Imperial College London

11/03/2023

Partial Fraction Decomposition

and its Formalisation in Lean

Sidharth Hariharan
sidharth.hariharan21@imperial.ac.uk

Motivations

How would you simplify the following?

1. $\displaystyle\int_0^1 \frac{1}{x^2+5x+6}\,\mathrm{d}x$
2. $\displaystyle\sum_{x=2}^{n} \frac{1}{x-x^2}$

Motivations

How would you simplify the following?

1. $$\int_0^1 \frac{1}{x^2 + 5x + 6}\,\mathrm{d}x = \int_0^1 \left(\frac{1}{x+2} - \frac{1}{x+3}\right)\mathrm{d}x$$
2. $$\sum_{x=2}^{n} \frac{1}{x - x^2} = \sum_{x=2}^{n} \left(\frac{1}{x} - \frac{1}{x-1}\right)$$

Motivations

How would you simplify the following?

1. $$\int_0^1 \frac{1}{x^2+5x+6}\,\mathrm{d}x = \int_0^1 \left(\frac{1}{x+2} - \frac{1}{x+3}\right)\mathrm{d}x = \log\left(\frac{1}{2}\right)$$
2. $$\sum_{x=2}^{n} \frac{1}{x-x^2} = \sum_{x=2}^{n} \left(\frac{1}{x} - \frac{1}{x-1}\right) = \frac{1}{n} - 1$$

Discussion

At this point, we ask a very natural question:

Discussion

At this point, we ask a very natural question:

Under what conditions can we thus decompose fractions of polynomials?

Imperial College London

Discussion

Our goal today will be:

- To understand what properties are sufficient for a decomposition to exist.
- To prove that a decomposition exists when said properties are satisfied.
- To explore the intricacies of formalising of this result in Lean.

Discussion

Our goal today will be:

- To understand what properties are sufficient for a decomposition to exist.
- To prove that a decomposition exists when said properties are satisfied.
- To explore the intricacies of formalising of this result in Lean.

But first: a bit of background on Lean.

Lean: Some Background

Lean is a dependent type theory-based functional programming language that's being used to formalise and verify proofs.

Lean: Some Background

Lean is a dependent type theory-based functional programming language that's being used to formalise and verify proofs.

In Lean, one can have objects, definitions, theorems and proofs.

Lean: Some Background

Lean is a dependent type theory-based functional programming language that's being used to formalise and verify proofs.

In Lean, one can have objects, definitions, theorems and proofs.

The long-term goal is to create a unified digital repository—mathlib—consisting of all the mathematics we know.

Lean: Some Background

Lean is a dependent type theory-based functional programming language that's being used to formalise and verify proofs.

In Lean, one can have objects, definitions, theorems and proofs.

The long-term goal is to create a unified digital repository—mathlib—consisting of all the mathematics we know.

mathlib conventions include

- Inheriting rather than duplicating code
- Formalising to maximum generality
- Optimising code to minimise compile time

Why are Lean proofs different from mathematical proofs?

Mathematics is usually done quite informally.

Why are Lean proofs different from mathematical proofs?

Mathematics is usually done quite informally.

We gloss over a lot of details: we say things like $\mathbb{Z} \subset \mathbb{R}$, when what we really mean is that there's an injective homomorphism from $\mathbb{Z}$ to $\mathbb{R}$.

Why are Lean proofs different from mathematical proofs?

Mathematics is usually done quite informally.

We gloss over a lot of details: we say things like $\mathbb{Z} \subset \mathbb{R}$, when what we really mean is that there's an injective homomorphism from $\mathbb{Z}$ to $\mathbb{R}$.

The Lean compiler checks proofs down to the *axiomatic level!*

Why are Lean proofs different from mathematical proofs?

Mathematics is usually done quite informally.

We gloss over a lot of details: we say things like $\mathbb{Z} \subset \mathbb{R}$, when what we really mean is that there's an injective homomorphism from $\mathbb{Z}$ to $\mathbb{R}$.

The Lean compiler checks proofs down to the *axiomatic level!*

So, if we want to "teach" it a proof, we need to lay out our argument *in a manner it can understand*.

Our Strategy

With this in mind, we can now plan our strategy to prove our "partial fraction decomposition theorem" the way we would in Lean.

Our Strategy

With this in mind, we can now plan our strategy to prove our "partial fraction decomposition theorem" the way we would in Lean.

We will:

- Define the relevant terminology
- State (without proof) a few results we'll need
- Prove the theorem!

