One-Dimensional Derivatives in Mathlib

Analysis.Calculus.Deriv.Basic

A Study Guide with Natural Language Explanations

Contents

1. Introduction

This module develops the theory of derivatives for functions $f : \mathbb{k} \to F$ where $\mathbb{k}$ is a normed field (like $\mathbb{R}$ or $\mathbb{C}$) and F is a normed space over that field. The derivative at a point gives us the instantaneous rate of change, represented as an element of F.

The key insight is that one-dimensional derivatives are special cases of Fréchet derivatives, which allows us to leverage the more general theory while providing specialized, simpler notation for the one-dimensional case.

2. Core Definitions

2.1. HasDerivAtFilter

Definition: `HasDerivAtFilter f f' x L`

Natural Language: The function f has derivative f' at point x when we approach x along the filter L.

Intuition: This is the most general form of having a derivative. A filter L describes a way of approaching the point x. The derivative f' tells us how fast f changes as we move through x along paths specified by L.

Mathematical Meaning: As x' approaches x along filter L: $f(x') = f(x) + (x' - x) \cdot f' + o(x' - x)$

where $o(x' - x)$ represents terms that vanish faster than linearly.

2.2. HasDerivWithinAt

Definition: `HasDerivWithinAt f f' s x`

Natural Language: The function f has derivative f' at point x when we only consider points within the set s.

Intuition: Sometimes we can only define derivatives by looking at nearby points in a restricted set. For example, when f is only defined on $[0, 1]$, we need this notion at the endpoints.

Real-world Example: Consider the speed of a car that can only drive on a specific road (the set s). The derivative at a point tells us the instantaneous velocity, but we can only measure it using positions along that road.

Mathematical Meaning: This is `HasDerivAtFilter` with $L = \mathcal{N}[s]x$ (the neighborhood filter restricted to s).

2.3. HasDerivAt

Definition: `HasDerivAt f f' x`

Natural Language: The function f has derivative f' at point x in the usual sense - approaching from all directions.

Intuition: This is the standard derivative from calculus. The function changes at rate f' when passing through x, regardless of the direction of approach.

Connection to Calculus: For $f : \mathbb{R} \to \mathbb{R}$, this means: $\lim_{h \to 0} \frac{f(x+h)-f(x)}{h} = f'$

Mathematical Meaning: This is `HasDerivAtFilter` with $L = \mathcal{N}x$ (the full neighborhood filter).

2.4. HasStrictDerivAt

Definition: `HasStrictDerivAt f f' x`

Natural Language: The function f has derivative f' at point x in a strict sense - the linear approximation works for comparing any two nearby points, not just comparing to x.

Intuition: This stronger notion says that near x, the function behaves almost linearly with slope f'. The difference between any two nearby values can be approximated by f' times their distance.

Why "Strict"?: Regular derivatives compare $f(y)$ to $f(x)$. Strict derivatives can compare $f(y)$ to $f(z)$ for any y, z near x: $f(y) - f(z) = (y - z) \cdot f' + o(y - z)$ as $y, z \to x$

Use Case: Strict differentiability is preserved under uniform limits, making it useful in analysis.

2.5. derivWithin

Definition: `derivWithin f s x`

Natural Language: The actual value of the derivative of f at x within set s, or zero if it doesn't exist.

Intuition: While `HasDerivWithinAt` is a proposition (true/false), `derivWithin` is a function that returns the actual derivative value. It's designed to always return something (zero as default) to avoid partial functions.

Practical Use: You can write expressions like `derivWithin f s x = 3` without worrying about existence - if the derivative doesn't exist, both sides will be zero.

Relationship: `HasDerivWithinAt f f' s x ↔ derivWithin f s x = f'` (when the derivative exists uniquely).

2.6. deriv

Definition: `deriv f x`

Natural Language: The actual value of the derivative of f at x, or zero if it doesn't exist.

Intuition: This is what you compute when you "take the derivative" in calculus. It's a function $\mathbb{R} \to \mathbb{R}$ (or more generally $\mathbb{k} \to F$) that gives the derivative at each point.

Example: If $f(x) = x^2$, then deriv $f x = 2x$.

