An Introduction to Formal Real Analysis, Rutgers University, Fall 2025, Math 311H

Lecture 8: Advanced Limit Theorems and Induction

Prof. Alex Kontorovich

This text is automatically generated by LLM from "Real Analysis, The Game", Lecture 8

1 Introduction: Mathematical Induction

SIMPLICIO: Hey Socrates, I've been thinking about something that's been bothering me. When we prove things in mathematics, we usually prove a specific statement. But what if I want to prove something is true for *all* natural numbers? Like, how do I prove a statement for 0, 1, 2, 3, 4, and so on... forever?

SOCRATES: Ah, an excellent question! You're right that we can't just check each case one by one—that would take infinitely long. Tell me, Simplicio, have you ever climbed a ladder?

SIMPLICIO: Of course! What does that have to do with anything?

SOCRATES: Well, imagine an infinitely tall ladder reaching up to the sky. If I wanted to convince you that you *could* climb to any rung on this ladder, what would I need to show you?

SIMPLICIO: Hmm... I guess you'd need to show me that I can reach the bottom-most rung?

SOCRATES: Good start! And what else?

SIMPLICIO: Well, if I'm standing on any particular rung, I'd need to know I can reach the next one up. So if I can always step from one rung to the next...

SOCRATES: Exactly! So if you can reach the first rung, and you can always step from rung k to rung $k+1$, then what can you conclude?

SIMPLICIO: Oh! Then I can reach *any* rung I want! If I want to reach rung 100, I just start at rung 0, step to rung 1, then to rung 2, and keep going until I reach rung 100. And the same works for any number!

SOCRATES: Precisely! This is the essence of **mathematical induction**. To prove something is true for all natural numbers n, you need exactly two things:

- A **base case**: prove it's true for $n = 0$
- An **inductive step**: prove that *if* it's true for $n = k$, *then* it's true for $n = k + 1$

SIMPLICIO: Wait, but in the inductive step, aren't we assuming what we're trying to prove? Isn't that circular reasoning?

SOCRATES: An astute observation! But no, it's not circular. We're not assuming the statement is true for all n. We're only assuming it's true for one particular value k, and using that assumption to prove that it's true for $k + 1$. We're proving an implication: "if $P(k)$ then $P(k + 1)$". Combined with the base case, this creates a chain reaction that reaches any natural number.

SIMPLICIO: Hmm, I think I see. So the assumption "it's true for k" is called the inductive hypothesis?

SOCRATES: Exactly! And that hypothesis is your most powerful tool. It's like having a foothold on rung k that you can push off from to reach rung $k + 1$.

SIMPLICIO: But why does this work? I mean, why should I believe this principle?

SOCRATES: Ah, a deep question! It comes from the very definition of the natural numbers themselves. How do you think the natural numbers are constructed?

SIMPLICIO: Well... I guess we start with 0. And then we have 1, which is $0 + 1$. And 2 is $1 + 1$. So each number is the "successor" of the previous one?

SOCRATES: Beautiful! The natural numbers are defined by exactly this process:

- Zero is a natural number
- If k is a natural number, then $k + 1$ is also a natural number
- These are the *only* natural numbers

Do you see how this mirrors the structure of induction?

SIMPLICIO: Oh wow! The base case corresponds to "zero is a natural number," and the inductive step corresponds to "if k is a natural number, then so is $k + 1$." Induction is just the construction of the natural numbers turned into a proof technique!

SOCRATES: Precisely! There are no "gaps" in the natural numbers—no number that can't be reached by starting at 0 and repeatedly adding 1. This is why induction works.

In fact, this is not just a philosophical observation—this is *literally* how the natural numbers are implemented in Lean! In Lean's type theory, a natural number is defined inductively as either:

- `zero : ℕ`, the base case, or
- `succ n : ℕ`, the successor of another natural number n

Here's how this looks in the core of Lean:

```
inductive Nat where
  | zero : Nat
  | succ (n : Nat) : Nat
```

You simply declare the existence of a natural number called `zero`, and then we declare that, given any natural number `n`, there's another one called `succ n`. (The word `succ` is here just a name; we could have called it `Alice n`. The important thing is that we're giving a way to construct a new natural number from a previously existing one.)

