

Abel's Limit Theorem in Isabelle/HOL

Kangfeng Ye
University of York, UK

kangfeng.ye@york.ac.uk

February 10, 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	1
2	Example application: boundary cases of binomial theorem	11
2.1	Binomial series	12
2.2	Alternating series	15
2.3	Binomial sqrt series with the boundary cases	18

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:

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

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  $a :: \text{nat} \Rightarrow \text{real}$ 
defines  $f1 \equiv (\lambda(x::\text{real})\ n.\ a\ n * x \wedge n)$ 
defines  $f \equiv (\lambda(x::\text{real}).\ \sum n.\ f1\ x\ n)$ 
assumes summable-a: summable  $a$  and
           conv-radius-1: conv-radius  $a = 1$ 
shows  $(f \longrightarrow (\sum n.\ a\ n))\ (at\text{-}left\ 1)$ 
proof —
  — This is the partial sum of coefficients up to  $n$ .
  let  $?s = \lambda n.\ (\sum k \leq n.\ a\ k)$ 

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

  let  $?fs\text{-}S = \lambda x\ n.\ ((?s\ n - S) * x \wedge n)$ 
  let  $?fs\text{-}S\text{-}sum = \lambda x.\ (\sum n.\ ?fs\text{-}S\ x\ n)$ 

  have s-limit-S:  $?s \longrightarrow S$ 
    using  $P$ - $S$  summable-a summable-LIMSEQ' by blast

  have summable-f1:  $\forall x.\ \text{norm } x < 1 \longrightarrow \text{summable } (\lambda n.\ f1\ x\ n)$ 
    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:  $\forall x::\text{real}.\ \text{norm } x < 1 \longrightarrow (\sum n.\ x \wedge n) = 1 / (1 - x)$ 
    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  $n$ th term is the partial sum up to  $n$ , that is,
   $?s\ n$ .
  have cauchy-product-to-partial-sum:  $(\sum n.\ x \wedge n) * f\ x = (\sum n.\ ?s\ n * x \wedge n)$ 
    if  $a1$ :  $\text{norm } x < 1$  for  $x :: \text{real}$ 
  proof —
    from  $a1$  have x-lt-1:  $|x| < 1$ 
      by (simp)

    show  $(\sum n.\ x \wedge n) * f\ x = (\sum n.\ ?s\ n * x \wedge n)$ 
  proof —

```

```

have f0:  $\forall n. \forall i \leq n. x^{\wedge} i * (a (n - i) * x^{\wedge} (n - i)) = a (n - i) * x^{\wedge} n$ 
  apply auto[1]
  by (metis le-add-diff-inverse power-add)

have f1:  $\forall n. (\sum i \leq n. x^{\wedge} i * (a (n - i) * x^{\wedge} (n - i))) = (\sum i \leq n. a (n - i) * x^{\wedge} n)$ 
  by (auto simp: f0)

have f2:  $\forall n. (\sum i \leq n. a (n - i)) = ?s\ n$ 
proof (rule allI)
  fix n::nat
  show  $(\sum i \leq n. a (n - i)) = ?s\ n$ 
    by (rule sum.reindex-bij-witness[of -  $\lambda i. n - i$   $\lambda i. n - i$ ]) auto
qed

show ?thesis
  unfolding f-def f1-def
proof (subst Cauchy-product)
  show summable ( $\lambda k. \text{norm } (x^{\wedge} k)$ )
    by (simp add: power-abs x-lt-1)
next
  show summable ( $\lambda k. \text{norm } (a\ k * x^{\wedge} k)$ )
    by (metis abs-summable-in-conv-radius conv-radius-1 ereal-less(3) real-norm-def x-lt-1)
next
  show  $(\sum k. \sum i \leq k. x^{\wedge} i * (a (k - i) * x^{\wedge} (k - i))) = (\sum n. \text{sum } a \ \{..n\} * x^{\wedge} n)$ 
    by (subst f1) (use f2 in <simp-all flip: sum-distrib-right>)
qed
qed
qed

