Stirling's formula

Manuel Eberl

February 6, 2026

Abstract

This work contains a proof of Stirling's formula both for the factorial $n! \sim \sqrt{2\pi n}(n/e)^n$ on natural numbers and the real Gamma function $\Gamma(x) \sim \sqrt{2\pi/x}(x/e)^x$. The proof is based on work by Graham Jameson [3].

This is then extended to the full asymptotic expansion

$$\log\Gamma(z) = \left(z - \tfrac{1}{2}\right)\log z - z + \tfrac{1}{2}\log(2\pi) + \sum_{k=1}^{n-1} \frac{B_{k+1}}{k(k+1)} z^{-k} - \frac{1}{n}\int_0^\infty B_n([t])(t+z)^{-n}\,\mathrm{d}t$$

uniformly for all complex $z \neq 0$ in the cone $\arg(z) \leq \alpha$ for any $\alpha \in (0, \pi)$, with which the above asymptotic relation for Γ is also extended to complex arguments.

Contents

1 Stirling's Formula

theory *Stirling-Formula*
imports
HOL−Analysis.Analysis
Landau-Symbols.Landau-More
HOL−Real-Asymp.Real-Asymp
begin

```
context
begin
```

First, we define the S_n^* from Jameson's article:

```
qualified definition S′ :: nat ⇒ real ⇒ real where
  S′ n x = 1/(2∗x) + (∑ r=1..<n. 1/(of-nat r+x)) + 1 /(2∗(n + x))
```

Next, the trapezium (also called T in Jameson's article):

```
qualified definition T :: real ⇒ real where
  T x = 1/(2∗x) + 1/(2∗(x+1))
```

Now we define The difference $\Delta(x)$:

```
qualified definition D :: real ⇒ real where
  D x = T x − ln (x + 1) + ln x

qualified lemma S′-telescope-trapezium:
  assumes n > 0
  shows   S′ n x = (∑ r<n. T (of-nat r + x))
proof (cases n)
  case (Suc m)
  hence m: Suc m = n by simp
  have (∑ r<n. T (of-nat r + x)) =
          (∑ r<Suc m. 1 / (2 ∗ real r + 2 ∗ x)) + (∑ r<n. 1 / (2 ∗ real (Suc r)
+ 2 ∗ x))
    unfolding m by (simp add: T-def sum.distrib algebra-simps)
  also have (∑ r<Suc m. 1 / (2 ∗ real r + 2 ∗ x)) =
               1/(2∗x) + (∑ r<m. 1 / (2 ∗ real (Suc r) + 2 ∗ x)) (is - = ?a + ?S)
    by (subst sum.lessThan-Suc-shift) simp
  also have (∑ r<n. 1 / (2 ∗ real (Suc r) + 2 ∗ x)) =
               ?S + 1 / (2∗(real m + x + 1)) (is - = - + ?b) by (simp add: Suc)
  also have ?a + ?S + (?S + ?b) = 2∗?S + ?a + ?b by (simp add: add-ac)
  also have 2 ∗ ?S = (∑ r=0..<m. 1 / (real (Suc r) + x))
    unfolding sum-distrib-left by (intro sum.cong) (auto simp add: divide-simps)
  also have (∑ r=0..<m. 1 / (real (Suc r) + x)) = (∑ r=Suc 0..<Suc m. 1 /
(real r + x))
    by (subst sum.atLeast-Suc-lessThan-Suc-shift) simp-all
  also have ... = (∑ r=1..<n. 1 / (real r + x)) unfolding m by simp
  also have ... + ?a + ?b = S′ n x by (simp add: S′-def Suc)
  finally show ?thesis ..
qed (insert assms, simp-all)

qualified lemma stirling-trapezium:
  assumes x: (x::real) > 0
  shows   D x ∈ {0 .. 1/(12∗x^2) − 1/(12 ∗ (x+1)^2)}
proof −
  define y where y = 1 / (2∗x + 1)
  from x have y: y > 0 y < 1 by (simp-all add: divide-simps y-def)
```

