AD FALCON API Manual

Automatic Time Increment

Javad Ghorbani

March 26, 2026

Contents

1 Automatic Time Increment

FALCON's auto-increment controller adaptively adjusts the sub-step size Δt within each simulation step to balance robustness, accuracy, and efficiency. This document describes the algorithm, all user-configurable parameters, and tuning strategies.

1.1 Overview

The controller:
- **Enables automatic sub-stepping** within each simulation step
- **Adapts Δt** based on a convergence/error metric (Implicit: normalized Newton residual; Explicit/IMEX: embedded error estimate)
- **Retries with reduced Δt** on failure with sophisticated backoff strategies
- **Grows Δt cautiously** when convergence is strong, subject to multiple safety caps
- **Tracks streaks** (success/failure) to inform growth/shrinkage decisions
- **Maintains a dynamic max** that can relax when the system is safely below target

1.2 Implicit vs Explicit/IMEX

ModernAutoInc is used in two different ways depending on `@@TimeIntegration:`. The same input keys control time-step bounds and retry limits, but the **accept/reject metric** differs.

Time integration	Accept/reject metric	Meaning of `@@Error Target:`
`Implicit`	Newton convergence (`fnorm`)	Target normalized residual
`Explicit / IMEX`	Embedded local error estimate (`err_est`)	Target error estimate

- For `Explicit/IMEX` details (predictor/corrector, θ-method for flow, mass-solve), see Explicit and IMEX Time Integration.

1.3 Syntax

Automatic time incrementation is configured inside `% Step Definitions` (per step). Keys are written as `@@<Key>: <Value>`.

```
% Step Definitions
@Step 1:
  @@StartStep: 0
  @@StepTime: 100.0
  @@NumberSteps: 100
  @@ModernAutoInc: Yes
  @@InitialStepIncrement: 0.5
  @@MinTimeStep: 1e-6
  @@MaxTimeStep: 2.0
  @@ErrorTarget: 1e-3
  @@MaxRetry: 10
%%%
```

Step keys are case-insensitive; the spellings below are the canonical forms used in this manual.

1.4 Enabling the Controller

Input Key	Values	Mandatory?	Default	Description
`@@ModernAutoInc:`	`Yes` / `No`	**Yes**	`No`	Enable auto-increment controller (preferred key)
`@@AutomaticStepControl:`	`Yes` / `No`	**Yes** (alias)	`No`	Alternative key to enable the controller

Note: You must set one of these to `Yes` to activate the controller. If both are `No`, FALCON uses the legacy stepping logic.

1.5 Primary Time-Step Controls

These parameters control the fundamental time-step bounds and error targets. All are **optional** but should be tuned for your problem.

Input Key	Mandatory?	Default	Type	Description
`@@InitialStep Increment:`	Optional	`1.0`	`double`	Initial Δt for the first sub-step. If ≤ 0 or not provided, computed as `stepTime / InitialStep FallbackDivisor`
`@@MinTimeStep:`	Optional	`1.0e-6`	`double`	Lower bound `delta TMin` (must be > 0)
`@@MaxTimeStep:`	Optional	`1.0`	`double`	Upper bound `delta TMax` (must be > `deltaTMin`)
`@@ErrorTarget:`	Optional	`1.0e-2`	`double`	Target for the controller metric: Implicit uses normalized Newton residual (`fnorm`); Explicit/IMEX uses embedded error estimate (`err_est`).
`@@RetryTol Relaxation:`	Optional	`No`	`Yes/No`	If `Yes`, temporarily relax `adaptive Target` during retries, then tighten after recovery
`@@MaxSubStep:`	Optional	`100000`	`int`	Maximum number of sub-steps allowed (infinite-loop guard)
`@@MaxRetry:`	Optional	`30`	`int`	Maximum solver retries per sub-step before aborting

Note: While defaults exist, you should **always tune** `MinTimeStep`, `MaxTimeStep`, and `ErrorTarget` for your specific problem to ensure proper scaling and convergence behavior.

