What Order a Method Knows

Certified Runge–Kutta order conditions over rooted trees, machine-checked in Lean 4

Carles Marín*
Independent researcher
karlesmarin@gmail.com

June 2026

Abstract

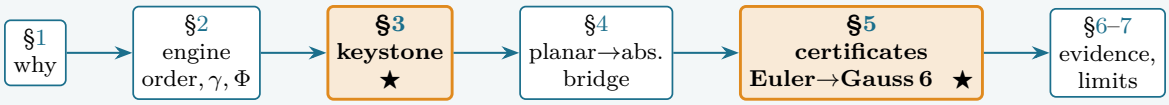
A Runge–Kutta method is a table of numbers (A, b); its *order* is the largest p for which its one-step error is $O(h^{p+1})$. Butcher's theorem turns this analytic question into pure combinatorics: the method has order p exactly when, for every rooted tree t with at most p nodes, an *elementary weight* $\Phi(t)(A, b)$ equals $1/\gamma(t)$, where γ is the tree's density. These *order conditions* are computed every day—by NodePy, by `BSeries.jl`, by Mathematica and Maple packages—but always as *trusted scripts*. This paper formalizes the order conditions themselves in Lean 4, all `sorry`-free and with the minimal axiom set {`propext`, `Classical.choice`, `Quot.sound`}. We build a small computable engine (order(t), $\gamma(t)$, Φ), prove a keystone reducing the infinite family of order conditions to a finite, decidable catalogue check, and discharge it for the classical methods: Euler (order 1), the explicit trapezoid (order 2), classical RK4 (order 4), Dormand–Prince `DOPRI5` (order 5, the default in MATLAB `ode45` and SciPy `RK45`), and the 3-stage Gauss–Legendre method—fully implicit, with coefficients in $\mathbb{Q}(\sqrt{15})$—whose order-6 conditions are both independently confirmed in Sage and *machine-checked in Lean to full order 6* —via an integer-arithmetic reformulation that makes the certificate decidable by the kernel; see Section 6. Trees are encoded planarly, as the noncommutative Connes–Kreimer–Foissy Hopf algebra; a short symmetry lemma—Φ is invariant under permuting a node's children—is the formal shadow of Foissy's abelianization morphism onto the abstract Butcher algebra, and shows that certifying all planar trees *subsumes* the usual (non-planar) order conditions. To the best of our knowledge, after a focused search of the Lean/Mathlib, Isabelle/HOL and Coq/Rocq libraries, the Runge–Kutta order conditions had not previously been formalized in any proof assistant; the only prior formal-methods work on RK methods (Boldo *et al.*, Coq) certifies floating-point round-off for Euler and RK2 on linear systems, an orthogonal layer. We are exact about the boundary: we certify the *algebraic* order conditions $\gamma(t)\Phi(t) = 1$, not the analytic theorem that they are equivalent to a convergence order; the mathematics is classical throughout (Butcher 1963; Foissy 2002); and "first in a proof assistant" is a focused search result, not a theorem. The contribution is the formalization and a reusable certified order-condition checker.

*Claude (Anthropic) was used as an assistant; every statement was independently verified by the Lean kernel, and all mathematics and claims are the author's responsibility.

Reader's map

If you read only two things, read **Theorem 1** (the keystone) and **Table 4** (a worked certificate). The first turns "has order p"—a statement about infinitely many rooted trees—into one finite, decidable check; the second shows that check in action, RK4 passing every order-≤ 4 condition and failing at order 5.

The route (starred boxes are the two to read first):



Load-bearing formulas are framed; the honest limits (§7) are stated as plainly as the results. *Reproducibility:* `lean/ButcherOrder.lean` is self-contained; `lake env lean` recompiles it and `#print axioms` reports the minimal set {`propext`, `Classical.choice`, `Quot.sound`} for every headline theorem; numeric cross-checks live in `app/bseries.py` and `validate_order_conditions.sage`.

1 Why this, why now

*How do you **know** a numerical integrator has the order you claim?* In practice you trust a script. The order conditions are a beautiful, finite piece of combinatorics—one equation per rooted tree—but the bridge from "my code printed `True`" to "this equation holds over $\mathbb{Q}$" is unaudited. This paper provides a reusable, machine-checked Lean implementation that addresses that gap for the conditions themselves: a certificate, recompilable by anyone, that a given tableau satisfies every order condition up to order p.