from x **have** $D\ x = T\ x - ln\ ((x + 1)\ /\ x)$ **by** (*subst ln-div*) (*simp-all add*: *D-def*)
also from x **have** $(x + 1)\ /\ x = 1 + 1\ /\ x$ **by** (*simp add*: *field-simps*)
finally have D: $D\ x = T\ x - ln\ (1 + 1/x)$ **.**

from y **have** $(\lambda n.\ y * y\hat{}n)$ *sums* $(y * (1\ /\ (1 - y)))$
by (*intro geometric-sums sums-mult*) *simp-all*
hence $(\lambda n.\ y\ \hat{}\ Suc\ n)$ *sums* $(y\ /\ (1 - y))$ **by** *simp*
also from x **have** $y\ /\ (1 - y) = 1\ /\ (2*x)$ **by** (*simp add*: *y-def divide-simps*)
finally have $*$: $(\lambda n.\ y\ \hat{}\ Suc\ n)$ *sums* $(1\ /\ (2*x))$ **.**

from y **have** $(\lambda n.\ (-y) * (-y)\hat{}n)$ *sums* $((-y) * (1\ /\ (1 - (-y))))$
by (*intro geometric-sums sums-mult*) *simp-all*
hence $(\lambda n.\ (-y)\ \hat{}\ Suc\ n)$ *sums* $(-(y\ /\ (1 + y)))$ **by** *simp*
also from x **have** $y\ /\ (1 + y) = 1\ /\ (2*(x+1))$ **by** (*simp add*: *y-def divide-simps*)
finally have $**$: $(\lambda n.\ (-y)\ \hat{}\ Suc\ n)$ *sums* $(-(1\ /\ (2*(x+1))))$ **.**

from *sums-diff*[*OF* $*$ $**$] **have** *sum1*: $(\lambda n.\ y\ \hat{}\ Suc\ n - (-y)\ \hat{}\ Suc\ n)$ *sums* $T\ x$
by (*simp add*: *T-def*)

from y **have** *abs* $y < 1$ *abs* $(-y) < 1$ **by** *simp-all*
from *sums-diff*[*OF this*[*THEN ln-series'*]]
have $(\lambda n.\ y\ \hat{}\ n\ /\ real\ n - (-y)\ \hat{}\ n\ /\ real\ n)$ *sums* $(ln\ (1 + y) - ln\ (1 - y))$ **by** *simp*
also from y **have** $ln\ (1 + y) - ln\ (1 - y) = ln\ ((1 + y)\ /\ (1 - y))$ **by** (*simp add*: *ln-div*)
also from x **have** $(1 + y)\ /\ (1 - y) = 1 + 1/x$ **by** (*simp add*: *divide-simps y-def*)
finally have $(\lambda n.\ y\hat{}n\ /\ real\ n - (-y)\hat{}n\ /\ real\ n)$ *sums* $ln\ (1 + 1/x)$ **.**
hence *sum2*: $(\lambda n.\ y\hat{}Suc\ n\ /\ real\ (Suc\ n) - (-y)\hat{}Suc\ n\ /\ real\ (Suc\ n))$ *sums* $ln\ (1 + 1/x)$
by (*subst sums-Suc-iff*) *simp*

have $ln\ (1 + 1/x) \leq T\ x$
proof (*rule sums-le* [*OF - sum2 sum1*])
fix n :: *nat*
show $y\ \hat{}\ Suc\ n\ /\ real\ (Suc\ n) - (-y)\ \hat{}\ Suc\ n\ /\ real\ (Suc\ n) \leq y\hat{}Suc\ n - (-y)\ \hat{}\ Suc\ n$
proof (*cases even n*)
case *True*
hence *eq*: $A - (-y)\ \hat{}\ Suc\ n\ /\ B = A + y\ \hat{}\ Suc\ n\ /\ B$ $A - (-y)\ \hat{}\ Suc\ n = A + y\ \hat{}\ Suc\ n$
for $A\ B$ **by** *simp-all*
from y **show** *?thesis* **unfolding** *eq*
by (*intro add-mono*) (*auto simp*: *divide-simps*)
qed *simp-all*
qed
hence $D\ x \geq 0$ **by** (*simp add*: D)

