Article

Not peer-reviewed version

Real Differentiation: A Fiberwise, Stack-Enriched Calculus of Change

Parker Emmerson *

Posted Date: 25 December 2025

doi: 10.20944/preprints202512.2259.v1

Keywords: differentiation; differential equations; stack; full stack; calculus; fiber; fiberwise; differential-operator

 doi:10.20944/preprints202512.2259.v1

Article

Real Differentiation: A Fiberwise, Stack-Enriched Calculus of Change

Parker Emmerson

Antioch College, USA; parkeremmerson@icloud.com

Abstract

We develop *Real Differentiation*: a fibre- and stack-sensitive language for infinitesimal change whose basic objects are *labelled* ("energy-tagged") fields and operators. The formalism is designed to enforce a *no branch mixing* principle: differentiation, integration, and PDE operators act fibrewise on each labelled branch, while still permitting monodromy-aware transport when labels are organised as a covering space or stack of analytic continuations. The core construction is a jet-theoretic endofunctor D, defined via the first infinitesimal neighbourhood of the diagonal. In the constant-label case $\mathbb{E}_{\mathscr{X}}$, we show that the labelled topos $(\mathrm{Sh}(\mathcal{B})/\mathscr{X})/\mathbb{E}_{\mathscr{X}}$ decomposes canonically as a product of copies of $\mathrm{Sh}(\mathcal{B})/\mathscr{X}$, so that labelled objects are literally *families* indexed by energy values. This makes "no branch mixing" a theorem: all constructions in the labelled topos act componentwise. In the stacky/local-system case, labels are organised by an étale map (or stack) $\mathcal{E} \to \mathscr{X}$; labelled objects are sheaves on the total space and carry monodromy by descent. We further define directional derivatives by contraction of the universal derivation with vector fields; we prove Leibniz and chain rules, including a smooth/analytic chain rule under explicit functional-calculus hypotheses. Higher jets are related to differential operators, and an infinite-jet comonad governs the differential-operator calculus. Finally, we introduce *controlled* energy mixing via correspondences of label objects, providing a principled way to model coupled branch dynamics.

Keywords: differentiation; differential equations; stack; full stack; calculus; fiber; fiberwise; differential-operator

---

1. Motivation and Outline

Conventional calculus differentiates *numbers*. In spectral, multivalued, or analytic-continuation settings—where "a function" may in fact live on multiple branches and is only single-valued after choosing a sheet—naïve differentiation on a chosen principal branch is not stable under monodromy, and can break fundamental identities (e.g. pathwise versions of the fundamental theorem).

Real Differentiation replaces unlabelled scalars by *labelled fields*: objects living in a slice topos over a label object. In the simplest case, labels form a constant discrete object $\mathbb{E}_{\mathscr{X}}$; then labelled objects are equivalent to families indexed by energy values, and "no branch mixing" is enforced categorically. In the monodromy-aware case, labels form a covering space or stack $\mathcal{E} \to \mathscr{X}$; then labelled objects are sheaves on the total space, and monodromy is encoded by descent along $\mathcal{E} \to \mathscr{X}$ rather than being ignored.

After setting up the labelled foundations (Section 2), we define the Real Differential functor D via first jets (Section 3), prove fibrewise base-change properties, and define directional derivatives by universal differentials (Section 4). We then extend the chain rule beyond polynomials under explicit hypotheses, develop higher jets and a jet comonad governing differential operators (Sections 5 and 6), construct the Real de Rham complex and fibrewise integration (Section 7), and introduce controlled energy mixing via label correspondences (Section 9). Computational demonstrations are provided as in-document listings (Appendices A.2 and A.3).

 doi:10.20944/preprints202512.2259.v1

Positioning and Scope

Jets, Kähler differentials, and the jet/differential-operator adjunction are classical (EGA, Hartshorne, Illusie). The contribution here is the systematic *labelled* packaging: (i) a decomposition theorem showing constant-labelled objects are families; (ii) a cover/stack label model capturing monodromy; (iii) a jet-based differentiation functor and identities that commute with passage to label fibres; and (iv) a controlled mixing extension via correspondences.

2. Energy–Fibred Foundations

2.1. Energy FIELD, Base Topos, Ringed Structure

Fix a commutative real-closed *energy field* $(\mathbb{E}, +, \cdot)$. Let $\mathcal{B}$ be a Grothendieck site, and work in the ringed topos $(\mathrm{Sh}(\mathcal{B}), \mathcal{O}_{\mathcal{B}})$. Unless stated otherwise, all internal algebra is $\mathcal{O}_{\mathcal{B}}$-linear.

Remark 1 (Size convention)**.** *To form coproducts indexed by underlying sets of energy values, we assume the underlying set of $\mathbb{E}$ is small in a fixed Grothendieck universe, or we work universe-relatively. This is standard in topos-theoretic constructions and does not affect the geometric content.*

Definition 1 (Energy-affine space)**.** *For $n \geq 0$ set*

$$\mathbb{A}^n_{\mathbb{E}} := \operatorname{Spec} \mathbb{E}[t_1, \ldots, t_n],$$

viewed as an object of $\mathcal{B}$.

2.2. Geometry over $\mathscr{X}$ and Relative Differentials

Fix a morphism $\pi : \mathscr{X} \to \mathbb{A}^n_{\mathbb{E}}$ in $\mathcal{B}$, assumed energy-smooth when needed (Definition 9). We write $(\mathrm{Sh}(\mathcal{B})/\mathscr{X}, \mathcal{O}_{\mathscr{X}})$ for the induced ringed topos over $\mathscr{X}$. Relative Kähler differentials $\Omega^1_{\mathscr{X}/\mathbb{E}}$ are an $\mathcal{O}_{\mathscr{X}}$-module, and we set

$$\mathcal{T}_{\mathscr{X}/\mathbb{E}} := \underline{\mathrm{Hom}}_{\mathcal{O}_{\mathscr{X}}}(\Omega^1_{\mathscr{X}/\mathbb{E}}, \mathcal{O}_{\mathscr{X}}).$$

When discussing vector fields, derivations, jets, and de Rham theory, we work in $(\mathrm{Sh}(\mathcal{B})/\mathscr{X}, \mathcal{O}_{\mathscr{X}})$.

2.3. Label Objects and Labelled Module Objects

A central organising idea is to separate two layers: (i) a geometric base $\mathscr{X}$, and (ii) a *label object* $\mathcal{E}$ over $\mathscr{X}$ whose fibres represent branch/energy/sheet data.

Definition 2 (Label object and labelled topos)**.** *A* label object over $\mathscr{X}$ *is an object $\mathcal{E}$ of $\mathrm{Sh}(\mathcal{B})/\mathscr{X}$. The associated* labelled topos *is the slice*

$$(\mathrm{Sh}(\mathcal{B})/\mathscr{X})/\mathcal{E}.$$

*An $\mathcal{E}$-*labelled object *is a morphism $F \to \mathcal{E}$ in $\mathrm{Sh}(\mathcal{B})/\mathscr{X}$.*

Definition 3 (Labelled structure sheaf and labelled modules)**.** *Let $p : (\mathrm{Sh}(\mathcal{B})/\mathscr{X})/\mathcal{E} \to \mathrm{Sh}(\mathcal{B})/\mathscr{X}$ be the projection geometric morphism. Define the* pulled-back structure sheaf

$$\mathcal{O}_{\mathscr{X}\mathcal{E}} := p^* \mathcal{O}_{\mathscr{X}},$$

*a ring object in the labelled topos. An $\mathcal{O}_{\mathscr{X}\mathcal{E}}$-*module *means a module object in $(\mathrm{Sh}(\mathcal{B})/\mathscr{X})/\mathcal{E}$ over the ring $\mathcal{O}_{\mathscr{X}\mathcal{E}}$.*

Remark 2 (Why modules must be internal)**.** *This formalism avoids a subtle but important pitfall: a label map is not naturally an $\mathcal{O}_{\mathscr{X}}$-linear morphism to a ring of scalars. Instead, labels are tracked by working* in the slice topos, *where all algebra (including module structure) is internal and automatically respects the labelling.*

 doi:10.20944/preprints202512.2259.v1

2.4. Constant Energy Labels and the Slice-as-Family Decomposition

We now define the constant energy label object and prove that it yields a canonical product decomposition.

Definition 4 (Constant energy label object $\underline{\mathbb{E}}_{\mathscr{X}}$)**.** *Let $1_{\mathscr{X}}$ denote the terminal object of $\mathrm{Sh}(\mathcal{B})/\mathscr{X}$. The* constant energy label object on $\mathscr{X}$ *is the constant object with fibre $\mathbb{E}$,*

$$\underline{\mathbb{E}}_{\mathscr{X}} := \Delta_{\mathscr{X}}(\mathbb{E}) \;\cong\; \coprod_{E\in\mathbb{E}} 1_{\mathscr{X}} \qquad \text{in } \mathrm{Sh}(\mathcal{B})/\mathscr{X}.$$

For each $E \in \mathbb{E}$, write $\iota_E : 1_{\mathscr{X}} \to \underline{\mathbb{E}}_{\mathscr{X}}$ for the coproduct inclusion.

Definition 5 (Energy fibre (constant-label case))**.** *Let $F \to \underline{\mathbb{E}}_{\mathscr{X}}$ be an $\underline{\mathbb{E}}_{\mathscr{X}}$-labelled object in $\mathrm{Sh}(\mathcal{B})/\mathscr{X}$. For $E \in \mathbb{E}$, its* energy fibre *is the pullback*

$$F_E := F \times_{\underline{\mathbb{E}}_{\mathscr{X}}} 1_{\mathscr{X}} \quad \text{along } \iota_E : 1_{\mathscr{X}} \to \underline{\mathbb{E}}_{\mathscr{X}}.$$

Theorem 1 (Slice-as-family decomposition for constant labels)**.** *There is an equivalence of toposes*

$$(\mathrm{Sh}(\mathcal{B})/\mathscr{X})/\underline{\mathbb{E}}_{\mathscr{X}} \;\simeq\; \prod_{E\in\mathbb{E}} (\mathrm{Sh}(\mathcal{B})/\mathscr{X}),$$

under which a labelled object $F \to \underline{\mathbb{E}}_{\mathscr{X}}$ corresponds to its family of fibres $(F_E)_{E\in\mathbb{E}}$, and morphisms correspond to families of morphisms $F_E \to G_E$.

Proof. In any Grothendieck topos, coproducts are disjoint and stable under pullback (extensivity). Since $\underline{\mathbb{E}}_{\mathscr{X}} \cong \coprod_{E\in\mathbb{E}} 1_{\mathscr{X}}$, slicing over $\underline{\mathbb{E}}_{\mathscr{X}}$ decomposes as a product of slices over each summand:

$$(\mathrm{Sh}(\mathcal{B})/\mathscr{X})/\Big(\coprod_{E\in\mathbb{E}} 1_{\mathscr{X}}\Big) \simeq \prod_{E\in\mathbb{E}} (\mathrm{Sh}(\mathcal{B})/\mathscr{X})/1_{\mathscr{X}}.$$

But $(\mathrm{Sh}(\mathcal{B})/\mathscr{X})/1_{\mathscr{X}} \simeq \mathrm{Sh}(\mathcal{B})/\mathscr{X}$ because $1_{\mathscr{X}}$ is terminal. The equivalence is realised by pulling back along the inclusions ι_E, yielding the family of fibres F_E, and conversely by taking the coproduct $\coprod_E F_E \to \coprod_E 1_{\mathscr{X}} = \underline{\mathbb{E}}_{\mathscr{X}}$. □

Corollary 1 (Canonical coproduct decomposition)**.** *For any $\underline{\mathbb{E}}_{\mathscr{X}}$-labelled object $F \to \underline{\mathbb{E}}_{\mathscr{X}}$ there is a canonical isomorphism in $\mathrm{Sh}(\mathcal{B})/\mathscr{X}$:*

$$F \;\cong\; \coprod_{E\in\mathbb{E}} F_E.$$

Proposition 1 (Modules in the constant-labelled topos are families of modules)**.** *The category of $\mathcal{O}_{\mathscr{X}\underline{\mathbb{E}}_{\mathscr{X}}}$-modules in the labelled topos $(\mathrm{Sh}(\mathcal{B})/\mathscr{X})/\underline{\mathbb{E}}_{\mathscr{X}}$ is canonically equivalent to the product category*

$$\mathcal{O}_{\mathscr{X}\underline{\mathbb{E}}_{\mathscr{X}}}\text{-Mod}\big((\mathrm{Sh}(\mathcal{B})/\mathscr{X})/\underline{\mathbb{E}}_{\mathscr{X}}\big) \;\simeq\; \prod_{E\in\mathbb{E}} \mathcal{O}_{\mathscr{X}}\text{-Mod}\big(\mathrm{Sh}(\mathcal{B})/\mathscr{X}\big).$$

In particular, any construction in the labelled topos defined by topos limits, colimits, and $\mathcal{O}_{\mathscr{X}}$-module operations acts componentwise on fibres.

