRatio Material TruncatedNormal 0.3 0.05 0.0 0.49
```

6. Use logarithmic parameters for permeability

```
# Permeability median = 1e-7 m/s, spanning 1e-9 to 1e-5
# ln(1e-7) = -16.12
# Standard deviation in log space: ~2.3 covers the range
@Parameter: k_sat Material LogNormal -16.12 2.3
```

!!! success "QoI Selection"

7. Record diverse response types

Capture both local and global behavior: "' # Local responses at critical points @QoI: CrestSettlement DOF 7 DisY @QoI: ToeStress StateVar 5.0 0.0 StressXX

```
# Global extrema
@QoI: MaxSettlement MinState DisY
```

```
@QoI: MaxStress MaxState StressYY

# Integrated quantities
@QoI: TotalReaction Reaction 1,2,3,4 DisY
```

8. Include failure indicators

```
@QoI: MinMeanStress MinState MeanStress  # Tension check
@QoI: MaxDevStress MaxState DeviatoricStress  # Yielding check
```

!!! warning "Common Mistakes"

1. Case mismatch in parameter names

```
# Input file:  YoungsModulus
# Parameter:   youngsmodulus   WILL FAIL
# Correct:     YoungsModulus
```

2. Truncated normal with mean outside bounds

```
# Bad: mean=25 is outside [30, 40]
@Parameter: phi Material TruncatedNormal 25 5 30 40

# Good: mean=35 is centered in [30, 40]
@Parameter: phi Material TruncatedNormal 35 2.5 30 40
```

3. Forgetting placeholders

```
# InitialStepAssignment uses:
@PW: H 0.0 values PW_top H 10.0 values PW_bot

# Must define BOTH:
@Parameter: PW_top StateVar ...
@Parameter: PW_bot StateVar ...
```

4. Incompatible parameter combinations

```
# Dangerous: very soft + near-incompressible = numerical issues
@Parameter: E Material Uniform 1000 5000
@Parameter: nu Material Uniform 0.48 0.499
```

1.12 Advanced Applications

1.12.1 Latin Hypercube Sampling

While FALCON uses pseudo-random sampling by default, you can post-process with Latin Hypercube for better space-filling:

```
from scipy.stats import qmc
import pandas as pd

# Generate Latin Hypercube samples (better coverage than random)
num_samples = 100
num_params = 3

sampler = qmc.LatinHypercube(d=num_params, seed=42)
samples = sampler.random(n=num_samples)

# Transform to parameter distributions
from scipy.stats import uniform, norm, lognorm

E_samples = uniform(loc=50000, scale=150000).ppf(samples[:, 0])
nu_samples = norm(loc=0.3, scale=0.03).ppf(samples[:, 1])
k_samples = lognorm(s=1.5, scale=np.exp(-16.12)).ppf(samples[:, 2])

# Write to input file or modify parameters externally
```

1.12.2 Multi-Fidelity Surrogate Training

For expensive high-fidelity models, use cheap low-fidelity runs for initial exploration:

```
% SurrogateTraining
# Phase 1: Coarse mesh, 1000 runs (low-fidelity)
@NumRuns: 1000
@OutputFile: low_fidelity_data.csv
%%%
```

Then refine with fewer high-fidelity runs and combine using co-kriging or multi-fidelity Gaussian processes.

1.12.3 Sobol Sensitivity Analysis

Generate datasets for variance-based sensitivity:

```
from SALib.sample import saltelli
from SALib.analyze import sobol
```

```
problem = {
    'num_vars': 3,
    'names': ['YoungsModulus', 'PoissonRatio', 'k_sat'],
    'bounds': [[50000, 200000], [0.2, 0.4], [1e-9, 1e-5]]
}

# Generate Saltelli samples (N*(2D+2) total runs)
param_values = saltelli.sample(problem, 1024)

# Run FALCON for each sample, then analyze
Y = results['TopSettlement'].values
Si = sobol.analyze(problem, Y)

print("First-order indices:", Si['S1'])
print("Total-order indices:", Si['ST'])
```

1.12.4 Bayesian Calibration

Use output for MCMC-based parameter estimation:

```
import pymc as pm

# Observed data
obs_settlement = -0.0234  # from field measurement
obs_std = 0.002  # measurement uncertainty

# Prior from parametric study bounds
with pm.Model() as model:
    E = pm.Uniform('E', lower=50000, upper=200000)
    nu = pm.TruncatedNormal('nu', mu=0.3, sigma=0.03, lower=0, upper=0.49)

    # Surrogate model (trained Gaussian process as likelihood)
    predicted = gp_surrogate(E, nu)  # from ML surrogate

    obs = pm.Normal('obs', mu=predicted, sigma=obs_std,
observed=obs_settlement)

    trace = pm.sample(2000, return_inferencedata=True)

# Posterior distribution on E and nu
```

1.13 Summary Reference Table

Configuration	Syntax	Example
Basic Settings		
Number of runs	`@NumRuns: <N>`	`@NumRuns: 1000`
Random seed	`@Seed: <S>`	`@Seed: 42`
Output file	`@OutputFile: <path>`	`@OutputFile: uq_data.csv`
Continue on error	`@ContinueOnError: Yes/No`	`@ContinueOnError: Yes`
Parallel Execution		
Enable parallel	`@Parallel: Yes/No`	`@Parallel: Yes`
Max workers	`@MaxJobs: <N>`	`@MaxJobs: 8`
Statistical Distributions		
Uniform	`Uniform <min> <max>`	`Uniform 50000 200000`
Normal	`Normal <mean> <std>`	`Normal 0.3 0.03`
Truncated Normal	`TruncatedNormal <mean> <std> <min> <max>`	`TruncatedNormal 30 5 20 40`
Log-Normal	`LogNormal <ln_mean> <ln_std>`	`LogNormal -16.12 1.5`
Fixed	`Fixed <value>`	`Fixed 0.0`
Interval	`Interval <min> <max>`	`Interval 50000 200000`
Parameter Targets		
Material property	`Material` (or `Mat`, `MaterialProperty`)	`@Parameter: E Material ...`
Initial state	`StateVar` (or `Custom`)	`@Parameter: PW_top StateVar ...`
Boundary condition	`Boundary` (or `BC`)	`@Parameter: BC_val Boundary ...`
Load magnitude	`Load` (or `Force`)	`@Parameter: Q Load ...`
Quantities of Interest		
DOF at node	`DOF <nodeID> <dof>`	`@QoI: Disp DOF 7 DisY`
Reaction force	`Reaction <nodes> <dof>`	`@QoI: RF Reaction 1,3,5 DisY`

Configuration	Syntax	Example
State at point	`StateVar <x> <y> <var>`	`@QoI: Stress State Var 1.0 1.0 Stress XX`
Maximum	`MaxState <var>`	`@QoI: MaxPW Max State PW`
Minimum	`MinState <var>`	`@QoI: MinStress Min State StressYY`

1.14 Keywords for Search

Monte Carlo simulation, uncertainty quantification, UQ, stochastic analysis, parametric study, parameter study, sensitivity analysis, surrogate modeling, machine learning training data, Gaussian process, neural network training, probabilistic analysis, reliability analysis, random sampling, Latin hypercube, design of experiments, DOE, response surface, metamodeling, variance-based sensitivity, Sobol indices, Bayesian calibration, inverse analysis, data-driven modeling, digital twin, model predictive control, ensemble simulation, uncertainty propagation, risk assessment, probability of failure, soil property uncertainty, parallel computing, parallel execution, multi-threaded simulation, HPC, high-performance computing, batch simulation, concurrent analysis