Abel's Limit Theorem in Isabelle/HOL

Kangfeng Ye
University of York, UK

kangfeng.ye@york.ac.uk

February 6, 2026

Abstract

This theory proves the Abel's limit theorem on power series of real numbers, and then an example is shown to use the theorem to cover the boundary cases of binomial series.

Contents

1 Abel's limit theorem on real power series

theory *Abel-Limit-Theorem*
imports *HOL−Analysis.Generalised-Binomial-Theorem*
begin

Abel's theorem or Abel's limit theorem [3] provides a crucial link between the behavior of a power series inside its interval of convergence (such as $(-1, 1)$) and its value at the boundary such as -1 or 1.

This section presents the proof of Abel's limit theorem, which relates a limit of a power series to the sum of its real coefficients, as shown below:

$$\lim_{x \to 1^-} f(x) = f(1) = \sum_{k=0}^{\infty} a_k \qquad \text{where } f(x) = \sum_{k=0}^{\infty} a_k x^k$$

if the power series has its radius of convergence equal to 1 and $\sum_{k=0}^{\infty} a_k$ converges, where a_k is the coefficient of the k-th term.

That is, $f(x)$ is continuous from the left at 1.

The proof of continuity or the limit of $f(x)$ is based on the ε-δ definition. This proof uses summation by parts or Abel transformation to express the power series $f(x)$ as a power series whose coefficients are the partial sums $(\sum_{k=0}^{n} a_k)$ of the coefficients of $f(x)$, instead of a_k. Then the new power series is split into two parts. The goal is to show that each part contributes to $\varepsilon/2$ for any x satisfying $(1 - x) < \delta$.

Several references [3, 1, 2] are used to construct this proof.

theorem *Abel-limit-theorem*:
  fixes *a* :: *nat* ⇒ *real*
  defines *f1* ≡ (λ(*x*::*real*) *n*. *a n* ∗ *x* ^ *n*)
  defines *f* ≡ (λ(*x*::*real*). ∑ *n*. *f1 x n*)
  assumes *summable-a*: *summable a* **and**
      conv-radius-1: *conv-radius a* = *1*
  shows (*f* ⟶ (∑ *n*. *a n*)) (*at-left 1*)
proof −
  — This is the partial sum of coefficients up to *n*.
  let *?s* = λ*n*. (∑ *k*≤*n*. *a k*)

  — *S* is the infinite sum of the coefficients.
  obtain *S* **where** *P-S*: (∑ *n*. *a n*) = *S*
    using *summable-a* **by** *simp*

  let *?fs-S* = λ*x n*. ((*?s n* − *S*) ∗ *x* ^ *n*)
  let *?fs-S-sum* = λ*x*. (∑ *n*. *?fs-S x n*)

  have *s-limit-S*: *?s* ⟶ *S*
    using *P-S summable-a summable-LIMSEQ'* **by** *blast*

  have *summable-f1*: ∀ *x*. *norm x* < *1* ⟶ *summable* (λ*n*. *f1 x n*)
   by (*simp only*: *f1-def*, *auto*, *rule summable-in-conv-radius*, *simp add*: *conv-radius-1*)

  — A geometric series sums to *1* / (*1* − *x*). Therefore, (*1*−*x*)∗(*1* / (*1* − *x*)) = *1*.
  have *geometric*: ∀ *x*::*real*. *norm x* < *1* ⟶ (∑ *n*. *x* ^ *n*) = *1* / (*1* − *x*)
    by (*auto simp add*: *suminf-geometric*)

  — The Cauchy product of a geometric series and a convergent power series is a power series whose coefficient for the nth term is the partial sum up to n, that is, *?s n*.
  have *cauchy-product-to-partial-sum*: (∑ *n*. *x* ^ *n*) ∗ *f x* = (∑ *n*. *?s n* ∗ *x* ^ *n*)
    if *a1*: *norm x* < *1* **for** *x* :: *real*
  proof −
    from *a1* **have** *x-lt-1*: |*x*| < *1*
      by (*simp*)

    show (∑ *n*. *x* ^ *n*) ∗ *f x* = (∑ *n*. *?s n* ∗ *x* ^ *n*)
    proof −

```
    have f0: ∀ n. ∀ i≤n. x ^ i * (a (n − i) * x ^ (n − i)) = a (n − i) * x ^ n
      apply auto[1]
      by (metis le-add-diff-inverse power-add)

    have f1: ∀ n. (∑ i≤n. x ^ i * (a (n − i) * x ^ (n − i))) = (∑ i≤n. a (n −
i) * x ^ n)
      by (auto simp: f0)

    have f2: ∀ n. (∑ i≤n. a (n − i)) = ?s n
    proof (rule allI)
      fix n::nat
      show (∑ i≤n. a (n − i)) = ?s n
        by (rule sum.reindex-bij-witness[of - λi. n − i λi. n − i]) auto
    qed

    show ?thesis
      unfolding f-def f1-def
    proof (subst Cauchy-product)
      show summable (λk. norm (x ^ k))
        by (simp add: power-abs x-lt-1)
    next
      show summable (λk. norm (a k * x ^ k))
     by (metis abs-summable-in-conv-radius conv-radius-1 ereal-less(3) real-norm-def
x-lt-1)
    next
      show (∑ k. ∑ i≤k. x ^ i * (a (k − i) * x ^ (k − i))) =
            (∑ n. sum a {..n} * x ^ n)
        by (subst f1) (use f2 in ‹simp-all flip: sum-distrib-right›)
    qed
  qed
qed

  have summable-s-n-x-n: ∀ x::real. norm x < 1 ⟶ (summable (λn. ?s n * x ^
n))
  proof (rule allI, rule impI)
    fix x::real

    assume a1: norm x < 1
    from a1 have x-lt-1: |x| < 1
      by (simp)

    have (∑ n. ?s n * x ^ n) = (∑ n. x ^ n) * f x
      using cauchy-product-to-partial-sum real-norm-def x-lt-1 by presburger

    have f0: (λn. ?s n * x ^ n) = (λn. ∑ i≤n. a i * x ^ n)
      using sum-distrib-right by blast

    have f1: ... = (λn. ∑ i≤n. (a i * x ^ i) * (x ^ (n − i)))
      apply (simp only: mult.assoc)
```