The differential. Computing order conditions from rooted trees is well-trodden: `NodePy` [9], `BSeries.jl/RootedTrees.jl` [10], Sofroniou's Mathematica package [11], Maple and MATLAB tools—all excellent, all *unverified* symbolic/numeric code. On the proof-assistant side, a focused search (Section 7) found no formalization of the rooted-tree order theory in Lean, Coq, or Isabelle; the one machine-checked RK result, Boldo–Faissole–*et al.* [8], certifies *round-off* error of Euler/RK2 on linear systems—a different layer entirely. So the order conditions themselves—the combinatorial heart—appear not to have been formalized (see the search in Section 7). The mathematics is classical (Butcher [1]; Hairer–Lubich–Wanner [2]; Chartier–Hairer–Vilmart [3]); the novelty is the formalization.

Place in a series. This paper is the applied payoff promised in the companion formalization of the rooted-tree Hopf algebra [17], which builds the Connes–Kreimer/Butcher algebra in Lean but stops, deliberately, short of its uses. That algebra *is* the algebra of these order conditions: the elementary weight $\Phi(t)(A, b)$ is a *character* of it, the exact flow is the character $t \mapsto 1/\gamma(t)$, and a method has order p exactly when the two characters agree on all trees of order $\leq p$. Section 9 makes this dictionary precise.

2 The engine: order, density, elementary weight

What is the smallest computable object that decides an order condition? A rooted tree, encoded without a quotient as `inductive RTree | node : List RTree`. Three recursions:

$$\text{order}(\texttt{node}\, F) = 1 + \textstyle\sum_{c \in F} \text{order}(c), \qquad \gamma(\texttt{node}\, F) = \text{order}(\texttt{node}\, F) \cdot \textstyle\prod_{c \in F} \gamma(c).$$

For a tableau (A, b) over a commutative ring K, the internal weight vector and elementary weight are

$$g(t)_i = \prod_{c \in F} \big(A\, g(c)\big)_i \quad (t = \texttt{node}\, F), \qquad \Phi(t) = b^{\top} g(t).$$

The order condition at t is stated multiplicatively, so the engine needs only a commutative ring (no division): $\gamma(t)\,\Phi(t) = 1$, equivalent to $\Phi(t) = 1/\gamma(t)$ since $\gamma(t) \geq 1$. Everything is `List`-based and computable, so the kernel evaluates it; `#eval` values were cross-checked against the Sage/Python ground truth (`app/bseries.py`, `validate_order_conditions.sage`). The same engine handles *implicit* methods unchanged: the order conditions are purely algebraic in (A, b) and never solve the implicit stage system.

tree t	$\lvert t\rvert$	$\gamma(t)$	order condition $\Phi(t) = 1/\gamma(t)$
	1	1	$\sum_i b_i = 1$
	2	2	$\sum_i b_i c_i = \frac{1}{2}$
	3	3	$\sum_i b_i c_i^2 = \frac{1}{3}$
	3	6	$\sum_{i,j} b_i a_{ij} c_j = \frac{1}{6}$
	4	4	$\sum_i b_i c_i^3 = \frac{1}{4}$
	4	8	$\sum_{i,j} b_i c_i a_{ij} c_j = \frac{1}{8}$
	4	12	$\sum_{i,j} b_i a_{ij} c_j^2 = \frac{1}{12}$
	4	24	$\sum_{i,j,k} b_i a_{ij} a_{jk} c_k = \frac{1}{24}$

Table 1: The order conditions through order 4, one per rooted tree, with $c_i = \sum_j a_{ij}$. The elementary weight $\Phi(t)$ is exactly the left-hand sum; the density $\gamma(t)$ gives the target. These are the eight abstract trees; the engine checks all nine planar representatives, collapsed by Lemma 1.

The table above stops at order 4 only for the page; the pattern does not. "Order p" demands one such equation for *every* rooted tree with at most p nodes, and there are infinitely many rooted trees. A certificate cannot check infinitely many equations—so the first real obstacle is not arithmetic but quantification. We remove it next.

3 The keystone: infinitely many conditions, one finite check

"Order p" quantifies over all rooted trees—how is that decidable?