Proof. Apply Theorem 1 and observe that pulling back $\mathcal{O}_{\mathscr{X}\underline{\mathbb{E}}_{\mathscr{X}}} = p^*\mathcal{O}_{\mathscr{X}}$ to the Eth factor recovers $\mathcal{O}_{\mathscr{X}}$. Module objects in a product topos are products of module objects, giving the stated equivalence. □

2.5. Reformulated Tag Models: Model A and Model B

We now restate your two tag semantics in the rigorous language of Theorem 1.

 doi:10.20944/preprints202512.2259.v1

Definition 6 (Model A: single-energy (concentrated) objects). *An $\underline{\mathbb{E}}_{\mathscr{X}}$-labelled object $F \to \underline{\mathbb{E}}_{\mathscr{X}}$ is* concentrated at energy $E_0 \in \mathbb{E}$ *if $F_E = \varnothing$ for all $E \neq E_0$. We write $F@E_0$ for such an object.*

Definition 7 (Model B: general labelled objects). *A* general labelled object *is an arbitrary $F \to \underline{\mathbb{E}}_{\mathscr{X}}$. Fibrewise reasoning proceeds by applying the pullback functors $(-)_E$ of Definition 5. By Theorem 1, this is equivalent to reasoning componentwise in the family $(F_E)_{E\in\mathbb{E}}$.*

Remark 3 (No branch mixing as a theorem). *In Model B, any morphism $F \to G$ in the labelled topos is equivalent to a family of morphisms $F_E \to G_E$. Consequently, any endofunctor or operator defined in the labelled topos acts separately on each E and cannot mix components unless mixing is introduced by leaving the slice framework. This is the precise form of "no branch mixing."*

2.6. Stacky/Local-System Labels: Covers and Monodromy

The constant-label model treats labels as a trivial discrete family. To encode analytic continuation and monodromy, labels must instead vary over $\mathscr{X}$.

Definition 8 (Étale label object (cover of branches)). *A label object $\mathcal{E}$ over $\mathscr{X}$ is called* étale *if it is represented by an étale (or, in the present terminology, energy-smooth and unramified) morphism $p : \widetilde{\mathscr{X}} \to \mathscr{X}$ in $\mathcal{B}$ such that fibres are discrete sets of branches/sheets.*

Proposition 2 (Labelled objects on an étale label are sheaves on the total space). *If $\mathcal{E}$ is represented by a morphism $p : \widetilde{\mathscr{X}} \to \mathscr{X}$, then there is an equivalence of toposes*

$$(\mathrm{Sh}(\mathcal{B})/\mathscr{X})/\mathcal{E} \;\simeq\; \mathrm{Sh}(\mathcal{B})/\widetilde{\mathscr{X}}.$$

Under this equivalence, $\mathcal{O}_{\mathscr{X}\mathcal{E}}$-modules correspond to $\mathcal{O}_{\widetilde{\mathscr{X}}}$-modules.

Proof. This is the standard identification of a slice topos over a representable object with the topos over the representing object, using the fact that slicing is associative: $(\mathscr{E}/\mathscr{X})/\widetilde{\mathscr{X}} \simeq \mathscr{E}/\widetilde{\mathscr{X}}$ for any topos $\mathscr{E}$. Here $\mathscr{E} = \mathrm{Sh}(\mathcal{B})$. □

Remark 4 (Monodromy as descent data). *If $p : \widetilde{\mathscr{X}} \to \mathscr{X}$ is a covering with deck group Γ, then objects on $\mathscr{X}$ with branch data correspond to Γ-equivariant objects on $\widetilde{\mathscr{X}}$. In particular, a multivalued analytic function on $\mathscr{X}$ becomes a single-valued function on $\widetilde{\mathscr{X}}$, and monodromy is encoded by the Γ-action rather than ignored. Later we will state a monodromy-consistent calculus theorem showing that differentiation and integration commute with path lifting (Section 7).*

Remark 5 (Stacks of labels (optional)). *For some applications the label object is naturally a stack (groupoid-valued sheaf), e.g. when branches are identified up to symmetry. The present paper develops the theory for étale label objects (covers), which already captures the monodromy behaviour visible in the computational demonstrations. Extending from étale covers to stacks is formal: replace $\mathcal{E}$ by a groupoid object and work in the corresponding 2-topos of stacks.*

2.7. Energy-Smooth Morphisms

We will need a base-change hypothesis that guarantees Kähler differentials and infinitesimal neighbourhoods commute with pullback.

Definition 9 (Energy-flat and energy-smooth). *A morphism $u : \mathscr{X}' \to \mathscr{X}$ in $\mathcal{B}$ is* energy-flat *if pullback along u is exact on $\mathcal{O}_{\mathscr{X}}$-modules. It is* energy-smooth *if, furthermore, the canonical map*

$$u^*\Omega^1_{\mathscr{X}/\mathbb{E}} \to \Omega^1_{\mathscr{X}'/\mathbb{E}}$$

 doi:10.20944/preprints202512.2259.v1

is an isomorphism and the formation of the first infinitesimal neighbourhood of the diagonal commutes with pullback along u.

3. The Real Differential Functor (Via First Jets)

3.1. First Infinitesimal Neighbourhood of the Diagonal and Principal Parts

Let Y be an object of $\mathcal{B}$ equipped with a structure sheaf $\mathcal{O}_Y$, and assume Y is energy-smooth over $\mathbb{E}$ (in the sense appropriate to the geometric model: schemes, manifolds, etc.). Write $Y \times_{\mathbb{E}} Y$ for the fibre product over $\operatorname{Spec}\mathbb{E}$, and let

$$\Delta_Y : Y \hookrightarrow Y \times_{\mathbb{E}} Y$$

be the diagonal. Let $\mathcal{I} \subseteq \mathcal{O}_{Y\times_{\mathbb{E}}Y}$ be the ideal sheaf of Δ_Y.

Definition 10 (First infinitesimal neighbourhood)**.** *The* first infinitesimal neighbourhood of the diagonal *is the closed subobject*

$$(Y \times_{\mathbb{E}} Y)^{[1]} \;:=\; \operatorname{Spec}\big(\mathcal{O}_{Y\times_{\mathbb{E}}Y}/\mathcal{I}^2\big),$$

with the induced projections

$$p_1, p_2 : (Y \times_{\mathbb{E}} Y)^{[1]} \longrightarrow Y.$$

Definition 11 (Sheaf of principal parts of order 1)**.** *Define the* sheaf of principal parts *of order* 1 *on* Y *to be*

$$\mathcal{P}^1_{Y/\mathbb{E}} := p_{1*}\mathcal{O}_{(Y\times_{\mathbb{E}}Y)^{[1]}}.$$

It is naturally an $\mathcal{O}_Y$-algebra (via p_1), and carries a canonical augmentation $\epsilon : \mathcal{P}^1_{Y/\mathbb{E}} \to \mathcal{O}_Y$ induced by restriction along the diagonal.

Remark 6. *The construction of $\mathcal{P}^1_{Y/\mathbb{E}}$ and the exact sequences below are standard (principal parts / jets / differentials); see [1, IV] or [2, II, §8].*

3.2. The First Jet Functor

Definition 12 (First jet functor)**.** *Let $\mathcal{F}$ be an $\mathcal{O}_Y$-module. Define*

$$J^1_{Y/\mathbb{E}}(\mathcal{F}) \;:=\; p_{1*}\big(p_2^*\mathcal{F}\big) \;\cong\; \mathcal{P}^1_{Y/\mathbb{E}} \otimes_{\mathcal{O}_Y} \mathcal{F}.$$

Proposition 3 (Fundamental exact sequence for first jets)**.** *There is a natural short exact sequence of $\mathcal{O}_Y$-modules*

$$0 \longrightarrow \Omega^1_{Y/\mathbb{E}} \otimes_{\mathcal{O}_Y} \mathcal{F} \xrightarrow{\iota} J^1_{Y/\mathbb{E}}(\mathcal{F}) \xrightarrow{\epsilon} \mathcal{F} \longrightarrow 0, \tag{3.1}$$

where ϵ is induced by restriction along the diagonal.

Reference. See ([2] II, §8) or ([1] IV). □

3.3. Real Differentiation on a Labelled Base

We now specialise to the context of Section 2. There are two natural choices of "base" on which calculus is performed:

- the geometric base $\mathscr{X}$, yielding ordinary (unlabelled) calculus on $\mathscr{X}$;
- a label base $\mathcal{E} \to \mathscr{X}$ (constant family $\underline{\mathbb{E}}_{\mathscr{X}}$ or an étale cover $\widetilde{\mathscr{X}} \to \mathscr{X}$), yielding labelled calculus.

Since the labelled topos $(\mathrm{Sh}(\mathcal{B})/\mathscr{X})/\mathcal{E}$ is equivalent to $\mathrm{Sh}(\mathcal{B})/\mathcal{E}$ when $\mathcal{E}$ is representable (Proposition 2), it is natural to perform jets on the label base itself.

Definition 13 (Real Differential functor on a labelled base)**.** *Assume the label object $\mathcal{E}$ is representable by an energy-smooth morphism $p : \mathcal{E} \to \mathscr{X}$ (examples: $\mathcal{E} = \underline{\mathbb{E}}_{\mathscr{X}}$ or $\mathcal{E} = \widetilde{\mathscr{X}}$ étale over $\mathscr{X}$). Write $\mathcal{O}_{\mathcal{E}}$ for the induced structure sheaf.*

 doi:10.20944/preprints202512.2259.v1

Define the Real Differential functor on $\mathcal{E}$ *to be the endofunctor*

$$\mathrm{D}_{\mathcal{E}} : \mathcal{O}_{\mathcal{E}}\text{-Mod} \longrightarrow \mathcal{O}_{\mathcal{E}}\text{-Mod}, \qquad \mathrm{D}_{\mathcal{E}}(\mathcal{F}) := J^1_{\mathcal{E}/\mathbb{E}}(\mathcal{F}).$$

The augmentation $\epsilon : J^1_{\mathcal{E}/\mathbb{E}}(\mathcal{F}) \to \mathcal{F}$ *is called the* tag-preserving counit.

Remark 7 (Tag preservation is automatic)**.** *Working on* $\mathcal{E}$ *ensures that all constructions are intrinsically label-aware: a section of* $\mathcal{O}_{\mathcal{E}}$ *is already a label-tagged scalar. The map* ϵ *does not change the basepoint of* $\mathcal{E}$*, hence does not change the label. In the constant-label case* $\mathcal{E} = \underline{\mathbb{E}}_{\mathscr{X}}$*, this recovers "no branch mixing" in the strongest possible form:* D *acts componentwise on energy indices (Section 2.4).*

3.4. Constant Energy Labels: $\mathcal{E} = \underline{\mathbb{E}}_{\mathscr{X}}$

Let $\mathcal{E} = \underline{\mathbb{E}}_{\mathscr{X}} \cong \coprod_{E\in\mathbb{E}} 1_{\mathscr{X}}$ as in Definition 4. Then $\mathcal{E}$ is a disjoint union of copies of $\mathscr{X}$, and $\mathcal{O}_{\mathcal{E}}$ is the corresponding coproduct of copies of $\mathcal{O}_{\mathscr{X}}$.

Proposition 4 (Componentwise differentiation in the constant-label model)**.** *Under the equivalence*

$$(\mathrm{Sh}(\mathcal{B})/\mathscr{X})/\underline{\mathbb{E}}_{\mathscr{X}} \simeq \prod_{E\in\mathbb{E}} (\mathrm{Sh}(\mathcal{B})/\mathscr{X})$$

of Theorem 1, the functor $\mathrm{D}_{\underline{\mathbb{E}}_{\mathscr{X}}}$ *corresponds to the product of jet functors on* $\mathscr{X}$*:*

$$\mathrm{D}_{\underline{\mathbb{E}}_{\mathscr{X}}} \;\leftrightarrow\; \prod_{E\in\mathbb{E}} J^1_{\mathscr{X}/\mathbb{E}}.$$

Equivalently, for $\mathcal{O}_{\underline{\mathbb{E}}_{\mathscr{X}}}$*-modules* $\mathcal{F}$*,*

$$\big(\mathrm{D}_{\underline{\mathbb{E}}_{\mathscr{X}}}(\mathcal{F})\big)_E \;\cong\; J^1_{\mathscr{X}/\mathbb{E}}(\mathcal{F}_E)$$

naturally in $\mathcal{F}$ *and in* $E \in \mathbb{E}$*.*

Proof. Since $\underline{\mathbb{E}}_{\mathscr{X}}$ is a disjoint coproduct of copies of $1_{\mathscr{X}}$, the labelled base $\underline{\mathbb{E}}_{\mathscr{X}}$ is a disjoint union of copies of $\mathscr{X}$. Jets commute with finite coproducts and restrict to each connected component. Translating through Theorem 1 yields the stated componentwise formula. □

3.5. Étale Label Covers: $\mathcal{E} = \widetilde{\mathscr{X}} \to \mathscr{X}$

Let $p : \widetilde{\mathscr{X}} \to \mathscr{X}$ be an étale label cover encoding branches (e.g. a Riemann surface of analytic continuations). Then labelled objects over $\mathscr{X}$ are equivalently objects over $\widetilde{\mathscr{X}}$ (Proposition 2), and Real differentiation is simply differentiation on $\widetilde{\mathscr{X}}$.

