Jerk-Limited Trajectory Planning in LinuxCNC (PLANNER_TYPE=2): Mathematical Specification for Community Review

apogee fork — branch automata/jerksplan

August 3, 2026 — draft for review

Abstract

This document collects, derives, and cross-references the mathematics used by the jerk-limited ("S-curve") trajectory planning in the apogee LinuxCNC fork, and the mathematics *proposed* for its successor. Planner numbering: PLANNER_TYPE 0 = trapezoidal and 1 = S-curve/Ruckig are **frozen**; all new work lands under a new selectable PLANNER_TYPE=2, which begins life as a behavioral clone of planner 1 and then receives the corrections marked PROPOSED or "implemented (planner 2)" throughout — the reverse-run direction fix (§6) and the jerk-limited degraded mode (§5) have already landed — with one deliberate exception: spindle *position* sync retains the trapezoidal law (decision recorded in §13). Every implemented equation carries a source reference (file:line on the branch above); proposed equations are marked PROPOSED. Numbered questions for reviewers are collected in §20. Nothing in this document changes planner 0/1 behavior.

1 Notation and conventions

Symbol	Meaning
$s(t)$, $v(t)$, $a(t)$, $j(t)$	path position, velocity, acceleration, jerk (arc-length parameterization)
$v_{\max}$, $a_{\max}$, $j_{\max}$	configured bounds (machine units, s, s^2, s^3)
T	servo period (tc->cycle_time), typically 10^{-3} s
R	radius of a circular segment; $\kappa = 1/R$ curvature; $\kappa' = d\kappa/ds$ sharpness
φ	total included angle of an arc (rad)
a_t, a_n	tangential / normal (centripetal) acceleration components
$\hat{T}, \hat{N}, \hat{B}$	Frenet frame (tangent, normal, binormal)
$c_n = \sqrt{3}/2$	BLEND_ACC_RATIO_NORMAL (normal-acceleration budget ratio)
$c_t = 1/2$	BLEND_ACC_RATIO_TANGENTIAL
f	feed override factor, $f \in [0, f_{\max}]$, $f_{\max}$ = [DISPLAY]MAX_FEED_OVERRIDE

The coordinated planner is one-dimensional in arc length: the Ruckig/cruckig solver plans $s(t)$ subject to $(v_{\max}, a_{\max}, j_{\max})$ and never sees geometry (tp.c:2909--2919); geometry enters only through velocity caps (§7) and the tangential acceleration budget. Per-axis motion is obtained by evaluating the segment geometry at s.

2 One-dimensional jerk-limited kinematics

Within any constant-jerk phase (`sp_scurve.c:642--653`):

$$s(t) = s_0 + v_0 t + \tfrac{1}{2} a_0 t^2 + \tfrac{1}{6} j t^3 \tag{1}$$

$$v(t) = v_0 + a_0 t + \tfrac{1}{2} j t^2 \tag{2}$$

$$a(t) = a_0 + j t \tag{3}$$

A full profile concatenates up to seven such phases (Ruckig's canonical form); the general boundary-value problem (arbitrary $v_0, a_0 \to v_f, a_f$ over distance L) is solved numerically by the vendored cruckig library. The closed forms below are the analytic special cases the planner uses directly.

2.1 Acceleration request toward a velocity target

Free-mode stepping (`sp_scurve.c:516--546`) chooses the acceleration that can still be ramped to zero exactly when the velocity error closes. Requiring $\Delta v = \int a\,dt$ with a ramping down at $j_{\max}$ from a_r gives $\Delta v = a_r^2/(2j_{\max})$ plus the in-cycle term, yielding

$$a_r \;=\; -j_{\max} T + \sqrt{2 j_{\max}\,\Delta v + (j_{\max} T)^2}, \qquad \Delta v = v_{\text{target}} - v \;>\; 0, \tag{4}$$

mirrored exactly for $\Delta v < 0$, with a small dead-band ($|\Delta v| \le 10^{-3} j_{\max} T^2 \Rightarrow a_r = 0$), then clamped to $\pm a_{\max}$ and slew-limited by $|\dot a| \le j_{\max}$.

2.2 Jerk-limited stopping distance

From state (v, a) the stop proceeds in up to four phases (`sp_scurve.c:587--640`): (i) if $a > 0$, ramp $a \to 0$ at $-j_{\max}$; (ii) ramp to peak deceleration

$$a^* \;=\; -\min\Big(a_{\max},\ \sqrt{v j_{\max} + \tfrac{1}{2} a^2}\Big), \tag{5}$$

(iii) hold a^* until the remaining velocity equals the phase-(iv) budget $\Delta v_{\text{iv}} = (a^*)^2/(2j_{\max})$; (iv) ramp $a \to 0$ at $+j_{\max}$, reaching $v = 0$. Distances accumulate via (1). Equation (5) is the peak deceleration for which phases (ii)+(iv) exactly consume v ("triangular" profile) when it is below $a_{\max}$.

2.3 Deceleration time

For a stop from cruise ($a_0 = 0$) (`sp_scurve.c:388--434`):

$$T_1 = \frac{a_{\max}}{j_{\max}}, \qquad T_2 = \max\Big(0,\ \frac{v - a_{\max}^2/j_{\max}}{a_{\max}}\Big), \qquad t_{\text{stop}} = 2T_1 + T_2, \tag{6}$$

with the triangular case $v < a_{\max}^2/j_{\max} \Rightarrow T_1 = \sqrt{v/j_{\max}},\ T_2 = 0$.

2.4 Peak velocity over a distance; velocity in a time

`findSCurveVPeak`$(a_{\max}, j_{\max}, L)$ = the maximum velocity reachable from rest within distance L (used for blend arcs, §9) — implemented as the cruckig-solved peak additionally clamped by the trapezoidal closed form, $\min(v_{\text{cruckig}},\ \sqrt{a_{\max} L})$ (`blendmath.h:261--280` wrapping `sp_scurve.c:288--340`) — and `calcSCurveSpeedWithT`$(a_{\max}, j_{\max}, t)$ = velocity reachable from rest in time t, cruckig-evaluated (`sp_scurve.c:453--505`). The exact triangular-case (jerk-only, symmetric) closed forms used in derivations below are

$$v(t) = \tfrac{1}{4} j_{\max} t^2 \;\; (t \le 2a_{\max}/j_{\max}), \qquad L(v) = \frac{v^{3/2}}{\sqrt{j_{\max}}} \quad (\text{exact: } T_1{=}\sqrt{v/j_{\max}},\ L = vT_1). \tag{7}$$

Caution for reviewers: a recurring in-code quantity is $s_1 = \frac{1}{6} j_{\max} T_1^3 = \frac{1}{6}\, v^{3/2}/\sqrt{j_{\max}}$ — the distance of a *single* jerk phase, exactly 1/6 of the true triangular stopping distance $L(v)$. Formulas derived from s_1 ((30)) are therefore *permissive* approximations; see the note there.

3 How constraints are applied: per-axis vs. trajectory-wide

Two constraint families exist and are applied at *different pipeline stages*:

	Per-axis / per-joint	Trajectory-wide
Source	`[AXIS_L]` / `[JOINT_N]` keys	`[TRAJ]` keys, F-word, override
Applied	**queue time** (canon projection)	**run time** (per-cycle clamps)
Produces	per-segment scalars $(v, a, j)_{\text{seg}}$	effective $(v, a, j)_{\text{eff}}$ each cycle

The 1-DOF solver (§2) only ever sees the product of the two: per-segment scalars further clamped by the trajectory-wide values.