1.6 User-Adjustable Advanced Parameters

The controller has **only 5 user-adjustable** advanced parameters for fine-tuning. All other internal parameters use fixed default values that have been carefully calibrated (see Appendix A for complete list).

Input Key	Default	Type	Category	Role	Tuning Guidance
`@@HardSafeFactor:`	`0.5`	`double`	Retry/Safety	Hard safety cap: new dt ≤ `factor * lastSuccessfulDt`	Conservative: 0.4–0.5; Aggressive: 0.6–0.7
`@@GrowthCooldownMax:`	8	`int`	Retry/Safety	Maximum cooldown counter (during cooldown, Δt cannot grow)	Conservative: 10–15; Aggressive: 5–6
`@@AdjustClampMax:`	`1.1`	`double`	Growth	Maximum adjustment factor per success (limits growth)	Conservative: 1.05–1.08; Aggressive: 1.15–1.2
`@@GrowthCapLimit:`	`1.05`	`double`	Growth	Absolute growth cap limit (caps total growth regardless of streak)	Conservative: 1.03–1.05; Aggressive: 1.08–1.1

Input Key	Default	Type	Category	Role	Tuning Guidance
`@@Smoothing Historical Weight:`	`0.8`	`double`	Smoothing	EMA weight for last successful dt (0.8 = 80% history, 20% new)	Conservative: 0.85–0.9; Aggressive: 0.6–0.7

Note: These 5 parameters give users the ability to trade off between stability (conservative) and efficiency (aggressive) without exposing unnecessary complexity. All 28 other internal parameters use fixed default values that have been extensively tested.

1.7 Implicit (Newton-based) algorithm

This section describes the **Implicit** (Newton-based) ModernAutoInc controller and highlights where the 5 user-adjustable parameters affect behavior. For `Explicit/IMEX`, the accept/reject metric and update rules differ; see Explicit and IMEX Time Integration.

1.7.1 Initialization

1. **Determine initial sub-step size:**
   - If `InitialStepIncrement` > 0: use that value
   - Otherwise: `Δt₀ = stepTime / 1e5` (fallback)
2. **Clamp to bounds:** `Δt₀ = clamp(Δt₀, MinTimeStep, MaxTimeStep)`
3. **Initialize tracking variables:**
   - `historicalSubStepIncrement = Δt₀` (seed for exponential moving average)
   - `dynamicMax = MaxTimeStep` (adaptive upper bound that can shrink on trouble)
   - `lastSuccessfulDt = Δt₀`
   - `successStreak = 0`, `failureStreak = 0`, `growthCooldown = 0`

1.7.2 Sub-Step Loop

The controller attempts sub-steps until the entire step time is covered. Each sub-step either succeeds (solution converges) or fails (divergence, Jacobian singular, stress integration failure).

```
While (remainingTime > 1e-12):
    actualIncrement = min(historicalSubStepIncrement, remainingTime)

    Attempt to solve with Δt = actualIncrement

    IF FAILURE → go to Section 5.3
    IF SUCCESS → go to Section 5.4
```

1.7.3 Failure Path: Aggressive Shrinkage with Safety Caps

When a sub-step fails, the controller aggressively reduces Δt to recover.

Step 1: Roll Back and Track Failure

- Restore mesh state to before the failed attempt
- `retryCount++`, `failureStreak++`, `successStreak = 0`

Step 2: Compute Retry Backoff Factor The controller uses exponential decay combined with residual-based scaling:

```
backoffFactor = (0.9 / 2^retryCount) × √(ErrorTarget / fnorm)
```

- **1st retry**: factor ≈ 0.45 × residual_ratio (cuts Δt by ~half)
- **2nd retry**: factor ≈ 0.225 × residual_ratio (cuts Δt by ~¾)
- **3rd retry**: factor ≈ 0.1125 × residual_ratio (cuts Δt by 7/8)

The backoff is clamped to `[0.5, 0.9]` to prevent too-slow or too-fast reduction.