apply (*subst power-add*[*symmetric*])
by *simp*

show *summable* (λ*n*. *?s n* ∗ *x* ^ *n*)
apply (*simp only*: *f0 f1*)
apply (*rule summable-Cauchy-product*[**where** *a* = λ*n*. (*a n*) ∗ *x* ^ *n* **and** *b* = λ*n*. *x* ^ *n*])
apply (*metis abs-summable-in-conv-radius conv-radius-1 ereal-less*(*3*) *real-norm-def x-lt-1*)
by (*simp add*: *power-abs x-lt-1*)
qed

— The power series *f x* is expressed as a convex combination of the partial sums.
have *f-x-to-partial-sum*: ∀ *x*::*real*. *norm x* < *1* ⟶ *f x* = (*1* − *x*) ∗ (∑ *n*. *?s n* ∗ *x* ^ *n*)
proof (*rule allI*, *rule impI*)
fix *x*::*real*

assume *a1*: *norm x* < *1*
from *a1* **have** *x-lt-1*: |*x*| < *1*
by (*simp*)

— Rewrite *f x* because (*1* − *x*) ∗ (∑ *n*. *x* ^ *n*) = *1*.
have *f-rewrite*: *f x* = (*1* − *x*) ∗ (∑ *n*. *x* ^ *n*) ∗ *f x*
using *geometric x-lt-1* **by** *fastforce*

— According to the Cauchy product result.
then show *f x* = (*1* − *x*) ∗ (∑ *n*. *?s n* ∗ *x* ^ *n*)
using *cauchy-product-to-partial-sum mult.assoc* **by** (*metis real-norm-def x-lt-1*)
qed

— The difference between *f x* and *S*, therefore, can be expressed as a convex combination of the partial sum minus *S*. So the goal is to show RHS tends to 0 when *x* approaches 1 from left.
have *f-x-minus-S*: ∀ *x*::*real*. *norm x* < *1* ⟶ *f x* − *S* = (*1* − *x*) ∗ *?fs-S-sum x*
proof (*rule allI*, *rule impI*)
fix *x*::*real*

assume *a1*: *norm x* < *1*
from *a1* **have** *x-lt-1*: |*x*| < *1*
by (*simp*)

have *f0*: (*1* − *x*) ∗ (∑ *n*. *?s n* ∗ *x* ^ *n*) − *S* = (*1* − *x*) ∗ (∑ *n*. *?s n* ∗ *x* ^ *n*) − (*1* − *x*) ∗ *S* ∗ (∑ *n*. *x* ^ *n*)
apply (*simp add*: *geometric*)
using *geometric x-lt-1* **by** *auto*

have *f1*: ... = (*1* − *x*) ∗ ((∑ *n*. *?s n* ∗ *x* ^ *n*) − (∑ *n*. *S* ∗ *x* ^ *n*))

apply (*subst suminf-mult*)
apply (*rule summable-geometric*)
apply (*simp add*: *x-lt-1*)
by (*simp add*: *right-diff-distrib*)

show $f\ x - S = (1 - x) * ?fs\text{-}S\text{-}sum\ x$
apply (*simp only*: *f-x-to-partial-sum a1*)
apply (*simp only*: *f0 f1*)
apply (*subst suminf-diff*)
using *real-norm-def summable-s-n-x-n x-lt-1* **apply** *presburger*
apply (*rule summable-mult*)
apply (*simp add*: *x-lt-1*)
by (*simp add*: *left-diff-distrib'*)
qed

have *summable-norm-s-S*: $\forall x{::}real.\ norm\ x < 1 \longrightarrow summable\ (\lambda n{::}nat.\ norm\ (?s\ n - S) * (norm\ x)\ \hat{}\ (n))$
proof (*rule allI*, *rule impI*)
fix *x*::*real*
assume *x-lt-1*: $norm\ x < 1$
obtain *M* **where** *P-m*: $\forall n.\ norm\ (?s\ n) \leq M$
using *convergent-imp-bounded*[*of ?s*] **by** (*metis UNIV-I bounded-iff imageI s-limit-S*)
have $\forall n.\ norm\ (?s\ n - S) * (norm\ x)\ \hat{}\ (n) \leq (M + norm\ S) * (norm\ x)\hat{}n$
proof (*rule allI*)
fix *n* :: *nat*
have $norm\ (?s\ n - S) * (norm\ x)\ \hat{}\ (n) \leq (norm\ (?s\ n) + norm\ S) * (norm\ x)\ \hat{}\ (n)$
by (*simp add*: *mult-mono*)
also have ... $\leq (M + norm\ S) * (norm\ x)\hat{}n$
by (*metis P-m add.commute add-le-cancel-left mult.commute mult-left-mono norm-ge-zero norm-power*)
finally show $norm\ (?s\ n - S) * (norm\ x)\ \hat{}\ (n) \leq (M + norm\ S) * (norm\ x)\ \hat{}\ n$
by *blast*
qed
moreover have *summable* $(\lambda n.\ (M + norm\ S) * (norm\ x)\hat{}n)$
using *x-lt-1* **by** (*simp add*: *summable-mult summable-geometric*)
ultimately show *summable* $(\lambda n{::}nat.\ norm\ (?s\ n - S) * norm\ x\ \hat{}\ n)$
using *summable-comparison-test*[*of* $\lambda n.\ norm\ (?s\ n - S) * (norm\ x)\ \hat{}\ (n)$ $\lambda n.\ (M + norm\ S) * (norm\ x)\hat{}n$]
by *fastforce*
qed