3.1 Queue time: projecting axis limits onto a straight segment

For a straight move with per-axis displacements d_k and axis limits (v_k, a_k, j_k), canon computes the *characteristic time* each axis needs under its own limit, takes the slowest, and converts back to a path scalar (`emccanon.cc:663--1000`):

$$t_v^* = \max_k \frac{|d_k|}{v_k}, \quad v_{\text{seg}} = \frac{d_{\text{tot}}}{t_v^*}; \qquad t_a^* = \max_k \sqrt{\frac{|d_k|}{a_k}}, \quad a_{\text{seg}} = \frac{d_{\text{tot}}}{(t_a^*)^2}; \qquad j_{\text{seg}} = \frac{d_{\text{tot}}}{(t_j^*)^3} \text{ per (38)}, \tag{8}$$

with d_{tot} the Cartesian path length (falling back to the UVW norm, then the ABC norm for pure-angular moves — the same fallback selects length vs. angle unit conversion). **Why this is exact for lines:** on a straight segment every axis moves in fixed ratio d_k/d_{tot}, so axis k's peak velocity is $v_{\text{seg}}\,|d_k|/d_{\text{tot}} = |d_k|/t_v^* \le v_k$ by construction of the max — and identically for acceleration and jerk, since scaling a 1-DOF profile scales all derivatives by the same ratio. The per-axis box constraint is therefore *provably satisfied at queue time*, which is why no per-axis check is needed at run time for straight segments. Mixed linear+angular moves compare characteristic times across unit systems (a known wart, §15).

3.2 Queue time: projecting axis limits onto an arc

For arcs the fixed-ratio argument fails (axis participation is sinusoidal), so canon uses the conservative plane bound (`emccanon.cc:3200--3300`):

$$v_{\text{axes}} = \min(v_1, v_2), \qquad a_{\text{axes}} = \min(a_1, a_2), \qquad j_{\text{seg}} = \min(j_1, j_2), \tag{9}$$

over the two in-plane axes (third axis folded in under an active coordinate rotation). Because a circle's axis velocity is $v\sin(\cdot) \le v$, bounding the *path* speed by the smaller axis limit bounds every axis. Canon then splits the acceleration budget between centripetal and tangential use, $a_n \le \frac{\sqrt{3}}{2} a_{\text{axes}} \Rightarrow v_{\text{radial}} = \sqrt{\frac{\sqrt{3}}{2} a_{\text{axes}} R}$, and bounds the helical/auxiliary contribution by the straight-move projection (8) applied to the chord: the segment velocity is the total arc length divided by the slowest of the planar and helical characteristic times. The TP re-derives the a_n/a_t split at its own level via (20) using the *effective minimum* radius. Note the asymmetry (defect, §15): velocity and acceleration get the full helical treatment, jerk gets only the planar min.

3.3 Run time: trajectory-wide clamps

Each cycle, the per-segment scalars are reduced — never raised — by the trajectory-wide chain:

$$v_{\text{eff}} = \min\left(f \cdot v_{\text{req}},\ \ v_{\text{seg}},\ \ v_{\lim}\right) \qquad \text{(15): F-word, override, slider} \tag{10}$$

$$a_{\text{eff}} = a_{\text{seg}} \cdot \left(1 - \max(\rho_{\text{kink}}, \rho'_{\text{kink}})\right) \cdot \left[\tfrac{1}{2} \text{ if parabolic}\right] \cdot \left[r_a \text{ if circular}\right] \qquad \texttt{tc.c:67--97} \tag{11}$$

$$j_{\text{eff}} = \min\left(j_{\text{seg}},\ J_{\text{traj}}\right) \qquad \texttt{tp.c:2758} \tag{12}$$

where J_{traj} is the `[TRAJ]MAX_LINEAR_JERK` scalar, ρ_{kink} the reserved kink-acceleration fractions of §10, and r_a the tangential ratio of (20). The geometric caps of §7–9 currently use J_{traj} *alone* rather than (12) — so two different jerk values can govern one segment (planner-2 fix: use j_{eff} uniformly).

3.4 What is deliberately not enforced at run time

(i) *Coordinated per-axis limits*: exact by construction for lines (§above), conservative for arcs; the only true runtime per-axis computation is the kink projection (31), which tests the acceleration *step* against per-axis bounds (`tp.c:188--209`) — there is no per-axis *jerk* accessor at all (defect C4, planner-2 item). (ii) *Per-joint limits under non-trivial kinematics*: coordinated motion is planned in axis space; joint velocity/acceleration/jerk through a nonlinear Jacobian is monitored, not enforced (consistent with stock LinuxCNC; integrators derate axis limits). (iii) *Free mode is the opposite regime*: jogging/homing runs one 1-DOF profile per joint (or axis for teleop) using the `[JOINT_N]`/`[AXIS_L]` values *directly* — per-joint constraints are exact there, and the trajectory-wide values enter only as clamps (including the destructive `control.c:1223--1226` clamp planner 2 replaces with non-mutating evaluation).

4 Look-ahead: backward reachability and the override invariant

For consecutive segments the optimizer computes the maximum entry velocity of the downstream segment that still permits reaching its exit constraint (`tp.c:1708--1755`):

$$v_s \;=\; \min\Big(\,\texttt{findSCurveMaxStartSpeed}(L,\; v_f,\; a_t,\; j_e),\;\; \sqrt{v_f^2 + 2a_tL}\,\Big), \qquad j_e = \min(j_{\text{seg}},\; j_{\max}), \tag{13}$$

i.e. the jerk-aware backward-reachable speed, additionally clamped by the trapezoidal bound. The previous segment's planned exit velocity is further capped by the peak velocity realizable *within that segment* under jerk (`tp.c:1746--1751`):

$$v_{\text{final}}^{(i-1)} \;\le\; \texttt{findSCurveVPeak}\big(a_t^{(i-1)},\, j_e^{(i-1)},\, L^{(i-1)}\big). \tag{14}$$

4.1 Curved segments in the backward pass

The optimizer never handles geometry directly: a curved segment presents itself to (13)–(14) as three scalars, all curvature-derived at queue time:

1. **Velocity ceiling** $v_{\max}^{\text{seg}}$: the output of §7's caps (20), (22)–(24) (applied to programmed arcs *and* blend arcs at finalization, `tc.c:936--979`).
2. **Tangential acceleration budget**: the backward pass uses $a_t = r_a\, a_{\max}$ with $r_a = \sqrt{1-(a_n/a_{\max})^2}$ evaluated at the capped speed (`acc_ratio_tan`, `tc.c:85--97`) — deceleration capacity along a curve is reduced by exactly the acceleration spent turning.
3. **Boundary ceilings**: at approximately-tangent joins the optimizer's v_f limit is $\min(v_{\max}^{\text{seg}},\; v_{\text{kink}})$ (`tp.c:1727--1734`), so corner physics caps what look-ahead may promise across the join.

For spiral/helical segments the radius entering the caps is the *effective minimum* radius over the span (`pmCircleEffectiveMinRadius`, `blendmath.c:1854--1870`) and the segment length is the conservative (over-estimating) quadratic fit (`SpiralArcLengthFit`), so reachability computed by (13) errs on the safe side: true speed along the curve is never faster than planned. This scalar reduction is what makes the look-ahead recursion geometry-independent — and it is exactly why its guarantees are only as good as the per-segment caps that produced the scalars.

Override invariant (existing; to be asserted, not changed). Planned boundary velocities are *absolute* caps; execution applies feed/rapid override only as

$$v_{\text{exec,final}} \;=\; \min\big(v_{\text{final}},\; \min(\tilde{v}_{\text{this}},\, \tilde{v}_{\text{next}})\big), \qquad \tilde{v}_i = \min\big(f\, v_i,\; v_i^{\max f},\; v_{\text{lim}}\big) \qquad (\texttt{tp.c:280--346}), \tag{15}$$

where $v_i^{\max f}$ is the segment cap evaluated at $f_{\max}$ and v_{lim} the machine velocity-slider limit; $v_{\text{exec,final}}$ is forced to zero when single-stepping, in reverse run, or at any non-tangent termination. Worst-case planning uses $f_{\max}$ (`tp.c:295--317`). Since every additional clamp only reduces velocity, the look-ahead result is valid for every $f \in [0, f_{\max}]$ and override changes never require queue replanning; they are absorbed by per-segment replans from the current (s, v, a) state with jerk limits intact.