define *c* **where** $c = (\lambda n.\ \mathit{if\ even\ n\ then\ 2} * (\mathit{1} - \mathit{1} / \mathit{real}\ (\mathit{Suc}\ n))\ \mathit{else\ 0})$
note *sums-diff*[*OF sum1 sum2*]
also have $(\lambda n.\ y \hat{}\ \mathit{Suc}\ n - (-y)\ \hat{}\ \mathit{Suc}\ n - (y\ \hat{}\ \mathit{Suc}\ n / \mathit{real}\ (\mathit{Suc}\ n) -$
$(-y)\ \hat{}\ \mathit{Suc}\ n / \mathit{real}\ (\mathit{Suc}\ n))) = (\lambda n.\ c\ n * y\ \hat{}\ \mathit{Suc}\ n)$
by (*intro ext*) (*simp add*: *c-def algebra-simps*)
finally have *sum3*: $(\lambda n.\ c\ n * y\ \hat{}\ \mathit{Suc}\ n)$ *sums D x* **by** (*simp add*: *D*)

from *y* **have** $(\lambda n.\ y\hat{}2 * (\mathit{of\text{-}nat}\ (\mathit{Suc}\ n) * y\hat{}n))$ *sums* $(y\hat{}2 * (1 / (1 - y)\hat{}2))$
by (*intro sums-mult geometric-deriv-sums*) *simp-all*
hence $(\lambda n.\ \mathit{of\text{-}nat}\ (\mathit{Suc}\ n) * y\hat{}(n+2))$ *sums* $(y\hat{}2 / (1 - y)\hat{}2)$
by (*simp add*: *algebra-simps power2-eq-square*)
also from *x* **have** $y\hat{}2 / (1 - y)\hat{}2 = 1 / (4*x\hat{}2)$ **by** (*simp add*: *y-def divide-simps*)
finally have ∗: $(\lambda n.\ \mathit{real}\ (\mathit{Suc}\ n) * y\ \hat{}\ (\mathit{Suc}\ (\mathit{Suc}\ n)))$ *sums* $(1 / (4 * x^2))$ **by** *simp*

from *y* **have** $(\lambda n.\ y\hat{}2 * (\mathit{of\text{-}nat}\ (\mathit{Suc}\ n) * (-y)\hat{}n))$ *sums* $(y\hat{}2 * (1 / (1 - -y)\hat{}2))$
by (*intro sums-mult geometric-deriv-sums*) *simp-all*
hence $(\lambda n.\ \mathit{of\text{-}nat}\ (\mathit{Suc}\ n) * (-y)\hat{}(n+2))$ *sums* $(y\hat{}2 / (1 + y)\hat{}2)$
by (*simp add*: *algebra-simps power2-eq-square*)
also from *x* **have** $y\hat{}2 / (1 + y)\hat{}2 = 1 / (2\hat{}2*(x+1)\hat{}2)$
unfolding *power-mult-distrib* [*symmetric*] **by** (*simp add*: *y-def divide-simps add-ac*)
finally have ∗∗: $(\lambda n.\ \mathit{real}\ (\mathit{Suc}\ n) * (-\ y)\ \hat{}\ (\mathit{Suc}\ (\mathit{Suc}\ n)))$ *sums* $(1 / (4 * (x + 1)^2))$ **by** *simp*