So the number 3, for instance, is literally represented as `succ (succ (succ zero))`. The principle of induction doesn't just *resemble* this construction, it

directly exploits it! When you prove something by induction in Lean, you're working with the actual computational structure of how natural numbers exist in the system.

SIMPLICIO: That's amazing! So induction isn't just a proof technique, it's baked into the very fabric of how Lean understands numbers?

SOCRATES: Exactly. The principle of mathematical induction is a theorem in many mathematical frameworks, but in type theory, it's a fundamental consequence of how the natural numbers are defined.

SIMPLICIO: Okay, I'm convinced this is a legitimate proof technique. Can you give me an example?

SOCRATES: Certainly.

2 Big Boss: Reciprocals of Convergent Sequences

One of the most important limit theorems concerns reciprocals: if a sequence converges to a nonzero limit, then the sequence of reciprocals converges to the reciprocal of the limit. This result is crucial for proving theorems about quotients and rational functions.

This is a Big Boss level—it requires synthesizing multiple techniques: working with nonzero limits, manipulating complex algebraic expressions, and carefully choosing epsilon strategies.

2.1 The Mathematical Setup

Theorem: If $a : \mathbb{N} \to \mathbb{R}$ converges to L with $L \neq 0$, and $b : \mathbb{N} \to \mathbb{R}$ is defined by $b(n) = 1/a(n)$ for all n, then b converges to $1/L$.

This is the most technically challenging proof in this lecture series.

2.2 New Tools

abs_div: For any real numbers x and y (with $y \neq 0$), we have $|x/y| = |x|/|y|$.

nonzero_of_abs_pos: If $0 < |x|$, then $x \neq 0$.

2.3 Strategic Approach

The key challenges are:

- Ensuring that $a(n) \neq 0$ eventually, so the reciprocals are well-defined
- Bounding $|1/a(n) - 1/L|$ by getting a common denominator
- Choosing the right epsilon when applying the convergence of a to L
- Using the lower bound on $|a(n)|$ to control the reciprocals

The critical insight is that the "right" epsilon isn't the obvious choice. We use $\varepsilon \cdot |L|^2/2$, which is precisely engineered to make the final inequalities work out.

2.4 Lean Solution

```
Statement InvLim (a : ℕ → ℝ) (L : ℝ) (aToL : SeqLim a L)
    (LneZero : L ≠ 0) (b : ℕ →
ℝ) (bEqInva : ∀ n, b n = 1 / a n) :
    SeqLim b (1 / L) := by
choose NhalfL hNhalfL using EventuallyGeHalfLimPos a L
   aToL LneZero
intro ε hε
have : 0 < |L| := by apply abs_pos_of_nonzero LneZero
specialize aToL (ε * |L| * |L| / 2) (by bound)
choose Na hNa using aToL
use Na + NhalfL
intro n hn
specialize bEqInva n
rewrite [bEqInva]
have hnHalfL : NhalfL ≤ n := by bound
have hna : Na ≤ n := by bound
specialize hNhalfL n hnHalfL
specialize hNa n hna
have : 0 < |a n| := by bound
have : a n ≠ 0 := by apply nonzero_of_abs_pos this
have l1 : |1 / a n - 1 / L| = |(L - a n) / (a n * L)| :=
    by field_simp
have l2 :  |(L - a n) / (a n * L)| =  |(L - a n)| / |(a
   n * L)| := by apply abs_div
have l3 : |(L - a n)| / |(a n * L)| = |(L - a n)| / (|a
   n| * |L|) := by bound
have l4 : |L - a n| = |-(a n - L)| := by ring_nf
have l5 : |-(a n - L)| = |(a n - L)| := by apply abs_neg
have l6 : |a n - L| / (|a n| * |L|) < (ε * |L| * |L| /
   2) / (|a n| * |L|) := by field_simp; bound
have l10 : |(L - a n)| / (|a n| * |L|) = |-(a n - L)| /
   (|a n| * |L|) := by rewrite [l4]; rfl
have l11 : |-(a n - L)| / (|a n| * |L|) = |(a n - L)| /
   (|a n| * |L|) := by rewrite [l5]; rfl
have l13 : ε * |L| * |L| / 2 / (|a n| * |L|) = ε * |L| /
    2 / |a n| := by field_simp
have l14 : ε * |L| / 2 / |a n| ≤ ε := by field_simp;
   bound
```

```
linarith [l1, l2, l3, l10, l11, l6, l13, l14]
```

2.5 Natural Language Proof

Proof: Assume $a(n) \to L$ with $L \neq 0$, and let $b(n) = 1/a(n)$. We must show $b(n) \to 1/L$.