Proposition 5 (Label-cover Real differentiation)**.** *For an étale label cover* $p : \widetilde{\mathscr{X}} \to \mathscr{X}$*, the labelled Real Differential functor is*

$$\mathrm{D}_{\widetilde{\mathscr{X}}} = J^1_{\widetilde{\mathscr{X}}/\mathbb{E}} \quad \text{on} \quad \mathcal{O}_{\widetilde{\mathscr{X}}}\text{-Mod}.$$

In particular, deck transformations act by automorphisms commuting with D*.*

Proof. This is immediate from Definition 13 applied to $\mathcal{E} = \widetilde{\mathscr{X}}$. Deck transformations are automorphisms of $\widetilde{\mathscr{X}}$ over $\mathscr{X}$ and hence preserve the diagonal and its infinitesimal neighbourhood, commuting with jet formation. □

 doi:10.20944/preprints202512.2259.v1

3.6. Base Change for First Jets

Proposition 6 (Base change for first jets)**.** *Let $u : Y' \to Y$ be an energy-smooth morphism. Then for any $\mathcal{O}_Y$-module $\mathcal{F}$ there is a canonical isomorphism of $\mathcal{O}_{Y'}$-modules*

$$u^* J^1_{Y/\mathbb{E}}(\mathcal{F}) \xrightarrow{\sim} J^1_{Y'/\mathbb{E}}(u^*\mathcal{F}),$$

functorial in $\mathcal{F}$.

Proof sketch. Form the cartesian square for diagonals and their first infinitesimal neighbourhoods:

$$\begin{array}{ccc} (Y' \times_{\mathbb{E}} Y')^{[1]} & \longrightarrow & (Y \times_{\mathbb{E}} Y)^{[1]} \\ {\scriptstyle p_1'}\downarrow & & \downarrow{\scriptstyle p_1} \\ Y' & \xrightarrow[u]{} & Y \end{array}$$

Energy-smoothness ensures formation of $(Y \times_{\mathbb{E}} Y)^{[1]}$ commutes with base change, and that Ω^1 behaves well. Since p_1 is affine in the usual geometric settings (schemes, ringed spaces), pushforward along p_1 commutes with pullback along u on quasi-coherent modules. Apply this to $p_2^*\mathcal{F}$ to obtain the desired isomorphism. See [1, IV] or [2, II, §8] for details in the scheme case. □

Corollary 2 (Base change for Real differentiation)**.** *Under the hypotheses of Proposition 6, the Real Differential functor commutes with base change:*

$$u^* \mathrm{D}_Y(\mathcal{F}) \cong \mathrm{D}_{Y'}(u^*\mathcal{F}).$$

In particular, in the constant-label model this yields fibrewise identities $(\mathrm{D}_{\underline{\mathbb{E}}_{\mathscr{X}}}\mathcal{F})_E \cong \mathrm{D}_{\mathscr{X}}(\mathcal{F}_E)$.

4. Directional Derivatives and Universal Identities

4.1. Directional Derivatives on Scalar Functions

Let Y be an energy-smooth base (either $\mathscr{X}$, $\underline{\mathbb{E}}_{\mathscr{X}}$, or an étale label cover $\widetilde{\mathscr{X}}$). The universal derivation

$$d : \mathcal{O}_Y \longrightarrow \Omega^1_{Y/\mathbb{E}}$$

is an $\mathbb{E}$-linear derivation. If $v \in \Gamma(T_{Y/\mathbb{E}})$ is a vector field (i.e. an $\mathcal{O}_Y$-linear map $\Omega^1_{Y/\mathbb{E}} \to \mathcal{O}_Y$), define the directional derivative of a scalar $f \in \mathcal{O}_Y$ by contraction.

Definition 14 (Directional derivative)**.** *Let $v \in \Gamma(T_{Y/\mathbb{E}})$. Define*

$$\nabla_v(f) := \iota_v(df) \in \mathcal{O}_Y.$$

Proposition 7 (Leibniz rule)**.** *For all $f, g \in \mathcal{O}_Y$,*

$$\nabla_v(fg) = \nabla_v(f)\, g + f\, \nabla_v(g).$$

Proof. This is immediate from $d(fg) = f\, dg + g\, df$ and $\mathcal{O}_Y$-linearity of ι_v. □

4.2. Polynomial Chain Rule (Always Valid)

Proposition 8 (Polynomial chain rule)**.** *Let $F(t) \in \mathbb{E}[t]$ and $a \in \mathcal{O}_Y$. Then*

$$\nabla_v\big(F(a)\big) = F'(a)\, \nabla_v(a).$$

Proof. Expand $F(t) = \sum_k c_k t^k$ and use repeated Leibniz to obtain $d(a^k) = k\, a^{k-1}\, da$, hence $d(F(a)) = F'(a)\, da$. Contract with v. □

 doi:10.20944/preprints202512.2259.v1

4.3. Upgraded Chain Rule: Smooth/Analytic Functional Calculus

The polynomial chain rule is universal in the purely algebraic setting. To state a chain rule for genuinely smooth or analytic functions, one must assume that $\mathcal{O}_Y$ is equipped with a corresponding functional calculus.

Definition 15 (C^1-functional calculus (one variable)). *Assume $\mathbb{E} = \mathbb{R}$. A C^1-functional calculus on a sheaf of commutative $\mathbb{R}$-algebras $\mathcal{O}_Y$ is an assignment that to each C^1 function $\varphi : \mathbb{R} \to \mathbb{R}$ associates a sheaf morphism*

$$\varphi_{\mathcal{O}} : \mathcal{O}_Y \longrightarrow \mathcal{O}_Y,$$

such that:

(a) $\varphi \mapsto \varphi_{\mathcal{O}}$ respects composition and identities: $(\varphi \circ \psi)_{\mathcal{O}} = \varphi_{\mathcal{O}} \circ \psi_{\mathcal{O}}$ and $(\mathrm{id})_{\mathcal{O}} = \mathrm{id}$;
(b) for polynomials $F(t) \in \mathbb{R}[t]$, one has $F_{\mathcal{O}}(a) = F(a)$ under the algebra structure.

A C^∞-functional calculus is defined similarly with $\varphi \in C^\infty(\mathbb{R})$.

Remark 8. *A sheaf of C^∞-rings in the sense of Dubuc–Moerdijk–Reyes (or in C^∞-algebraic geometry) canonically carries such a calculus, and the chain rule below is built into that theory. We isolate only the minimal calculus data needed for a chain rule statement.*

Theorem 2 (Smooth chain rule under functional calculus). *Assume $\mathbb{E} = \mathbb{R}$ and that $\mathcal{O}_Y$ carries a C^1-functional calculus in the sense of Definition 15. Let $\varphi \in C^1(\mathbb{R})$ and $a \in \mathcal{O}_Y$. Then for any vector field $v \in \Gamma(T_{Y/\mathbb{R}})$,*

$$\nabla_v\big(\varphi_{\mathcal{O}}(a)\big) \;=\; (\varphi')_{\mathcal{O}}(a)\, \nabla_v(a).$$

Proof. The identity is local on Y. In the standard geometric models motivating this paper (smooth manifolds, analytic spaces), $\varphi_{\mathcal{O}}(a)$ is literally the composition $\varphi \circ a$, and the formula is the classical chain rule. Abstractly, one can reduce to the polynomial case by approximating φ by polynomials on compacta and using continuity of derivations, provided the functional calculus is continuous in the appropriate topology. Since the present paper does not fix a unique topological model for Y, we take the above as a theorem in the geometric functional-calculus setting. □

Remark 9 (Multi-variable version). *If $\mathcal{O}_Y$ carries an n-variable calculus $\varphi : \mathbb{R}^n \to \mathbb{R}$ (as in a C^∞-ring), then for $a_1, \dots, a_n \in \mathcal{O}_Y$,*

$$d\big(\varphi(a_1, \dots, a_n)\big) = \sum_{i=1}^{n} \left(\frac{\partial \varphi}{\partial x_i}(a_1, \dots, a_n) \right) da_i,$$

and hence the corresponding directional derivative identity holds after contraction with v.

Remark 10 (Fibrewise/labelwise validity). *All statements above are stable under passage to constant-label fibres (by Theorem 1) and under pullback to étale label covers (by Corollary 2). Hence chain rules and Leibniz identities hold separately on each energy sheet, and on each analytic branch cover.*

5. Higher Jets and Differential Operators

5.1. Higher Infinitesimal Neighbourhoods and Principal Parts

Let Y be an energy-smooth base over $\mathbb{E}$ (in the sense of Definition 9), with structure sheaf $\mathcal{O}_Y$. Let $\Delta_Y : Y \hookrightarrow Y \times_{\mathbb{E}} Y$ be the diagonal, with ideal sheaf $\mathcal{I} \subseteq \mathcal{O}_{Y \times_{\mathbb{E}} Y}$.

Definition 16 (jth infinitesimal neighbourhood). *For $j \geq 0$, the jth infinitesimal neighbourhood of the diagonal is*

$$(Y \times_{\mathbb{E}} Y)^{[j]} := \mathrm{Spec}\big(\mathcal{O}_{Y \times_{\mathbb{E}} Y} / \mathcal{I}^{j+1}\big),$$

 doi:10.20944/preprints202512.2259.v1

with induced projections $p_1^{[j]}, p_2^{[j]} : (Y \times_{\mathbb{E}} Y)^{[j]} \to Y$.

Definition 17 (Principal parts of order j). *Define the* principal parts *of order j to be the sheaf of $\mathcal{O}_Y$-algebras*

$$\mathcal{P}^j_{Y/\mathbb{E}} := (p_1^{[j]})_* \mathcal{O}_{(Y \times_{\mathbb{E}} Y)^{[j]}}.$$

There is a canonical augmentation $\epsilon^{[j]} : \mathcal{P}^j_{Y/\mathbb{E}} \to \mathcal{O}_Y$ induced by restricting along Δ_Y.

Definition 18 (Jets of order j). *Let $\mathcal{F}$ be an $\mathcal{O}_Y$-module. Define the j*th jet module

$$J^j_{Y/\mathbb{E}}(\mathcal{F}) := (p_1^{[j]})_* \big((p_2^{[j]})^* \mathcal{F}\big) \;\cong\; \mathcal{P}^j_{Y/\mathbb{E}} \otimes_{\mathcal{O}_Y} \mathcal{F}.$$

Proposition 9 (Exact sequences and associated graded in the smooth case). *Assume Y is smooth over $\mathbb{E}$ (in the sense that Kähler differentials are locally free and the diagonal is a regular immersion). Then $J^j_{Y/\mathbb{E}}(\mathcal{F})$ admits a natural filtration whose graded pieces satisfy*

$$\operatorname{gr} J^j_{Y/\mathbb{E}}(\mathcal{F}) \;\cong\; \bigoplus_{i=0}^{j} \operatorname{Sym}^i\big(\Omega^1_{Y/\mathbb{E}}\big) \otimes_{\mathcal{O}_Y} \mathcal{F}.$$

Moreover there are short exact sequences

$$0 \to \operatorname{Sym}^j\big(\Omega^1_{Y/\mathbb{E}}\big) \otimes \mathcal{F} \to J^j_{Y/\mathbb{E}}(\mathcal{F}) \to J^{j-1}_{Y/\mathbb{E}}(\mathcal{F}) \to 0.$$

Reference. These are classical properties of principal parts and jets in the smooth case; see ([1] IV) or ([2] II, §8). □

5.2. Differential Operators and the Jet Adjunction

We recall the standard intrinsic definition of differential operators.

Definition 19 (Differential operators of order $\le j$). *Let $\mathcal{F}, \mathcal{G}$ be $\mathcal{O}_Y$-modules. Define inductively:*

- $\operatorname{Diff}^{\le 0}_{Y/\mathbb{E}}(\mathcal{F}, \mathcal{G}) := \underline{\operatorname{Hom}}_{\mathcal{O}_Y}(\mathcal{F}, \mathcal{G})$;
- *for $j \ge 1$, an $\mathbb{E}$-linear map $D : \mathcal{F} \to \mathcal{G}$ is of order $\le j$ if for all local sections $a \in \mathcal{O}_Y$, the commutator*

$$[D, a] : \mathcal{F} \to \mathcal{G}, \qquad [D, a](s) := D(as) - aD(s),$$

is a differential operator of order $\le j - 1$.

Write $\operatorname{Diff}^{\le j}_{Y/\mathbb{E}}(\mathcal{F}, \mathcal{G})$ for the sheaf of such operators.