define *d* **where** $d = (\lambda n.\ \mathit{if\ even\ n\ then\ 2} * \mathit{real}\ n\ \mathit{else\ 0})$
note *sums-diff*[*OF* ∗ ∗∗]
also have $(\lambda n.\ \mathit{real}\ (\mathit{Suc}\ n) * y\hat{}(\mathit{Suc}\ (\mathit{Suc}\ n)) - \mathit{real}\ (\mathit{Suc}\ n) * (-y)\hat{}(\mathit{Suc}\ (\mathit{Suc}\ n))) =$
$(\lambda n.\ d\ (\mathit{Suc}\ n) * y\ \hat{}\ \mathit{Suc}\ (\mathit{Suc}\ n))$
by (*intro ext*) (*simp-all add*: *d-def*)
finally have $(\lambda n.\ d\ n * y\ \hat{}\ \mathit{Suc}\ n)$ *sums* $(1 / (4 * x^2) - 1 / (4 * (x + 1)^2))$
by (*subst* (*asm*) *sums-Suc-iff*) (*simp add*: *d-def*)
from *sums-mult*[*OF this*, *of 1/3*] *x*
have *sum4*: $(\lambda n.\ d\ n / 3 * y\ \hat{}\ \mathit{Suc}\ n)$ *sums* $(1 / (12 * x\hat{}2) - 1 / (12 * (x + 1)\hat{}2))$
by (*simp add*: *field-simps*)

have $D\ x \le (1 / (12 * x\hat{}2) - 1 / (12 * (x + 1)\hat{}2))$
proof (*intro sums-le* [*OF - sum3 sum4*] *allI*)
fix *n* :: *nat*
define c' :: $\mathit{nat} \Rightarrow \mathit{real}$
where $c' = (\lambda n.\ \mathit{if\ odd}\ n \lor n = 0\ \mathit{then\ 0\ else\ if}\ n = 2\ \mathit{then}\ 4/3\ \mathit{else}\ 2)$
show $c\ n * y\ \hat{}\ \mathit{Suc}\ n \le d\ n / 3 * y\ \hat{}\ \mathit{Suc}\ n$
proof (*intro mult-right-mono*)
have $c\ n \le c'\ n$ **by** (*simp add*: *c-def c'-def*)
also consider $n = 0 \mid n = 1 \mid n = 2 \mid n \ge 3$ **by** *force*

hence $c'\ n \leq d\ n\ /\ 3$ **by** *cases* (*simp-all add*: *c'-def d-def*)
finally show $c\ n \leq d\ n\ /\ 3$ **.**
qed (*insert y*, *simp*)
qed

with ‹$D\ x \geq 0$› **show** *?thesis* **by** *simp*
qed

The following functions correspond to the $p_n(x)$ from the article. The special case $n = 0$ would not, strictly speaking, be necessary, but it allows some theorems to work even without the precondition $n \neq 0$:

qualified definition $p :: nat \Rightarrow real \Rightarrow real$ **where**
$p\ n\ x = (if\ n = 0\ then\ 1/x\ else\ (\sum r{<}n.\ D\ (real\ r + x)))$

We can write the Digamma function in terms of S':