First, using the theorem EventuallyGeHalfLimPos, there exists N_1 such that for all $n \geq N_1$, we have $|a(n)| \geq |L|/2 > 0$, ensuring $a(n) \neq 0$ and $b(n)$ is well-defined.

Given $\varepsilon > 0$, since $a(n) \to L$, there exists N_2 such that for all $n \geq N_2$:

$$|a(n) - L| < \frac{\varepsilon |L|^2}{2}$$

Let $N = N_1 + N_2$. For any $n \geq N$, we have:

$$\begin{aligned}
|b(n) - 1/L| &= |1/a(n) - 1/L| \\
&= |(L - a(n))/(a(n) \cdot L)| \\
&= |L - a(n)|/(|a(n)| \cdot |L|) \\
&= |a(n) - L|/(|a(n)| \cdot |L|) \\
&< \frac{\varepsilon |L|^2/2}{|a(n)| \cdot |L|} \\
&= \frac{\varepsilon |L|}{2|a(n)|} \\
&\leq \frac{\varepsilon |L|}{2 \cdot |L|/2} \\
&= \varepsilon
\end{aligned}$$

Therefore, $b(n) \to 1/L$. **QED**

2.6 Applications and Extensions

With this theorem, we now have a complete toolkit for limits of rational functions. Combined with results on sums and products, we can prove:

Quotient Theorem: If $a(n) \to L$ and $c(n) \to M$ with $M \neq 0$, then $a(n)/c(n) \to L/M$.

The proof is straightforward: first show $1/c(n) \to 1/M$ using the reciprocal theorem, then use the product theorem to show $a(n) \cdot (1/c(n)) \to L \cdot (1/M) = L/M$.

This completes the fundamental arithmetic of limits: sums, products, and quotients—the building blocks for analyzing polynomials, rational functions, and complex expressions throughout calculus and analysis.

3 By Cases: Case Analysis Without Hypotheses

When you already have a hypothesis $h : P \vee Q$, you can use `cases'` to break the goal into two subgoals. But how do you break into cases when you don't already have a hypothesis? That's where the `by_cases` tactic comes in.

3.1 The by_cases Tactic

The `by_cases` tactic has syntax `by_cases h : fact`, where `h` is your name for a new hypothesis, and `fact` is the fact claimed in the hypothesis. Calling `by_cases` creates two subgoals:

- One with the additional hypothesis `h : fact`
- One with the hypothesis `h : ¬fact`

This allows you to perform case analysis on any proposition, not just ones you already have as hypotheses.

3.2 Application: Eventually Greater than Half the Limit

We can use `by_cases` to prove a more general version of EventuallyGeHalfLimPos that doesn't require the assumption $L \neq 0$.

Theorem: If $a : \mathbb{N} \to \mathbb{R}$ converges to L (with no assumption that $L \neq 0$), then there exists N such that for all $n \geq N$, we have $|a(n)| \geq |L|/2$.

3.3 Lean Solution

```
Statement EventuallyGeHalfLim (a : ℕ → ℝ) (L : ℝ) (aToL
   : SeqLim a L) :
    ∃ N, ∀ n ≥ N, |L| / 2 ≤ |a (n)| := by
by_cases h : L = 0
use 1
intro n hn
rewrite [h]
bound
apply EventuallyGeHalfLimPos a L aToL h
```

3.4 Natural Language Proof

Proof: We proceed by cases on whether $L = 0$.

Case 1: If $L = 0$, then $|L|/2 = 0$. Since $|a(n)| \geq 0$ for all n, we can take $N = 1$, and the inequality $|L|/2 \leq |a(n)|$ holds trivially for all $n \geq N$.

Case 2: If $L \neq 0$, then we can apply the theorem EventuallyGeHalfLimPos (which requires $L \neq 0$ as a hypothesis) to obtain the desired N. **QED**

3.5 Importance

The `by_cases` tactic is essential for handling propositions where the truth value matters but isn't already known. It allows us to:

- Handle edge cases systematically
- Prove theorems with weaker hypotheses
- Apply different proof strategies depending on which case holds

This technique is ubiquitous in mathematical reasoning, from analysis to algebra to combinatorics.