5 Feed and rapid override under planner 2

Planner 2 keeps the two-level architecture: overrides never touch the queue-time plan (§4), and enter execution only as target changes absorbed by jerk-limited replanning.

5.1 Execution mechanics (implemented, inherited from planner 1)

Each servo cycle the effective target velocity is recomputed, $\tilde{v} = \min(f\, v_{\text{req}},\; v^{\max f},\; v_{\text{lim}})$; any change beyond 10^{-8} triggers a position-mode replan of the 1-DOF profile from the *current* state (s, v, a) toward $\big(s_{\text{end}},\; v_{\text{final}},\; a{=}0\big)$ with the segment's $(a_{\max}, j_e)$ (`tp.c:2894--2919`). Consequently an override step is acceleration-continuous by construction and its jerk is bang-bang bounded by j_e — *provided the replan succeeds.* The exception is the plan-failure path: on a failed solve planner 1 falls back to a raw trapezoidal cycle (unbounded jerk), and aggressive override steps taken from high-acceleration states are its most likely trigger. Planner 2 replaces that fallback with an analytic jerk-limited stop (implemented), which closes the unbounded-jerk hole; the stale-plan continuation cycle remains outstanding. Subject to that, no override change can violate the jerk limit. Feed hold and override-to-zero switch the profile to velocity mode — a jerk-limited deceleration to $(v, a) = (0, 0)$ — and resume replans from rest (`tp.c:2782--2887`). Rapid override uses the same machinery: the rapid scale is folded into the active motion's factor f (`control.c:376`), so (15) covers G0 segments unchanged, and a feed↔rapid boundary with unequal overrides is just another bounded target step at a segment join (the boundary cap takes the minimum of both scaled targets).

Two intentional scale discontinuities exist and are absorbed by the same replans: blending clamps $f \le 1$ during parabolic overlap (`tp.c:253--255, 298--301`), and the active-motion-type scale switch at feed/rapid joins.

Validation range. The regression matrix (plan, §P0.1) exercises override steps across the *full* $f \in [0, 2]$ range on all three planners: $100\% \to 0$ (override-to-zero must behave exactly as the velocity-mode hold above), $0 \to 200\%$ (the largest admissible target step — standstill toward $\min(2v_{\text{req}}, v_{\text{seg}}, v_{\text{lim}})$ — the worst case for replan severity), intermediate steps (25, 50, 150%), the rapid-override analogues including RO=0 hold-and-resume, and mixed FO≠RO feed↔rapid joins with one scale at 0 while the other is at 200%. The thrice-differentiated $|j| \le j_e$ assertion is carried by the *planner-2* runs only (report-only until the corresponding D2/D3 fixes land, hard thereafter); planner-0/1 runs are frozen-baseline characterization with telemetry recorded and no jerk bound asserted, because these very scenarios are the documented triggers of the frozen planner's defects. Delivered so far: the feed sweep (`tests/planner2/override-range`) and the rapid sweep (`tests/planner2/rapid-override`); the latter confirms (15) empirically — at 200% the commanded speed stays pinned at the planned rapid cap rather than rising through it.

5.2 proposed (planner 2) additions

(a) Override slew shaping. The scale signal itself is applied unramped today; a step $0.6 \to 2.0$ becomes a single maximum-jerk transient. Planner 2 adds optional shaping of $f(t)$ with a configurable slew rate $\dot{f}_{\max}$ (HAL parameter, %/s; default off), bounding each replan's severity and making override feel rate-independent of when the operator turns the knob.

(b) Horizon sizing at maximum override. The look-ahead window must always cover the jerk-limited stopping distance from the *worst-case* speed $v^* = f_{\max} v_{\max}$ (clamped by $v_{\lim}$). From (6), average velocity over a symmetric stop is $v/2$, giving the closed form

$$s_{\text{stop}}(v) \;=\; \begin{cases} \dfrac{v^2}{2a_{\max}} + \dfrac{v\,a_{\max}}{2j_{\max}}, & v \geq a_{\max}^2/j_{\max} \quad \text{(full S-curve)} \\[2ex] \dfrac{v^{3/2}}{\sqrt{j_{\max}}}, & v < a_{\max}^2/j_{\max} \quad \text{(triangular)} \end{cases} \tag{16}$$

— the jerk term $va_{\max}/2j_{\max}$ is exactly the extra distance versus a trapezoidal stop. Look-ahead depth validation and starvation telemetry are evaluated at $s_{\text{stop}}(v^*)$, not at programmed speed.

6 Reverse run under jerk limitation

Reverse run — retracing the executed path backward, driven by a negative adaptive-feed value — is a recovery feature (plasma/torch restarts, operator back-out) that planner 2 implements and planner 1 refuses. The mathematics is a direction-mapping layer over the unchanged 1-DOF machinery; it is presented here because both of its defects were found on exactly this path, one statically and one empirically.

6.1 Semantics and the engage sequence (implemented)

The scale and the direction are separated at the source (`control.c:381--408`): the applied factor is $f_{\text{out}} = |f_{\text{in}}|$ always, and a *sign change* requests a direction switch, which is accepted only at standstill:

$$f_{\text{in}} < 0 \;\wedge\; \text{dir} = \text{FWD} \;\Rightarrow\; \begin{cases} \text{switch accepted} & \text{if } v = 0 \\ f_{\text{out}} := 0 & \text{otherwise (jerk-limited stop first)} \end{cases} \tag{17}$$

so a direction change always decomposes into: velocity-mode stop (§5, duration and distance per (6)/(16)) → direction flip at rest → position-mode plan in the new direction. Segment bookkeeping is direction-abstracted: the reverse target of a segment is 0 and the distance-to-go is sign-positive in both directions (`tc.c:395--407`); completed segments are retained as history and re-entered backward (`tcqBackStep`, 200-segment margin).

6.2 The direction mapping (implemented, planner 2)

Let $\sigma = +1$ forward, -1 reverse. Planner 2 plans on the signed progress axis:

$$s_{\text{target}} = s + \sigma\, d_{\text{togo}} \;=\; \begin{cases} L & \sigma = +1 \\ 0 & \sigma = -1 \end{cases} \qquad (v,a)_{\text{plan}} = \sigma\,(v,a)_{\text{state}}, \qquad (v,a,j)_{\text{out}} = \sigma\,(v,a,j)_{\text{plan}}, \tag{18}$$

with the commanded position taken from the plan directly. Stored state keeps positive-domain magnitudes, so every downstream consumer (status, gating, blend logic) is unaffected, and $\sigma = +1$ reduces (18) to the identity — planner-2 forward behavior is bit-identical to planner 1 on non-position-synced motion (held continuously by the parity gate, whose program corpus therefore

excludes G33/G33.1 and rigid tap; those diverge deliberately per §13). Additionally the solver's admissible velocity interval must contain the signed cruise:

$$v \in [\, v_{\min},\ f\, v_{\text{req}} \,], \qquad v_{\min} = \begin{cases} 0 & \sigma = +1 \\ -f\, v_{\text{req}} & \sigma = -1. \end{cases} \tag{19}$$

The defect pair this fixes (both planner-1-latent, hence the refusal gate there): D1a — the unsigned target $s + d_{\text{togo}}$ evaluates to $2s$ in reverse and plans *forward*; D1b — found empirically only after fixing D1a: the "unidirectional" interval $v_{\min} = 0$ makes every backward plan infeasible at the solver level, which the degraded mode (§2 fallback) masks as a safe permanent hold. The lesson generalizes: direction correctness requires *both* the boundary conditions (18) and the constraint set (19); auditing one without exercising the solver misses the other.