Step 3: Apply Safety Caps Multiple safety mechanisms limit how small Δt can become:

a) Exponential shrink based on failure streak:

```
shrinkCeiling = lastFailureDt × 0.6^failureStreak
```

- 1 failure: 60% of last failure dt
- 2 failures: 36% of last failure dt
- 3 failures: 21.6% of last failure dt

b) Hard safety cap [USER-ADJUSTABLE: `HardSafeFactor`]

```
hardSafe = lastSuccessfulDt × HardSafeFactor
```

Even if the backoff formula suggests a larger Δt, never exceed this fraction of the last known-good dt.

- **Default**: `0.5` (no more than 50% of last success)
- **Conservative tuning**: `0.4`–`0.5` (more aggressive cap, safer recovery)
- **Aggressive tuning**: `0.6`–`0.7` (less restrictive, faster recovery but riskier)

Effect: Lower values make failure recovery more cautious; higher values allow faster recovery.

c) Update dynamic maximum:

```
dynamicMax = min(dynamicMax, shrinkCeiling, firstTryHistorical)
```

The dynamic max shrinks based on failures, preventing future attempts from being too large.

Step 4: Apply Growth Cooldown [USER-ADJUSTABLE: `GrowthCooldownMax`] After failure, increment the cooldown counter:

```
growthCooldown = min(growthCooldown + 3, GrowthCooldownMax)
```

While `growthCooldown > 0`, Δt is **blocked from growing** even if convergence is good.

- **Default**: `8` steps of cooldown
- **Conservative tuning**: `10`–`15` (longer caution period after failure)
- **Aggressive tuning**: `5`–`6` (shorter caution, resume growth sooner)

Effect: Higher values keep Δt conservative for longer after failures; lower values resume growth sooner.

Step 5: Optional Tolerance Relaxation If `RetryTolRelaxation: Yes`:

```
adaptiveTarget = min(ErrorTarget × 10^retryCount, 1.0)
```

Temporarily relax the residual target to help convergence recover (tightened back after success).

Step 6: Abort Guards

- If `retryCount > MaxRetry` → **Abort** with error `[SSH-3112]`
- If new $\Delta t \leq$ `max(MinTimeStep, 1e-20)` → **Abort** with error `[SSH-3116]`

Otherwise, retry the sub-step with the reduced Δt.

1.7.4 Success Path: Cautious Growth with Multiple Limiters

When a sub-step succeeds, the controller adjusts Δt for the next attempt.

Step 1: Update Streak Counters

```
successStreak++
failureStreak = 0
if (growthCooldown > 0) growthCooldown--
```

Step 2: Get Conservative Estimate from Watcher The global convergence watcher examines:
- Mesh quality (no severely distorted elements)
- Residual trends (fnorm not growing)
- Iteration count (converged quickly or slowly?)

Returns: `conservativeDt` (a safe estimate for next Δt)

Step 3: Compute Adjustment Factor [USER-ADJUSTABLE: `AdjustClampMax`] Based on residual quality:

```
IF fnorm > 1e-14:
    adjustmentFactor = 0.9 × √(adaptiveTarget / fnorm)
ELSE:
    adjustmentFactor = 1.0  (residual negligible, no penalty)
```

Clamp to bounds:

```
adjustmentFactor = clamp(adjustmentFactor, 0.001, AdjustClampMax)
```

- **Default**: `1.1` (max 10% growth per sub-step)
- **Conservative tuning**: `1.05–1.08` (limit to 5–8% growth, very stable)
- **Aggressive tuning**: `1.15–1.2` (allow 15–20% growth, faster but less stable)

Effect: This is your **primary growth control**. Lower = slower, more stable; higher = faster, less stable.