4 Mathematical Induction

Induction is one of the most powerful proof techniques for statements about natural numbers. It allows us to prove infinitely many statements (one for each natural number) using only two steps: a base case and an inductive step.

4.1 The Principle of Mathematical Induction

To prove that a property $P(n)$ holds for all natural numbers n, it suffices to prove:

1. **Base Case:** $P(0)$ is true
2. **Inductive Step:** For all k, if $P(k)$ is true, then $P(k+1)$ is true

The intuition is like climbing an infinite ladder: if you can reach the first rung (base case), and if being on any rung allows you to reach the next rung (inductive step), then you can reach every rung.

4.2 The induction' Tactic

The syntax for induction in Lean is: `induction' n with k hk`. This means:

- Apply induction on the variable `n`
- Use `k` for the new dummy variable in the inductive step
- Use `hk` for the induction hypothesis on `k`

4.3 New Tool: ge_one_of_nonzero

If a natural number $n \neq 0$, then $1 \leq n$. This simple lemma is useful for handling the case when we're past the base case.

4.4 Example: Exponential Growth

Theorem: For all natural numbers n, we have $n < 2^n$.

This theorem captures the idea that exponential functions grow faster than linear functions—a fundamental fact with applications throughout mathematics and computer science.

4.5 Lean Solution

```
Statement (n : ℕ) : n < 2 ^ n := by
induction' n with k hk
norm_num
by_cases hk0 : k = 0
rewrite [hk0]
norm_num
have : 1 ≤ k := by apply ge_one_of_nonzero hk0
have f1 : k + 1 ≤ 2 * k := by bound
have f2 : 2 * k < 2 * 2 ^ k := by linarith [hk]
have f3 : 2 * 2 ^ k = 2 ^ (k + 1) := by ring_nf
linarith [f1, f2, f3]
```

4.6 Natural Language Proof

Proof: We proceed by induction on n.

Base Case ($n = 0$): We have $0 < 2^0 = 1$, which is true.

Inductive Step: Assume $k < 2^k$ (induction hypothesis). We must show $k + 1 < 2^{k+1}$.

We proceed by cases on whether $k = 0$.

If $k = 0$, then $k + 1 = 1 < 2 = 2^1 = 2^{k+1}$.

If $k \neq 0$, then $k \geq 1$. Therefore:

$$\begin{aligned} k + 1 &\leq 2k && \text{(since } k \geq 1\text{)} \\ &< 2 \cdot 2^k && \text{(by induction hypothesis)} \\ &= 2^{k+1} \end{aligned}$$

This completes the induction. **QED**

4.7 Applications

Mathematical induction is essential for proving:

- Formulas for sums: $\sum_{i=1}^{n} i = n(n+1)/2$
- Inequalities: $n! > 2^n$ for large n
- Divisibility properties: $n^3 - n$ is divisible by 6

- Recursive definitions: Fibonacci numbers, factorials, etc.
- Algorithm correctness: proving loop invariants

The combination of induction with other proof techniques (like case analysis) creates a powerful toolkit for discrete mathematics and computer science.

4.8 Key Insight

This proof demonstrates an important strategy: within an inductive proof, you may need to perform additional case analysis (using `by_cases`). The inductive hypothesis gives you leverage, but you often need to be clever about how to apply it, especially when dealing with boundary cases or when the algebra requires additional constraints.

Thm: $a_n \to L \neq 0$, $1/a_n \to 1/L$.

a) Scratchwork: Can assume $|a_n - L| < \delta$.

Want: $\left|\frac{1}{a_n} - \frac{1}{L}\right| < \varepsilon$. (after making N large).

Goal: $\left|\frac{L - a_n}{a_n \cdot L}\right| < \varepsilon$.

Eventually $|a_n| \geq |L|/2$.

Goal: $\left|\frac{a_n - L}{|L/2| \cdot |L|}\right| < \varepsilon$. ✓. So $\delta = \frac{\varepsilon |L|^2}{2}$.

---

have: aTol: $a \to L$, Lneteq: $L \neq 0$.

btq Inv a: $b = 1/a$.

