Coefficient Path Calculation

The F-Node Formula in PolyPaint

PolyPaint Technical Documentation

Abstract

This document gives the exact formula for the **final coefficient vector** $\mathbf{F}(t) \in \mathbb{C}^n$ — the complex polynomial coefficients fed to the Ehrlich–Aberth solver at each time step. Every root trajectory the application produces is determined entirely by this vector. A degree-d polynomial has $n = d + 1$ coefficients indexed $i = 0, \ldots, n-1$.

Master Formula

$$\boxed{F_i(t) \;=\; M\big(\,C_i(t),\; D_i(t),\; \theta(t)\big) \;+\; J_i(s) \;\cdot\; \mathbf{1}_{[i \in S]}} \tag{1}$$

Symbol	Meaning
$F_i(t) \in \mathbb{C}$	Final coefficient i at time t. This is what the solver sees.
$C_i(t) \in \mathbb{C}$	C-node i position at time t, sampled from its precomputed curve.
$D_i(t) \in \mathbb{C}$	D-node i position at time t, sampled from its precomputed curve.
$M(c, d, \theta)$	Morph interpolation function (identity when morph disabled).
$\theta(t)$	Morph phase angle $= 2\pi \cdot r_{\text{morph}} \cdot t$.
$J_i(s) \in \mathbb{C}$	Jiggle offset for coefficient i at jiggle step s (additive).
s	Jiggle step $= \lfloor t/\Delta t_{\text{jiggle}} \rfloor$.
$S \subseteq \{0, \ldots, n-1\}$	Set of selected coefficient indices (jiggle targets).
$\mathbf{1}_{[i \in S]}$	Iverson bracket: 1 if $i \in S$, 0 otherwise.

The order of operations is:

$$\text{animate } C \;\longrightarrow\; \text{animate } D \;\longrightarrow\; \text{morph blend} \;\longrightarrow\; \text{add jiggle} \;\longrightarrow\; \text{solve}$$

Jiggle is never applied to C or D individually—it is applied to the already-blended result. When morph is disabled, M is the identity on C: $M(c, d, \theta) = c$. When jiggle is off, the jiggle term is zero.

Node Parameters

The formula (1) involves two sets of animated nodes, each of length $n = d+1$:

- **C-nodes** (`coefficients[]`): primary coefficients. Always active.
- **D-nodes** (`morphTargetCoeffs[]`): morph targets. Active only when morph is enabled.

Both node types carry the *same* set of per-node animation parameters. D-nodes additionally support $\tau_i =$ `"follow-c"` (see D-Node Animation below).

Per-node parameters

Symbol	Meaning	Range	Default	Unit
H_i	Home position ($= \gamma_i[0]$)	$\mathbb{C}$	(from pattern)	complex
γ_i	Precomputed closed curve	—	$[\mathrm{H}_i]$	N points in $\mathbb{C}$
τ_i	Path type (curve shape)	see below	`"none"`	—
R_i	Path radius	[1, 100]	25	% of E
v_i	Traversal speed	[0.001, 1.0]	1.0	cycles/s
α_i	Rotation of path shape	[0, 1)	0	turns
δ_i	Direction: -1 if CCW, $+1$ if CW	$\{-1, +1\}$	$+1$	—
ε_i	Extra parameters (path-specific)	varies	{}	—
N	Curve sample count	{200, 1500}	200	points

Here $E =$ `coeffExtent`: the maximum pairwise distance among all coefficient home positions. The absolute radius used for curve generation is $R_{\text{abs}} = (R_i/100) \cdot E$.

The curve γ_i is generated once from (H_i, τ_i, R_{abs}, α_i, ε_i) and recomputed whenever any parameter changes. When $\tau_i =$ `"none"`, the curve is the single point $[\mathrm{H}_i]$ (no animation).

Path types

There are 22 base path shapes in three groups. Each animated type also has a **dithered variant** (suffix `-dither`) that adds per-frame Gaussian noise.

Group	Types
Non-animated	`none`, `follow-c` †
Basic	circle, horizontal, vertical, spiral *, random ‡
Curves	lissajous, figure-8, cardioid, astroid, deltoid, rose, spirograph, hypotrochoid, butterfly, star, square, c-ellipse *
Space-filling	hilbert, peano, sierpinski

†D-node only: $D_i(t) = C_i(t)$. *Orbital paths: curve points are absolute positions (orbiting the origin), not offsets from H_i. ‡Cloud: pre-generated Gaussian point cloud; no interpolation between samples.