Step 4: Apply Success-Based Growth Cap [USER-ADJUSTABLE: `GrowthCapLimit`] The growth cap rewards consecutive successes:

```
growthCap = min(1.0 + 0.02 × (successStreak - 1), GrowthCapLimit)
adjustmentFactor = min(adjustmentFactor, growthCap)
```

- After 1 success: cap = 1.0 (no growth yet)
- After 2 successes: cap = 1.02 (2% growth allowed)
- After 3 successes: cap = 1.04 (4% growth allowed)
- After 5+ successes: cap = `GrowthCapLimit`

Parameters:
- **Default**: `1.05` (max 5% total growth from streak)
- **Conservative tuning**: `1.03–1.05` (cap at 3–5%)
- **Aggressive tuning**: `1.08–1.1` (cap at 8–10%)

Effect: Lower values make the controller earn growth slowly; higher values allow faster ramping.

Step 5: Block Growth During Cooldown

```
if (growthCooldown > 0):
    adjustmentFactor = min(adjustmentFactor, 1.0)  // no growth allowed
```

This prevents Δt from growing immediately after recovering from failure.

Step 6: Relax Dynamic Max (Safe Region Logic) If the system is safely below target with a good success streak:

```
IF cooldown == 0 AND successStreak ≥ 5 AND fnorm < 0.2 × adaptiveTarget:
    safety = fnorm / adaptiveTarget
    relaxFactor = 1 + 0.01 × min(successStreak - 4, 6) × (1 - safety)
    dynamicMax = min(originalMax, dynamicMax × relaxFactor)
```

This gradually lifts the dynamic max back toward `MaxTimeStep` when the system is converging easily.

Step 7: Compute New Historical Δt with Smoothing [USER-ADJUSTABLE: `SmoothingHistoricalWeight`] Combine the proposed new Δt with history using exponential moving average:

```
newHistorical = conservativeDt × adjustmentFactor
newHistorical = min(newHistorical, dynamicMax, 1.02 × lastSuccessfulDt)

historicalSubStepIncrement = SmoothingHistoricalWeight × lastSuccessfulDt
                            + SmoothingNewWeight × newHistorical
```

- **Default**: `0.8` (80% weight on history, 20% on new)
- **Conservative tuning**: `0.85–0.9` (more smoothing, very stable but slower to adapt)
- **Aggressive tuning**: `0.6–0.7` (less smoothing, faster adaptation but more reactive)

Effect: Higher values make Δt change slowly and smoothly; lower values make it respond quickly to changes.

Step 8: Optional Tolerance Tightening If `RetryTolRelaxation: Yes` and it was previously relaxed:

```
adaptiveTarget = max(adaptiveTarget / 1.01, ErrorTarget)
```

Gradually tighten the residual target back to the original.

Step 9: Update for Next Sub-Step

```
mesh.dt = historicalSubStepIncrement
lastSuccessfulDt = historicalSubStepIncrement
remainingTime -= actualIncrement
```

Step 10: Abort Guard If `numberofSubSteps > MaxSubStep` → **Abort** with error `[SSH-3111]`

1.7.5 Final Time Correction (Optional)

After covering the full step time, check for accumulated error:

```
IF |actualTotalPassedTime - targetEndTime| > 1e-12:
    Perform up to 3 tiny correction sub-steps
```

1.7.6 Summary: How the 5 User Parameters Work Together

Parameter	What It Controls	Impact on Δt Evolution
`HardSafeFactor`	Safety cap after failure	Lower = more cautious recovery after failure
`GrowthCooldownMax`	How long to block growth after failure	Higher = longer conservative period post-failure
`AdjustClampMax`	Maximum growth per sub-step	**Primary growth knob**: lower = slower growth
`GrowthCapLimit`	Maximum growth from success streak	Limits how fast Δt can ramp up over time
`SmoothingHistoricalWeight`	How much history vs. new affects Δt	Higher = smoother, more stable Δt changes