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

  assume a1:  $\text{norm } x < 1$ 
  from a1 have x-lt-1:  $|x| < 1$ 
    by (simp)

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

  have f0:  $(\lambda n. ?s\ n * x^{\wedge} n) = (\lambda n. \sum i \leq n. a\ i * x^{\wedge} n)$ 
    using sum-distrib-right by blast

  have f1:  $\dots = (\lambda n. \sum i \leq n. (a\ i * x^{\wedge} i) * (x^{\wedge} (n - i)))$ 
    apply (simp only: mult.assoc)

```

```

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

show summable ( $\lambda n. ?s\ n * x^{\wedge} n$ )
apply (simp only: f0 f1)
apply (rule summable-Cauchy-product[where  $a = \lambda n. (a\ n) * x^{\wedge} n$  and  $b$ 
=  $\lambda n. x^{\wedge} n$ ])
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 $f\text{-}x\text{-to-partial-sum}$: $\forall x::real. \text{norm } x < 1 \longrightarrow f\ x = (1 - x) * (\sum n. ?s\ n * x^{\wedge} n)$

```

proof (rule allI, rule impI)
fix  $x::real$ 

```

```

assume  $a1$ :  $\text{norm } x < 1$ 
from  $a1$  have  $x\text{-lt-1}$ :  $|x| < 1$ 
by (simp)

```

— Rewrite $f\ x$ because $(1 - x) * (\sum n. x^{\wedge} n) = 1$.
have $f\text{-rewrite}$: $f\ x = (1 - x) * (\sum n. x^{\wedge} n) * f\ x$
using $\text{geometric } x\text{-lt-1}$ **by** fastforce

— According to the Cauchy product result.
then show $f\ x = (1 - x) * (\sum n. ?s\ n * x^{\wedge} n)$
using $\text{cauchy-product-to-partial-sum}$ mult.assoc **by** (metis real-norm-def
 $x\text{-lt-1}$)
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  $f\text{-}x\text{-minus-}S$ :  $\forall x::real. \text{norm } x < 1 \longrightarrow f\ x - S = (1 - x) * ?fs\text{-}S\text{-sum } x$ 
proof (rule allI, rule impI)
fix  $x::real$ 

```

```

assume  $a1$ :  $\text{norm } x < 1$ 
from  $a1$  have  $x\text{-lt-1}$ :  $|x| < 1$ 
by (simp)

```

```

have  $f0$ :  $(1 - x) * (\sum n. ?s\ n * x^{\wedge} n) - S = (1 - x) * (\sum n. ?s\ n * x^{\wedge} n)$ 
-  $(1 - x) * S * (\sum n. x^{\wedge} n)$ 
apply (simp add:  $\text{geometric}$ )
using  $\text{geometric } x\text{-lt-1}$  by  $\text{auto}$ 

```

```

have  $f1$ :  $\dots = (1 - x) * ((\sum n. ?s\ n * x^{\wedge} n) - (\sum n. S * x^{\wedge} n))$ 

```

```

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

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

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

```

```

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

```

— Use the ε - δ definition of a continuous function or a limit.

```

have  $S\text{-is-}f\text{-limit-from-left}$ :  $(f \longrightarrow S) \text{ (at-left } 1)$ 
proof ( $\text{simp only: tendsto-iff eventually-at-left-field, simp only: dist-norm, rule}$ 
 $\text{allI, rule impI}$ )
  —  $r$  is the difference of function  $f\ x$  from  $f\ 1$ 
  fix  $r :: \text{real}$ 
  assume  $r\text{-gt-0}$ :  $(0 :: \text{real}) < r$ 

```

— Try to make both tail and head parts contribute $r/2$, so finally r .

```

define  $e$  where  $e = r/2$ 

```

```

have  $e\text{-gt-0}$ :  $0 < e$ 
  by ( $\text{simp add: r-gt-0 e-def}$ )

```

— M is not necessary to be positive and can be 0.