Curve sample count: $N = 200$ for most paths; $N = 1500$ for space-filling paths and `spiral` (Hilbert, Peano, Sierpiński, spiral).

Notable extra parameters (ε_i). `spiral`: multiplier $m \in [0, 2]$ (default 1.5), turns $T \in [0.5, 5]$ (default 2). `lissajous`: frequencies $a \in [1, 8]$ (default 3), $b \in [1, 8]$ (default 2). `c-ellipse`: minor-axis width $w \in [1, 100]\%$ (default 50). `random`: sigma $\in [0, 10]$ (default 2). Dithered variants: $\sigma_{\%} \in [0, 1]$ (default 0.2).

$C_i(t)$: C-Node Animation

Each C-node i carries a precomputed curve γ_i generated from the parameters in the preceding table. At each time step, $C_i(t)$ is obtained by sampling this curve at a phase determined by the speed v_i and direction δ_i.

Curve Sampling

Define the **phase** of coefficient i at time t:

$$\varphi_i(t) \;=\; t \cdot v_i \cdot \delta_i \qquad \text{where} \quad \delta_i = \begin{cases} -1 & \text{if CCW} \\ +1 & \text{if CW} \end{cases}$$

and v_i is the speed parameter. Then the normalized phase:

$$u_i \;=\; \big((\varphi_i \bmod 1) + 1\big) \bmod 1 \;\in\; [0, 1)$$

and the raw index into the curve array:

$$\rho \;=\; u_i \cdot N$$

Smooth curves. Linear interpolation between adjacent samples (wrapping at N):

$$\ell = \lfloor \rho \rfloor \bmod N, \qquad h = (\ell + 1) \bmod N, \qquad f = \rho - \lfloor \rho \rfloor \tag{2}$$

$$C_i(t) = \gamma_i[\ell]\,(1 - f) \;+\; \gamma_i[h]\,f \tag{3}$$

Cloud curves (random path, `_isCloud` flag): no interpolation—snap to the nearest point:

$$C_i(t) = \gamma_i\big[\lfloor \rho \rfloor \bmod N\big]$$

Dither. If the curve carries a dither parameter $\sigma_{\%} > 0$:

$$C_i(t) \;\leftarrow\; C_i(t) \;+\; \sigma_{\text{abs}}\,\big(\mathcal{N}(0,1) + i\,\mathcal{N}(0,1)\big), \qquad \sigma_{\text{abs}} = \frac{\sigma_{\%}}{100} \cdot E$$

where $\mathcal{N}(0,1)$ is a standard Gaussian (Box–Muller) and $E =$ `coeffExtent` is the maximum pairwise distance between any two coefficients.

Static case. When pathType = `"none"`: $C_i(t) = \mathrm{H}_i$ for all t.

$D_i(t)$: D-Node Animation

D-nodes use the **identical curve-sampling formula** as C-nodes, with their own independent pathType, speed, radius, angle, direction, and precomputed curve.

Follow C. When D_i.pathType = `"follow-c"`:

$$D_i(t) = C_i(t)$$

The D-node copies the already-animated C-node position. This is evaluated *after* C-node advancement.

Static case. When pathType = `"none"`: $D_i(t)$ stays at the D-node's home position (which may differ from H_i if the user has dragged it).

$M(c, d, \theta)$: Morph Interpolation

When morph is **disabled**: $M(c, d, \theta) = c$ (D-nodes are ignored).

When morph is **enabled**, the behaviour depends on the **C–D path type**. All paths share a local coordinate frame.

Setup

Given $c, d \in \mathbb{C}$ (the animated C- and D-node positions):

$$\boldsymbol{\delta} = d - c \qquad L = |\boldsymbol{\delta}| \tag{4}$$

$$\mathbf{u} = \boldsymbol{\delta}/L \qquad \mathbf{v} = i\,\mathbf{u} \quad (90° \text{ CCW rotation}) \tag{5}$$

$$\mathbf{m} = \tfrac{1}{2}(c + d) \qquad a = L/2 \quad (\text{semi-major}) \tag{6}$$

Let $\sigma = +1$ if CCW, -1 if CW. When $L < 10^{-15}$, all paths return c.

The morph phase angle:

$$\theta(t) = 2\pi \cdot r_{\text{morph}} \cdot t \tag{7}$$

where r_{morph} is the morph rate in Hz (default 0.01, range $[0, 0.1]$).