qualified lemma *S'-LIMSEQ-Digamma*:
assumes x: $x \neq 0$
shows $(\lambda n.\ ln\ (real\ n) - S'\ n\ x - 1/(2{*}x)) \longrightarrow Digamma\ x$
proof −
define c **where** $c = (\lambda n.\ ln\ (real\ n) - (\sum r{<}n.\ inverse\ (x + real\ r)))$
have *eventually* $(\lambda n.\ 1\ /\ (2 * (x + real\ n)) = c\ n - (ln\ (real\ n) - S'\ n\ x - 1/(2{*}x)))$ *at-top*
using *eventually-gt-at-top*[*of 0::nat*]
proof *eventually-elim*
fix $n :: nat$
assume n: $n > 0$
have $c\ n - (ln\ (real\ n) - S'\ n\ x - 1/(2{*}x)) =$
$-(\sum r{<}n.\ inverse\ (real\ r + x)) + (1/x + (\sum r{=}1..{<}n.\ inverse\ (real\ r + x))) + 1/(2{*}(real\ n + x))$
using x **by** (*simp add*: *S'-def c-def field-simps*)
also have $1/x + (\sum r{=}1..{<}n.\ inverse\ (real\ r + x)) = (\sum r{<}n.\ inverse\ (real\ r + x))$
unfolding *lessThan-atLeast0* **using** n
by (*subst* (*2*) *sum.atLeast-Suc-lessThan*) (*simp-all add*: *field-simps*)
finally show $1\ /\ (2 * (x + real\ n)) = c\ n - (ln\ (real\ n) - S'\ n\ x - 1/(2{*}x))$ **by** *simp*
qed
moreover have $(\lambda n.\ 1\ /\ (2 * (x + real\ n))) \longrightarrow 0$
by *real-asymp*
ultimately have $(\lambda n.\ c\ n - (ln\ (real\ n) - S'\ n\ x - 1/(2{*}x))) \longrightarrow 0$
by (*blast intro*: *Lim-transform-eventually*)
from *tendsto-minus*[*OF this*] **have** $(\lambda n.\ (ln\ (real\ n) - S'\ n\ x - 1/(2{*}x)) - c\ n) \longrightarrow 0$ **by** *simp*
moreover from *Digamma-LIMSEQ*[*OF x*] **have** $c \longrightarrow Digamma\ x$ **by** (*simp add*: *c-def*)
ultimately show $(\lambda n.\ ln\ (real\ n) - S'\ n\ x - 1/(2{*}x)) \longrightarrow Digamma\ x$
by (*rule Lim-transform* [*rotated*])
qed

Moreover, we can give an expansion of S' with the p as variation terms.

qualified lemma *S′-approx*:
  S′ n x = ln (real n + x) − ln x + p n x
proof (*cases n = 0*)
  case *True*
  thus *?thesis* **by** (*simp add*: *p-def S′-def*)
next
  case *False*
  hence S' *n x* = ($\sum$ *r<n. T (real r + x)*)
    by (*subst S′-telescope-trapezium*) *simp-all*
  also have ... = ($\sum$ *r<n. ln (real r + x + 1) − ln (real r + x) + D (real r + x)*)
    by (*simp add*: *D-def*)
  also have ... = ($\sum$ *r<n. ln (real (Suc r) + x) − ln (real r + x)) + p n x*
    using *False* **by** (*simp add*: *sum.distrib add-ac p-def*)
  also have ($\sum$ *r<n. ln (real (Suc r) + x) − ln (real r + x)) = ln (real n + x) − ln x*
    by (*subst sum-lessThan-telescope*) *simp-all*
  finally show *?thesis* **.**
qed

We define the limit of the p (simply called $p(x)$ in Jameson's article):

qualified definition *P* :: *real* ⇒ *real* **where**
  P x = ($\sum$ *n. D (real n + x)*)

qualified lemma *D-summable*:
  assumes *x*: *x > 0*
  shows *summable (λn. D (real n + x))*
proof −
  have ∗: *summable (λn. 1 / (12 ∗ (x + real n)*^{2}*) − 1 / (12 ∗ (x + real (Suc n))*^{2}*))*
    by (*rule telescope-summable′*) *real-asymp*
  show *summable (λn. D (real n + x))*
  proof (*rule summable-comparison-test*[*OF - ∗*], *rule exI*[*of - 2*], *safe*)
    fix *n* :: *nat* **assume** *n ≥ 2*
    show *norm (D (real n + x)) ≤ 1 / (12 ∗ (x + real n)^2) − 1 / (12 ∗ (x + real (Suc n))^2)*
      using *stirling-trapezium*[*of real n + x*] *x* **by** (*auto simp*: *algebra-simps*)
  qed
qed