Our Strategy

With this in mind, we can now plan our strategy to prove our "partial fraction decomposition theorem" the way we would in Lean.

We will:

- Define the relevant terminology
- State (without proof) a few results we'll need
- Prove the theorem!

Then, we'll look at some of the intricacies of proving it *in Lean*.

Imperial College London

On Domains

Polynomial coefficients generally come from Rings, from which are inherited polynomial addition and multiplication.

Typically, they come from Integral Domains, which are a very broad class of rings that tend to be "well-behaved"—whatever that means.

Definition (Domain)

A Ring R is called a Domain if $\forall a, b \in R \setminus \{0\}$, we have $ab \neq 0$.

Definition (Integral Domain)

An Integral Domain is a Commutative Ring that's also a Domain.

Polynomial Rings

We're now ready for the following:

Definition (Polynomial Ring over an Integral Domain)

If R is an Integral Domain, the set of polynomials with coefficients in R under addition $(p+q)(X) = p(X) + q(X)$ and multiplication $(pq)(X) = p(X)q(X)$ is called the Polynomial Ring $R[X]$. [a]

[a] In fact, $R[X]$ is *also* an Integral Domain.

Note that X does **not** have to belong to R. It is called an "indeterminate" or a "variable" and is treated somewhat like a constant when one manipulates polynomials.

Can you think of an example where one can apply a polynomial to an object not in the base ring?

More Definitions

Definition (Monic)

Let R be an ID. $f \in R[X]$ is monic if its leading coefficient is 1, ie, if f is of the form

$$f(X) = a_0 + a_1 X + a_2 X^2 + \cdots + a_{n-1} X^{n-1} + X^n$$

where $n = \deg(f)$ and $a_i \in R$ for all i.

Definition (Coprime)

Let R be an ID. Then, $f, g \in R[X]$ are coprime if $\exists a, b \in R[X]$ s.t.

$$af + bg = 1$$

Useful Results

Theorem

Let R be an ID. For $f, g \in R[X]$ with g monic, $\exists Q, R \in R[X]$ s.t. $R + Qg = f$.

This is given by `polynomial.mod_by_monic_add_div` in Lean's math library mathlib. Note also that Q and R are respectively denoted `f /ₘ g` and `f %ₘ g` in Lean. Note $\deg(R) < \deg(g)$, given by `polynomial.degree_mod_by_monic_lt`.

Lemma

Let R be an ID. Fix $f, g, h \in R[X]$ and let f, g and f, h be coprime. Then, f and gh are coprime.

In mathlib, this exists (more generally) as `is_coprime.mul_right`.

Main Result

Theorem (Partial Fraction Decomposition)

Let R be an Integral Domain. Fix $f, g_1, g_2, \cdots, g_n \in R[X]$ and let the g_is be *monic* and *pairwise coprime*. Then, $\exists q, r_1, r_2, \cdots, r_n \in R[X]$ such that $\deg(r_i) < \deg(g_i)$ for all i, and

$$\frac{f}{\prod_{i=1}^{n} g_i} = q + \sum_{i=1}^{n} \frac{r_i}{g_i}$$

Imperial College London

Proof Sketch

We start by proving the $n = 2$ case and then proceed by induction.

Proof Sketch: $n = 2$

By coprimality of g_1 and g_2, we know $\exists c, d \in R[X]$ s.t. $cg_1 + dg_2 = 1$. Then, we write $f = f \cdot 1$, and then get

$$\frac{f}{g_1 g_2} = \frac{f\left(cg_1 + dg_2\right)}{g_1 g_2} = \frac{cf}{g_2} + \frac{df}{g_1}$$

We know that $\exists q_1, q_2, r_1, r_2 \in R[X]$ s.t.

$$\frac{cf}{g_2} + \frac{df}{g_1} = \left(q_2 + \frac{r_2}{g_2}\right) + \left(q_1 + \frac{r_1}{g_1}\right) = (q_1 + q_2) + \frac{r_1}{g_1} + \frac{r_2}{g_2} \qquad \square$$

Imperial College London

Proof Sketch: General n

We proceed by induction on n. The $n = 1$ base case follows directly from the quotient-remainder result[1].