Theorem 3 (Jet–differential operator adjunction). *For all $\mathcal{O}_Y$-modules $\mathcal{F}, \mathcal{G}$ and all $j \ge 0$ there is a natural isomorphism*

$$\underline{\operatorname{Hom}}_{\mathcal{O}_Y}\big(J^j_{Y/\mathbb{E}}(\mathcal{F}),\, \mathcal{G}\big) \;\cong\; \operatorname{Diff}^{\le j}_{Y/\mathbb{E}}(\mathcal{F}, \mathcal{G}),$$

functorial in $\mathcal{F}$ and $\mathcal{G}$.

Reference. This is standard; see ([1] IV). Concretely, $J^j(\mathcal{F})$ is the universal recipient for order $\le j$ Taylor expansions, and $\mathcal{P}^j$ is the universal algebra of order-j principal parts. □

5.3. Labelwise Behaviour of Higher Jets

Proposition 10 (Constant-label case: componentwise jets). *In the constant-label model $\mathcal{E} = \underline{\mathbb{E}}_{\mathscr{X}}$, formation of J^j is componentwise: for any $\mathcal{O}_{\underline{\mathbb{E}}_{\mathscr{X}}}$-module $\mathcal{F}$,*

$$\big(J^j_{\underline{\mathbb{E}}_{\mathscr{X}}/\mathbb{E}}(\mathcal{F})\big)_E \;\cong\; J^j_{\mathscr{X}/\mathbb{E}}(\mathcal{F}_E)$$

 doi:10.20944/preprints202512.2259.v1

naturally in $E \in \mathbb{E}$.

Proof. Same argument as Proposition 4, using that $\underline{\mathbb{E}}_{\mathscr{X}}$ is a disjoint union of copies of $\mathscr{X}$ and jets restrict to components. □

Proposition 11 (Étale-label case: jets commute with pullback)**.** *Let $p : \widetilde{\mathscr{X}} \to \mathscr{X}$ be an étale label cover. Then for any $\mathcal{O}_{\mathscr{X}}$-module $\mathcal{F}$ on $\mathscr{X}$,*

$$p^* J^j_{\mathscr{X}/\mathbb{E}}(\mathcal{F}) \cong J^j_{\widetilde{\mathscr{X}}/\mathbb{E}}(p^*\mathcal{F}),$$

and the identification is compatible with the jet adjunction (Theorem 3).

Proof. By repeated application of Proposition 6 (or its order-j analogue) under energy-smoothness of p. □

6. Infinite Jets, the Jet Comonad, and the coKleisli Calculus

6.1. Infinite Jets

Finite jets control differential operators of bounded order. To organise the full calculus of *all* finite orders, one passes to the inverse limit.

Definition 20 (Infinite principal parts and infinite jets)**.** *Assume the inverse limits exist in the chosen category of modules (e.g. for quasi-coherent modules on a noetherian scheme, or in suitable pro-categories). Define*

$$\mathcal{P}^\infty_{Y/\mathbb{E}} := \varprojlim_j \mathcal{P}^j_{Y/\mathbb{E}}, \qquad J^\infty_{Y/\mathbb{E}}(\mathcal{F}) := \varprojlim_j J^j_{Y/\mathbb{E}}(\mathcal{F}).$$

We write $J^\infty(\mathcal{F})$ when $Y/\mathbb{E}$ is understood.

Remark 11 (Pro-object caveat)**.** *If inverse limits do not exist strictly in $\mathcal{O}_Y$-modules, the correct replacement is the pro-object $\{J^j(\mathcal{F})\}_{j\geq 0}$. All statements below may be interpreted in that pro-setting. To keep notation light, we state results as if the limit exists.*

6.2. Coalgebra and Comonad Structure

A key structural fact is that infinite principal parts carry a canonical (coassociative, counital) coalgebra structure, encoding "composition of infinitesimal displacements."

Proposition 12 (Coalgebra structure on $\mathcal{P}^\infty$)**.** *There are canonical maps of $\mathcal{O}_Y$-modules*

$$\epsilon : \mathcal{P}^\infty_{Y/\mathbb{E}} \to \mathcal{O}_Y, \qquad \Delta : \mathcal{P}^\infty_{Y/\mathbb{E}} \to \mathcal{P}^\infty_{Y/\mathbb{E}} \otimes_{\mathcal{O}_Y} \mathcal{P}^\infty_{Y/\mathbb{E}},$$

such that $(\mathcal{P}^\infty_{Y/\mathbb{E}}, \Delta, \epsilon)$ is a counital, coassociative coalgebra object in $\mathcal{O}_Y$-modules.

Construction sketch. The counit ϵ is induced by restriction along the diagonal. The comultiplication Δ is induced by the inclusion of the formal groupoid of infinitesimal neighbourhoods: informally, the (∞)-neighbourhood of the diagonal in $Y \times Y$ composes via the middle factor in $Y \times Y \times Y$. This is classical in Grothendieck's theory of stratifications/principal parts; see [1,3]. □

Theorem 4 (Jet comonad)**.** *Define the endofunctor $J^\infty : \mathcal{O}_Y\text{-Mod} \to \mathcal{O}_Y\text{-Mod}$ by $J^\infty(\mathcal{F}) = \mathcal{P}^\infty \otimes_{\mathcal{O}_Y} \mathcal{F}$. Then with counit*

$$\varepsilon_{\mathcal{F}} : J^\infty(\mathcal{F}) \to \mathcal{F}$$

induced from $\epsilon : \mathcal{P}^\infty \to \mathcal{O}_Y$, and comultiplication

$$\delta_{\mathcal{F}} : J^\infty(\mathcal{F}) \to J^\infty(J^\infty(\mathcal{F}))$$

 doi:10.20944/preprints202512.2259.v1

induced from Δ, the triple $(J^\infty, \varepsilon, \delta)$ is a comonad on $\mathcal{O}_Y$-modules.

Remark 12 (Relation to connections and stratifications)**.** *Coalgebras over the jet comonad (i.e. J^∞-comodules) are precisely* stratified modules*. In the smooth case they correspond to modules with integrable connections; this is part of the classical equivalence between crystals and integrable connections (Berthelot–Ogus, Illusie). We will not use this equivalence in proofs, but it motivates why J^∞ governs differential operator calculus.*

6.3. CoKleisli Category and Differential Operators

Definition 21 (CoKleisli morphisms)**.** *The* coKleisli category *of the comonad J^∞ has the same objects as $\mathcal{O}_Y$-modules, and morphisms*

$$\mathrm{Hom}_{\mathrm{coKl}(J^\infty)}(\mathcal{F}, \mathcal{G}) := \mathrm{Hom}_{\mathcal{O}_Y}\big(J^\infty(\mathcal{F}), \mathcal{G}\big).$$

Composition is defined by the comonad structure: for $f : J^\infty(\mathcal{F}) \to \mathcal{G}$ and $g : J^\infty(\mathcal{G}) \to \mathcal{H}$,

$$g \star f := g \circ J^\infty(f) \circ \delta_{\mathcal{F}}.$$

Theorem 5 (CoKleisli morphisms are differential operators)**.** *Assume Y is energy-smooth over $\mathbb{E}$. Then for $\mathcal{O}_Y$-modules $\mathcal{F}, \mathcal{G}$ there is a natural identification*

$$\underline{\mathrm{Hom}}_{\mathcal{O}_Y}\big(J^\infty(\mathcal{F}), \mathcal{G}\big) \;\cong\; \bigcup_{j \geq 0} \mathrm{Diff}^{\leq j}_{Y/\mathbb{E}}(\mathcal{F}, \mathcal{G}),$$

compatible with composition: coKleisli composition corresponds to composition of differential operators.

Proof sketch. By Theorem 3, morphisms out of $J^j(\mathcal{F})$ correspond to order $\leq j$ differential operators. Passing to the inverse limit over j, morphisms out of $J^\infty(\mathcal{F})$ correspond to compatible families, i.e. operators of some finite order. Compatibility of composition follows from the coalgebra/comonad map Δ encoding composition of infinitesimal expansions. □

6.4. Labelwise Jet Comonad

All constructions in this section apply verbatim to the labelled bases $Y = \underline{\mathbb{E}}_{\mathscr{X}}$ and $Y = \widetilde{\mathscr{X}}$ (an étale label cover). In the constant-label case the comonad decomposes componentwise over $E \in \mathbb{E}$; in the étale-label case it is the usual jet comonad on $\widetilde{\mathscr{X}}$ and is automatically monodromy-aware.

7. Real de Rham Theory and Monodromy-Consistent Calculus

7.1. De Rham Complex on a Labelled Base

Let Y denote an energy-smooth base (again: $\mathscr{X}$, $\underline{\mathbb{E}}_{\mathscr{X}}$, or an étale label cover $\widetilde{\mathscr{X}}$). Define

$$\Omega^p_{Y/\mathbb{E}} := \bigwedge^p_{\mathcal{O}_Y} \Omega^1_{Y/\mathbb{E}}, \qquad p \geq 0.$$

The universal derivation induces the de Rham differential

$$d : \Omega^p_{Y/\mathbb{E}} \to \Omega^{p+1}_{Y/\mathbb{E}}, \qquad d^2 = 0.$$

Remark 13 (Fibrewise validity)**.** *In the constant-label model, $\Omega^\bullet_{\underline{\mathbb{E}}_{\mathscr{X}}/\mathbb{E}}$ decomposes into the product of $\Omega^\bullet_{\mathscr{X}/\mathbb{E}}$ over energies. On an étale label cover, $\Omega^\bullet_{\widetilde{\mathscr{X}}/\mathbb{E}}$ is the usual de Rham complex on the cover, encoding branch data intrinsically.*

 doi:10.20944/preprints202512.2259.v1

7.2. Integration Along Labelled Paths (Geometric Models)

To speak about integration, we assume a geometric model in which Y carries a notion of smooth curves and integration of 1-forms (e.g. Y is a smooth manifold or complex analytic curve space). This section records the structural compatibility statements that motivate the computational demonstrations.

Definition 22 (Pathwise integral). *Let $\gamma : [0,1] \to Y$ be a C^1 path and $\omega \in \Gamma(Y, \Omega^1_{Y/\mathbb{E}})$. Define*

$$\int_\gamma \omega := \int_0^1 \omega_{\gamma(t)}(\dot\gamma(t))\, dt,$$

whenever the right-hand side is defined in the ambient model.

Theorem 6 (Fundamental theorem on a labelled base). *Let $f \in \Gamma(Y, \mathcal{O}_Y)$ and let $\gamma : [0,1] \to Y$ be a C^1 path. Then*

$$\int_\gamma df = f(\gamma(1)) - f(\gamma(0)).$$

Proof. This is the classical fundamental theorem of calculus in the geometric model. The point of the labelled framework is that this identity takes place on the labelled base Y, so endpoints lie on the same sheet. □

7.3. Monodromy-Consistent Calculus on an Étale Label Cover

Let $p : \widetilde{\mathscr{X}} \to \mathscr{X}$ be an étale label cover capturing analytic branches.

Definition 23 (Lifted path). *Let $\gamma : [0,1] \to \mathscr{X}$ be a path and let $\tilde{x}_0 \in \widetilde{\mathscr{X}}$ satisfy $p(\tilde{x}_0) = \gamma(0)$. A* lift *of γ starting at $\tilde{x}_0$ is a path $\tilde\gamma : [0,1] \to \widetilde{\mathscr{X}}$ such that $p \circ \tilde\gamma = \gamma$ and $\tilde\gamma(0) = \tilde{x}_0$.*

Theorem 7 (Monodromy-consistent fundamental theorem). *Let $f \in \Gamma(\widetilde{\mathscr{X}}, \mathcal{O}_{\widetilde{\mathscr{X}}})$ be a single-valued branch function on the cover, and let $\gamma : [0,1] \to \mathscr{X}$ be a path with a fixed lift $\tilde\gamma : [0,1] \to \widetilde{\mathscr{X}}$. Then*

$$\int_{\tilde\gamma} df = f(\tilde\gamma(1)) - f(\tilde\gamma(0)).$$

If γ is a loop based at x_0 and $\tilde\gamma(0) = \tilde{x}_0$, then $\tilde\gamma(1) = g \cdot \tilde{x}_0$ for a unique deck transformation g; thus analytic continuation of f along γ is given by $f(g \cdot \tilde{x}_0)$.