Goal: $b \to 1/L$.

Intro ε hε. hε: $0 < \varepsilon$.

have f0: $0 < |L| :=$ by apply abs_pos_of_nonzero Lneteq

have f1: $0 < \varepsilon \cdot |L|^2/2 :=$ by bound

Specialize aTol $(\varepsilon |L|^2/2)$ f1

have f0': $\exists N2, \forall n \geq N2, |a_n| \geq |L|/2$:= by apply Eventually_Ge_atTop_Nonzero.

gToL: $\exists N1, \forall n \geq N1, |a_n - L| < \varepsilon \cdot |L|^2/2$.

Choose N1 hN1 using gToL

Choose N2 hN2 using f0'.

Use N1+N2.

Goal: $\forall n \geq N1+N2, \left|b_n - \frac{1}{L}\right| < \varepsilon$.

intro n hn. $n \geq N_1+N_2$.

have hn1: $N1 \leq n$:= by bound.

specialize hN1 n hn1. hN1: $|a_n - L| < \varepsilon |L|^2/2$

specialize hN2 n hn2 hN2: $|L|/2 \leq |a_n|$

rewrite [b Eq Inva n]. Goal: $\left|\frac{1}{a_n} - \frac{1}{L}\right| < \varepsilon$.

have f2: $\frac{1}{a_n} - \frac{1}{L} = \frac{L - a_n}{a_n \cdot L}$:= by field_simp.

have absa: $0 < |a_n|$:= by have [hN2, f0].

have anezero: $a_n \neq 0$:= by apply nonzero_of_abspos absa.

rewrite [f2] Goal: $\left|\frac{L-a_n}{a_n \cdot L}\right| < \varepsilon$.

have f3: $\left|\frac{L-a_n}{a_n \cdot L}\right| = \frac{|L-a_n|}{|a_n \cdot L|}$:= by apply abs_div

have f4: $|a_n \cdot L| = |a_n| \cdot |L|$:= by apply abs_mul

rewrite [f4] at f3.

rewrite [f3] Goal: $\frac{|L-a_n|}{|a_n| \cdot |L|} < \varepsilon$.

have f5: $|L - a_n| = |-(a_n - L)|$:= by ring_nf

have f6: $|-(a_n - L)| = |a_n - L|$:= by apply abs_neg,

rewrite [f5] [f6] Goal: $\frac{|a_n - L|}{|a_n| \cdot |L|} < \varepsilon$.

have f7: $\frac{|a_n - L|}{|a_n| \cdot |L|} < \left(\frac{\varepsilon \cdot |L|^2}{\varepsilon |a_n| \cdot |L|}\right)$:= by bound

have f8: $\left(\frac{\varepsilon |L|^2}{2}\right) / |a_n| \cdot |L| = \frac{\varepsilon |L|}{2|a_n|}$:= by field_simp; bound.

have f9: $\frac{\varepsilon \cdot |L|}{2|a_n|} \leq \varepsilon$:= by field_simp; bound.

linarith [f7, f8, f9].

Part c: Natural language proof that 1/ an -> 1/ L (L != 0)
let epsilon be given and assume epsilon > 0. Then epsilon |L|^2 /2 > 0. since a n -> L, we can make n large enough so that |a n - L| < epsilon |L|^2 /2. We can also make n even larger (if need be) so that |a n| >= |L|/2.
Taking n large enough for both conditions to hold, we can bound:
|1 / an - 1/ L| = |(L - an)/(an L)| < epsilon |L|^2 /2 / (L * (|L|/2)) = epsilon. Done.

Quiz: Prove by induction that for any $n : \mathbb{N}$, $n < 2^n$.

Argue by induction.

Base case n=0: $0 < 2^0 = 1$ ✓. commutativity

Scratchwork for induction: assume (hi): $n < 2^n$.

GOAL: $n+1 < 2^{n+1}$

by-cases $n \stackrel{?}{=} 0$:

- $0+1 < 2^{0+1}$ ✓.
- $n \neq 0 \Rightarrow 1 \leq n$.

$n+1 \leq 2 \cdot n$.

$< 2 \cdot 2^n = 2^{n+1}$.

(hi)

$1 \leq 2^n$.

$n+1 < 2^n + 1$

$n+1 \leq 2^n$