We assume the result for general n. Then, we write

$$
\begin{aligned}
\frac{f}{\prod_{i=1}^{n+1} g_i} &= \frac{f}{\left(\prod_{i=1}^{n} g_i\right) \cdot g_{n+1}} \\
&= q' + \frac{r_{n+1}}{g_{n+1}} + \frac{f}{\prod_{i=1}^{n} g_i} \\
&= \left(q' + Q\right) + \frac{r_{n+1}}{g_{n+1}} + \sum_{i=1}^{n} \frac{r_i}{g_i} \qquad \square
\end{aligned}
$$

[1] In informal mathematics, yes; in Lean, *not quite!*

Things are not so easy in Lean!!

Things are not so easy in Lean!!

There's (mainly) two things that are different in Lean:

Things are not so easy in Lean!!

There's (mainly) two things that are different in Lean:

- We use a different style of induction
- The distinction between an Integral Domain and its Field of Fractions is significantly more pronounced

The Use of Finsets

In Lean, rather than indexing over $\mathbb{N}$, we index over an arbitrary type ι and look at an arbitrary finite subset `s` of ι. This allows for more generality.

Induction therefore looks a bit different:

- Base Case: True over ($\emptyset$ `: finset` ι)
- Inductive Case: True over (`b : finset` ι) $\implies$ true over (`insert a b : finset` ι), where (`a :` ι) and (`a` $\notin$ `b`).

Proving this base case and inductive case amounts to proving the result for any finite subset of ι.

The Distinction between an ID and its Field of Fractions

Roughly speaking, the Field of Fractions K over an Integral Domain R is

$$\text{"}\,\{p/q : p \in R, q \in R \setminus \{0\}\}\,\text{"}$$

But, this is a pretty terrible definition... can anyone see why?

Fields of Fractions

To fix the problem, we make use of the following two observations:

- Each element "p/q" consists of two ring elements: p and q
- Each "p/q" would also be expected to be the same as "px/qx" for some $x \in R \setminus 0$. So, each element is not necessarily represented by a *unique* pair (p, q).

Field of Fractions

Let R be an ID. Consider the set $J := R \times (R \setminus \{0\})$. Define an equivalence relation $\sim$ on J by $(a, b) \sim (c, d)$ iff $ad = bc$. This loosely models what we would intuitively expect:

$$"\frac{a}{b} = \frac{c}{d} \iff ad = bc"$$

Definition (Field of Fractions)

Given an ID R, we construct a set J as above. Then, the Field of Fractions K of R is the set of Equivalence Classes on J given by $\sim$, which forms a field under the operations

$$[(a, b)] + [(c, d)] = [(ad + bc, bd)]$$
$$[(a, b)] \cdot [(c, d)] = [(ac, bd)]$$

Field of Fractions

At this point, we note the following things:

- The construction of the Field of Fractions mirrors identically the construction of $\mathbb{Q}$ from $\mathbb{Z}$.
- The reason we need R to be an ID is so that we don't get 0 in the denominator when we multiply two "fractions".
- It is **improper**[2] to say that $R \subseteq K$ (even though we were taught in school that $\mathbb{Z} \subseteq \mathbb{Q}$). What is true, however, is that there is an injective "inclusion map" going from R to K.

(Note: we usually drop the $[(p, q)]$ notation and just use normal fraction notation p/q.)

[2] At least, from a *strictly formal perspective*

Why does this matter?

In Lean, all results we have about polynomials in $R[X]$ are *in* $R[X]$.
However, the theorem we have to prove is in the *field of fractions!*

Why does this matter?

In Lean, all results we have about polynomials in $R[X]$ are *in* $R[X]$. However, the theorem we have to prove is in the *field of fractions!*

This means that when we use these results in $R[X]$ to prove our theorem, we have to apply the injective homomorphism from $R[X]$ to the fraction field.

Why does this matter?

In Lean, all results we have about polynomials in $R[X]$ are *in* $R[X]$. However, the theorem we have to prove is in the *field of fractions!*

This means that when we use these results in $R[X]$ to prove our theorem, we have to apply the injective homomorphism from $R[X]$ to the fraction field.

This is typically expressed as a **coercion**.

Concluding Remarks

Today, we have seen

- How to rigorously prove the existence of a partial fraction decomposition, given some conditions
- What Lean is all about, and what's involved in formalising mathematics in Lean
- Some of the intricacies involved in formalising and proving this theorem in Lean

Concluding Remarks

Today, we have seen

- How to rigorously prove the existence of a partial fraction decomposition, given some conditions
- What Lean is all about, and what's involved in formalising mathematics in Lean
- Some of the intricacies involved in formalising and proving this theorem in Lean

Any questions?

Further Resources

SCAN ME

(a) The mathlib file I wrote

SCAN ME

(b) My GitHub page with these slides