have *summable-norm-fs-S*: $\forall x{::}real.\ norm\ x < 1 \longrightarrow summable\ (\lambda n.\ norm\ (?fs\text{-}S\ x\ n))$
proof (*rule allI*, *rule impI*)
fix *x*::*real*

assume *x-lt-1*: *norm x < 1*
obtain *M* **where** *P-m*: $\forall n.$ *norm* (*?s n*) $\leq M$
using *convergent-imp-bounded*[*of ?s*] **by** (*metis UNIV-I bounded-iff imageI s-limit-S*)
have $\forall n.$ *norm* (*?fs-S x n*) $\leq (M +$ *norm S*$) *$ (*norm x*)^n
proof (*rule allI*)
fix *n* :: *nat*
have *norm* ((*?s n* − *S*) ∗ *x* ^ *n*) ≤ *norm* ((*?s n* − *S*)) ∗ *norm* (*x* ^ *n*)
using *norm-mult-ineq* **by** *blast*
also have ... ≤ (*norm* (*?s n*) + *norm S*) ∗ *norm* (*x* ^ *n*)
by (*simp add*: *mult-mono*)
also have ... ≤ (*M* + *norm S*) ∗ (*norm x*)^*n*
by (*metis P-m add.commute add-le-cancel-left mult.commute mult-left-mono norm-ge-zero norm-power*)
finally show *norm* ((*?s n* − *S*) ∗ *x* ^ *n*) ≤ (*M* + *norm S*) ∗ *norm x* ^ *n*
by *blast*
qed
moreover have *summable* (λ*n*. (*M* + *norm S*) ∗ (*norm x*)^*n*)
using *x-lt-1* **by** (*simp add*: *summable-mult summable-geometric*)
ultimately show *summable* (λ*n*. *norm* (*?fs-S x n*))
using *summable-comparison-test*[*of* λ*n*. *norm* (*?fs-S x n*) λ*n*. (*M* + *norm S*) ∗ (*norm x*)^*n*]
by *fastforce*
qed

— Use the *ε*-*δ* definition of a continuous function or a limit.
have *S-is-f-limit-from-left*: (*f* ⟶ *S*) (*at-left* (*1*))
proof (*simp only*: *tendsto-iff eventually-at-left-field*, *simp only*: *dist-norm*, *rule allI*, *rule impI*)
— *r* is the difference of function *f x* from *f 1*
fix *r*::*real*
assume *r-gt-0*: (*0*::*real*) < *r*

— Try to make both tail and head parts contribute r/2, so finally r.
define *e* **where** *e* = *r*/*2*

have *e-gt-0*: *0* < *e*
by (*simp add*: *r-gt-0 e-def*)

— *M* is not necessary to be positive and can be 0.
obtain *M* **where** *P-M*: (∀ *n* ≥ *M*. *norm* (*?s n* − *S*) < *e*)
using *s-limit-S LIMSEQ-iff e-gt-0 real-norm-def* **by** *metis*

— Make *N* be positive, which will be used later to ensure *norm x* ^ *N* < *1*
define *N* **where** *N* = *M* + *1*

— *C* is the sum of differences up to *N*.
define *C* **where** *C* = (∑ *k*<*N*. *norm* (*?s k* − *S*))

have *P-N*: $(\forall\, n \geq N.\ norm\ (?s\ n - S) < e)$
unfolding *N-def* **using** *P-M* **by** *simp*

— Split the sum into two parts based on its index: $\{0..N-1\}$ and $\{N..\infty\}$, also called the head part and the tail part.
have *fs-S-split*: $\forall\, x{::}real.\ norm\ x < 1 \longrightarrow (1 - x) * ?fs\text{-}S\text{-}sum\ x = (1 - x) * (\sum n < N.\ ?fs\text{-}S\ x\ n) + (1 - x) * (\sum n.\ ?fs\text{-}S\ x\ (n + N))$
proof (*rule allI*, *rule impI*)
fix *x*::*real*

assume *a1*: *norm x* < *1*
from *a1* **have** *x-lt-1*: |*x*| < *1*
by (*simp*)

have $(\sum n.\ ?fs\text{-}S\ x\ n) = (\sum n{<}N.\ ?fs\text{-}S\ x\ n) + (\sum n.\ ?fs\text{-}S\ x\ (n + N))$ (**is** ... = *?fs-S-sum-hd* + *?fs-S-sum-tl*)
apply (*subst suminf-split-initial-segment*[**where** *k* = *N*])
using *summable-norm-fs-S real-norm-def summable-norm-cancel x-lt-1* **apply** *fastforce*
by *linarith*

then show $(1 - x) * (\sum n.\ ?fs\text{-}S\ x\ n) = (1 - x) * (\sum n{<}N.\ ?fs\text{-}S\ x\ n) + (1 - x) * (\sum n.\ ?fs\text{-}S\ x\ (n + N))$
by (*simp add*: *distrib-left*)
qed