Line (default)

$$\mu = \frac{1 - \cos\theta}{2} \qquad \Longrightarrow \qquad M(c, d, \theta) = c\,(1 - \mu) + d\,\mu \tag{8}$$

Raised-cosine oscillation. μ sweeps smoothly from 0 (at c) to 1 (at d) and back. Period $= 1/r_{\text{morph}}$ seconds. The path is a straight line segment.

Circle

The c–d segment is the **diameter** of a circle:

$$M = \mathbf{m} \;+\; (-a\cos\theta)\,\mathbf{u} \;+\; (\sigma\, a \sin\theta)\,\mathbf{v} \tag{9}$$

At $\theta = 0$: position $= c$. At $\theta = \pi$: position $= d$. Full revolution $= 2\pi$.

Ellipse

Same as circle, with a shorter semi-minor axis $b = p \cdot a$ where $p \in [0.1, 1.0]$ is the minor-axis percentage:

$$M = \mathbf{m} \;+\; (-a\cos\theta)\,\mathbf{u} \;+\; (\sigma\, b \sin\theta)\,\mathbf{v} \tag{10}$$

When $p = 1$, this reduces to the circle. The major axis lies along $c \to d$.

Figure-8

A Lissajous 1:2 curve crossing at the midpoint:

$$M = \mathbf{m} \;+\; (-a\cos\theta)\,\mathbf{u} \;+\; (\sigma\, \tfrac{a}{2} \sin 2\theta)\,\mathbf{v} \tag{11}$$

At $\theta = 0$: at c. At $\theta = \pi/2$: crosses midpoint. At $\theta = \pi$: at d. At $\theta = 3\pi/2$: crosses midpoint again. The two lobes are symmetric about $\mathbf{m}$.

Summary Table

Path	**Local-frame displacement** (l_x, l_y)	**Period**
Line	$c + (d - c)\,\dfrac{1 - \cos\theta}{2}$	2π
Circle	$(-a\cos\theta,\ \sigma\, a \sin\theta)$	2π
Ellipse	$(-a\cos\theta,\ \sigma\, b \sin\theta)$	2π
Figure-8	$\left(-a\cos\theta,\ \sigma\, \tfrac{a}{2} \sin 2\theta\right)$	2π

Position in global frame: $M = \mathbf{m} + l_x\,\mathbf{u} + l_y\,\mathbf{v}$ (except Line, which is computed directly).

$J_i(s)$: Jiggle Offsets

Jiggle produces per-coefficient **additive** complex offsets with these properties:

- Computed from **home positions** $\mathrm{H}_i = \gamma_i[0]$, *not* from the current animated position $C_i(t)$.
- **Piecewise-constant** in time: offsets change only at step boundaries, every Δt_{jiggle} seconds.
- Applied **only to selected** coefficients ($i \in S$).
- Applied **after** morph blending.

$$s = \lfloor t \,/\, \Delta t_{\text{jiggle}} \rfloor, \qquad J_i(s) = 0 \;\text{ if }\; i \notin S \text{ or mode} = \texttt{"none"}$$

Notation (in addition to the per-node symbols from the Node Parameters table):

Symbol	Meaning	Used by
$\mathbf{c}$	Centroid $= \frac{1}{\lvert S\rvert}\sum_{i\in S}\mathrm{H}_i$	rotate, scale-cen., spiral-cen., breathe, wobble
σ	$(\texttt{jiggleSigma}/10)\cdot E$	random, walk
g	$(\texttt{scaleStep}/100)\cdot E$	scale, spiral
N_{angle}	Angle steps per full rotation	rotate, spiral, wobble
N_{circle}	Steps per full rotation (about origin)	circle
A	Amplitude parameter (0–100)	breathe, lissajous
P	Period in jiggle steps	breathe, wobble, lissajous
f_x, f_y	Frequency multipliers	lissajous
R_α	Rotation operator by angle α	all rotation modes

Random

$$J_i(s) = \sigma\,\mathcal{N}_1(0,1) + i\,\sigma\,\mathcal{N}_2(0,1) \tag{12}$$

Independent Gaussian noise, freshly sampled each step.

Walk

$$J_i(s) = J_i(s{-}1) + \sigma\,\mathcal{N}_1(0,1) + i\,\sigma\,\mathcal{N}_2(0,1), \qquad J_i(0) = 0 \tag{13}$$

Cumulative random walk. The *only* jiggle mode with memory.

Rotate

$$\alpha = \frac{2\pi s}{N_{\text{angle}}}, \qquad J_i(s) = R_\alpha\,(\mathrm{H}_i - \mathbf{c}) - (\mathrm{H}_i - \mathbf{c}) \tag{14}$$

Rigid rotation of the selected coefficients about their centroid.