Default Value: Returns zero when the derivative doesn't exist, making it a total function. This simplifies many proofs and computations.

3. Key Relationships

3.1. Connection to Fréchet Derivatives

One-dimensional derivatives are special cases of Fréchet derivatives where the linear map is simply scalar multiplication:

- `HasDerivAt f f' x` is equivalent to `HasFDerivAt f (smulRight 1 f') x`
- The `smulRight 1 f'` creates a linear map that multiplies its input by f'

This relationship allows us to:
1. Prove theorems about derivatives using the general Fréchet theory
2. Specialize Fréchet results to get cleaner one-dimensional versions
3. Switch between the two viewpoints as convenient

3.2. Hierarchy of Derivative Concepts

HasStrictDerivAt (strongest) ↓ HasDerivAt ↓ HasDerivWithinAt ↓ HasDerivAtFilter (most general)

4. Computational Rules

The module establishes derivatives for basic operations:

4.1. Elementary Functions

- **Constants:** deriv $(\lambda x, c)x = 0$ - Constants don't change, so zero rate of change
- **Identity:** deriv $(\lambda x, x)x = 1$ - Linear function with slope 1
- **Linear maps:** deriv $(\lambda x, L(x))x = L(1)$ - Linear maps have constant derivative

4.2. Arithmetic Operations

- **Addition:** deriv $(f + g)x =$ deriv $f x +$ deriv $g x$ - Derivatives add
- **Scalar multiplication:** deriv $(c \cdot f)x = c \cdot$ deriv $f x$ - Constants factor out
- **Product rule:** deriv $(f \cdot g)x =$ deriv $f x \cdot g(x) + f(x) \cdot$ deriv $g x$
- **Chain rule:** deriv $(f \circ g)x =$ deriv $f(g(x)) \cdot$ deriv $g x$

4.3. Advanced Operations

- **Powers:** deriv $(\lambda x, x^n)x = n \cdot x^{n-1}$
- **Inverse:** deriv $(\lambda x, x^{-1})x = -x^{-2}$ (when $x \neq 0$)
- **Division:** Combination of product rule and inverse rule

5. Practical Examples

5.1. Example: Calculating a Derivative

Consider $f(x) = \cos(\sin(x)) \cdot e^x$. The module's simplification rules allow:

```
example (x : ℝ) :
    deriv (fun x ↦ cos (sin x) * exp x) x =
    (cos (sin x) - sin (sin x) * cos x) * exp x := by
  simp; ring
```

The simplifier automatically:
1. Applies the product rule
2. Uses chain rule for $\cos(\sin(x))$
3. Knows $\text{deriv}(\exp) = \exp$
4. Combines terms algebraically

5.2. Example: One-sided Derivatives

For a function only defined on $[0, \infty)$, we use `HasDerivWithinAt`:

```
HasDerivWithinAt sqrt (1 / (2 * sqrt x)) (Ici 0) x
```

This captures that $\sqrt{x}$ has derivative $\frac{1}{2\sqrt{x}}$ when approaching from the right at any $x > 0$.

6. Design Philosophy

6.1. Total Functions via Defaults

Making `deriv` return zero for non-differentiable points might seem odd, but it:
- Avoids partial functions and `Option` types
- Simplifies algebraic manipulation
- Matches the convention that "zero is the most boring value"
- Works well with the simplifier

6.2. Strict vs. Non-strict

Having both strict and non-strict versions allows:
- Stronger theorems when strictness is available
- More general applicability with non-strict versions
- Choice of the right tool for each proof

6.3. Filter-based Foundations

Using filters as the foundation provides:
- Unified treatment of limits from different directions
- Clean handling of one-sided derivatives
- Natural generalization to more exotic spaces
- Compatibility with Mathlib's topology library

7. Conclusion

This module provides a complete foundation for one-dimensional calculus in a formally verified setting. The definitions balance:
- Mathematical precision with practical usability
- Generality with specialized efficiency
- Connection to advanced theory with elementary accessibility

The result is a system where both simple calculus computations and sophisticated analytical arguments can be expressed naturally and verified rigorously.