Recommended tuning approach:
1. Start with defaults
2. If you see too many failures: reduce `AdjustClampMax` and `GrowthCapLimit`, increase `GrowthCooldownMax`
3. If analysis is too slow: increase `AdjustClampMax` and `GrowthCapLimit`, reduce `SmoothingHistoricalWeight`
4. If Δt oscillates: increase `SmoothingHistoricalWeight`
5. If recovery from failure is too slow: increase `HardSafeFactor`

1.8 Error Messages and Abort Codes

Code	Message	Cause	Action
[SSH-3111]	Exceeded maximum number of sub-steps	numberofSub Steps > MaxSub Step	Increase MaxSubStep, check for stalled convergence, or reduce ErrorTarget
[SSH-3112]	Exceeded maximum number of retries	retryCount > Max Retry	Increase MaxRetry, relax ErrorTarget, enable RetryTol Relaxation, or check material/BC setup
[SSH-3115]	Conservative dt at/below minimum	conservativeDt <= deltaTMin	Increase MinTime Step, relax Error Target, or check problem conditioning
[SSH-3116]	Sub-step increment at/below minimum threshold	historicalSub StepIncrement <= max(delta TMin, Min Increment Absolute)	Increase MinTime Step, relax Error Target, or enable RetryTolRelaxation

Explicit/IMEX note: automatic substepping in Explicit/IMEX time integration uses different abort codes: [SSH-3210] (max sub-steps) and [SSH-3211] (max retries).

1.9 Practical Tuning Strategies

These ranges are intended for Implicit (Newton-based) solves. For Explicit/IMEX, start with ErrorTarget around 1e-3 and tune based on whether err_est is consistently above/below the target.

1.9.1 Conservative (Stability First)

Goal: Ensure robust convergence, accept slower progress.

Parameter	Conservative Value	Rationale
ErrorTarget:	1e-3 to 1e-6	Tighter tolerance
AdjustClampMax:	1.05 to 1.08	Limit growth to 5-8% per step

Parameter	Conservative Value	Rationale
`GrowthCapLimit:`	`1.03` to `1.05`	Cap total growth at 3-5%
`Smoothing HistoricalWeight:`	`0.85` to `0.9`	Heavy smoothing (85-90% history)
`GrowthCooldownMax:`	`10` to `15`	Longer cooldown after failure
`HardSafeFactor:`	`0.4` to `0.5`	More aggressive safety cap
`RetryTolRelaxation:`	Yes	Help recover from failure bursts

1.9.2 Aggressive (Efficiency First)

Goal: Faster analysis, accept occasional retries.

Parameter	Aggressive Value	Rationale
`ErrorTarget:`	`1e-2` to `1e-3`	Looser tolerance
`AdjustClampMax:`	`1.15` to `1.2`	Allow 15-20% growth per step
`GrowthCapLimit:`	`1.08` to `1.1`	Cap total growth at 8-10%
`SmoothingHistorical Weight:`	`0.6` to `0.7`	Less smoothing (60-70% history)
`GrowthCooldownMax:`	`5` to `6`	Shorter cooldown
`HardSafeFactor:`	`0.6` to `0.7`	Less aggressive safety cap

1.9.3 Balanced (Default)

Goal: Good stability with reasonable efficiency.
Use the default values shown in Section 3 and Section 4 above.

1.10 Notes and Recommendations

1. **Always tune** `InitialStepIncrement`, `MinTimeStep`, `MaxTimeStep`, and `ErrorTarget` for your specific problem. While defaults exist (`1.0`, `1e-6`, `1.0`, `1e-2`), they may not be appropriate for your time/length scales.