6.3 Validated behavior and current conservatism

Empirically locked by runtime tests: in-segment reversal (smooth, jerk-limited, completes after restore), refusal-and-hold under planner 1, and *backward traversal across a completed segment boundary* — a full back-step through a blended corner to program start with clean recovery. Known conservatisms, stated plainly: (i) boundary velocity in reverse is forced to zero ((15) preconditions), so reverse executes stop-and-go at every segment boundary rather than blending backward; (ii) the engage transition inherits the end-of-trajectory snap discontinuity (D7 class) visible in sampled $|\Delta a|/T$ telemetry; (iii) spindle-synchronized and rigid-tap segments refuse reverse activation (correct — the spindle cannot be un-threaded by adaptive feed).

6.4 proposed: reverse look-ahead

Because reverse is a recovery feature, stop-and-go may be acceptable — but the machinery for better exists: the backward-reachability pass of §4 applies symmetrically to the *history* chain. Running (13)–(14) over the retained segments with reverse targets (0-end boundaries, same tangent/kink caps at the joins traversed backward) yields nonzero reverse boundary velocities with identical safety guarantees, at the cost of maintaining a second optimization direction. Whether reverse deserves that investment is reviewer question **Q10**.

7 Jerk constraints for circular motion (implemented)

Steady circular motion at speed v has $a_n = \kappa v^2 = v^2/R$. The acceleration budget is split (`tc.c:869--885`):

$$a_n \le c_n a_{\max} \ \Rightarrow\ v \le \sqrt{c_n a_{\max} R}, \qquad \left.\frac{a_t}{a_{\max}}\right|_{\text{allowed}} = \sqrt{1 - (a_n/a_{\max})^2}. \tag{20}$$

For the S-curve family three further caps apply (`tc.c:887--929`). They follow from the Frenet jerk decomposition: for a path with curvature $\kappa(s)$ and speed v,

$$\frac{d\vec{a}}{dt} \;=\; \big(\dot{a}_t - \kappa^2 v^3\big)\hat{T} \;+\; \big(3\kappa v a_t + \kappa' v^3\big)\hat{N} \;+\; \kappa\tau v^3 \hat{B}. \tag{21}$$

Constraint 1 — steady rotational jerk with transition budget. On a circle ($\kappa' = 0$, $\tau = 0$, $a_t = 0$) the rotating a_n contributes $|\, d\vec{a}/dt \,| = \kappa^2 v^3 = v^3/R^2$. The implementation shares the jerk budget between the sustained rotation (weight φ) and the two entry/exit transitions (weight 2):

$$v \;\le\; \left(\frac{R^2\, j_{\max}\, \varphi}{2 + \varphi}\right)^{1/3} \qquad (\texttt{tc.c:898}). \tag{22}$$

Constraint 2 — tangential–normal coupling. With $a_t \neq 0$ on a circle, the normal jerk component of (21) is $3\kappa v a_t$. Bounding it by $j_{\max}$ with the tangential budget $a_t \le c_t a_{\max}$:

$$3\,\frac{v\,a_t}{R} \le j_{\max} \;\Rightarrow\; v \;\le\; \frac{j_{\max} R}{3\,c_t a_{\max}} \qquad (\texttt{tc.c:903--904}). \tag{23}$$

Constraint 3 — entry/exit curvature step. At a line–arc join, curvature steps $0 \to 1/R$, so a_n steps by v^2/R within one servo period; requiring the implied rate to stay within $j_{\max}$:

$$\frac{v^2/R}{T} \le j_{\max} \;\Rightarrow\; v \;\le\; \sqrt{j_{\max} R\,T} \qquad (\texttt{tc.c:909}). \tag{24}$$

The final cap is min of (22)–(24) and (20); a_t's allowed ratio is then recomputed at the capped speed.

Known deficiencies (to be fixed in planner 2, §11). (i) (24) is servo-period-dependent: halving T cuts permitted corner speed by $\sqrt{2}$ for identical G-code. (ii) (24) bounds each side separately; an arc–arc join with curvature change $\Delta\kappa$ admits a combined one-cycle step up to $\approx 2 j_{\max} T$. (iii) These caps bound magnitudes but the executed profile still *steps* a_n discontinuously — the 1-DOF planner cannot shape normal jerk. (iv) The scalar $j_{\max}$ here is the global `[TRAJ]` value, not the per-segment or per-axis limit (see §15).

8 Jerk while moving along a curve

The previous section's caps are chosen at queue time; this section examines what the *executed* jerk looks like while the 1-DOF profile changes speed on curved geometry — the case look-ahead creates constantly (decelerating along an arc toward a corner, accelerating out of one).

8.1 Why the queue-time caps remain valid during traversal

With $v(t) \le v_{\max}^{\text{seg}}$ throughout the segment, every speed-dependent term of (21) is monotone increasing in v: $\kappa^2 v^3$, $3\kappa v a_t$, and $\kappa' v^3$ all attain their maxima at $v_{\max}^{\text{seg}}$. Hence caps (22)–(23) evaluated at the ceiling bound those components for the *entire* traversal, including all intermediate speeds commanded by look-ahead — no per-cycle re-evaluation is needed. The tangential ramp itself is bang-bang bounded, $|\dot a_t| \le j_e$, by the 1-DOF solver.

8.2 The component-sum gap (honest statement, planner-2 refinement)

Individually bounded components do not bound the *vector* jerk. During an in-curve speed change the magnitude is

$$\left|\tfrac{d\vec a}{dt}\right| \;=\; \sqrt{\left(\dot a_t - \kappa^2 v^3\right)^2 + \left(3\kappa v a_t + \kappa' v^3\right)^2}, \tag{25}$$

and with $\dot a_t = \pm j_e$ while the geometric terms are near their caps, the sum can transiently exceed j_e (bounded by $\sqrt{2}\,j_e$ when each family is capped at j_e — and note constraint 1's $\varphi/(2+\varphi)$ apportionment already under-allocates the sustained term, mitigating but not eliminating this). PROPOSED (planner 2): budget the components jointly, mirroring the acceleration treatment. Define a jerk partition $r_j \in (0,1)$ and enforce

$$\underbrace{|\dot a_t| \;\le\; r_j\, j_e}_{\text{1-DOF solver budget}} \qquad \text{and} \qquad \underbrace{3\kappa v a_t + \kappa' v^3 \;\le\; \sqrt{1-r_j^2}\; j_e}_{\text{velocity cap (23)/(39) with reduced budget}} \qquad \Rightarrow \quad \left|\tfrac{d\vec a}{dt}\right| \le j_e \;(+\,\kappa^2 v^3 \text{ term via (22)}), \tag{26}$$

i.e. a `jerk_ratio_tan` companion to `acc_ratio_tan`: on curved segments the solver receives $r_j j_e$ instead of j_e, and the curvature caps tighten by $\sqrt{1-r_j^2}$. The natural default mirrors the acceleration split ($r_j = c_t = 1/2$ on blends); whether r_j should be fixed, curvature-dependent, or simply folded into a documented $\sqrt{2}$ allowance is reviewer question **Q9**.

8.3 Execution mechanics on curves

Per cycle the scalar progress s maps to pose through the segment's arc-length parameterization: for circles/spirals the quadratic fit inverts by the numerically-stable Citardauq form and self-validates with a round-trip check at insertion (`blendmath.c:1695--1846`); the fit over-estimates total length, so commanded speed along the true curve errs slow. Replans triggered mid-curve (override change, look-ahead update) restart from the current (s, v, a) with acceleration continuity — geometry never enters the replan, so in-curve replan behavior is identical to straight-line behavior in profile space. The remaining executed discontinuities are exactly the boundary events of §14 (curvature steps and kinks at segment joins), handled today by the per-cycle bounds (24)/(31) and in planner 2 by the transition windows (33)/(32).