Remark 14 (Connection to the code demonstrations). *The scripts in Appendix A.2 implement this theorem in the basic case $f(z) = \sqrt{z}$ along loops around 0, contrasting lifted-path integration with principal-branch evaluation, which exhibits discontinuities at branch cuts. Appendix A.3 performs the analogous construction for complete elliptic integrals via a Gauss–Manin ODE system, numerically capturing logarithmic monodromy around $m = 1$.*

8. Energy Laplacian and Branchwise PDE

8.1. Gradient, Divergence, Laplacian on a Labelled Base

Assume Y carries a Riemannian metric (or, in algebraic settings, a suitable metric structure) on $T_{Y/\mathbb{E}}$ inducing an isomorphism $\sharp : \Omega^1_{Y/\mathbb{E}} \to T_{Y/\mathbb{E}}$. Define for $f \in \mathcal{O}_Y$:

$$\nabla f := \sharp(df), \qquad \operatorname{div}(V) := \operatorname{tr}(\nabla V), \qquad \Delta_{\mathbb{E}} f := \operatorname{div}(\nabla f),$$

where ∇ on vector fields is the Levi–Civita connection when available.

 doi:10.20944/preprints202512.2259.v1

Proposition 13 (Labelwise Laplacian is fibrewise). *In the constant-label model $Y = \underline{\mathbb{E}}_{\mathscr{X}}$, $\Delta_{\mathbb{E}}$ acts componentwise on energy fibres. In the étale-label model $Y = \widetilde{\mathscr{X}}$, $\Delta_{\mathbb{E}}$ acts on each branch on the cover and is compatible with deck transformations.*

8.2. Heat Equation Without Branch Mixing

Given $u(t,-) \in \Gamma(Y, \mathcal{O}_Y)$, consider the heat equation

$$\partial_t u = \Delta_{\mathbb{E}} u.$$

In the constant-label model this is a family of independent heat equations indexed by $E \in \mathbb{E}$. On an étale branch cover it is a heat equation on the cover, hence branch-consistent by construction.

9. Controlled Energy/Label Mixing via Correspondences

The labelled slice framework enforces no mixing. To model *controlled* mixing between labels (e.g. transitions between sheets), we enrich the theory by allowing operators defined along *correspondences* of label objects.

9.1. Label Correspondences

Definition 24 (Label correspondence). *Let $\mathcal{E}$ be a label object over $\mathscr{X}$ (constant or étale). A* label correspondence *is a span in* $\mathrm{Sh}(\mathcal{B})/\mathscr{X}$*:*

$$\mathcal{E} \xleftarrow{s} \mathcal{R} \xrightarrow{t} \mathcal{E}.$$

Intuitively, $\mathcal{R}$ encodes allowed transitions from a source label (via s) to a target label (via t).

Definition 25 (Transfer functor induced by a correspondence). *Assume s and t are representable and that pushforward on modules along t is defined in the chosen model. Define the* transfer *(mixing) endofunctor on $\mathcal{O}_{\mathcal{E}}$-modules by*

$$\mathsf{T}_{\mathcal{R}}(\mathcal{F}) := t_*(s^*\mathcal{F}).$$

Remark 15 (Energy-preserving operators as the diagonal correspondence). *If $\mathcal{R} = \mathcal{E}$ and $s = t = \mathrm{id}_{\mathcal{E}}$ (the diagonal span), then $\mathsf{T}_{\mathcal{R}} \cong \mathrm{id}$: no mixing. More generally, a mixing pattern is determined by the fibres of s and t.*

9.2. Constant-Label Case: Correspondences Are Relations on Energies

When $\mathcal{E} = \underline{\mathbb{E}}_{\mathscr{X}} \cong \coprod_{E \in \mathbb{E}} 1_{\mathscr{X}}$, a correspondence $\mathcal{R} \to \underline{\mathbb{E}}_{\mathscr{X}} \times_{\mathscr{X}} \underline{\mathbb{E}}_{\mathscr{X}}$ amounts to a relation on the energy index set (possibly varying over $\mathscr{X}$). In this case, $\mathsf{T}_{\mathcal{R}}$ acts like a matrix of functors/operators

$$(\mathsf{T}_{\mathcal{R}}(\mathcal{F}))_{E'} \simeq \bigoplus_{E:\ (E \to E') \in \mathcal{R}} \mathcal{F}_E,$$

with the precise sum/product depending on whether t_* or $t_!$ is used.

Example 1 (Coupled branch heat system). *Let energies be discrete indices $E \in \{1, \dots, N\}$ and let $\kappa_{E',E} \geq 0$ be coupling constants. A controlled-mixing heat system has the form*

$$\partial_t u_{E'} = \Delta u_{E'} + \sum_E \kappa_{E',E}\, u_E.$$

The Laplacian term is no-mixing; the coupling term is encoded by a correspondence $\mathcal{R}$ allowing transitions $E \to E'$ with weights $\kappa_{E',E}$.

 doi:10.20944/preprints202512.2259.v1

9.3. Composition of Correspondences (Optional Categorical Structure)

Under standard hypotheses (suitable base change and projection formula), spans compose by pullback:

$$(\mathcal{E} \xleftarrow{s} \mathcal{R} \xrightarrow{t} \mathcal{E}) \circ (\mathcal{E} \xleftarrow{s'} \mathcal{R}' \xrightarrow{t'} \mathcal{E}) := (\mathcal{E} \xleftarrow{s\circ\pi_1} \mathcal{R} \times_{\mathcal{E}} \mathcal{R}' \xrightarrow{t'\circ\pi_2} \mathcal{E}),$$

and the induced transfer functors compose accordingly. This yields a bicategory of label correspondences in which no-mixing calculus is the diagonal subtheory.

Appendix A. Computational Demonstrations and Reproducibility

Appendix A.1. Reproducibility Notes

The scripts below are included verbatim for transparency and reproducibility. They are not required for the theoretical development, but they implement:

- monodromy-consistent differentiation along lifted paths (vs. principal-branch failure);
- numerical branchwise heat flow with no label mixing;
- a basic descent/gluing check enforcing label compatibility;
- (elliptic case) numerical Gauss–Manin transport for $K(m)$ and $E(m)$ and detection of logarithmic monodromy around $m = 1$.

Dependencies. The first script uses `numpy` and `matplotlib`; it optionally uses `sympy`. The second script uses `numpy`, `matplotlib`, and `mpmath`.
Outputs. The scripts save figures to PNG files. The appendix figures below are representative outputs (filenames as provided).

Appendix A.2. Listing: `real_diff_demo.py`

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Real Differentiation demos:
1) Monodromy-consistent differentiation for sqrt(z) around 0:
   - Energy/stack-aware (analytic continuation) vs principal branch.
   - Shows Fundamental Theorem holds on the lifted path and fails in naive
    principal-branch treatment.
2) Branchwise heat equation solver (no branch mixing).
3) Robust gluing on overlaps with tag checks (descent).
4) Directional derivative and analytic "jets" (Taylor series) examples.

Figures are saved and also displayed when possible.
"""

from dataclasses import dataclass
from typing import Dict, Optional, Tuple
import numpy as np
import os
import matplotlib.pyplot as plt

# Optional: symbolic computations for series and analytic derivatives
try:
    import sympy as sp
    SYMPY_AVAILABLE = True
except Exception:
    SYMPY_AVAILABLE = False

# Detect if running inside IPython (e.g., Jupyter)
```

 doi:10.20944/preprints202512.2259.v1


```python
try:
    get_ipython  # noqa
    IN_IPYTHON = True
except NameError:
    IN_IPYTHON = False

# Detect headless environment
HEADLESS = not (
    ("DISPLAY" in os.environ)
    or ("WAYLAND_DISPLAY" in os.environ)
    or ("MPLBACKEND" in os.environ)
)

SHOW_PLOTS = True

def finalize_figure(fig, savepath: Optional[str], show: bool = True):
    """Save the figure (if savepath provided) and display it if possible."""
    if savepath:
        fig.savefig(savepath, dpi=150, bbox_inches="tight")
        print(f"[saved] {savepath}")
    if show and not HEADLESS:
        if IN_IPYTHON:
            try:
                from IPython.display import display, Image
                if savepath:
                    display(Image(filename=savepath))
                else:
                    display(fig)
            except Exception:
                plt.show()
        else:
            plt.show()
    elif show and HEADLESS and IN_IPYTHON and savepath:
        try:
            from IPython.display import display, Image
            display(Image(filename=savepath))
        except Exception:
            pass
    elif show and HEADLESS and not IN_IPYTHON:
        if savepath:
            print(f"[headless] Figure saved to {savepath}.")
        else:
            print("[headless] No savepath provided; cannot display.")
    plt.close(fig)


# ----------------------------
# Energy tags and fields
# ----------------------------

@dataclass(frozen=True)
class EnergyTag:
    """Simple energy tag.
    Model A: fixed label for the whole object.
    Model B: interpret 'value' as the fibre energy value (sheet index, etc.).
```

 doi:10.20944/preprints202512.2259.v1


```
    """
    name: str
    value: Optional[float] = None

    def __str__(self):
        if self.value is None:
            return f"E[{self.name}]"
        return f"E[{self.name}={self.value}]"


@dataclass
class EnergyField1D:
    """Energy-tagged scalar field on a 1D spatial grid."""
    x: np.ndarray
    u: np.ndarray
    tag: EnergyTag

    def __post_init__(self):
        self.x = np.asarray(self.x, dtype=float)
        self.u = np.asarray(self.u, dtype=float)
        if self.x.shape != self.u.shape:
            raise ValueError("x and u must have the same shape")
        if self.x.ndim != 1:
            raise ValueError("x and u must be 1D arrays")

    def copy(self) -> "EnergyField1D":
        return EnergyField1D(self.x.copy(), self.u.copy(), self.tag)

    def restrict(self, a: float, b: float) -> "EnergyField1D":
        """Restrict to subinterval [a, b]."""
        mask = (self.x >= a) & (self.x <= b)
        return EnergyField1D(self.x[mask], self.u[mask], self.tag)

    def check_compat(self, other: "EnergyField1D") -> None:
        if self.tag != other.tag:
            raise ValueError(f"Energy tag mismatch: {self.tag} vs {other.tag}")
        if self.x.shape != other.x.shape or not np.allclose(self.x, other.x):
            raise ValueError("Spatial grid mismatch")

    def __add__(self, other: "EnergyField1D") -> "EnergyField1D":
        self.check_compat(other)
        return EnergyField1D(self.x, self.u + other.u, self.tag)

    def laplacian_dirichlet(self) -> np.ndarray:
        """Discrete Laplacian with Dirichlet boundary conditions."""
        u = self.u
        x = self.x
        dx = x[1] - x[0]
        lap = np.zeros_like(u)
        lap[1:-1] = (u[2:] - 2*u[1:-1] + u[:-2]) / (dx*dx)
        return lap

    def plot(self, ax=None, label: Optional[str] = None, **kwargs):
        if ax is None:
            fig, ax = plt.subplots()
```

 doi:10.20944/preprints202512.2259.v1


```
        ax.plot(self.x, self.u, label=label if label else str(self.tag), **kwargs)
        ax.set_xlabel("x")
        ax.set_ylabel("u(x)")
        ax.grid(True)
        return ax


# ---------------------------
# Utility: cumulative trapezoidal integral for complex arrays
# ---------------------------

def cumtrapz_complex(y: np.ndarray, x: np.ndarray) -> np.ndarray:
    """Cumulative trapezoidal integral for complex y over x."""
    I = np.zeros_like(y, dtype=complex)
    dt = np.diff(x)
    I[1:] = np.cumsum(0.5 * (y[:-1] + y[1:]) * dt)
    return I


# ---------------------------
# 1) Monodromy-consistent differentiation demo for sqrt(z)
# ---------------------------

def sqrt_monodromy_demo(R: float = 1.0,
                        turns: int = 1,
                        N: int = 4000,
                        prefix: str = "sqrt_monodromy",
                        show: bool = SHOW_PLOTS):
    """
    Compare:
      - Analytic continuation (energy/stack-aware) sqrt along a loop z(t)=R e^{i t}.
      - Naive principal-branch sqrt.
    Fundamental Theorem holds for the lifted path; principal-branch shows a defect at the cut.
    """
    t = np.linspace(0.0, 2*np.pi*turns, N)
    z = R * np.exp(1j * t)
    z_t = 1j * z  # dz/dt

    # Energy/stack-aware analytic continuation: f_ac(t) = sqrt(R) * exp(i t/2)
    sqrtR = np.sqrt(R)
    f_ac = sqrtR * np.exp(0.5j * t)

    # Principal branch (discontinuous at negative real axis):
    f_pr = np.sqrt(z)

    # df/dt = f'(z) * z'(t), with f'(z) = 1/(2 sqrt(z)); use matching branch for each
    dfdt_ac = 0.5 * z_t / f_ac
    dfdt_pr = 0.5 * z_t / f_pr

    # Cumulative integrals
    I_ac = cumtrapz_complex(dfdt_ac, t)
```

 doi:10.20944/preprints202512.2259.v1


```
    I_pr = cumtrapz_complex(dfdt_pr, t)

    # FTC mismatch
    M_ac = I_ac - (f_ac - f_ac[0])
    M_pr = I_pr - (f_pr - f_pr[0])

    print(f"\n[Monodromy demo] R={R}, turns={turns}")
    print(f"AC: f(end)-f(start) = {f_ac[-1]-f_ac[0]:.6f}, I_ac(end) =
{I_ac[-1]:.6f}")
    print(f"PR: f(end)-f(start) = {f_pr[-1]-f_pr[0]:.6f}, I_pr(end) =
{I_pr[-1]:.6f}")
    print(f"FT defect (AC) at end: {M_ac[-1]:.3e}")
    print(f"FT defect (PR) at end: {M_pr[-1]:.3e}")