— For the tail part, it is less than *e*.
have *fs-S-tail*: $\forall\, x{::}real.\ 0 < x \wedge norm\ x < 1 \longrightarrow norm\ ((1 - x) * (\sum n.\ ?fs\text{-}S\ x\ (n + N))) < e$
proof (*rule allI*, *rule impI*)
fix *x*::*real*
assume *x-lt-1*: (*0*::*real*) < *x* ∧ *norm x* < *1*
have *x-N-le-1*: *norm x* ^ *N* < *1*
using *power-Suc-less-one P-N N-def x-lt-1* **by** *fastforce*
have $norm\ ((1 - x) * (\sum n.\ ?fs\text{-}S\ x\ (n + N))) \leq norm\ ((1 - x)) * norm\ (\sum n.\ ?fs\text{-}S\ x\ (n + N))$
using *norm-mult-ineq* **by** *blast*
also have ... $\leq (1 - x) * (\sum n.\ norm\ (?fs\text{-}S\ x\ (n + N)))$
apply (*subgoal-tac norm* (*1*−*x*) = *1*−*x*)
apply (*subgoal-tac norm* $(\sum n.\ ?fs\text{-}S\ x\ (n + N)) \leq (\sum n.\ norm\ (?fs\text{-}S\ x\ (n + N))))$
subgoal by (*simp add*: *mult-mono*)
apply (*subst summable-norm*)
apply (*subst summable-iff-shift*)
using *summable-norm-fs-S x-lt-1* **apply** *blast*
apply *simp*
using *x-lt-1* **by** *fastforce*
also have ... $\leq (1 - x) * (\sum n.\ norm\ (?s\ (n + N) - S) * (norm\ x)\ \hat{}\ (n + N))$

```
      by (smt (z3) norm-mult norm-power suminf-cong)
    also have ... ≤ (1 − x) * (∑ n. e * (norm x) ^ (n + N))
      apply (rule mult-mono)
        apply simp
       apply (rule suminf-le)
      apply (smt (verit) P-N le-add2 mult-right-mono norm-ge-zero zero-le-power)
        apply (subst summable-iff-shift)
        using summable-norm-s-S x-lt-1 apply blast
       apply (subst summable-iff-shift)
       using x-lt-1 apply force
      using x-lt-1 apply auto[1]
      by (smt (z3) calculation real-norm-def x-lt-1 zero-le-mult-iff)
    also have ... = (1 − x) * e * (norm x) ^ N * (∑ n. (norm x) ^ (n))
      apply (subst suminf-mult)
       using x-lt-1 apply force
      apply (simp only: power-add)
      apply (subgoal-tac (∑ n::nat. norm x ^ n * norm x ^ N) = norm x ^ N *
(∑ n::nat. norm x ^ n))
       apply simp
      apply (subst suminf-mult[symmetric])
       using x-lt-1 apply auto[1]
      by (meson mult.commute)
    also have ... = (1 − x) * e * (norm x) ^ N * 1 / (1 − norm x)
      apply (subst suminf-geometric)
       using x-lt-1 apply fastforce
      using times-divide-eq-right by blast
    also have ... = e * (norm x) ^ N
      using x-lt-1 by fastforce
    also have ... < e
      using x-N-le-1 using e-gt-0 by force
    finally show norm ((1 − x) * (∑ n. ?fs-S x (n + N))) < e
      by blast
  qed

  — For the head part, it is bounded by C.
  have fs-S-head: ∀ x::real. 0 < x ∧ norm x < 1 ⟶ norm ((1 − x) * (∑ n <
N. ?fs-S x n)) ≤ (1−x)*C
  proof (rule allI, rule impI)
    fix x::real
    assume x-lt-1: (0::real) < x ∧ norm x < 1
     have norm ((1 − x) * (∑ n < N. ?fs-S x n)) ≤ norm ((1 − x)) * norm
(∑ n < N. ?fs-S x n)
      using norm-mult-ineq by blast
    also have ... ≤ (1 − x) * (∑ n < N. norm (?fs-S x (n)))
      apply (subgoal-tac norm (1−x) = 1−x)
       apply (subgoal-tac norm (∑ n<N. ?fs-S x (n)) ≤ (∑ n<N. norm (?fs-S x
(n))))
        apply (simp add: mult-mono)
       using norm-sum apply blast
```

```
    using x-lt-1 by fastforce
  also have ... ≤ (1 − x) ∗ (∑ n<N. norm (?s (n) − S) ∗ (norm x) ^ (n))
    by (smt (verit) norm-mult norm-power sum.cong)
  also have ... ≤ (1 − x) ∗ (∑ n<N. norm (?s (n) − S))
    apply (rule mult-mono)
    subgoal by simp
     apply (rule sum-le-included[where i = λx. x])
    subgoal by simp
    subgoal by simp
    subgoal by simp
     apply (smt (verit) mult-left-le power-le-one-iff real-norm-def x-lt-1)
    using x-lt-1 apply force
    by (simp add: sum-nonneg)
  finally show norm ((1 − x) ∗ (∑ n < N. ?fs-S x n)) ≤ (1−x)∗C
    by (simp add: C-def)
qed

have C-nonneg: C ≥ 0
  by (simp add: C-def N-def)

show ∃ b<1::real. ∀ y>b. y < (1::real) ⟶ norm (f y − S) < r
proof (cases C = 0)
  case True
  then show ?thesis
    — Any x between 0 and 1 because the head part is 0 and only consider the
tail part
    proof (intro exI[of - 0.9])
      show (9::real) / (10::real) < 1 ∧ (∀ y>(9::real) / (10::real). y < 1 ⟶
norm (f y − S) < r)
      proof (rule conjI)
        show (9::real) / (10::real) < 1
          by simp
        show ∀ y>(9::real) / (10::real). y < 1 ⟶ norm (f y − S) < r
        proof (rule allI, rule impI, rule impI)
          fix y::real
          assume y-gt-0: (9::real) / (10::real) < y
          assume y-lt-1: y < (1::real)

          have norm ((1 − y) ∗ (∑ n::nat<N. (?s n − S) ∗ y ^ n) +
                (1 − y) ∗ (∑ n::nat. (?s (n + N) − S) ∗ y ^ (n + N))) < r
          proof (rule norm-triangle-lt)
            have norm ((1 − y) ∗ (∑ n < N. ?fs-S y n)) ≤ (1−y)∗C
              apply (subst fs-S-head)
               using y-gt-0 y-lt-1 apply force
              by simp
            also have ... = 0
              by (simp add: True)
            also have head-0: norm ((1 − y) ∗ (∑ n < N. ?fs-S y n)) = 0
              using calculation by force
```

```
          also have tail-lt-e: norm ((1 − y) ∗ (∑ n. ?fs-S y (n + N))) < e
            apply (subst fs-S-tail)
             using y-gt-0 y-lt-1 apply force
            by simp
          finally show norm ((1 − y) ∗ (∑ n < N. ?fs-S y n)) +
                       norm ((1 − y) ∗ (∑ n. ?fs-S y (n + N))) < r
            using e-def e-gt-0 head-0 tail-lt-e by linarith
        qed
        then show norm (f y − S) < r
          apply (subst f-x-minus-S)
           using y-gt-0 y-lt-1 apply simp
          apply (subst fs-S-split)
           using y-gt-0 y-lt-1 apply simp
           by blast
       qed
      qed
    qed
   next
    case False
    then show ?thesis
    — This witness is to ensure (1−x)∗C ≤ e.
    proof (intro exI[of - 1 − min (e/C) 1])
     show 1 − min (e / C) 1 < 1 ∧ (∀ y>1 − min (e / C) 1. y < 1 ⟶ norm
(f y − S) < r)
      proof (rule conjI)
       show 1 − min (e / C) 1 < 1
        using C-nonneg r-gt-0 False e-gt-0 by fastforce
       show ∀ y>1 − min (e / C) 1. y < 1 ⟶ norm (f y − S) < r
       proof (rule allI, rule impI, rule impI)
        fix y::real
        assume y-gt-0: (1::real) − min (e / C) (1::real) < y
        assume y-lt-1: y < (1::real)

        have norm ((1 − y) ∗ (∑ n < N. ?fs-S y n)) ≤ (1−y)∗C
          apply (subst fs-S-head)
           using y-gt-0 y-lt-1 apply force
          by simp
        also have ... ≤ e
          by (smt (verit) C-nonneg False e-def pos-divide-less-eq y-gt-0)
        also have tail-lt-e: norm ((1 − y) ∗ (∑ n. ?fs-S y (n + N))) < e
          apply (subst fs-S-tail)
           using y-gt-0 y-lt-1 apply force
          by simp
        show norm (f y − S) < r
          apply (subst f-x-minus-S)
           using y-lt-1 y-gt-0 apply force
          apply (subst fs-S-split)
           using y-lt-1 y-gt-0 apply force
          apply (rule norm-triangle-lt)
```

```
          using calculation e-def tail-lt-e by linarith
        qed
      qed
    qed
  qed
qed

show ?thesis
  using P-S S-is-f-limit-from-left by blast
qed

lemma filterlim-at-right-at-left-eq:
  shows ((λx. f (−x)) ⟶ l) (at-right (−1)) ⟷ ((λx. f (x)) ⟶ l) (at-left (1::real))
  apply (rule iffI)
   apply (simp add: at-left-minus)
   apply (simp add: filterlim-filtermap)
  apply (subst at-right-minus)
  by (simp add: filterlim-filtermap)
```