9 Blend-arc sizing under jerk (implemented)

For a corner of half-angle θ between two lines with blend tolerance ε (`blendmath.c:1137--1210`):

$$h_{\text{tol}} = \frac{\varepsilon}{1-\sin\theta}, \qquad d_{\text{tol}} = h_{\text{tol}}\cos\theta, \qquad d = \min(L_1, L_2, d_{\text{tol}}), \qquad R_{\text{geom}} = d\,\tan\theta. \tag{27}$$

The normal-limited speed on the blend arc is

$$v_{\text{normal}} = \begin{cases} \texttt{findSCurveVPeak}(a_{n,\max},\ j_{\max},\ R_{\text{geom}}) & \text{S-curve family} \\ \sqrt{a_{n,\max} R_{\text{geom}}} & \text{trapezoidal} \end{cases} \qquad (\texttt{blendmath.c:1156--1160}). \tag{28}$$

A parabolic-equivalent consumption metric sizes the radius to the segment (`blendmath.c:1172--1176`), and the jerk floor follows from (22) in the sustained limit (φ large, $v^3 \le R^2 j_{\max}$):

$$R_{\text{jerk,min}} = \frac{v_{\text{plan}}^{3/2}}{\sqrt{j_{\max}}} \qquad (\texttt{blendmath.c:1181--1186}), \qquad R_{\text{plan}} = \max\left(R_{\text{accel}},\ \max(R_{\text{blend}}, R_{\text{jerk,min}})\right), \tag{29}$$

with $R_{\text{accel}} = v_{\text{plan}}^2 / a_{n,\max}$.

For parabolic-blend triggering with a G64 tolerance, the implemented S-curve tolerance velocity solves the *single-jerk-phase* distance $s_1 = \frac{1}{6} v^{3/2}/\sqrt{j_{\max}}$ (see the caution at (7)) for v at $s_1 = 2\varepsilon/\cos\theta$:

$$v_{\text{tol}} = \left(6\,L\,\sqrt{j_e}\right)^{2/3}, \qquad j_e = \min(j_{\text{seg}}, j_{\max}) \qquad (\texttt{tp.c:2446--2459}). \tag{30}$$

Because s_1 under-states the true stopping distance by $6\times$, this velocity is *permissive* by $6^{2/3} \approx 3.30$ relative to the exact $v = (L\sqrt{j_e})^{2/3}$; the in-code comment's "lower bound" refers to distance, not velocity. Whether to keep the permissive form in planner 2 is reviewer question **Q8**. Note also this path uses the per-segment j_e, unlike the `tc.c`/`blendmath.c` caps which use the global scalar (§15).

10 Tangent (kink) joins

10.1 Implemented (planner 0/1, frozen)

An approximately tangent join with unit tangents $\hat{t}_1, \hat{t}_2$ is modeled as a one-cycle velocity-direction change (`tp.c:1946--2000`). With $v_j = \min(v_1^{\max f}, v_2^{\max f})$ the lesser of the two segments' maximum target velocities at maximum feed scale, and a_{seg} the incoming segment's acceleration limit:

$$a_{\text{inst}} = \frac{v_j}{T} + a_{\text{seg}}, \qquad \vec{a}_{\text{diff}} = a_{\text{inst}}\,(\hat{t}_2 - \hat{t}_1), \qquad \sigma = \max_k \frac{|a_{\text{diff},k}|}{a_{\text{bound},k}} \left[\ \sigma \to \sigma/c_t \text{ if either segment is circular}\right]. \tag{31}$$

If $\sigma < \rho$ (`kink_ratio`, default clamp $[10^{-3}, 0.7071]$, `tp.c:185--189`) the join is declared tangent at $v_{\text{kink}} = v_j$ with acceleration reservation σ; otherwise $v_{\text{kink}} = v_j\,\rho/\sigma$. **No jerk term appears anywhere in** (31): the implied one-cycle jerk $|\vec{a}_{\text{diff}}|/T$ is unbounded relative to $j_{\max}$.

10.2 proposed (planner 2): jerk-aware kink velocity

Spread the direction change over a fixed transition window τ (not one period), and bound the per-axis jerk of the resulting acceleration pulse. Modeling the velocity rotation $\Delta\hat{t} = \hat{t}_2 - \hat{t}_1$ at speed v as an acceleration pulse of duration τ ramped at the axis jerk limit:

$$a_k(\tau)\text{ pulse peak} \approx \frac{2\,v\,|\Delta t_k|}{\tau}, \qquad j_k \approx \frac{2a_k}{\tau} = \frac{4\,v\,|\Delta t_k|}{\tau^2} \;\le\; j_{\text{bound},k} \;\Rightarrow\; v_{\text{kink}} \;\le\; \min_k \frac{j_{\text{bound},k}\;\tau^2}{4\,|\Delta t_k|}. \tag{32}$$

combined with the frozen acceleration criterion (31) evaluated over τ instead of T. The window τ is a machine constant (proposed default $\tau = \max(4T,\ a_{\max}/j_{\max})$), making corner speeds servo-rate-independent. *Review requested: pulse-shape constant (the factors 2/4 above assume a triangular acceleration pulse); interaction with the existing ρ tradeoff.*

11 Fixed-time transition windows for curvature steps

PROPOSED (planner 2). Replace the per-cycle step bounds (24) by the same window τ:

$$v \;\le\; \sqrt{\frac{j_{\max}\,R\,\tau}{c_{\text{step}}}}, \qquad \text{and at arc–arc joins} \qquad \frac{v^2\,|\Delta\kappa|}{\tau} \le j_{\max} \;\Rightarrow\; v \le \sqrt{\frac{j_{\max}\tau}{|\Delta\kappa|}}, \tag{33}$$

with c_{step} a shape constant (1 for a linear ramp of a_n across the window; review requested), $|\Delta\kappa| = |\kappa_2 - \kappa_1|$ accounting both sides of the join. Execution spreads the a_n engagement across $N = \lceil \tau/T \rceil$ cycles (a planner-2 execution change; the 1-DOF profile is unchanged, the geometric evaluation blends κ over the window).

12 Parabolic blend overlap

Implemented (frozen): during a parabolic blend the outgoing and incoming segments run concurrent 1-DOF profiles; the *acceleration* budget is halved per side (`tc.c:72--77`) so the vector sum respects $a_{\max}$, but each side plans with full $j_{\max}$: worst-case combined path jerk is $2j_{\max}$.

PROPOSED (planner 2): apply the same split to jerk,

$$j_{\text{side}} = \tfrac{1}{2}\,j_{\max} \qquad (\text{or } j_{\text{side}} = j_{\max}/\sqrt{2} \text{ if the blend directions are nearly orthogonal — review requested}), \tag{34}$$

or disallow parabolic fallback entirely under planner 2 (forcing stop or blend-arc).