2. **Start with defaults** for the 5 adjustable advanced parameters, then tune based on performance.
3. **Use `RetryTolRelaxation: Yes`** on tough problems (e.g., contact, large deformation, unstable materials).
4. **Monitor diagnostics**: If you see frequent `[SSH-3112]` or `[SSH-3116]`, relax `ErrorTarget` or increase safety factors.
5. **Balance smoothing**: Higher `SmoothingHistoricalWeight` (0.8-0.9) = more stable but slower; lower (0.6-0.7) = faster but more responsive to transients.
6. **Growth caps matter**: `GrowthCapLimit` and `AdjustClampMax` are your main knobs for controlling how aggressively dt grows.
7. **Default `ErrorTarget = 1e-2`** is relatively loose; tighten to `1e-6` to `1e-8` for accuracy-critical analyses.
8. **Most parameters use fixed defaults**: The controller uses additional internal parameters with fixed default values (see Appendix A) that have been carefully tuned based on FALCON's development team's experience and are not adjustable by the user.

1.11 Convergence / Error Metric

1.11.1 Implicit (Newton residual: `fnorm`)

For `TimeIntegration: Implicit`, the controller uses the same residual norm that the main Newton solver employs. In each iteration:

- $\mathbf{F}_{\text{unb}}$ is the **unbalanced force** (global residual) on all active DOFs — external loads minus internal forces, dynamic terms and incremental corrections.
- $\mathbf{F}_{\text{ext,eff}}$ is the **effective external load** on active DOFs (the currently applied external forces, including incremental contributions).
- $\mathbf{F}_{\text{rest}}$ is the vector of **reaction forces** on restrained DOFs.
- ε is a small regularization constant (on the order of 10^{-9}) used to avoid division by zero when the denominator becomes very small.

Conceptually, the (undamped) residual norm is

$$\text{fnorm}_{\text{raw}} = \sqrt{\frac{\|\mathbf{F}_{\text{unb}}\|^2}{\|\mathbf{F}_{\text{ext,eff}}\|^2 + \|\mathbf{F}_{\text{rest}}\|^2 + \varepsilon}}.$$

For practical purposes you can view `fnorm` as this normalized residual (possibly with a small amount of internal smoothing between iterations). Convergence is declared when

$\text{fnorm} \le \text{adaptiveTarget}.$

`adaptiveTarget` starts from `ErrorTarget` and may be temporarily relaxed or tightened by `RetryTolRelaxation` during automatic step control.

1.11.2 Explicit/IMEX (embedded error estimate: `err_est`)

For `TimeIntegration: Explicit` and `TimeIntegration: IMEX`, ModernAutoInc uses a **dimensionless embedded local error estimate** (reported as `err_est` in diagnostics) derived from the Modified Euler / Heun predictor–corrector pair.

A predictor state (Euler) and a corrector state (Heun) are computed over the same sub-step, and the raw embedded error is the predictor–corrector difference (e.g., $\Delta \mathbf{u} = \mathbf{u}^{\text{corr}} - \mathbf{u}^{\text{pred}}$ and similarly for rates). FALCON then forms an infinity-norm measure over active DOF groups and applies a mixed absolute/relative scaling:

$$\|\Delta x\|_\infty \le \text{atol} + \text{rtol}\,\|x\|_\infty, \qquad \text{with rtol} = \texttt{@@ErrorTarget}.$$

In terms of the scalar that is printed, FALCON computes an infinity-norm predictor–corrector jump per DOF group and reports

$$\text{err\_est} = \max_g \left(\text{rtol}\,\frac{\|\Delta x_g\|_\infty}{\text{atol} + \text{rtol}\,s_g} \right), \qquad s_g = \max(\|x_g^n\|_\infty, \|x_g^{\text{corr}}\|_\infty, 10^{-30}).$$

For `Explicit`, the groups correspond to displacement and velocity DOFs. For `IMEX`, the pore-pressure and pore-rate groups are also included.