```

obtain  $M$  where  $P\text{-M}$ :  $(\forall n \geq M. \text{norm } (?s\ n - S) < e)$ 
  using  $s\text{-limit-}S \text{ LIMSEQ-iff } e\text{-gt-0 real-norm-def}$  by  $\text{metis}$ 

```

— Make N be positive, which will be used later to ensure $\text{norm } x^{\wedge}N < 1$

```

define  $N$  where  $N = M + 1$ 

```

— C is the sum of differences up to N .

```

define  $C$  where  $C = (\sum k < N. \text{norm } (?s\ k - S))$ 

```

have $P-N$: $(\forall n \geq N. \text{norm } (?s\ n - S) < e)$
unfolding $N\text{-def}$ **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 $fs\text{-}S\text{-split}$: $\forall x::\text{real}. \text{norm } x < 1 \longrightarrow (1 - x) * ?fs\text{-}S\text{-sum } x$
 $= (1 - x) * (\sum n < N. ?fs\text{-}S\ x\ n) + (1 - x) * (\sum n. ?fs\text{-}S\ x\ (n + N))$
proof (rule allI , rule impI)
fix $x::\text{real}$

assume $a1$: $\text{norm } x < 1$
from $a1$ **have** $x\text{-lt-1}$: $|x| < 1$
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**
 $\dots = ?fs\text{-}S\text{-sum-hd} + ?fs\text{-}S\text{-sum-tl}$)
apply ($\text{subst suminf-split-initial-segment}[\text{where } k = N]$)
using $\text{summable-norm-fs-S}$ real-norm-def $\text{summable-norm-cancel } x\text{-lt-1}$
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 ($\text{simp add: distrib-left}$)
qed

— For the tail part, it is less than e .

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

```

    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
            end
          end
        end
      end
    end
  end

```

```

    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  $((\lambda x. f \ (-x)) \longrightarrow l) \ (at\text{-}right \ (-1)) \longleftrightarrow ((\lambda x. f \ (x)) \longrightarrow l) \ (at\text{-}left \ (1::real))$ 
  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  $\Rightarrow$  real
  defines f1  $\equiv (\lambda(x::real) \ n. \ a \ n * x ^ n)$ 
  defines f  $\equiv (\lambda(x::real). \ \sum n. \ f1 \ x \ n)$ 
  assumes summable-a: summable a and
    conv-radius-1: conv-radius a = 1
  shows  $((\lambda x. f \ (-x)) \longrightarrow (\sum n. \ a \ n)) \ (at\text{-}right \ (-1))$ 
  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 $|?z| < 1 \implies (\lambda n. \ (1 \ / \ 2 \ gchoose \ n) * ?z^n) \ sums \ sqrt \ (1 + ?z)$. 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 $x :: \text{real}$
assumes $|x| < 1$
shows $\text{suminf } (\lambda n. ((1/2) \text{ gchoose } n) * x^n) = \text{sqrt } (1 + x)$
apply ($\text{subst sums-unique}[\text{where } s = \text{sqrt } (1 + x) \text{ and } f = (\lambda n. ((1/2) \text{ gchoose } n) * x^n)]$)
apply ($\text{rule sqrt-series}[\text{where } z = x]$)
using *assms* **apply** *blast*
by *simp*

The generalized binomial coefficient $a \text{ gchoose } n$ 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*: $((1/2) \text{ gchoose } (\text{Suc } n)) = ((-1)^n / (2^{2n+1})) * \text{real } (\text{catalan } n)$
by ($\text{subst catalan-closed-form-gbinomial}$) (*simp add: power-mult power-minus'*)

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

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

lemma *gbinomial-1-2-simp*:

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

lemma *summable-real-powr-iff'*: $\text{summable } (\lambda n. 1 / \text{of-nat } n \text{ powr } s :: \text{real}) \longleftrightarrow s > 1$