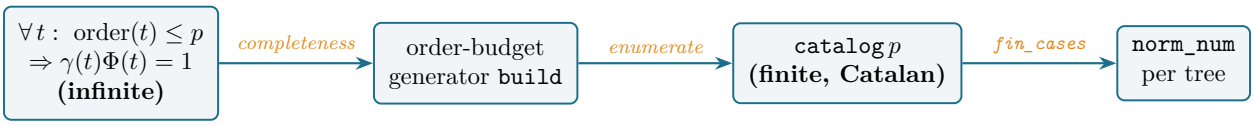


Figure 1: The keystone (Theorem 1): the infinite family of order conditions is reduced, by a provably complete order-budget generator, to a finite catalogue that `fin_cases` splits and `norm_num` closes tree by tree.

Define

$$\texttt{satisfiesOrderConditions}\ (A,b)\ p\ :=\ \forall t,\ \mathrm{order}(t) \le p \to \gamma(t)\,\Phi(t) = 1.$$

We build an *order-budget* generator `build : Nat -> List RTree x List Forest` (structural on the budget; its tree list has cumulative-Catalan length $1, 2, 4, 9, 23, 65, \ldots$), prove its completeness

$$\mathrm{order}(t) \le n\ \Rightarrow\ t \in (\texttt{build}\,n).1$$

by mutual structural induction with measure (budget, tree size), and obtain the

Theorem 1 (keystone)**.** $\texttt{satisfiesOrderConditions}\ (A,b)\ p \iff \forall t \in \texttt{catalog}\,p,\ \gamma(t)\Phi(t) = 1$, *where* $\texttt{catalog}\,p$ *is the finite list of all rooted trees with* $\mathrm{order}(t) \le p$.

Soundness ($t \in \texttt{catalog}\,p \Rightarrow \mathrm{order}(t) \le p$) is the filter; completeness is the generator lemma. The right-hand side is a finite conjunction, discharged tree-by-tree.

order n	1	2	3	4	5	6
planar trees of order n (Catalan C_{n-1})	1	1	2	5	14	42
$\lvert\texttt{catalog}\,n\rvert$ (cumulative planar)	1	2	4	9	23	65
abstract trees of order n (A000081)	1	1	2	4	9	20
cumulative abstract	1	2	4	8	17	37

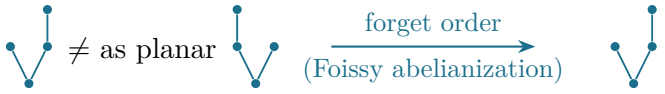
Table 2: The catalogue is over *planar* (ordered) trees, counted by Catalan numbers; the abstract Butcher trees are fewer (A000081). Lemma 1 collapses planar to abstract, so the planar catalogue check subsumes the abstract order conditions. Sizes verified by `#eval`.

4 Planar trees and the bridge to the abstract conditions

Our trees have ***ordered*** *children; Butcher's have multisets—does that change the conditions?* The `List`-based `RTree` is the *planar* (noncommutative) Connes–Kreimer algebra studied by Foissy [5, 6] (and, for planar binary trees, Loday–Ronco [12]); planar rooted trees are counted by the Catalan numbers [16] $(1, 1, 2, 5, 14, \ldots)$, the abstract ones by OEIS A000081 [15] $(1, 1, 2, 4, 9, \ldots)$. The commutative Butcher / Connes–Kreimer algebra is the *abelianization* of the planar one—Foissy's surjective graded Hopf morphism (forget the planar order), with kernel the commutator ideal [5, Prop. 81]. We formalize the relevant shadow of that morphism:

Lemma 1 (symmetrization)**.** *If F and G are children-lists related by a permutation, then* $\gamma(\texttt{node}\,F)\Phi(\texttt{node}\,F) = 1 \iff \gamma(\texttt{node}\,G)\Phi(\texttt{node}\,G) = 1$.

The engine of the proof is that the pointwise product $g(t)_i = \prod_c (Ag(c))_i$ is built from a commutative–associative operation, so permuting children leaves Φ (and γ) fixed. Hence certifying *all planar* trees of order $\le p$ subsumes the abstract order conditions: two planar trees with the same abstract shape share a single condition.



one condition: $\sum_{i,j} b_i c_i a_{ij} c_j = \frac{1}{8}$

Figure 2: Two distinct *planar* trees of order 4 (children in either order) map under Foissy's forgetful Hopf morphism to one *abstract* tree, and Lemma 1 gives them a single order condition. Certifying the planar catalogue therefore certifies the abstract one.