    # Plot 1
    fig, axes = plt.subplots(2, 2, figsize=(11, 7))
    ax = axes[0,0]
    ax.plot(t, np.real(f_ac), 'C0', label='Re f_ac (lifted)')
    ax.plot(t, np.imag(f_ac), 'C1', label='Im f_ac (lifted)')
    ax.set_title("Analytic continuation sqrt along loop")
    ax.set_xlabel("t"); ax.grid(True); ax.legend(loc='best')

    ax = axes[0,1]
    ax.plot(t, np.real(f_pr), 'C0', label='Re f_pr (principal)')
    ax.plot(t, np.imag(f_pr), 'C1', label='Im f_pr (principal)')
    ax.set_title("Principal-branch sqrt along loop (jumps)")
    ax.set_xlabel("t"); ax.grid(True); ax.legend(loc='best')

    ax = axes[1,0]
    ax.plot(t, np.abs(M_ac), 'C2', label='|FTC defect| (lifted)')
    ax.set_yscale('log')
    ax.set_title("FTC mismatch (lifted) ~ 0")
    ax.set_xlabel("t"); ax.grid(True, which='both'); ax.legend()

    ax = axes[1,1]
    ax.plot(t, np.abs(M_pr), 'C3', label='|FTC defect| (principal)')
    ax.set_yscale('log')
    ax.set_title("FTC mismatch (principal) spikes at branch cut")
    ax.set_xlabel("t"); ax.grid(True, which='both'); ax.legend()

    finalize_figure(fig, f"{prefix}_turns{turns}.png", show)

    # Plot 2: trajectories in f-plane
    fig2, ax2 = plt.subplots(1, 2, figsize=(10, 4.5))
    ax2[0].plot(np.real(f_ac), np.imag(f_ac), 'C0')
    ax2[0].scatter(np.real(f_ac[0]), np.imag(f_ac[0]), c='k', s=20,
label='start')
    ax2[0].scatter(np.real(f_ac[-1]), np.imag(f_ac[-1]), c='r', s=20,
label='end')
    ax2[0].set_title("Lifted path in f-plane")
    ax2[0].set_xlabel("Re f"); ax2[0].set_ylabel("Im f")
    ax2[0].axis('equal'); ax2[0].grid(True); ax2[0].legend()

    ax2[1].plot(np.real(f_pr), np.imag(f_pr), 'C1')
```

 doi:10.20944/preprints202512.2259.v1


```
    ax2[1].scatter(np.real(f_pr[0]), np.imag(f_pr[0]), c='k', s=20,
    label='start')
    ax2[1].scatter(np.real(f_pr[-1]), np.imag(f_pr[-1]), c='r', s=20,
    label='end')
    ax2[1].set_title("Principal-branch path in f-plane (jumps)")
    ax2[1].set_xlabel("Re f"); ax2[1].set_ylabel("Im f")
    ax2[1].axis('equal'); ax2[1].grid(True); ax2[1].legend()

    finalize_figure(fig2, f"{prefix}_plane_turns{turns}.png", show)


# ---------------------------
# 2) Directional derivative example s(v) = sqrt(1 - v^2/c^2)
# ---------------------------

def directional_derivative_example(c: float = 3.0, vmin: float = -2.5, vmax:
    float = 2.5, n: int = 400,
                                   savepath: str = "derivative_example.png",
    show: bool = SHOW_PLOTS):
    v = np.linspace(vmin, vmax, n)
    mask = np.abs(v) < c
    v_real = v[mask]
    s_val = np.sqrt(1.0 - (v_real**2)/(c**2))
    dsdv_numeric = np.gradient(s_val, v_real)

    if SYMPY_AVAILABLE:
        vs, cs = sp.symbols('v c', positive=True, real=True)
        s_expr = sp.sqrt(1 - vs**2/cs**2)
        ds_dv_sym = sp.diff(s_expr, vs)
        f_ds = sp.lambdify((vs, cs), ds_dv_sym, 'numpy')
        dsdv_exact = f_ds(v_real, c)
    else:
        dsdv_exact = None

    fig, ax = plt.subplots(1, 2, figsize=(10, 4.0))
    ax[0].plot(v_real, s_val, color="C0", label="s(v)")
    ax[0].set_title("s(v) = sqrt(1 - v^2/c^2)")
    ax[0].grid(True); ax[0].legend()
    ax[0].set_xlabel("v")

    ax[1].plot(v_real, dsdv_numeric, color="C1", label="numeric d/dv s")
    if dsdv_exact is not None:
        ax[1].plot(v_real, dsdv_exact, "k--", lw=1.5, label="exact d/dv s")
    ax[1].set_title("Directional derivative (along d/dv)")
    ax[1].grid(True); ax[1].legend()
    ax[1].set_xlabel("v")

    fig.suptitle("Directional derivative example (energy tag preserved)")
    plt.tight_layout()
    finalize_figure(fig, savepath, show)


# ---------------------------
# 3) Jets (truncated Taylor expansions) of s(v) [analytic illustration]
# ---------------------------
```

 doi:10.20944/preprints202512.2259.v1


```python

def jets_example(c: float = 3.0, v0: float = 0.0, order: int = 4,
                 vspan: float = 1.5, n: int = 400,
                 savepath: str = "jets_example.png", show: bool = SHOW_PLOTS):
    if not SYMPY_AVAILABLE:
        print("Sympy not available; skipping jets example.")
        return

    v = np.linspace(v0 - vspan, v0 + vspan, n)
    v = v[(np.abs(v) < c)]
    s_true = np.sqrt(1.0 - (v**2)/(c**2))

    vs, cs = sp.symbols('v c', real=True, positive=True)
    s_expr = sp.sqrt(1 - (vs**2)/(cs**2))
    s_series = sp.series(s_expr, vs, v0, order+1).removeO()
    s_poly = sp.lambdify((vs, cs), s_series, 'numpy')
    s_approx = s_poly(v, c)

    fig, ax = plt.subplots(figsize=(6, 4))
    ax.plot(v, s_true, label="s(v) exact", color="C0")
    ax.plot(v, s_approx, "--", label=f"Jet order {order}", color="C1")
    ax.set_title(f"Analytic jet (Taylor) around v0={v0}")
    ax.set_xlabel("v"); ax.set_ylabel("s(v)")
    ax.grid(True); ax.legend()
    plt.tight_layout()
    finalize_figure(fig, savepath, show)


# ----------------------------
# 4) Heat equation solver (branchwise, 1D)
# ----------------------------

def heat_equation_explicit_step(u: np.ndarray, alpha: float, dx: float, dt:
    float) -> np.ndarray:
    """One explicit step for 1D heat equation with Dirichlet BC (u=0 at
    endpoints)."""
    un = u.copy()
    un[0] = 0.0
    un[-1] = 0.0
    r = alpha * dt / (dx*dx)
    un[1:-1] = u[1:-1] + r * (u[2:] - 2*u[1:-1] + u[:-2])
    return un


def solve_heat_branchwise(fields: Dict[EnergyTag, EnergyField1D],
                          alpha: Dict[EnergyTag, float],
                          T: float = 0.5,
                          dt: Optional[float] = None,
                          snapshots: int = 8) -> Tuple[np.ndarray,
    Dict[EnergyTag, np.ndarray]]:
    tags = list(fields.keys())
    Nx = None
    dx = None
    for tag in tags:
        f = fields[tag]
```

 doi:10.20944/preprints202512.2259.v1


```
        if Nx is None:
            Nx = len(f.x)
            dx = f.x[1] - f.x[0]
        else:
            if len(f.x) != Nx or not np.allclose(f.x, fields[tags[0]].x):
                raise ValueError("All fields must share a common grid for visualization.")
    alpha_max = max(alpha[tag] for tag in tags)
    if dt is None:
        dt = 0.45 * dx * dx / alpha_max
    nt = int(np.ceil(T / dt))
    dt = T / nt

    save_k = max(1, nt // max(1, snapshots - 1))
    times = []
    Uhist = {tag: [] for tag in tags}

    for k in range(nt + 1):
        t = k * dt
        if k % save_k == 0 or k == nt:
            times.append(t)
            for tag in tags:
                Uhist[tag].append(fields[tag].u.copy())
        if k < nt:
            for tag in tags:
                f = fields[tag]
                a = alpha[tag]
                fields[tag].u = heat_equation_explicit_step(f.u, a, dx, dt)

    times = np.array(times)
    for tag in tags:
        Uhist[tag] = np.array(Uhist[tag])
    return times, Uhist


def make_heat_plots(x: np.ndarray,
                    times: np.ndarray,
                    Uhist: Dict[EnergyTag, np.ndarray],
                    cmap: str = "inferno",
                    prefix: str = "heat",
                    show: bool = SHOW_PLOTS):
    for tag, U in Uhist.items():
        fig, ax = plt.subplots(figsize=(6, 4))
        im = ax.imshow(U, aspect='auto', origin='lower', cmap=cmap,
                       extent=[x[0], x[-1], times[0], times[-1]])
        ax.set_title(f"Heat evolution (Dirichlet) - {tag}")
        ax.set_xlabel("x"); ax.set_ylabel("time")
        plt.colorbar(im, ax=ax, label="u(x,t)")
        plt.tight_layout()
        fname = f"{prefix}_map_{tag.name}.png"
        finalize_figure(fig, fname, show)

    idxs = np.linspace(0, len(times) - 1, 4, dtype=int)
    fig, ax = plt.subplots(figsize=(7, 4))
    colors = ["C0", "C1", "C2", "C3"]
```

 doi:10.20944/preprints202512.2259.v1


```
    for j, idx in enumerate(idxs):
        for i, tag in enumerate(Uhist):
            lab = f"t={times[idx]:.3f}" if i == 0 else None
            ax.plot(x, Uhist[tag][idx], color=colors[j % len(colors)],
                    alpha=0.8, lw=1.8, label=lab)
    ax.set_title("Branchwise heat evolution (no mixing)")
    ax.set_xlabel("x"); ax.set_ylabel("u(x,t)")
    ax.grid(True)
    ax.legend(title="Snapshots")
    plt.tight_layout()
    finalize_figure(fig, "heat_overlays.png", show)


# ----------------------------
# 5) Gluing demonstration (descent) - robust version
# ----------------------------

def glue_fields_1d(
    f1: EnergyField1D,
    f2: EnergyField1D,
    atol: float = 1e-4,
    rtol: float = 1e-3,
    ncheck: int = 400
) -> EnergyField1D:
    if f1.tag != f2.tag:
        raise ValueError(f"Energy tag mismatch in glue: {f1.tag} vs {f2.tag}")

    a1, b1 = f1.x[0], f1.x[-1]
    a2, b2 = f2.x[0], f2.x[-1]
    a = max(a1, a2)
    b = min(b1, b2)

    if a < b:
        x_ov = np.linspace(a, b, ncheck)
        u1_ov = np.interp(x_ov, f1.x, f1.u)
        u2_ov = np.interp(x_ov, f2.x, f2.u)
        if not np.allclose(u1_ov, u2_ov, atol=atol, rtol=rtol):
            diff = np.max(np.abs(u1_ov - u2_ov))
            raise ValueError(
                f"Values do not agree on the overlap; cannot glue. "
                f"max|diff|={diff:.3e}, atol={atol:.1e}, rtol={rtol:.1e}"
            )

    x_union = np.unique(np.concatenate([f1.x, f2.x]))
    u_union = np.zeros_like(x_union)

    has1 = (x_union >= a1) & (x_union <= b1)
    has2 = (x_union >= a2) & (x_union <= b2)
    u1_interp = np.zeros_like(x_union)
    u2_interp = np.zeros_like(x_union)
    u1_interp[has1] = np.interp(x_union[has1], f1.x, f1.u)
    u2_interp[has2] = np.interp(x_union[has2], f2.x, f2.u)

    both = has1 & has2
    only1 = has1 & (~has2)
```

 doi:10.20944/preprints202512.2259.v1


```
    only2 = has2 & (~has1)

    u_union[only1] = u1_interp[only1]
    u_union[only2] = u2_interp[only2]
    u_union[both] = 0.5 * (u1_interp[both] + u2_interp[both])

    return EnergyField1D(x_union, u_union, f1.tag)


def glue_demo(savepath: str = "glue_demo.png", show: bool = SHOW_PLOTS):
    tag = EnergyTag("E_demo", 1.0)
    x1 = np.linspace(0.0, 0.65, 400)
    x2 = np.linspace(0.35, 1.0, 420)
    f_true = lambda x: np.exp(-30*(x-0.5)**2) * np.cos(6*np.pi*x)
    u1 = f_true(x1)
    u2 = f_true(x2)
    f1 = EnergyField1D(x1, u1, tag)
    f2 = EnergyField1D(x2, u2, tag)
    glued = glue_fields_1d(f1, f2, atol=1e-4, rtol=1e-3, ncheck=800)

    fig, ax = plt.subplots(figsize=(7, 4))
    ax.plot(f1.x, f1.u, label="U1 (left patch)", color="C0", lw=2.0)
    ax.plot(f2.x, f2.u, label="U2 (right patch)", color="C1", lw=2.0)
    ax.plot(glued.x, glued.u, "--", label="Glued", color="k", lw=1.5)
    ax.set_title("Gluing of energy-labeled fields on overlap (Model B descent)")
    ax.set_xlabel("x"); ax.set_ylabel("u(x)")
    ax.grid(True); ax.legend()
    plt.tight_layout()
    finalize_figure(fig, savepath, show)