13 Spindle-synchronized motion

13.1 Execution architecture (implemented, inherited)

In the inherited architecture the sync layer runs before the per-cycle motion update and rewrites exactly one quantity — the segment's target velocity — which then flows into the *active planner's* ordinary replan machinery of §5:

$$\text{G95 (velocity sync):}\quad v_{\text{target}} = \omega_s\,d_{\text{rev}}, \qquad \text{G33/G33.1 (position sync):}\quad v_{\text{target}} = \omega_s\,d_{\text{rev}} \pm v_{\text{corr}}, \tag{35}$$

where ω_s is measured spindle speed (rev/s), d_{rev} the programmed distance per revolution, and v_{corr} the position-error correction below (`tp.c:3519--3605`). Because the spindle-derived target

changes essentially every servo cycle, synced segments replan every cycle. Under planner 1 that replan is jerk-limited, so the commanded motion keeps the jerk guarantee at the cost of added tracking lag; under planner 2 the response for *position* sync is deliberately second-order — the decision recorded in the next subsection. Protective interactions (implemented): feed override is forced to $f{=}1$ for position-synced segments (`tp.c:251--252`); pause never applies the velocity-mode stop to a position-synced segment (no mid-thread feed hold, `tp.c:243`); reverse run refuses synced and rigid-tap segments (`tp.c:3431--3434`); rigid-tap reversal rewrites the target mid-segment and is absorbed as a replan with continuous (v, a). The spindle itself has no motion profiling (drive-side concern; out of scope).

13.2 Tracking law and the planner-2 decision

Implemented (frozen): position-synced correction velocity

$$v_{\text{corr}} = \sqrt{|e|\; a_t^{\max}} \qquad (\texttt{tp.c:3595}), \tag{36}$$

the time-optimal second-order (acceleration-limited) surface.

Decision (adopted). Under planner 2, position-synchronized segments (G33, G33.1, rigid tap) keep both this correction law *and* the acceleration-limited (trapezoidal) per-cycle response — behavior identical to planner 0 inside the sync region. Rationale: during position sync the axis is slaved to the spindle, and a third-order (jerk-limited) response necessarily lags a second-order one whenever the spindle deviates from constant speed; during threading that lag *is* pitch error, the one quantity that cannot be traded away. Beating the lag would require spindle-acceleration feed-forward from a differentiated encoder signal (noise-amplifying), while the spindle's own inertia already band-limits the acceleration demanded of the axis, making the forgone jerk limiting partly redundant physically. The second-order law also carries decades of field validation on exactly this operation.

The resulting contract: **jerk limitation is suspended while the spindle owns the motion; it is enforced up to the sync engage point and re-established from the disengage state.** The engage and disengage transitions are ordinary planner-2 segments, so the acceleration state handed into and received back from the sync region is blended jerk-limited on the planner-2 side (the same accel-saturation spreading proposed for profile handoffs). The pitch-error A/B regression (`tests/planner2/sync-trapezoid`) asserts that planner-2 pitch error does not exceed the planner-0 baseline inside the sync region — a hard bound from day one, since both run the same correction law and the same second-order per-cycle response. Pitch error is evaluated intrinsically per run ($e = \Delta z + K\,\Delta n_{\text{rev}}$) under an injected spindle-speed disturbance, so no cross-run time alignment is needed. Representative runs: $\approx$0.0071 in (planner 2) versus 0.0069–0.0071 in (planner 0) — equal within the run-to-run spread of the wall-clock disturbance trigger — while planner 1's jerk-limited tracking measures $\approx$0.0202 in, about 2.8$\times$ worse on the same disturbance. That ratio is the empirical case for this decision. G95 *velocity* sync is the opposite case: feed-per-rev has no position lock and a transient deviation is a small, self-correcting surface-finish effect, so the spindle-speed-scaled target *does* stay on the jerk-limited profile — which additionally filters spindle-encoder speed noise out of axis acceleration. (RT note: the per-cycle replan-cost concern therefore applies to G95 segments only; the position-sync trapezoid update is cheap. The 1 kHz benchmark item in the plan stands for G95.)

CONSIDERED, NOT ADOPTED (retained for review): a third-order time-optimal surface — the acceleration-limited law clamped by the *exact* jerk-limited reachability of the error distance $|e|$ (exact triangular case of (7)):

$$v_{\text{corr}} \;=\; \min\!\Big(\sqrt{|e|\,a_t^{\max}},\;\; k_s\,\big(|e|\,\sqrt{j_{\max}}\big)^{2/3}\Big), \qquad k_s \in (0,1] \text{ damping gain}, \tag{37}$$

(deliberately not the permissive $6^{2/3}$-inflated form of (30)). Demoted to a research option: it trades pitch error for smoothness in exactly the regime where pitch error is the specification (see Q5).

14 Managing normal (geometry-induced) jerk

Normal jerk cannot be handled by the 1-DOF profile — it is curvature-driven and the executed profile is curvature-blind. Planner 2's management strategy is a three-mechanism decomposition, each targeting one term of the Frenet identity (21):

Sources. (i) *Within-curve, speed-dependent*: sustained rotation contributes $\kappa^2 v^3$ (rotating a_n) and speed change on curvature contributes $3\kappa v a_t$; on varying-curvature paths the sharpness term $\kappa' v^3$ adds. These scale with v and exist on *perfectly smooth* geometry — no preprocessor can remove them. (ii) *Boundary steps*: curvature discontinuities $\Delta\kappa$ at segment joins step a_n by $v^2\Delta\kappa$ (G1-but-not-G2 continuity). (iii) *Tangent kinks*: direction discontinuities demand an impulsive velocity rotation (§10).

Mechanism 1 — velocity caps (implemented). Bound each speed-dependent term by choosing v: constraints (22)–(23) for circles, the blend floor (29), and for general curves the span-minimum caps (39) including the sharpness bound. These guarantee magnitudes but not smoothness: the executed $a_n(t)$ still steps at joins.

Mechanism 2 — transition windows (proposed, the planner-2 execution change). Spread every residual discontinuity over the fixed time window τ: curvature steps via (33) with geometric κ-blending across the window, kinks via (32). This converts impulses into bounded-jerk pulses and makes corner behavior servo-rate-independent — the piece the current per-cycle bounds (24) approximate only at one fixed period.

Mechanism 3 — geometry improvement (upstream, complementary). The canon-layer segment fitter (liscio) reduces *event count*: micro-segment polylines (a kink per waypoint) become long primitives with zero interior joins, so Mechanism 2 fires per-primitive instead of per-waypoint. Division of labor, stated precisely: geometry reduces the *frequency and severity* of (ii)/(iii) — constant-curvature primitives still step κ at every boundary and the fitter propagates only tangent (G1) continuity — while the planner owns all of (i) and the residual steps. True elimination of (ii) requires curvature-continuous geometry (clothoid corner blends or G2-constrained curve fits, §17); even then Mechanism 1 remains, because (i) is speed-dependent physics, not geometry.

Normal-jerk source	Managed by	Status
$\kappa^2 v^3$ sustained rotation	cap (22)	implemented
$3\kappa v a_t$ accel coupling	cap (23)	implemented
$\kappa' v^3$ sharpness	cap (39)	proposed (curves)
$v^2\Delta\kappa$ boundary step	window (33); today (24)	proposed / partial
kink impulse	window (32); today (31)	proposed / partial
event frequency	upstream segment fitter	implemented (canon)
step elimination	G2 geometry (clothoid / curve fits)	future

15 Per-axis jerk propagation

Implemented (frozen): the segment scalar jerk is derived at queue time from per-axis limits. For a straight move with per-axis displacements d_k and axis jerks j_k (`emccanon.cc:663--772`):

$$t_k = \left(\frac{d_k}{j_k}\right)^{1/3}, \qquad t^* = \max_k t_k, \qquad j_{\text{seg}} = \frac{d_{\text{tot}}}{(t^*)^3}, \tag{38}$$

(the constant 6 from $d = \frac{1}{6}jt^3$ cancels in the ratio). For arcs the two in-plane axis jerks are considered, $j_{\text{seg}} = \min(j_1, j_2)$, with a third axis folded in only when a coordinate rotation is active on a non-XY plane (`emccanon.cc:3251--3267`; the segment-fitter arc path applies the same rule, `emccanon.cc:1332--1346`) — the helix axis and auxiliary axes are otherwise ignored (**defect**, planner-2 fix: include the helix axis and use the plane-projected form of (38)). At runtime the geometric caps (§7–9) use the global `[TRAJ]` scalar; planner 2 switches them to $\min(j_{\text{seg}}, j_{\max})$ and, where a plane is defined, to the projected per-axis bound analogous to the acceleration treatment.