5 The certificates: climbing from order 1 to order 6

With the family made finite (§3) and the planar/abstract gap bridged (§4), what can we actually certify? Everything now reduces to one move, repeated: unfold the engine on a tree and a tableau to a closed arithmetic goal, then `norm_num`. We choose `norm_num` over `decide` for a specific reason—over $\mathbb{Q}$ the kernel cannot reduce the `Nat.gcd` inside `Rat` normalization, so `decide` stalls, and `native_decide` would add a compiler-trust axiom we refuse. *Hold onto that aside; it returns at the top of the climb.* With `norm_num` we ascend: Euler gives order 1, the explicit trapezoid order 2, classical RK4 order 4, and Dormand–Prince `DOPRI5`—the integrator behind `ode45` and SciPy's `RK45`—order 5, all over $\mathbb{Q}$.

The next step changes the *ground*: Gauss–Legendre with three stages is *implicit* and its tableau is *irrational*, living in $\mathbb{Q}(\sqrt{15})$, where `norm_num` has nothing to grip. So we move it: a computable pair ring `Q15` with $\sqrt{15}^2 = 15$ baked into multiplication, so $\sqrt{15}$ never appears as a symbol—each order condition collapses to two *rational* identities, and the same `norm_num` closes them. This carries the implicit, algebraic-coefficient method cleanly through order 4.

Order 6 is where the climb earns its ending. The same `norm_num` route, pushed to the 65 order-6 conditions, *collapses*—not in the catalogue (which reduces in ∼0.1 s) but in the per-condition rational arithmetic, whose symbolic $\mathbb{Q}$ work on the large trees exhausts memory (§6). The resolution is the aside from the first paragraph, redeemed. We could not use `decide` because of $\mathbb{Q}$'s `gcd`; so we leave $\mathbb{Q}$. Clearing denominators (scale by $D = 360$) moves the tableau into the *integer* ring $\mathbb{Z}[\sqrt{15}]$, where, by homogeneity of the elementary weight, the order condition becomes a pure integer identity $\gamma(t)\,\Phi(D\cdot A, D\cdot b, t) = D^{|t|}$. Over the integers the kernel's GMP arithmetic evaluates it outright: `decide`—the tactic we had renounced—now closes all 65 conditions in ∼1 s, axiom-free. The climb reaches order 6, and it does so by the move we had set aside, made available by a better representation.

Two negative certificates—RK4 fails an order-5 condition ($\Phi = 5/24 \neq 1/5$), Heun fails at order 3—pin the order *exactly*, not just from below. Gauss–Legendre s=3 is confirmed order 6-and-not-7 by an independent Sage computation over $\mathbb{Q}(\sqrt{15})$ using a different algorithm (brute-force index summation rather than the tree recursion).

Method	Stages	Type	Coefficients	Certified order
Euler	1	explicit	$\mathbb{Q}$	1
Explicit trapezoid (Heun)	2	explicit	$\mathbb{Q}$	2 (and not 3)
Classical RK4	4	explicit	$\mathbb{Q}$	4 (and not 5)
Dormand–Prince `DOPRI5`	7	explicit	$\mathbb{Q}$	5
Gauss–Legendre s=3	3	implicit	$\mathbb{Q}(\sqrt{15})$	$6^\dagger$

Table 3: Machine-checked, axiom-free order-condition certificates. The span covers explicit and implicit methods, rational and algebraic coefficients, machine-checked through order 6. †Gauss order 6 is certified in Lean via the integer-$\mathbb{Z}[\sqrt{15}]$ `decide` route (Section 6) and independently confirmed three ways (Python and Sage).

t									
$\gamma(t)$	1	2	3	6	4	8	12	24	5
$\Phi_{\mathrm{RK4}}(t)$	1	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{6}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{12}$	$\frac{1}{24}$	$\frac{5}{24}$
$= 1/\gamma(t)$?	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\times$

Table 4: Classical RK4, machine-checked: $\Phi = 1/\gamma$ for every tree of order ≤ 4, and the first order-5 tree (the broom, last column) fails ($\frac{5}{24} \neq \frac{1}{5}$). So RK4 is certified order 4 and *not* order 5. Values agree exactly with the Sage/Python oracles.