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

lemma *summable-1-2-gchoose*: $\text{summable } (\lambda n. ((1::\text{real})/2) \text{ gchoose } n)$

proof –

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

```

    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  $(\sum n. (((1::real) / (2::real) \text{ gchoose } n))) = \text{sqrt } 2$  (is  $(\sum n. ?f-1 \ n) = -$ )
proof -
  let ?f =  $\lambda x. (\sum n. (((1::real) / (2::real) \text{ gchoose } n)) * x ^ n)$ 

  — Inside the disk: expansion gives sqrt(1+x)
  have eq-inside:  $\bigwedge x. \text{abs } x < 1 \implies ?f \ (x) = \text{sqrt } (1+x)$ 
    using sqrt-series sums-unique by force

  have (?f  $\longrightarrow \text{sqrt } 2$ ) (at-left 1))
proof -
  have  $((\lambda x. \text{sqrt } (1+x)) \longrightarrow \text{sqrt } 2)$  (at-left 1))
proof (intro tendsto-intros)
  have  $((+) (1::real) \longrightarrow 1 + 1)$  (at-left 1)
    using tendsto-add-const-iff by fastforce
  then show  $((+) (1::real) \longrightarrow 2)$  (at-left 1)
    by simp
qed
moreover have eventually  $(\lambda x. ?f \ (x) = \text{sqrt}(1+x))$  (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:  $(?f \longrightarrow \text{sqrt } 2)$  (at-left 1)) by simp

have lim-by-abel-from-left:  $(?f \longrightarrow (\sum n. ?f-1 \ n))$  (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 =  $(\sum n. ?f-1 \ n)$ 
    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 $a :: 'a :: \text{real-normed-field}$

assumes $a \notin \mathbb{N}$

shows $(\lambda n. ((a \text{ gchoose } n) * (-1) ^ n) / ((a \text{ gchoose } \text{Suc } n) * (-1) ^ (\text{Suc } n)))$
 $\longrightarrow 1$

proof –

have $(\lambda n. ((a \text{ gchoose } n) * (-1) ^ n) / ((a \text{ gchoose } \text{Suc } n) * (-1) ^ (\text{Suc } n)))$
 $= (\lambda n. - ((a \text{ gchoose } n) / (a \text{ gchoose } \text{Suc } n)))$

by *auto*

then show *?thesis*

using *gbinomial-ratio-limit assms tendsto-minus-cancel-left* **by** *fastforce*

qed

lemma *conv-radius-gchoose-alternating*:

fixes $a :: 'a :: \{\text{real-normed-field}, \text{banach}\}$

assumes $a \notin \mathbb{N}$

shows *conv-radius* $(\lambda n::\text{nat}. (a \text{ gchoose } n) * (-1) ^ n) = (1::\text{ereal})$

proof –

from *tendsto-norm[OF gbinomial-ratio-limit']*

have *conv-radius* $(\lambda n::\text{nat}. (a \text{ gchoose } n) * (-1) ^ n) = 1$

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 $(\lambda n::\text{nat}. (1 / 2 \text{ gchoose } n) * (-1) ^ n :: \text{real})$ (**is** *summable* *?f*)

proof –

have *f0*: $(\lambda n. ((1/2) \text{ gchoose } (\text{Suc } n))) \sim[at-top]$

$(\lambda n. (((-1) ^ n / 2) * (1/4 ^ n)) * (4 ^ n / ((\text{sqrt } \pi * n \text{ powr } (3/2))))))$

apply (*simp only: gbinomial-1-2-catalan'*)

apply (*subst asymp-equiv-mult*)

using *asymp-equiv-refl* **apply** *blast*

using *catalan-asympotics* **apply** *blast*

by *simp*

have *f1*: $\dots = (\lambda n. (-1) ^ n / (2 * (\text{sqrt } \pi * n \text{ powr } (3/2))))$

by *auto*

have *f2*: $\dots = (\lambda n. 1 / (2 * (\text{sqrt } \pi))) * ((-1) ^ n / (n \text{ powr } (3/2)))$ (**is** $= ?g$)

by *auto*

have *f3*: $(\lambda n. ?g (n) * (- (1::\text{real})) ^ (\text{Suc } n)) =$