Here the group index g denotes a **subset of DOFs/state components** used in the norm, i.e. x_g is the restriction of the state vector to that group. In practice the groups are:

- displacement DOFs u_{disp} (DisX/DisY/DisZ)
- pore-pressure DOFs u_{pore} (PW/PA)
- velocity/rate DOFs v_{disp} (time-derivative of Dis*)
- pore-pressure rates v_{pore} (time-derivative of PW/PA)

The reported scalar `err_est` is constructed so that the acceptance test is
err_est $\le$ adaptiveTarget
with `adaptiveTarget` initialized from `@@ErrorTarget`.

Practical interpretation: `@@ErrorTarget = 1e-3` targets a predictor–corrector difference on the order of **0.1% of the current solution magnitude** (with a small internal absolute floor when magnitudes are near zero). For IMEX, the indicator includes both the solid and pore-pressure parts of the state.

1.12 Appendix A: Fixed Internal Parameters

The following 28 parameters use **fixed default values** and are not user-adjustable. They have been carefully calibrated for robust performance across a wide range of problems. If you attempt to set these in your input file, FALCON will issue a warning and use the default values instead.

1.12.1 A.1 Global Controller Tolerances (5 parameters)

Parameter	Default	Type	Role
`AutoTolerance`	`1.0e-12`	`double`	Tolerance for determining when remaining step time is negligible
`SmallResidualFactor`	`0.95`	`double`	Small residual threshold = `errorTarget * 0.95` (used by global convergence watcher)
`InitialStepFallbackDivisor`	`1.0e5`	`double`	If `InitialStepIncrement` not provided, use `stepTime / divisor`
`LargeResidualFactor`	`2.0`	`double`	Large residual threshold = `errorTarget * 2.0` (indicates poor convergence)
`RefIterFraction`	`0.5`	`double`	Reference iteration = `fraction * maxIterations`. Convergence faster than this suggests Δt can grow

1.12.2 A.2 Retry and Backoff Logic (9 parameters)

Parameter	Default	Type	Role
`RetryClampUpper`	`0.9`	`double`	Upper clamp on adjustment factor after retry (prevents barely shrinking)
`RetryClampLower`	`0.5`	`double`	Lower clamp on adjustment factor after retry (prevents instant collapse)
`JustAfterRetryFactor`	`0.8`	`double`	Override adjustment factor for the first successful step after a retry
`RetryBackoffBaseCoeff`	`0.9`	`double`	Numerator in backoff formula: `coeff / decayBase^retryCount * sqrt(target/fnorm)`
`RetryBackoffDecayBase`	`2.0`	`double`	Denominator base (exponential decay): each retry halves reduction (1, 1/2, 1/4, 1/8...)
`ShrinkFailureBase`	`0.6`	`double`	Exponential shrink factor per failure streak: `base^failureStreak` (0.6, 0.36, 0.216...)

Parameter	Default	Type	Role
`GrowthCooldownInc`	`3`	`int`	Cooldown counter increment after each failure
`PrevFnormGrowthThreshold`	`1.05`	`double`	If current `fnorm > threshold * previous Fnorm`, shrink dynamic max and add cooldown
`MinIncrementAbsolute`	`1.0e-20`	`double`	Absolute lower bound for Δt. If reached, analysis aborts (prevents infinite shrinking)

1.12.3 A.3 Adjustment Factor and Growth Shaping (4 parameters)

Parameter	Default	Type	Role
`FnormSignificanceThreshold`	`1.0e-14`	`double`	If `fnorm > threshold`, compute adjustment from residual ratio; else use 1.0
`AdjustClampMin`	`1.0e-3`	`double`	Minimum adjustment factor (prevents dt collapse on single poor convergence)

Parameter	Default	Type	Role
`GrowthCapSlope`	`0.02`	`double`	Success-based growth cap slope: `1 + slope * (successStreak - 1)` (2% per success)
`SuccessStreakGrowthThreshold`	`5`	`int`	Number of consecutive successes required before relaxing the dynamic max cap