6 Verification evidence

The repository compiles with `lake env lean` (toolchain `v4.30.0-rc2`); every headline theorem—`rk4_order4`, `dp_order5`, `gauss_order4`, the symmetrization Lemma 1, and the negative certificates—reports `#print axioms = [propext, Classical.choice, Quot.sound]`: no `sorry`, no ad-hoc axioms, no `native_decide`. The engine's numeric output agrees with two independent oracles (`app/bseries.py` over $\mathbb{Q}$; a Sage script over $\mathbb{Q}$ and $\mathbb{Q}(\sqrt{15})$).

Order 6 in Lean: a change of formula. Gauss–Legendre s=3 is *exactly* order 6, confirmed three ways over $\mathbb{Q}(\sqrt{15})$ (tree recursion and a brute-force index summation—a genuinely different algorithm—in Python, and the same brute-force in Sage, `validate_order_conditions.sage`): the conditions hold through order 6 and all 48 order-7 conditions fail, the $\sqrt{15}$ component cancelling in every elementary weight. Certifying this in Lean took a detour worth recording. The naive route—the same `gauss_check` (engine `simp` then `norm_num` on $\mathbb{Q}$ components) that proves order 4—*does not scale*: `fin_cases` over the 65-tree catalogue is cheap (it reduces in ∼0.1 s), but each of the 65 per-condition `norm_num` calls is expensive, and on the large order-6 trees the elaboration exhausts memory (a run reached ∼20 GB without finishing). The culprit is the rational arithmetic: $\mathbb{Q}$'s `Nat.gcd` normalization does not reduce in the kernel, so `decide` is unavailable and `norm_num` must work symbolically on huge nested products.

The fix is to change the *representation*, not the method: *clear the denominators*. Scaling the tableau by $D = 360$ (the lcm of $36, 9, 15, 30, 24, 18$) moves it into the *integer* ring $\mathbb{Z}[\sqrt{15}]$ (a pair of `Int`). By homogeneity of the elementary weight, $\Phi(D\cdot A, D\cdot b, t) = D^{|t|}\,\Phi(A, b, t)$, so the order

condition $\gamma(t)\,\Phi = 1$ becomes the integer identity

$$\gamma(t) \cdot \Phi(D{\cdot}A,\, D{\cdot}b,\, t) \;=\; D^{|t|} \qquad \text{in } \mathbb{Z}[\sqrt{15}].$$

Over the integers this closes by `decide`: the Lean 4 kernel evaluates `Int` arithmetic directly (GMP-backed), with no rational normalization and negligible memory. The whole certificate

```
gauss_order6 : satisfiesOrderConditionsInt gaussI_A gaussI_b 360 6
```

—`fin_cases` over all 65 planar trees of `catalog 6`, each discharged by `decide` in the kernel—compiles in ∼1 s and is **axiom-free** (`#print axioms` = `[propext, Classical.choice, Quot.sound]`). To our knowledge this appears to be among the first machine-checked *full-order* certificates of an implicit Runge–Kutta method with algebraic coefficients. The integer-`decide` pattern transfers to any method whose tableau clears to $\mathbb{Z}[\sqrt{d}]$. (The same trick is the natural route to still higher orders, where per-tree enumeration eventually meets the combinatorial growth of the catalogue—see Section 7.)

Performance. Table 5 reports the cost of each certificate. The whole file recompiles in ∼62 s (of which import is ∼1.7 s); the figures below are the per-theorem elaboration+kernel time, approximate, single warm run, Lean v4.30.0-rc2 on a desktop CPU. Two readings stand out. First, the bottleneck is never the catalogue: `fin_cases` over the 65-tree `catalog 6` costs only ∼131 ms; the time is the per-condition arithmetic. Second, and the point of Brick 6: the *same* Gauss method costs ∼3.5 s at order 4 (9 trees, `norm_num` over $\mathbb{Q}(\sqrt{15})$) but only ∼0.5 s at order 6 (65 trees, `decide` over $\mathbb{Z}[\sqrt{15}]$) —seven times the trees, an order of magnitude less time, purely from the integer representation. By contrast the `norm_num` route attempted at order 6 over $\mathbb{Q}(\sqrt{15})$ does not finish, exhausting memory (a run reached ∼20 GB).