$(\lambda n. (-1 / (2 * (\text{sqrt } \pi))) * (1 / (n \text{ powr } (3/2))))$ (**is** $= ?g1$)

by *auto*

have *f4*: $(\lambda n. ?f (\text{Suc } n)) \sim[at-top] (\lambda n. ?g (n) * (- (1::\text{real})) ^ (\text{Suc } n))$ (**is** $\sim[at-top] ?g1$)

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:  $\forall n. (- (1::real)) ^ n / real\ n\ powr ((3::real) / (2::real)) =$ 
     $(- (1::real)) ^ n * real\ n\ powr (- ((3::real) / (2::real)))$ 
    using divide-powr-uminus by presburger

  have f2: summable ( $\lambda n::nat. - ((- (1::real)) ^ n * real\ (n + 1)\ powr (-$ 
     $((3::real) / (2::real))))$ )
    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 ( $\lambda n::nat. ((- (1::real)) ^ (n + 1) * real\ (n + 1)\ powr (-$ 
     $((3::real) / (2::real))))$ )
    using f2 by simp

  show summable ( $\lambda n::nat. (- (1::real)) ^ n / real\ n\ powr ((3::real) / (2::real))$ )
    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 ( $\lambda n::nat. (1::real) / real\ n\ powr ((3::real) / (2::real)) =$ 
     $(\lambda n::nat. real\ n\ powr (- (3::real) / (2::real)))$ )
    subgoal by (simp add: summable-real-powr-iff)
    by (simp add: inverse-eq-divide powr-minus)

have summable-norm-g1: summable ( $\lambda n. norm\ (?g1\ n)$ )
  apply (simp add: f3)
  apply (subgoal-tac ( $\lambda n::nat. (1::real) / ((2::real) * sqrt\ pi * real\ n\ powr (3 /$ 
     $2))) =$ 
     $(\lambda n::nat. (1::real) / ((2::real) * sqrt\ pi) * real\ n\ powr (-3 / 2)))$ )
    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  $(\sum n. ((1/2 \text{ gchoose } n) * (- (1::real)) ^ n)) = 0$  (is  $(\sum n. ?f-1 \ n) = 0$ )
proof -
  let ?f =  $\lambda x. (\sum n. (((1::real) / (2::real)) \text{ gchoose } n) * (-1) ^ n * x ^ n)$ 

  have f0:  $\forall x. ?f \ x = (\sum n. (((1::real) / (2::real)) \text{ gchoose } n) * (-x) ^ n)$ 
    by (metis (no-types, lifting) more-arith-simps(11) power-minus suminf-cong)

  — Inside the disk: expansion gives sqrt(1+x)
  have eq-inside:  $\bigwedge x. \text{abs } x < 1 \implies ?f \ (-x) = \text{sqrt } (1 - (-x))$ 
    apply (simp only: f0)
    using sqrt-series sums-unique by force

  have  $((\lambda x. ?f \ (-x)) \longrightarrow \text{sqrt } (1 + (-1))) \text{ (at-right } (-1))$ 
proof -
  have  $((\lambda x. \text{sqrt } (1+x)) \longrightarrow \text{sqrt } 0) \text{ (at-right } (-1))$ 
    apply (intro tendsto-intros)
    using filterlim-at-right-to-0 by fastforce
  moreover have eventually  $(\lambda x. ?f \ (-x) = \text{sqrt}(1+x)) \text{ (at-right } (-1))$ 
    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:  $((\lambda x. ?f \ (-x)) \longrightarrow 0) \text{ (at-right } (-1))$  by simp

have lim-by-abel-from-right:  $((\lambda x. ?f \ (-x)) \longrightarrow (\sum n. ?f-1 \ n)) \text{ (at-right } (-1))$ 
  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-distrib(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 =  $(\lambda x. ?f \ (-x))$  and  $a = (\sum n. ?f-1 \ n)$ 
    and  $F = \text{at-right } (-1)$  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  $|x| \leq (1 :: real)$ 
  shows  $\suminf (\lambda n. ((1/2) gchoose n) * x ^ n) = \text{sqrt } (1 + x)$ 
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.