qualified lemma *p-LIMSEQ*:
  assumes *x*: *x > 0*
  shows (*λn. p n x*) ⟶ *P x*
proof (*rule Lim-transform-eventually*)
  from *D-summable*[*OF x*] **have** (*λn. D (real n + x)*) *sums P x* **unfolding** *P-def*
    by (*simp add*: *sums-iff*)
  then show (*λn.* $\sum$ *r<n. D (real r + x)*) ⟶ *P x* **by** (*simp add*: *sums-def*)
  moreover from *eventually-gt-at-top*[*of 1*]

show *eventually* $(\lambda n.\ (\sum r{<}n.\ D\ (real\ r + x)) = p\ n\ x)$ *at-top*
by *eventually-elim* (*auto simp*: *p-def*)
qed

This gives us an expansion of the Digamma function:

lemma *Digamma-approx*:
assumes x: $(x :: real) > 0$
shows $Digamma\ x = ln\ x - 1\ /\ (2 * x) - P\ x$
proof –
have *eventually* $(\lambda n.\ ln\ (inverse\ (1 + x\ /\ real\ n)) + ln\ x - 1/(2{*}x) - p\ n\ x = ln\ (real\ n) - S'\ n\ x - 1/(2{*}x))$ *at-top*
using *eventually-gt-at-top*[*of* 1::*nat*]
proof *eventually-elim*
fix $n :: nat$ **assume** n: $n > 1$
have $ln\ (real\ n) - S'\ n\ x = ln\ ((real\ n)\ /\ (real\ n + x)) + ln\ x - p\ n\ x$
using *assms n* **unfolding** *S'-approx* **by** (*subst ln-div*) (*auto simp*: *algebra-simps*)
also from n **have** $real\ n\ /\ (real\ n + x) = inverse\ (1 + x\ /\ real\ n)$ **by** (*simp add*: *field-simps*)
finally show $ln\ (inverse\ (1 + x\ /\ real\ n)) + ln\ x - 1/(2{*}x) - p\ n\ x = ln\ (real\ n) - S'\ n\ x - 1/(2{*}x)$ **by** *simp*
qed
moreover have $(\lambda n.\ ln\ (inverse\ (1 + x\ /\ real\ n)) + ln\ x - 1/(2{*}x) - p\ n\ x) \longrightarrow ln\ (inverse\ (1 + 0)) + ln\ x - 1/(2{*}x) - P\ x$
by (*rule tendsto-intros p-LIMSEQ x real-tendsto-divide-at-top filterlim-real-sequentially* | *simp*)+
hence $(\lambda n.\ ln\ (inverse\ (1 + x\ /\ real\ n)) + ln\ x - 1/(2{*}x) - p\ n\ x) \longrightarrow ln\ x - 1/(2{*}x) - P\ x$ **by** *simp*
ultimately have $(\lambda n.\ ln\ (real\ n) - S'\ n\ x - 1\ /\ (2 * x)) \longrightarrow ln\ x - 1/(2{*}x) - P\ x$
by (*blast intro*: *Lim-transform-eventually*)
moreover from x **have** $(\lambda n.\ ln\ (real\ n) - S'\ n\ x - 1\ /\ (2 * x)) \longrightarrow Digamma\ x$
by (*intro S'-LIMSEQ-Digamma*) *simp-all*
ultimately show $Digamma\ x = ln\ x - 1\ /\ (2 * x) - P\ x$
by (*rule LIMSEQ-unique* [*rotated*])
qed

Next, we derive some bounds on P:

qualified lemma *p-ge-0*: $x > 0 \Longrightarrow p\ n\ x \geq 0$
using *stirling-trapezium*[*of real n + x* **for** *n*]
by (*auto simp add*: *p-def intro*!: *sum-nonneg*)

qualified lemma *P-ge-0*: $x > 0 \Longrightarrow P\ x \geq 0$
by (*rule tendsto-lowerbound*[*OF p-LIMSEQ*])
(*insert p-ge-0*[*of x*], *simp-all*)