Abel's limit theorem is also suitable for continuous from the right at -1.

```
corollary Abel-limit-theorem':
  fixes a :: nat ⇒ real
  defines f1 ≡ (λ(x::real) n. a n * x ^ n)
  defines f ≡ (λ(x::real). ∑ n. f1 x n)
  assumes summable-a: summable a and
      conv-radius-1: conv-radius a = 1
  shows ((λx. f (−x)) ⟶ (∑ n. a n)) (at-right (−1))
  apply (simp add: filterlim-at-right-at-left-eq)
  using assms Abel-limit-theorem by blast

end
```

2 Example application: boundary cases of binomial theorem

```
theory Binomial-Sqrt-Series-Boundary
  imports
    Abel-Limit-Theorem
    Catalan-Numbers.Catalan-Numbers
    HOL−Real-Asymp.Real-Asymp
begin
```

Newton's generalized binomial theorem is applicable to $|x| < 1$ as seen from this $|?z| < 1 \Longrightarrow (\lambda n.\ (1\ /\ 2\ gchoose\ n) * ?z^n)\ sums\ sqrt\ (1 + ?z)$. However, it doesn't apply to the boundary cases where $|x| = 1$ or $|x| = -1$. Here, Abel's limit theorem is applied to establish the binomial theorem for the boundary cases.