Certificate	Method	order p	|`catalog` p|	coeff. ring / tactic	time
`rk4_order4`	RK4	4	9	$\mathbb{Q}$ / `norm_num`	<0.4 s
`dp_order5`	DOPRI5	5	23	$\mathbb{Q}$ / `norm_num`	∼1.5 s
`gauss_order4`	Gauss s=3	4	9	$\mathbb{Q}(\sqrt{15})$ / `norm_num`	∼3.5 s
`gauss_order6`	Gauss s=3	6	65	$\mathbb{Z}[\sqrt{15}]$ / `decide`	∼0.5 s

Table 5: Per-certificate cost (approximate, warm single run). The catalogue check (`fin_cases`) is ∼131 ms even at order 6; the cost is the per-tree arithmetic. Integer `decide` over $\mathbb{Z}[\sqrt{15}]$ certifies the 65 order-6 conditions faster than `norm_num` over $\mathbb{Q}(\sqrt{15})$ certifies the 9 order-4 ones, while the `norm_num` route at order 6 does not finish (∼20 GB).

7 Scope and limits

We state the boundary exactly. **(1)** We certify the *algebraic* order conditions $\gamma(t)\Phi(t) = 1$; we do *not* formalize Butcher's analytic theorem that satisfying them is equivalent to a local error $O(h^{p+1})$ (that needs B-series convergence, elementary differentials, and Taylor expansion of the flow—future work). **(2)** The trees are planar; the bridge to the abstract conditions is Lemma 1, not a formalization of Foissy's full Hopf morphism.

(3) Formalization, not new mathematics. The mathematics is classical throughout (Butcher [1]; Foissy [5]). The keystone (Theorem 1) is a formalization-enabling *reduction*—a finite

catalogue check is how a textbook proof proceeds informally—and its content here is only that the reduction, its completeness, and the resulting certificates are machine-checked. The contribution is the reusable implementation, not a new theorem; a numerical analyst would rightly say the novelty lies in formalization rather than mathematics, and we agree.

(4) The Hopf-algebraic framing is contextual. Section 9 situates the order conditions as *characters* of the Connes–Kreimer/Butcher Hopf algebra and links the companion [17]; that perspective is motivational here. The certificates in this paper use only the trees, order, γ, Φ, and the permutation invariance of Lemma 1: the coproduct, antipode, Butcher-group law and pre-Lie bracket are *not* invoked—they are formalized in [17]. The title names rooted trees, not the full Hopf structure, for this reason.

(5) The "not previously formalized" claim is a focused search, not a theorem. A forgotten development could exist and we would be glad to be corrected. Our statement rests on the search below (conducted June 2026); it is focused, not exhaustive. In each system we searched the entry index and full text for the terms *Runge–Kutta*, *Butcher*, *rooted tree*, *order condition(s)* and *B-series*.

System	Sources searched
Lean 4 / Mathlib	Mathlib repository, Loogle, Moogle, `leanprover` Zulip
Isabelle/HOL	Archive of Formal Proofs (entry index + full text)
Coq/Rocq	`opam` index, coq-community, web search

The only hit on machine-checked RK methods was Boldo *et al.* [8] (Coq, floating-point round-off of Euler/RK2)—an orthogonal layer, not the order theory.

8 Prior art

The order theory: Butcher [1]; Hairer–Lubich–Wanner [2]; Chartier–Hairer–Vilmart [3]. The Hopf-algebraic reading: Connes–Kreimer [4]; the planar (noncommutative) algebra and its abelianization, Foissy [5, 6]; the planar binary-tree algebra, Loday–Ronco [12]; operadic planar/non-planar relations, Chapoton [7]. An informal recursive-tree proof of Butcher's theorem is in Bornemann [13]. Software: `NodePy` [9], `BSeries.jl` [10], Sofroniou [11]. Formal methods on RK: Boldo *et al.* [8] (Coq round-off, orthogonal). Companion formalizations: the planar Connes–Kreimer–Foissy Hopf algebra in Lean [17] and the Eulerian idempotent / backward error [18], both on Mathlib [14].