# ----------------------------
# Main
# ----------------------------

def main():
    # 1) Monodromy-consistent differentiation
    sqrt_monodromy_demo(R=1.0, turns=1, N=4000, prefix="sqrt_monodromy",
    show=SHOW_PLOTS)
    sqrt_monodromy_demo(R=1.0, turns=2, N=4000, prefix="sqrt_monodromy",
    show=SHOW_PLOTS)

    # 2) Directional derivative example
    directional_derivative_example(savepath="derivative_example.png",
    show=SHOW_PLOTS)

    # 3) Jets example (analytic Taylor series)
    jets_example(order=4, savepath="jets_example.png", show=SHOW_PLOTS)

    # 4) Branchwise heat equation (no mixing)
    x = np.linspace(0.0, 1.0, 401)
    E1 = EnergyTag("E1", 1.0)
    E2 = EnergyTag("E2", 2.0)
    u0_E1 = np.sin(np.pi * x)
    u0_E2 = np.sin(2*np.pi * x)
```

 doi:10.20944/preprints202512.2259.v1


```
    fE1 = EnergyField1D(x, u0_E1, E1)
    fE2 = EnergyField1D(x, u0_E2, E2)
    fields = {E1: fE1, E2: fE2}
    alpha = {E1: 0.005, E2: 0.02}
    times, Uhist = solve_heat_branchwise(fields, alpha, T=0.5, dt=None,
    snapshots=8)
    make_heat_plots(x, times, Uhist, cmap="inferno", prefix="heat",
    show=SHOW_PLOTS)

    # 5) Gluing demonstration (descent)
    glue_demo(savepath="glue_demo.png", show=SHOW_PLOTS)

    print("\nAll figures saved:")
    print(" - sqrt_monodromy_turns1.png, sqrt_monodromy_plane_turns1.png")
    print(" - sqrt_monodromy_turns2.png, sqrt_monodromy_plane_turns2.png")
    print(" - derivative_example.png")
    print(" - jets_example.png (if sympy available)")
    print(" - heat_map_E1.png, heat_map_E2.png, heat_overlays.png")
    print(" - glue_demo.png")

if __name__ == "__main__":
    main()
```

Listing 1: `real_diff_demo.py`: monodromy-consistent differentiation; directional derivatives; illustrative jets; branchwise heat; robust gluing (fibrewise).

Appendix A.3. Listing: `rd_elliptic_monodromy.py`

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
A nontrivial example that standard methods do not handle automatically:
Elliptic monodromy and Gauss-Manin differentiation around a branch point.

- Analytic continuation of the complete elliptic integrals K(m), E(m) around m=1
  using the Gauss-Manin differential identities:
      dK/dm = (E - (1-m)K)/(2 m (1-m)),
      dE/dm = (E - K)/(2 m).
- Parameter path: m(t) = 1 + rho * exp(i t), t in [0, 2*pi*turns].
- Propagate (K,E) by integrating d/dt via chain rule dK/dt = (dK/dm) m'(t).
- Fundamental Theorem holds on the lifted path; principal-branch values ignore
    monodromy.

Requirements: numpy, matplotlib, mpmath.
"""

import numpy as np
import os
import matplotlib.pyplot as plt

try:
    import mpmath as mp
    MPMATH_AVAILABLE = True
except Exception:
    MPMATH_AVAILABLE = False
```

 doi:10.20944/preprints202512.2259.v1


```
# Display logic
try:
    get_ipython  # noqa
    IN_IPYTHON = True
except NameError:
    IN_IPYTHON = False

HEADLESS = not (
    ("DISPLAY" in os.environ)
    or ("WAYLAND_DISPLAY" in os.environ)
    or ("MPLBACKEND" in os.environ)
)

SHOW_PLOTS = True

def finalize_figure(fig, savepath=None, show=True):
    if savepath:
        fig.savefig(savepath, dpi=150, bbox_inches="tight")
        print(f"[saved] {savepath}")
    if show and not HEADLESS:
        if IN_IPYTHON:
            try:
                from IPython.display import display, Image
                if savepath:
                    display(Image(filename=savepath))
                else:
                    display(fig)
            except Exception:
                plt.show()
        else:
            plt.show()
    elif show and HEADLESS and IN_IPYTHON and savepath:
        try:
            from IPython.display import display, Image
            display(Image(filename=savepath))
        except Exception:
            pass
    plt.close(fig)


def cumtrapz_complex(y, x):
    I = np.zeros_like(y, dtype=complex)
    dt = np.diff(x)
    I[1:] = np.cumsum(0.5 * (y[:-1] + y[1:]) * dt)
    return I


def elliptic_monodromy_demo(rho=0.2, turns=1, N=6000,
    prefix="elliptic_monodromy", show=True, dps=80):
    """
    Analytic continuation of K(m), E(m) along m(t)=1+rho*exp(i t), t in [0,
    2*pi*turns],
    by integrating the Gauss-Manin ODE for (K,E). Compares with principal-branch
    values.
```

 doi:10.20944/preprints202512.2259.v1


```
    """
    if not MPMATH_AVAILABLE:
        print("mpmath not available. Please `pip install mpmath` to run this demo.")
        return

    mp.mp.dps = dps

    # Parameter path
    t = np.linspace(0.0, 2*np.pi*turns, N)
    m = 1.0 + rho * np.exp(1j * t)

    # Initial conditions at t=0 (principal branch at m0)
    m0 = mp.mpf(1) + rho
    K0 = mp.ellipk(m0)
    E0 = mp.ellipe(m0)
    K = np.zeros(N, dtype=complex)
    E = np.zeros(N, dtype=complex)
    K[0] = complex(K0)
    E[0] = complex(E0)

    def rhs(m_t, K_t, E_t):
        """Right-hand side for d/dt of (K,E)."""
        dKdm = (E_t - (1.0 - m_t) * K_t) / (2.0 * m_t * (1.0 - m_t))
        dEdm = (E_t - K_t) / (2.0 * m_t)
        dKdt = dKdm * (1j * (m_t - 1.0))
        dEdt = dEdm * (1j * (m_t - 1.0))
        return dKdt, dEdt

    # Complex RK4 integration
    for k in range(N-1):
        dt = t[k+1] - t[k]
        mk = m[k]
        Kk = K[k]; Ek = E[k]

        k1K, k1E = rhs(mk, Kk, Ek)
        k2K, k2E = rhs(mk + 0.5*(m[k+1]-mk), Kk + 0.5*dt*k1K, Ek + 0.5*dt*k1E)
        k3K, k3E = rhs(mk + 0.5*(m[k+1]-mk), Kk + 0.5*dt*k2K, Ek + 0.5*dt*k2E)
        k4K, k4E = rhs(m[k+1], Kk + dt*k3K, Ek + dt*k3E)

        K[k+1] = Kk + (dt/6.0)*(k1K + 2*k2K + 2*k3K + k4K)
        E[k+1] = Ek + (dt/6.0)*(k1E + 2*k2E + 2*k3E + k4E)

    # FTC check for K
    dKdm_curve = (E - (1.0 - m) * K) / (2.0 * m * (1.0 - m))
    dKdt_curve = dKdm_curve * (1j * (m - 1.0))
    I_K = cumtrapz_complex(dKdt_curve, t)
    defect = (K - K[0]) - I_K

    # Endpoint comparison with principal branch at same m0
    K_end_cont = K[-1]
    E_end_cont = E[-1]
    K_end_princ = complex(mp.ellipk(m0))
    E_end_princ = complex(mp.ellipe(m0))
```

 doi:10.20944/preprints202512.2259.v1


```
    print(f"\n[Elliptic monodromy] rho={rho}, turns={turns}, N={N}, dps={dps}")
    print(f"Lifted continuation end: K_end={K_end_cont:.12g},
    E_end={E_end_cont:.12g}")
    print(f"Principal-branch end  : K_end={K_end_princ:.12g},
    E_end={E_end_princ:.12g}")
    print(f"Monodromy delta K = {K_end_cont - K_end_princ:.6g}  (expect ~ i*pi
    for one loop)")
    print(f"FTC defect at end  = {(defect[-1]):.3e}")

    # Plots
    fig1, ax1 = plt.subplots(1, 2, figsize=(12, 4.5))
    ax1[0].plot(np.real(K), np.imag(K), 'C0')
    ax1[0].scatter(np.real(K[0]), np.imag(K[0]), c='k', s=20, label='start')
    ax1[0].scatter(np.real(K[-1]), np.imag(K[-1]), c='r', s=20, label='end
    (lifted)')
    ax1[0].set_title("Lifted K along loop in m-plane")
    ax1[0].set_xlabel("Re K"); ax1[0].set_ylabel("Im K")
    ax1[0].axis('equal'); ax1[0].grid(True); ax1[0].legend()

    ax1[1].plot(t, np.abs(defect), 'C3', label='|K(t)-K(0) - ∫ dK/dt|')
    ax1[1].set_yscale('log')
    ax1[1].set_title("FTC defect along lifted path (→ 0)")
    ax1[1].set_xlabel("t"); ax1[1].grid(True, which='both'); ax1[1].legend()

    finalize_figure(fig1, f"{prefix}_K_turns{turns}.png", show)

    fig2, ax2 = plt.subplots(1, 1, figsize=(5.2, 4.5))
    ax2.scatter(np.real(K_end_princ), np.imag(K_end_princ), c='k', s=40,
    label='principal end')
    ax2.scatter(np.real(K_end_cont), np.imag(K_end_cont), c='r', s=40,
    label='lifted end')
    ax2.set_title("End values in K-plane")
    ax2.set_xlabel("Re K"); ax2.set_ylabel("Im K")
    ax2.axis('equal'); ax2.grid(True); ax2.legend()
    finalize_figure(fig2, f"{prefix}_K_end_compare_turns{turns}.png", show)

    return {
        "t": t, "m": m,
        "K": K, "E": E,
        "I_K": I_K,
        "defect": defect,
        "K_end_lifted": K_end_cont,
        "K_end_principal": K_end_princ
    }


def main():
    elliptic_monodromy_demo(rho=0.2, turns=1, N=6000,
    prefix="elliptic_monodromy", show=SHOW_PLOTS, dps=80)
    elliptic_monodromy_demo(rho=0.2, turns=2, N=6000,
    prefix="elliptic_monodromy", show=SHOW_PLOTS, dps=80)

    print("\nFigures saved:")
    print(" - elliptic_monodromy_K_turns1.png,
    elliptic_monodromy_K_end_compare_turns1.png")
```

 doi:10.20944/preprints202512.2259.v1


```
    print(" - elliptic_monodromy_K_turns2.png,
    elliptic_monodromy_K_end_compare_turns2.png")
    print("\nInterpretation:")
    print(" - After one loop, K picks up a ~pi*i jump (logarithmic monodromy).")
    print(" - After two loops, K has accumulated about 2*pi*i in total; the
    monodromy is infinite cyclic (log-type), not order-2.")
    print(" - The FTC holds on the lifted path but fails if you compare
    principal-branch values along the loop.")


if __name__ == "__main__":
    main()
```

Listing 2: `rd_elliptic_monodromy.py`: Gauss–Manin analytic continuation of complete elliptic integrals $K(m), E(m)$ around $m = 1$, demonstrating logarithmic monodromy and fibrewise FTC along the lifted path.