2.1 Binomial series

lemma *binomial-sqrt-series*:
fixes x :: *real*
assumes $|x| < 1$
shows *suminf* $(\lambda n.\ ((1/2)\ gchoose\ n) * x\ \hat{}\ n) = sqrt\ (1 + x)$
apply (*subst sums-unique*[**where** $s = sqrt\ (1 + x)$ **and** $f = (\lambda n.\ ((1/2)\ gchoose\ n) * x\ \hat{}\ n)$])
apply (*rule sqrt-series*[**where** $z = x$])
using *assms* **apply** *blast*
by *simp*

The generalized binomial coefficient $a\ gchoose\ n$ where $a = \frac{1}{2}$ can also be rewritten as an expression including a Catalan numbers. This is used to prove its summability using the property of Catalan numbers.

lemma *gbinomial-1-2-catalan*: $((1/2)\ gchoose\ (Suc\ n)) = ((-1)\hat{}(n)/(2\hat{}(2*n+1))) * real\ (catalan\ n)$
by (*subst catalan-closed-form-gbinomial*) (*simp add*: *power-mult power-minus'*)

lemma *gbinomial-1-2-catalan'*: $((1/2)\ gchoose\ (Suc\ n)) = ((-1)\hat{}n/2) * (1/4\hat{}n) * real\ (catalan\ n)$
by (*subst gbinomial-1-2-catalan*) (*simp-all add*: *power-mult*)

Rewrite the generalized binomial coefficient $a\ gchoose\ n$ where $a = \frac{1}{2}$ as a binomial coefficient.

lemma *gbinomial-1-2-simp*:
$((1/2)\ gchoose\ (Suc\ n)) = ((-1)\hat{}n\ /\ real\ (2\hat{}(2*n+1) * (Suc\ n))) * ((2*n)\ choose\ n)$
by (*subst gbinomial-1-2-catalan*, *subst of-nat-catalan-closed-form*)
(*auto simp*: *algebra-simps*)

lemma *summable-real-powr-iff'*: *summable* $(\lambda n.\ 1\ /\ of\text{-}nat\ n\ powr\ s :: real) \longleftrightarrow s > 1$
apply (*subgoal-tac* $\forall\ n.\ 1\ /\ of\text{-}nat\ n\ powr\ s = of\text{-}nat\ n\ powr\ (-s)$)
apply (*simp*)
using *summable-real-powr-iff* **apply** *auto*[*1*]
by (*simp add*: *powr-minus-divide*)

lemma *summable-1-2-gchoose*: *summable* $(\lambda n.\ ((1::real)/2)\ gchoose\ n)$
proof −
have *f0*: $(\lambda n.\ ((1/2)\ gchoose\ (Suc\ n))) \sim[at\text{-}top] (\lambda n.\ (((-1)\hat{}n/2) * (1/4\hat{}n)) * (4\hat{}n\ /\ ((sqrt\ pi * n\ powr\ (3/2)))))$
apply (*simp only*: *gbinomial-1-2-catalan'*)
apply (*subst asymp-equiv-mult*)
using *asymp-equiv-refl* **apply** *blast*
using *catalan-asymptotics* **apply** *blast*
by *simp*
have *f1*: $... = (\lambda n.\ (-1)\hat{}n\ /\ (2 * (sqrt\ pi * n\ powr\ (3/2))))$

```
  by auto
 have f2: ... = (λn. 1 / (2 * (sqrt pi)) * ((−1)^n / (n powr (3/2)))) (is - = ?g)
  by auto

 have summable-g: summable ?g
 proof (rule summable-mult)
  have f1: ∀ n. (− (1::real)) ^ n / real n powr ((3::real) / (2::real)) =
   (− (1::real)) ^ n * real n powr (− ((3::real) / (2::real)))
   using divide-powr-uminus by presburger

   have f2: summable (λn::nat. − ((− (1::real)) ^ n * real (n + 1) powr (−
((3::real) / (2::real)))))
   apply (rule summable-minus)
   apply (rule summable-Leibniz')
    apply (subst tendsto-neg-powr)
   subgoal by simp
    using filterlim-real-sequentially
   apply (metis filterlim-add-const-nat-at-top filterlim-sequentially-iff-filterlim-real)
    subgoal by simp
   subgoal by simp
   by (simp add: powr-mono2')

  have f3: summable (λn::nat. ((− (1::real)) ^ (n + 1) * real (n + 1) powr (−
((3::real) / (2::real)))))
   using f2 by simp

  show summable (λn::nat. (− (1::real)) ^ n / real n powr ((3::real) / (2::real)))
   apply (simp only: f1)
   apply (subst summable-Suc-iff[symmetric])
   using f3 Suc-eq-plus1 by presburger
 qed

 have summable-norm-g: summable (λn. norm(?g n))
 proof −
  have f0: ∀ n. norm(?g n) = ((1::real) / ((2::real) * sqrt pi) * (1 / real n powr
((3::real) / (2::real))))
   by auto
  show ?thesis
   apply (simp only: f0)
   apply (rule summable-mult)
   apply (subst summable-real-powr-iff')
   by simp
 qed

 show ?thesis
  apply (subst summable-Suc-iff[symmetric])
  apply (subst summable-comparison-test-bigo[where g = ?g])
   apply (simp only: summable-norm-g)
  apply (rule asymp-equiv-imp-bigo)
```

```
    using f0 f1 f2 apply metis
   by simp
qed

lemma gbinomial-1-2-gchoose-sum-sqrt-2:
  shows (∑ n. (((1::real) / (2::real) gchoose n))) = sqrt 2 (is (∑ n. ?f-1 n) = -)
proof −
  let ?f = λx. (∑ n. ((((1::real) / (2::real) gchoose n)) * x ^ n))

  — Inside the disk: expansion gives sqrt(1+x)
  have eq-inside: ⋀x. abs x < 1 ⟹ ?f (x) = sqrt (1+x)
    using sqrt-series sums-unique by force

  have (?f ⟶ sqrt (2)) (at-left (1))
  proof −
    have ((λx. sqrt (1+x)) ⟶ sqrt 2) (at-left (1))
    proof (intro tendsto-intros)
      have ((+) (1::real) ⟶ 1 + 1) (at-left 1)
        using tendsto-add-const-iff by fastforce
      then show ((+) (1::real) ⟶ 2) (at-left 1)
        by simp
    qed
    moreover have eventually (λx. ?f (x) = sqrt(1+x)) (at-left (1))
      apply(subst eventually-at)
      apply (rule exI[of - 0.1])
      apply (auto simp: dist-real-def)[1]
      using eq-inside by force
    ultimately show ?thesis
      by (simp add: filterlim-cong)
  qed
  hence lim: (?f ⟶ sqrt 2) (at-left (1)) by simp

  have lim-by-abel-from-left: (?f ⟶ (∑ n. ?f-1 n)) (at-left (1))
    apply (subst Abel-limit-theorem)
      using summable-1-2-gchoose apply simp
     apply (subst conv-radius-gchoose)
     apply (smt (verit, best) Nats-cases field-sum-of-halves nat-less-real-le of-nat-0
of-nat-0-less-iff)
    by auto

  from lim lim-by-abel-from-left show ?thesis
    apply (subst tendsto-unique[where f = ?f and a = (∑ n. ?f-1 n)
          and F = (at-left (1)) and b = sqrt 2])
      using trivial-limit-at-left-real apply blast
     apply blast
    apply blast
   by simp
qed
```