1.12.4 A.4 Dynamic Max Relaxation (6 parameters)

Parameter	Default	Type	Role
`LowFnormRatioForRelax`	`0.2`	`double`	Residual must be < `0.2 * adaptiveTarget` to trigger relaxation
`SafetyMinNumerator`	`1.0e-16`	`double`	Floor for safety ratio numerator: `safety = max(fnorm, numerator) / max(target, denominator)`
`SafetyMinDenominator`	`1.0e-12`	`double`	Floor for safety ratio denominator (prevents division issues)
`RelaxFactorSlope`	`0.01`	`double`	Slope in relax factor formula (1% per eligible success)

Parameter	Default	Type	Role
RelaxFactorSuccessOffset	4	int	Success count offset: max(successStreak - 4, 0) for relax computation
RelaxFactorSuccessCap	6	int	Cap on effective success count for relaxation (limits to 6)

1.12.5 A.5 Smoothing and Final Corrections (4 parameters)

Parameter	Default	Type	Role
MaxGrowthFromLastSuccessful	1.02	double	Cap new dt at 1.02 * lastSuccessfulDt (prevents single-step jumps > 2%)
SmoothingNewWeight	0.2	double	EMA weight for new proposed dt (complements SmoothingHistoricalWeight)
OvershootTolerance	0.0	double	Tolerance for clamping tiny time accumulation overshoots

Parameter	Default	Type	Role
`MaxFinalCorrections`	`3`	`int`	Maximum attempts for final time correction if cumulative time deviates from target

1.12.6 Summary Table: All Fixed Internal Parameters

#	Category	Parameter	Default	Type
1	Tolerance	`AutoTolerance`	`1.0e-12`	`double`
2	Tolerance	`SmallResidualFactor`	`0.95`	`double`
3	Tolerance	`LargeResidualFactor`	`2.0`	`double`
4	Tolerance	`InitialStepFallbackDivisor`	`1.0e5`	`double`
5	Tolerance	`RefIterFraction`	`0.5`	`double`
6	Retry	`RetryClampUpper`	`0.9`	`double`
7	Retry	`RetryClampLower`	`0.5`	`double`
8	Retry	`JustAfterRetryFactor`	`0.8`	`double`
9	Retry	`RetryBackoffBaseCoeff`	`0.9`	`double`
10	Retry	`RetryBackoffDecayBase`	`2.0`	`double`
11	Retry	`ShrinkFailureBase`	`0.6`	`double`
12	Retry	`GrowthCooldownInc`	`3`	`int`
13	Retry	`PrevFnormGrowthThreshold`	`1.05`	`double`

#	Category	Parameter	Default	Type
14	Retry	`MinIncrementAbsolute`	`1.0e-20`	`double`
15	Growth	`FnormSignificanceThreshold`	`1.0e-14`	`double`
16	Growth	`AdjustClampMin`	`1.0e-3`	`double`
17	Growth	`GrowthCapSlope`	`0.02`	`double`
18	Growth	`SuccessStreakGrowthThreshold`	`5`	`int`
19	Relax	`LowFnormRatioForRelax`	`0.2`	`double`
20	Relax	`SafetyMinNumerator`	`1.0e-16`	`double`
21	Relax	`SafetyMinDenominator`	`1.0e-12`	`double`
22	Relax	`RelaxFactorSlope`	`0.01`	`double`
23	Relax	`RelaxFactorSuccessOffset`	`4`	`int`
24	Relax	`RelaxFactorSuccessCap`	`6`	`int`
25	Smoothing	`MaxGrowthFromLastSuccessful`	`1.02`	`double`
26	Smoothing	`SmoothingNewWeight`	`0.2`	`double`
27	Smoothing	`OvershootTolerance`	`0.0`	`double`
28	Smoothing	`MaxFinalCorrections`	`3`	`int`

Total fixed internal parameters: 28

Note: These parameters are provided for reference and transparency. Users should focus on tuning the 5 adjustable parameters in Section 4 and the primary controls in Section 3.