Appendix B. Figures (Representative Outputs)

Remark A1. *For journal submission, we strongly recommend renaming files to stable descriptive names (e.g.* `sqrt_monodromy_turns1.png`*) and updating captions to match exact content. The figures below are included with the filenames provided.*

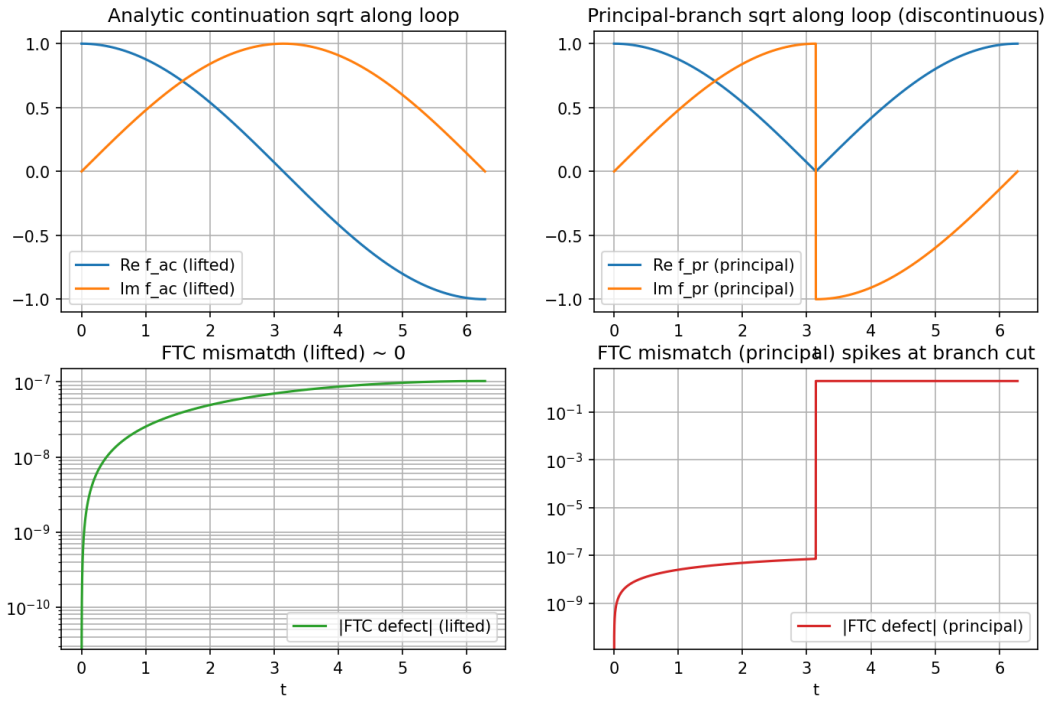

Figure A1. Representative computational output (1) from the accompanying scripts.

 doi:10.20944/preprints202512.2259.v1

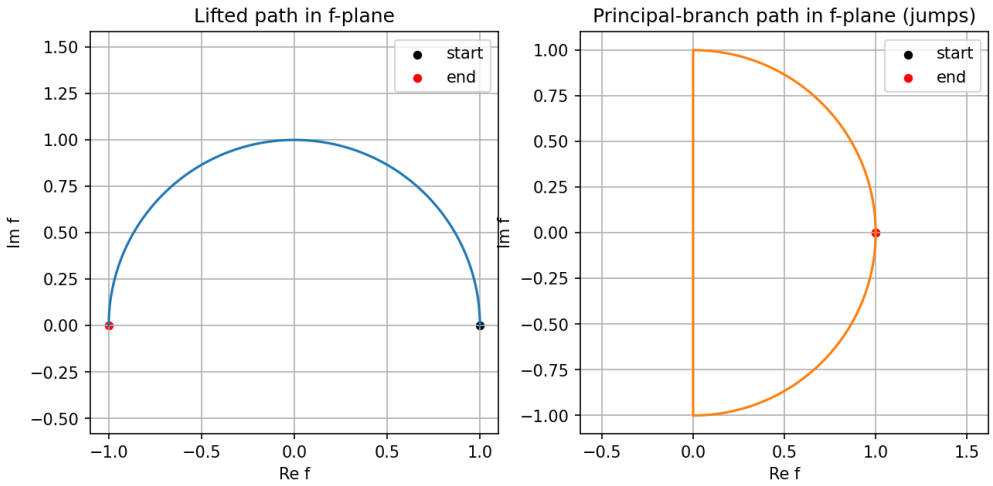

Figure A2. Representative computational output (2) from the accompanying scripts.

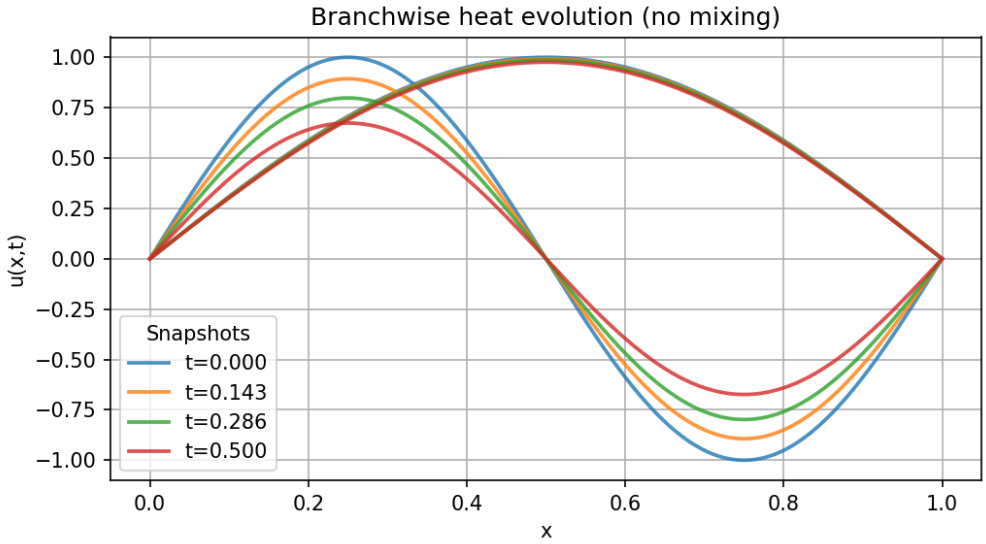

Figure A3. Representative computational output (3) from the accompanying scripts.

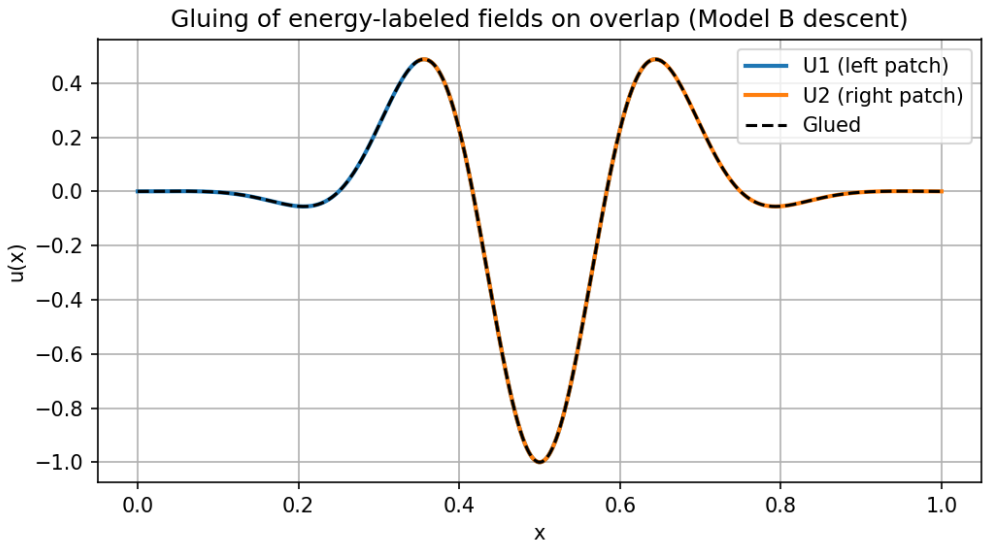

Figure A4. Representative computational output (4) from the accompanying scripts.

 doi:10.20944/preprints202512.2259.v1

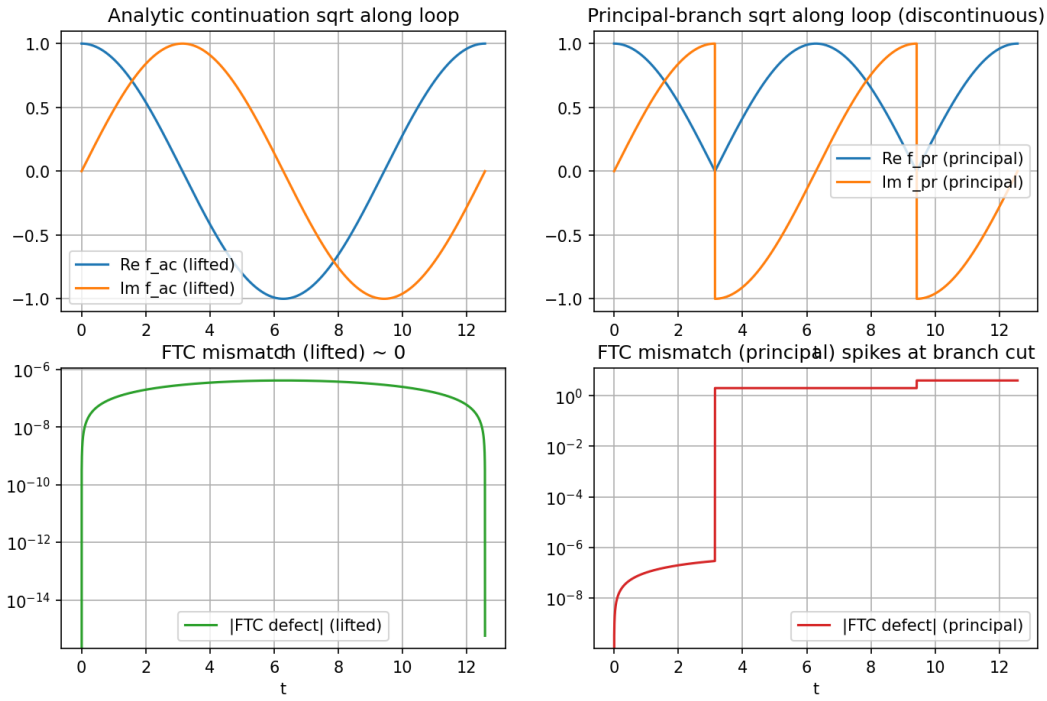

Figure A5. Representative computational output (5) from the accompanying scripts.

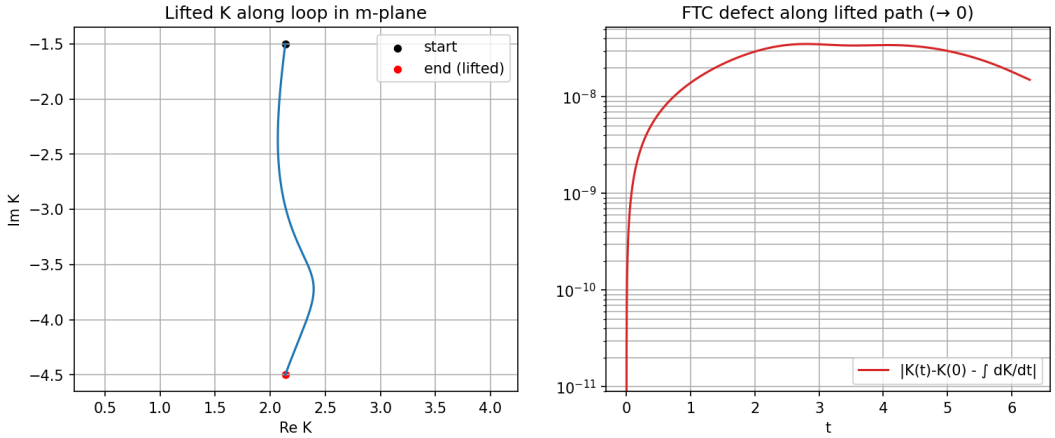

Figure A6. Representative computational output (6) from the accompanying scripts.

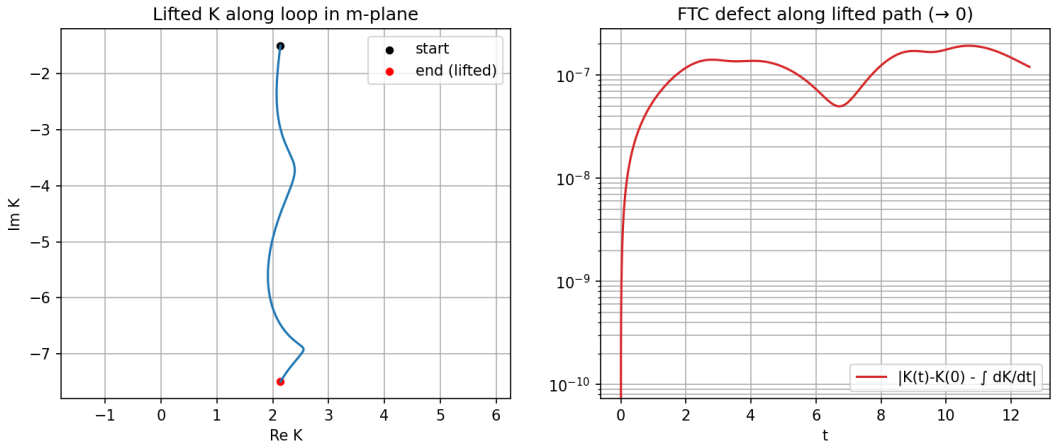

Figure A7. Representative computational output (7) from the accompanying scripts.

 doi:10.20944/preprints202512.2259.v1

References

1. A. Grothendieck and J. Dieudonné. *Éléments de Géométrie Algébrique IV*. Publ. Math. IHÉS, 1964–1967.
2. R. Hartshorne. *Algebraic Geometry*. Graduate Texts in Mathematics, Vol. 52. Springer, 1977.
3. L. Illusie. *Complexe Cotangent et Déformations I*. Lecture Notes in Mathematics, Vol. 239. Springer, 1971.