2.2 Alternating series

lemma *gbinomial-ratio-limit'*:
fixes *a* :: *'a* :: *real-normed-field*
assumes $a \notin \mathbb{N}$
shows $(\lambda n.\ ((a\ gchoose\ n) * (-1)\ \hat{}\ n)\ /\ ((a\ gchoose\ Suc\ n) * (-1)\ \hat{}\ (Suc\ n)))$
$\longrightarrow 1$
proof −
have $(\lambda n.\ ((a\ gchoose\ n) * (-1)\ \hat{}\ n)\ /\ ((a\ gchoose\ Suc\ n) * (-1)\ \hat{}\ (Suc\ n)))$
$= (\lambda n.\ -\ ((a\ gchoose\ n)\ /\ (a\ gchoose\ Suc\ n)))$
by *auto*
then show *?thesis*
using *gbinomial-ratio-limit assms tendsto-minus-cancel-left* **by** *fastforce*
qed

lemma *conv-radius-gchoose-alternating*:
fixes *a* :: *'a* :: {*real-normed-field*,*banach*}
assumes $a \notin \mathbb{N}$
shows *conv-radius* $(\lambda n{::}nat.\ (a\ gchoose\ n) * (-1)\ \hat{}\ n) = (1{::}ereal)$
proof −
from *tendsto-norm*[*OF gbinomial-ratio-limit'*]
have *conv-radius* $(\lambda n{::}nat.\ (a\ gchoose\ n) * (-1)\ \hat{}\ n) = 1$
apply (*intro conv-radius-ratio-limit-nonzero*[*of - 1*])
subgoal by (*simp add*: *norm-divide*)
subgoal by (*simp add*: *norm-divide*)
apply (*simp add*: *norm-divide*[*symmetric*])
using *assms* **by** *blast*
then show *?thesis* **by** *blast*
qed