Circle (jiggle)

$$\alpha = \frac{2\pi s}{N_{\text{circle}}}, \qquad J_i(s) = R_\alpha\,\mathrm{H}_i - \mathrm{H}_i \tag{15}$$

Rotation of each coefficient about the origin.

Scale–Center

$$g = \frac{\texttt{scaleStep}}{100} \cdot E, \qquad J_i(s) = \frac{\mathrm{H}_i}{|\mathrm{H}_i|} \cdot g \cdot s \tag{16}$$

Radial growth from the origin. Linear in step number.

Scale–Centroid

$$\mathbf{d}_i = \mathrm{H}_i - \mathbf{c}, \qquad J_i(s) = \frac{\mathbf{d}_i}{|\mathbf{d}_i|} \cdot g \cdot s \tag{17}$$

Spiral–Centroid

$$\alpha = \frac{2\pi s}{N_{\text{angle}}}, \qquad r = 1 + \frac{g\,s}{|\mathbf{d}_i|}, \qquad J_i(s) = r\,R_\alpha\,\mathbf{d}_i - \mathbf{d}_i \tag{18}$$

Combined rotation and radial scaling from centroid.

Spiral–Center

$$r = 1 + \frac{g\,s}{|\mathrm{H}_i|}, \qquad J_i(s) = r\,R_\alpha\,\mathrm{H}_i - \mathrm{H}_i \tag{19}$$

Breathe

$$\lambda = 1 + \frac{A}{100}\,\sin\!\Big(\frac{2\pi s}{P}\Big), \qquad J_i(s) = (\mathrm{H}_i - \mathbf{c})\,(\lambda - 1) \tag{20}$$

Sinusoidal radial pulsation from centroid.

Wobble

$$\alpha = \frac{2\pi}{N_{\text{angle}}}\,\sin\!\Big(\frac{2\pi s}{P}\Big), \qquad J_i(s) = R_\alpha\,(\mathrm{H}_i - \mathbf{c}) - (\mathrm{H}_i - \mathbf{c}) \tag{21}$$

Sinusoidal angular oscillation about centroid.

Lissajous

$$A_{\text{abs}} = \frac{A}{100} \cdot E, \qquad J_i(s) = A_{\text{abs}}\,\sin\!\Big(\frac{2\pi f_x s}{P} + \frac{\pi}{2}\Big) + i\,A_{\text{abs}}\,\sin\!\Big(\frac{2\pi f_y s}{P}\Big) \tag{22}$$

Uniform translation of *all* selected coefficients along a Lissajous curve. Same offset for every coefficient.

Complete Pipeline

1. **Animate C-nodes.** For each coefficient i:

$$C_i(t) = \begin{cases} \text{sample } \gamma_i \text{ at phase } \varphi_i(t) & \text{if pathType} \neq \texttt{"none"} \\ \mathrm{H}_i & \text{otherwise} \end{cases}$$

   If dither is enabled, add Gaussian noise.

2. **Animate D-nodes.** For each D-node i:

$$D_i(t) = \begin{cases} C_i(t) & \text{if pathType} = \texttt{"follow-c"} \\ \text{sample D-curve}_i \text{ at phase } \varphi_i^D(t) & \text{if pathType} \neq \texttt{"none"} \\ D_{\mathrm{H},i} & \text{otherwise} \end{cases}$$

3. **Morph blend.**

$$B_i = \begin{cases} M(C_i(t),\, D_i(t),\, \theta(t)) & \text{if morph enabled} \\ C_i(t) & \text{otherwise} \end{cases}$$

4. **Jiggle.** Compute step $s = \lfloor t/\Delta t_{\text{jiggle}} \rfloor$ and offsets $J_i(s)$ for $i \in S$.

5. **Assemble F.**

$$F_i(t) = B_i + J_i(s) \cdot \mathbf{1}_{[i \in S]}$$

6. **Solve.** Feed $\mathbf{F}(t)$ to the Ehrlich–Aberth root finder $\rightarrow$ roots.