9 The Hopf-algebraic reading: order conditions as characters

Why is this the same object as renormalization and the free pre-Lie algebra? Butcher's order theory and the Connes–Kreimer Hopf algebra of rooted trees are two readings of one structure [4, 1, 5]. The trees here are a basis of that Hopf algebra H; the forest product is its multiplication; the admissible-cut coproduct—formalized in the companion paper [17]—is dual to the *composition of numerical flows*, the group law of the Butcher group. A B-series method is a *character* of H (an algebra morphism $H \to \mathbb{R}$, i.e. multiplicative on forests); for a tableau (A, b) that character is exactly our elementary weight $t \mapsto \Phi(t)(A, b)$. The exact time-h flow is itself a character, $t \mapsto 1/\gamma(t)$. In this language the whole of order theory is one statement:

$$\text{a method has order } p \iff \Phi \text{ and } 1/\gamma \text{ agree on } H_{\le p},$$

the method's character and the exact-flow character coinciding on the graded pieces up to order p. The antipode formalized in [17] is the *inverse* (adjoint) method; its logarithm—the Eulerian idempotent of [18]—is the modified equation, the backward error of a method. The three formalizations are thus three faces of one Hopf algebra: its *structure* [17], its *characters* (the order conditions, this paper), and its *Lie/logarithm* side (backward error, [18]). What [17] listed as the applied payoff its agenda opened, this paper delivers.

10 What this unlocks

The reusable core—trees, order, γ, Φ, the order-budget generator, the keystone—is the substrate for two next steps already visible: certified *backward error* (the modified equation, via the Eulerian idempotent of Paper II) and the substitution/composition laws (Chapoton's operadic picture). The certified order-condition checker is the first rung.

References

[1] J. C. Butcher, *Coefficients for the study of Runge–Kutta integration processes*, J. Austral. Math. Soc. **3** (1963), 185–201.

[2] E. Hairer, C. Lubich, G. Wanner, *Geometric Numerical Integration*, Springer, 2006.

[3] P. Chartier, E. Hairer, G. Vilmart, *Algebraic structures of B-series*, Found. Comput. Math. **10** (2010), 407–427.

[4] A. Connes, D. Kreimer, *Hopf algebras, renormalization and noncommutative geometry*, Comm. Math. Phys. **199** (1998), 203–242.

[5] L. Foissy, *Les algèbres de Hopf des arbres enracinés décorés I*, Bull. Sci. Math. **126** (2002), 193–239. (arXiv:math/0105212; planar/noncommutative algebra, self-duality, abelianization Prop. 81.)

[6] L. Foissy, *Les algèbres de Hopf des arbres enracinés décorés II*, Bull. Sci. Math. **126** (2002), 249–288.

[7] F. Chapoton, *Operads and combinatorial Hopf algebras of trees* (and related works on planar vs. non-planar tree structures).

[8] S. Boldo, F. Faissole, *et al.*, *Formally-verified round-off error analysis of Runge–Kutta methods*, J. Automated Reasoning **68** (2024).

[9] D. I. Ketcheson, *NodePy*, J. Open Source Software **5**(55):2515, 2020.

[10] D. I. Ketcheson, H. Ranocha, *Computing with B-series*, ACM TOMS **49**(2), 2023.

[11] M. Sofroniou, *Symbolic derivation of Runge–Kutta methods*, J. Symbolic Comput. **18** (1994), 265–296.

[12] J.-L. Loday, M. O. Ronco, *Hopf algebra of the planar binary trees*, Adv. Math. **139** (1998), 293–309.

[13] F. Bornemann, *Runge–Kutta methods, trees, and Mathematica*, 2002, arXiv:math/0211049. (An informal recursive-tree proof of Butcher's order theorem.)

[14] The mathlib Community, *The Lean Mathematical Library*, CPP 2020; Lean 4 toolchain `v4.30.0-rc2`.

[15] OEIS Foundation, *Sequence A000081: unlabeled rooted trees*, https://oeis.org/A000081.

[16] OEIS Foundation, *Sequence A000108: Catalan numbers (ordered/planar rooted trees)*, https://oeis.org/A000108.

[17] C. Marín, *How a Tree Remembers Its Cuts: the Connes–Kreimer–Foissy Hopf algebra of rooted trees, machine-checked in Lean 4*, 2026 (companion paper).

[18] C. Marín, *The Eulerian Idempotent and backward error of B-series, in Lean 4*, 2026 (companion paper).