lemma *summable-1-2-gchoose-alternating*:
summable $(\lambda n{::}nat.\ (1\ /\ 2\ gchoose\ n) * (-1)\ \hat{}\ n :: real)$ (**is** *summable ?f*)
proof −
have *f0*: $(\lambda n.\ ((1/2)\ gchoose\ (Suc\ n))) \sim[at\text{-}top]$
$(\lambda n.\ (((-1)\hat{}n/2) * (1/4\hat{}n)) * (4\hat{}n\ /\ ((sqrt\ pi * n\ powr\ (3/2)))))$
apply (*simp only*: *gbinomial-1-2-catalan'*)
apply (*subst asymp-equiv-mult*)
using *asymp-equiv-refl* **apply** *blast*
using *catalan-asymptotics* **apply** *blast*
by *simp*
have *f1*: ... $= (\lambda n.\ (-1)\hat{}n\ /\ (2 * (sqrt\ pi * n\ powr\ (3/2))))$
by *auto*
have *f2*: ... $= (\lambda n.\ 1\ /\ (2 * (sqrt\ pi)) * ((-1)\hat{}n\ /\ (n\ powr\ (3/2))))$ (**is** - = *?g*)
by *auto*
have *f3*: $(\lambda n.\ ?g\ (n) * (-\ (1{::}real))\ \hat{}\ (Suc\ n)) =$
$(\lambda n.\ (-1\ /\ (2 * (sqrt\ pi))) * (1\ /\ (n\ powr\ (3/2))))$ (**is** - = *?g1*)
by *auto*
have *f4*: $(\lambda n.\ ?f\ (Suc\ n)) \sim[at\text{-}top]\ (\lambda n.\ ?g\ (n) * (-\ (1{::}real))\ \hat{}\ (Suc\ n))$ (**is** - $\sim[at\text{-}top]$ *?g1*)
apply (*subst asymp-equiv-mult*)

```
    using f0 f1 f2 subgoal by auto
   using asymp-equiv-refl apply blast
  by simp

 have summable-g: summable ?g
 proof (rule summable-mult)
  have f1: ∀ n. (− (1::real)) ^ n / real n powr ((3::real) / (2::real)) =
   (− (1::real)) ^ n * real n powr (− ((3::real) / (2::real)))
   using divide-powr-uminus by presburger

   have f2: summable (λn::nat. − ((− (1::real)) ^ n * real (n + 1) powr (−
((3::real) / (2::real)))))
   apply (rule summable-minus)
   apply (rule summable-Leibniz')
    apply (subst tendsto-neg-powr)
     subgoal by simp
    using filterlim-real-sequentially
   apply (metis filterlim-add-const-nat-at-top filterlim-sequentially-iff-filterlim-real)
    subgoal by simp
   subgoal by simp
   by (simp add: powr-mono2')

  have f3: summable (λn::nat. ((− (1::real)) ^ (n + 1) * real (n + 1) powr (−
((3::real) / (2::real)))))
   using f2 by simp

  show summable (λn::nat. (− (1::real)) ^ n / real n powr ((3::real) / (2::real)))
   apply (simp only: f1)
   apply (subst summable-Suc-iff[symmetric])
   using f3 Suc-eq-plus1 by presburger
 qed

 have summable-g1: summable ?g1
  apply (simp only: f3)
  apply (rule summable-mult)
  apply (subgoal-tac (λn::nat. (1::real) / real n powr ((3::real) / (2::real))) =
     (λn::nat. real n powr (− (3::real) / (2::real))))
  subgoal by (simp add: summable-real-powr-iff)
  by (simp add: inverse-eq-divide powr-minus)

 have summable-norm-g1: summable (λn. norm (?g1 n))
  apply (simp add: f3)
  apply (subgoal-tac (λn::nat. (1::real) / ((2::real) * sqrt pi * real n powr (3 /
2))) =
    (λn::nat. (1::real) / ((2::real) * sqrt pi) * real n powr (−3 / 2)))
  subgoal by (simp add: summable-real-powr-iff)
  by (simp add: inverse-eq-divide powr-minus)

 show ?thesis
```

```
    apply (subst summable-Suc-iff[symmetric])
    apply (subst summable-comparison-test-bigo[where g = ?g1])
     using summable-norm-g1 apply blast
    apply (rule asymp-equiv-imp-bigo)
    using f4 apply blast
   by simp
qed

lemma gbinomial-1-2-gchoose-alternating-sum-0:
  shows (∑ n. ((1/2 gchoose n) * (- (1::real)) ^ n)) = 0 (is (∑ n. ?f-1 n) = 0)
proof −
  let ?f = λx. (∑ n. ((((1::real) / (2::real) gchoose n) * (−1) ^ n) * x ^ n))

  have f0: ∀ x. ?f x = (∑ n. ((((1::real) / (2::real) gchoose n) * (−x) ^ n)))
    by (metis (no-types, lifting) more-arith-simps(11) power-minus suminf-cong)

  — Inside the disk: expansion gives sqrt(1+x)
  have eq-inside: ⋀x. abs x < 1 ⟹ ?f (−x) = sqrt (1−(−x))
    apply (simp only: f0)
    using sqrt-series sums-unique by force

  have ((λx. ?f (−x)) ⟶ sqrt (1 + (−1))) (at-right (−1))
  proof −
    have ((λx. sqrt (1+x)) ⟶ sqrt 0) (at-right (−1))
      apply (intro tendsto-intros)
      using filterlim-at-right-to-0 by fastforce
    moreover have eventually (λx. ?f (−x) = sqrt(1+x)) (at-right (−1))
      apply (subst eventually-at)
      apply (rule exI[of - 0.1])
      apply (auto simp: dist-real-def)[1]
      using eq-inside by force
    ultimately show ?thesis
      by (simp add: filterlim-cong)
  qed
  hence lim: ((λx. ?f (−x)) ⟶ 0) (at-right (−1)) by simp

  have lim-by-abel-from-right: ((λx. ?f (−x)) ⟶ (∑ n. ?f-1 n)) (at-right (−1))
    apply (subst Abel-limit-theorem')
     subgoal using summable-1-2-gchoose-alternating by simp
    apply (subst conv-radius-gchoose-alternating[where a = 1/2::real])
     apply (smt (verit, ccfv-threshold) Multiseries-Expansion.intyness-simps(1)
Nats-cases One-nat-def
      Rings.ring-distribs(2) divide-inverse inverse-eq-divide nat-less-real-le
     nonzero-mult-div-cancel-left of-nat-0-less-iff one-power2 plus-1-eq-Suc times-divide-eq-right)
    by auto

  from lim lim-by-abel-from-right show ?thesis
    apply (subst tendsto-unique[where f = (λx. ?f (−x)) and a = (∑ n. ?f-1 n)
        and F = (at-right (−1)) and b = 0])
```

```
    using trivial-limit-at-right-real apply blast
    apply blast
    apply blast
    by simp
qed
```

2.3 Binomial sqrt series with the boundary cases

This lemma incorporates the boundary values where $x = 1$ and $x = -1$.

```
theorem binomial-sqrt-series':
  assumes |x| ≤ (1 :: real)
  shows suminf (λn. ((1/2) gchoose n) * x ^ n) = sqrt (1 + x)
proof (cases |x| < 1)
  case True
  then show ?thesis using binomial-sqrt-series by presburger
next
  case abs-x-1: False
  then show ?thesis
  proof (cases x = 1)
    case True
    then show ?thesis
      by (simp add: gbinomial-1-2-gchoose-sum-sqrt-2)
  next
    case False
    then have x = -1
      using abs-x-1 assms by linarith
    then show ?thesis by (simp add: gbinomial-1-2-gchoose-alternating-sum-0)
  qed
qed

end
```

References

[1] Proof of Abel's limit theorem — planetmath.org. https://planetmath.org/proofofabelslimittheorem. [Accessed 11-11-2025].

[2] F. Holland. Abel's limit theorem, its converse, and multiplication formulae for $\Gamma(x)$. *Irish Math. Soc. Bull.*, 0089:57–64, 2022.

[3] Wikipedia contributors. Abel's theorem. [Accessed 11-11-2025]. URL: https://en.wikipedia.org/wiki/Abel%27s_theorem.