Key Invariants

1. **Jiggle never mutates C or D.** It only affects the final blend F.
2. **Jiggle uses home positions.** All jiggle offsets (except walk) are deterministic functions of (s, H_i, params), not the current animated position.
3. **Morph uses live positions.** The morph function receives the already-animated $C_i(t)$ and $D_i(t)$, not home positions.
4. **No feedback.** $F_i(t)$ depends only on the current state of $C_i(t)$, $D_i(t)$, and jiggle step s. It does not depend on F at any previous time (except jiggle walk mode, which accumulates).
5. **In-place mutation.** The `coefficients[i].re/im` fields hold the current animated $C_i(t)$ after step 1. `solveRoots()` copies before modifying.

Fast Mode (Bitmap Workers)

In bitmap rendering, the pipeline from Section 1 runs in parallel Web Workers. This section defines the execution model.

Definitions

Symbol	Meaning	Value / range
ΔT_{pass}	Virtual time per pass	1.0 s (constant)
K	Total steps per pass	user-selected: 10 to 10^6
W	Number of workers	min(HW threads, 16)
t_{off}	Elapsed offset (accumulated)	starts at 0, $+\Delta T_{\text{pass}}$ each pass

What is a pass

A **pass** is one complete execution of K solver steps distributed across W workers. Each pass advances virtual time by exactly $\Delta T_{\text{pass}} = 1.0\,\text{s}$. This is not wall-clock time; it is the simulated elapsed time used to advance coefficient animations along their curves.

Step distribution. Steps are divided evenly across workers. Let $b = \lfloor K/W \rfloor$ and $r = K \bmod W$. Worker w ($w = 0, \ldots, W-1$) executes steps $[\text{start}_w, \text{end}_w)$ where the first r workers receive $b+1$ steps and the remaining $W-r$ workers receive b steps.

Elapsed time per step. Within a pass, step k ($0 \le k < K$) runs at virtual time:

$$t(k) \;=\; t_{\text{off}} \;+\; \frac{k}{K}\,\Delta T_{\text{pass}} \tag{23}$$

So within a single pass, t ramps linearly from t_{off} to $t_{\text{off}} + \Delta T_{\text{pass}}$.

Per-step pipeline. Each step executes the full pipeline from the Complete Pipeline section (animate C, animate D, morph, jiggle, solve), then maps each root to a canvas pixel and emits it as a sparse (x, y, r, g, b) entry.

Pass lifecycle

1. **Init.** Main thread serializes all static data (curves, colors, morph config, jiggle offsets, WASM binaries) and sends it to each worker once.
2. **Run.** Main thread sends a `"run"` message to each worker with its step range $[\text{start}_w, \text{end}_w)$, the current t_{off}, and warm-start root positions.
3. **Compute.** Workers execute their steps in parallel. Each worker accumulates sparse pixel arrays (no full-canvas buffers).
4. **Done.** Each worker transfers its pixel arrays (`Int32Array` indices + `Uint8Array` RGB) to the main thread via zero-copy buffer transfer.
5. **Composite.** Main thread merges all workers' pixels into a persistent `ImageData` buffer and calls `putImageData` on the dirty rectangle only.
6. **Advance.** $t_{\text{off}} \leftarrow t_{\text{off}} + \Delta T_{\text{pass}}$. Root positions from the highest-step worker become the warm start for the next pass. Go to step 2.

Workers are **persistent**: they are created once at init and reused across passes (only a new `"run"` message is sent each pass).

Jiggle boundary

Jiggle offsets are **static within a pass**: the main thread computes $J_i(s)$ once and bakes it into the worker init data. Workers apply the same offset for every step in the pass.

A **jiggle boundary** occurs every Δt_{jiggle} virtual seconds (default 4 s = 4 passes). When the pass counter reaches the jiggle interval:

1. Compute the new jiggle step $s = \lfloor t_{\text{off}} / \Delta t_{\text{jiggle}} \rfloor$.
2. Compute $J_i(s)$ for all $i \in S$.
3. **Terminate all workers** and recreate them with the updated offsets (full reinit, because jiggle offsets are init-time data).

Solver and WASM

Workers use a 3-tier execution strategy:

1. **WASM step loop**: entire per-step pipeline compiled to WASM (∼15 KB). Used when morph is disabled or C–D path = `"line"`.
2. **WASM solver + JS loop**: only the Ehrlich–Aberth solver is WASM (∼2 KB); curve sampling, morph interpolation, and pixel emission run in JS. Used for non-line morph paths.
3. **Pure JS**: fallback when WASM is unavailable.

Worker solver parameters differ from the main thread:

	Main thread	**Workers**
MAX_ITER	100	64
TOL	10^{-12} (magnitude)	10^{-16} (squared)

The mathematical pipeline is identical to the main-thread path. The output differs only in solver precision and iteration count.