Under non-trivial kinematics, per-*joint* jerk cannot be bounded by any Cartesian scalar (the Jacobian is position-dependent); planner 2 scopes joint-space jerk to monitoring, matching the existing treatment of joint velocity/acceleration.

16 Free-mode (jog/home) planner

Implemented: per-joint/axis 1-DOF stepping using (4) with stopping distance (5)ff. deciding the deceleration switch (`src/emc/motion/simple_tp.c:93--236`). **Defect (planner-2 fix):** terminal settling falls back to the trapezoidal stepper (unbounded jerk) on overshoot/status conditions; the v2 form completes the stop analytically with phases (ii)–(iv) of §2 and a documented position tolerance.

17 General curves (future, Point B / P5)

From (21), a varying-curvature path adds the sharpness term $\kappa' v^3$ to normal jerk. The planner-2 velocity cap for a curve segment over its span $[s_0, s_1]$ becomes

$$v \;\le\; \min_{s} \left[\left(\frac{j_{\max}}{\kappa(s)^2}\right)^{1/3},\;\; \frac{j_{\max}}{3\,\kappa(s)\,a_t^{\max}},\;\; \left(\frac{j_{\max}}{|\kappa'(s)|}\right)^{1/3} \right] \tag{39}$$

(the third entry is the sharpness bound; clothoids have constant κ', cubic Béziers piecewise-smooth κ'). Arc-length parameterization follows the existing conservative-fit precedent (`SpiralArcLengthFit`, `blendmath.c:1774--1828`): a monotone $s(u)$ fit that over-estimates length so true speed errs slow, with a round-trip validity check.

18 Line-number fidelity (G-code provenance)

Not a kinematic topic, but an execution contract the planner owns and the community relies on daily: *the line number the machine reports should be the line whose G-code motion is actually executing* — during blends, split cycles, fitted windows, reverse run, and after abort. An audit of the full flow found the mechanism sound in principle and broken in specific, user-visible ways. Planner-2 adopts fixing them as a goal (plan, §P2.55); everything here is reporting/observability — no executed trajectory changes on any planner.

18.1 The two channels (implemented, audited)

The value widely believed to be a queue serial number is in fact the G-code source line. Channel 1: the interpreter counts physical lines (`interp_read.cc:3158`), canon stamps each queued message, task republishes via `emcTrajSetMotionId` (`emctaskmain.cc:2625--2628`), and motion calls `tpSetId` before every segment add — so `tc->id` *is* the source line (the `tc_types.h:155` "serial number" comment is stale). A single write point (`tpUpdateMovementStatus`, `tp.c:3464`) publishes the executing segment's id to the `motion.program-line` HAL pin, the status GUIs use for the current-line highlight, and the single-step gate. Channel 2: the *state tag* (`tc->tag`,

GM_FIELD_LINE_NUMBER) rides the NML message itself, is latched only at segment *activation* (tp.c:3820), and surfaces separately as motion.interp.line-number. Two channels, two update disciplines — and they disagree in exactly the situations below.

18.2 Audited defects (L-class, condensed)

ID	Defect	Where
L1	Blend arc carries the *later* segment's id but the *earlier* segment's tag: the two HAL line pins contradict each other while the arc runs; the GUI highlight jumps a line early	tp.c:1529 vs :863
L2	Blend arc and its successor share a duplicate id (inc_id=false); single-step cannot stop at the boundary	tp.c:2087
L3	Fork regression: split cycles always report the *next* segment one servo period early (an else scope lost in the S-curve commit; upstream 2014 code had it right)	tp.c:4299--4306
L4	execTag never cleared on program end/abort: motion.interp.line-number and the feed/arc pins freeze at the last executed line forever	tp.c:3535--3549
L5	execTag lags execId during every parabolic blend and split (tag updates only at activation)	tp.c:3820
L6	Optimizer-consumed segments vanish without their line ever being reported	tp.c:1108, 1503
L7	The tool-change move never stamps a line; it inherits the previous motion's number and poisons the next	emccanon.cc:3625--3636
L8	Fitted windows (segment fitter) report only the last absorbed line for the whole window; the fitter library's progress→line interpolation (liscio_primitive_line_at) has no caller	emccanon.cc:1269, 1319
L9	Planner diagnostics print tc->id labelled "segment %d" (it is a line number — and for blend arcs the wrong line); the reverse-refusal message prints no location at all	tp.c:3046, 4611
L10	Ids are non-monotonic across O-word subroutine boundaries and essentially unvalidated	interp_o_word.cc
L11	GM_FIELD_FLOAT_LINE_NUMBER is dead outside the fitter; zero-length id-bearing moves are dropped unreported	state_tag.h:97

18.3 proposed remedies

(i) Reporting-correctness fixes L3/L4/L5/L1/L7 — planner-neutral, and L3 simply restores the upstream scoping; (ii) diagnostics reworded to "G-code line n" and the reverse-refusal/history-exhaustion messages gain a line number; (iii) fitter provenance: wire the existing progress→line interpolation into the emitted arc's float-line tag field, keeping the integer channel at last-absorbed-line for compatibility; (iv) a line-number regression family sampling both HAL pins together with golden line sequences under blending, splits, fitted windows, reverse run (backward sequence), and abort (both pins clear). Reviewer input requested on the open semantic decisions (Q11 in §20).

19 Planner-2 v0 equivalence and validation

Planner 2 v0 is *defined* as: every equation in §2–§16 marked "implemented" evaluated identically to planner 1, selected via a dispatch predicate; no numeric change. The one deliberate exception is the position-sync per-cycle response (§13), which is second-order by decision. The

acceptance gate is empirical: identical MDI programs — excluding position-synced blocks — executed under `PLANNER_TYPE=1` and `2` must produce pointwise-identical sampled command trajectories (`tests/planner2/parity`). Jerk telemetry (max $|\Delta a|/T$) is recorded report-only; the equations marked PROPOSED throughout (including those before §10 — override slew shaping and horizon sizing in §5, the tangential/normal jerk partition in §8), and excluding the sync section's deliberately retained trapezoidal law (§13), then land one at a time, each tightening planner-2 assertions while planner-1 telemetry remains the frozen baseline.

20 Questions for reviewers

Q1. Constraint 1's budget split (22): is the $\varphi/(2+\varphi)$ apportionment between sustained rotation and transitions well-founded, or should transitions be handled solely by (33) with (22) reduced to $v^3 \le j_{\max} R^2$?

Q2. Constraint 2 (23) uses the fixed ratio $c_t = 1/2$; should it use the segment's actual a_t budget $\sqrt{1-(a_n/a_{\max})^2}\, a_{\max}$ evaluated self-consistently (a cubic in v)?

Q3. The proposed kink law (32): preferred pulse shape (and constants) for the transition-window model? Is a shared window τ for kinks and curvature steps (33) acceptable, and is $\tau = \max(4T, a_{\max}/j_{\max})$ a sane default?

Q4. Parabolic overlap (34): halve jerk per side, $\sqrt{2}$ partition, or prohibit parabolic under planner 2?

Q5. Sync decision (§13): position-synchronized moves stay trapezoidal under planner 2 — jerk limitation suspended inside the sync region, enforced at the engage/disengage handoffs; G95 velocity sync stays jerk-limited. Any objection to this contract? And is the considered third-order surface (37) — or an observer-based spindle-accel feedforward — worth pursuing as an *opt-in* for machines where sync-region smoothness matters (e.g. heavy tapping heads at 2000 rpm), given that it trades pitch error for smoothness?

Q6. Per-axis propagation (§15): the cube-root characteristic-time combination (38) under-approximates multi-axis coupling for arcs; is the plane-projected extension adequate, or should arcs get a worst-case sinusoidal per-axis analysis ($a_k = \kappa v^2 \sin(\cdot)$, $j_k = \kappa^2 v^3 \cos(\cdot)$)?

Q7. Are there community objections to the strict planner-2 config semantics (startup *error* on `PLANNER_TYPE=2` with `MAX_LINEAR_JERK` < 1, rather than silent trapezoidal fallback)?

Q8. The implemented tolerance velocity (30) is permissive by $6^{2/3} \approx 3.30$ relative to the exact triangular-stop bound (see (7)). Should planner 2 adopt the exact form $v = (L\sqrt{j_e})^{2/3}$ (more conservative corners under G64 P), keep the implemented form for planner-1 behavioral compatibility, or make the factor configurable?

Q9. The component-sum gap (25): is the proposed `jerk_ratio_tan` partition (26) worth its cost in curve speed (a fixed r_j=1/2 costs $\approx 13\%$ on (23)-limited arcs), or is a documented $\sqrt{2}$ transient allowance acceptable given that servo drives tolerate brief jerk excursions?

Q10. Reverse run currently executes stop-and-go at segment boundaries (§6); is that acceptable for a recovery feature, or should the backward-reachability pass be extended over the history chain (the proposed reverse look-ahead) to blend backward through joins?

Q11. Line-number fidelity (§18): three semantic decisions need community preference. (a) Which line should a blend arc report — the incoming line (our proposal: both channels agree on it), the outgoing line, or a convention like "the line owning the majority of the arc's time"? (b) Should blend arcs get *unique* ids so single-step can stop at the arc boundary, accepting that id-equality checks and any downstream consumers assuming one-id-per-programmed-line must be revisited

— or keep the duplicate-id convention and document that stepping treats arc+segment as one unit? (c) For fitted windows (many source lines fused into one segment): is last-absorbed-line on the integer channel plus interpolated progress on the float channel acceptable, or does the community want the fitter to split windows at step boundaries when single-stepping is active (a real motion change, out of scope for the reporting package)?

A Parameter provenance: INI key → delivery → equations

Parameter (INI)	Delivery path	Where it enters the equations
`[AXIS_L]MAX_VELOCITY`	`iniaxis` → `AxisConfig`; canon reads via `emcAxisGetMaxVelocity`; motion copy for kink bounds	v_k in the straight projection (8); plane min in (9); result becomes v_{seg} in (15)/(10)
`[AXIS_L]MAX_ACCELERATION`	same route; runtime axis bounds via `tp.c:188--209`	a_k in (8); a_{axes}, a_n split in (9); per-axis $a_{\text{bound},k}$ in the kink projection (31)
`[AXIS_L]MAX_JERK`	`iniaxis` → `emcAxisSetMaxJerk`; canon reads `emcAxisGetMaxJerk`	j_k in (38)/(8); planar min in (9) → j_{seg} in (12), (13), (30). *No runtime per-axis use (defect C4)*
`[JOINT_N]MAX_VELOCITY`, `MAX_ACCELERATION`	`inijoint` → joint struct	free-mode 1-DOF bounds ($v_{\max}, a_{\max}$ of §2 per joint); homing; *not used in coordinated planning*
`[JOINT_N]MAX_JERK`	`inijoint` → `joint->jerk_limit`	$j_{\max}$ in (4)–(6) for jog/home; clamped by `[TRAJ]` jerk each FREE cycle (planner-2 fix: non-mutating)
`[TRAJ]MAX_LINEAR_VELOCITY`	`initraj` → `tp->vLimit` (slider ceiling)	v_{lim} in (15)/(10)
`[TRAJ]MAX_LINEAR_ACCELERATION`	`initraj` → traj/status accel	free-mode clamp on joint accel; general accel ceiling (coordinated segments carry canon-projected a_{seg})
`[TRAJ]MAX_LINEAR_JERK`	`initraj` → `EMCMOT_SET_JERK` → J_{traj}	(12) min; the geometric caps (22)–(24), (28)–(29) (currently J_{traj} alone); optimizer (13)–(14) via j_e; planner-2 validity gate (≥ 1)
`[TRAJ]PLANNER_TYPE`	`initraj`/`inihal` → `EMCMOT_SET_PLANNER_TYPE`	selects trapezoid (0) vs S-curve family (1, 2): whether (13) S-branch, (22)–(24), (28) S-form, (30) apply
`[DISPLAY]MAX_FEED_OVERRIDE`	`emcmotConfig->maxFeedScale`	$f_{\max}$ in (15)/(10); worst-case $v^{\max f}$ caps; v_j in the kink model (31); horizon speed v^* in (16)
`[EMCMOT]SERVO_PERIOD`	cycle time T	entry-step cap (24); kink a_{inst} in (31); deadband in (4); replan cadence (§5)
`[TRAJ]ARC_BLEND_TANGENT_KINK_RATIO`	`tp.c:181--189`	ρ threshold/scale in (31)
`[TRAJ]ARC_BLEND_OPTIMIZATION_DEPTH`	optimizer walk length	how many segments (13) recurses over; horizon coverage vs (16)
G-code `F` word	canon feed rate	v_{req} in (10) (clamped by v_{seg})
G-code `G64 P`	canon tolerance	ε in blend geometry (27); L in (30)

(compile-time) `BLEND_ACC_RATIO_*`	`blendmath.h:21--22`	$c_n = \sqrt{3}/2$ in (20)/(9); $c_t = 1/2$ in (23), kink circular scaling

Sanity property worth asserting in tests: every row flows *downward* only — INI values are projected at queue time and clamped at run time; no runtime path can raise a limit above its INI origin. The two known leaks in that story are the geometric caps' use of J_{traj} instead of (12) and the absent runtime per-axis jerk bounds, both planner-2 items.

B Equation-to-source cross-reference

Eq.	Source (branch `automata/jerksplan`)	Status
(1)–(3)	`src/emc/tp/sp_scurve.c:642--653`	implemented
(4)	`src/emc/tp/sp_scurve.c:516--546`	implemented
(5)	`src/emc/tp/sp_scurve.c:587--640`	implemented
(6)	`src/emc/tp/sp_scurve.c:388--434`	implemented
(13),(14)	`src/emc/tp/tp.c:1708--1755`	implemented
(15)	`src/emc/tp/tp.c:295--346`	implemented (assert in P0)
(20)	`src/emc/tp/tc.c:869--885`	implemented
(22)–(24)	`src/emc/tp/tc.c:887--929`	implemented
(27)–(29)	`src/emc/tp/blendmath.c:1137--1210`	implemented
(30)	`src/emc/tp/tp.c:2446--2459`	implemented (permissive approx., Q8)
(31)	`src/emc/tp/tp.c:1946--2000`	implemented (frozen)
(32)	—	PROPOSED (planner 2, P3)
(33)	—	PROPOSED (planner 2, P2)
(34)	—	PROPOSED (planner 2, P2)
(16)	— (from (6))	PROPOSED (horizon sizing)
(17)	`src/emc/motion/control.c:381--408`	implemented
(18),(19)	`src/emc/tp/tp.c` (`tpCalculateSCurveAccelV2`)	implemented (planner 2)
(25)	— (from (21))	analysis
(26)	—	PROPOSED (planner 2, Q9)
(35)	`src/emc/tp/tp.c:3519--3605`	implemented
(36)	`src/emc/tp/tp.c:3595`	implemented (frozen; retained by planner 2, §1
(37)	—	considered, not adopted (research option, Q5)
(38)	`src/emc/task/emccanon.cc:663--772`	implemented
(39)	—	PROPOSED (Point B / P5)