qualified lemma *p-upper-bound*:
assumes $x > 0\ n > 0$

```
  shows p n x ≤ 1/(12*x^2)
proof −
  from assms have p n x = (∑r<n. D (real r + x))
    by (simp add: p-def)
  also have ... ≤ (∑r<n. 1/(12*(real r + x)^2) − 1/(12 * (real (Suc r) + x)^2))
    using stirling-trapezium[of real r + x for r] assms
    by (intro sum-mono) (simp add: add-ac)
  also have ... = 1 / (12 * x²) − 1 / (12 * (real n + x)²)
    by (subst sum-lessThan-telescope') simp
  also have ... ≤ 1 / (12 * x^2) by simp
  finally show ?thesis .
qed

qualified lemma P-upper-bound:
  assumes x > 0
  shows   P x ≤ 1/(12*x^2)
proof (rule tendsto-upperbound)
  show eventually (λn. p n x ≤ 1 / (12 * x^2)) at-top
    using eventually-gt-at-top[of 0]
    by eventually-elim (use p-upper-bound[of x] assms in auto)
  show (λn. p n x) ⟶ P x
    by (simp add: assms p-LIMSEQ)
qed auto
```

We can now use this approximation of the Digamma function to approximate *ln-Gamma* (since the former is the derivative of the latter). We therefore define the function g from Jameson's article, which measures the difference between *ln-Gamma* and its approximation:

```
qualified definition g :: real ⇒ real where
  g x = ln-Gamma x − (x − 1/2) * ln x + x

qualified lemma DERIV-g: x > 0 ⟹ (g has-field-derivative −P x) (at x)
  unfolding g-def [abs-def] using Digamma-approx[of x]
  by (auto intro!: derivative-eq-intros simp: field-simps)

qualified lemma isCont-P:
  assumes x > 0
  shows   isCont P x
proof −
  define D' :: real ⇒ real
    where D' = (λx. − 1 / (2 * x^2 * (x+1)^2))
  have DERIV-D: (D has-field-derivative D' x) (at x) if x > 0 for x
    unfolding D-def [abs-def] D'-def T-def
    by (insert that, (rule derivative-eq-intros refl | simp)+)
       (simp add: power2-eq-square divide-simps, (simp add: algebra-simps)?)
  note this [THEN DERIV-chain2, derivative-intros]

  have (P has-field-derivative (∑n. D' (real n + x))) (at x)
```

unfolding *P-def* [*abs-def*]
proof (*rule has-field-derivative-series′*)
show *convex* {*x*/*2*<..} **by** *simp*
next
fix *n* :: *nat* **and** *y* :: *real* **assume** *y*: *y* ∈ {*x*/*2*<..}
with *assms* **have** *y* > *0* **by** *simp*
thus ((λ*a*. *D* (*real n* + *a*)) *has-real-derivative D′* (*real n* + *y*)) (*at y within* {*x*/*2*<..})
by (*auto intro*!: *derivative-eq-intros*)
next
from *assms D-summable*[*of x*] **show** *summable* (λ*n*. *D* (*real n* + *x*)) **by** *simp*
next
show *uniformly-convergent-on* {*x*/*2*<..} (λ*n x*. ∑ *i*<*n*. *D′* (*real i* + *x*))
proof (*rule Weierstrass-m-test′*)
fix *n* :: *nat* **and** *y* :: *real*
assume *y*: *y* ∈ {*x*/*2*<..}
with *assms* **have** *y* > *0* **by** *auto*
have *norm* (*D′* (*real n* + *y*)) = (*1* / (*2* ∗ (*y* + *real n*)^*2*)) ∗ (*1* / (*y* + *real* (*Suc n*))^*2*)
by (*simp add*: *D′-def add-ac*)
also from *y assms* **have** ... ≤ (*1* / (*2* ∗ (*x*/*2*)^*2*)) ∗ (*1* / (*real* (*Suc n*))^*2*)
by (*intro mult-mono divide-left-mono power-mono*) *simp-all*
also have *1* / (*real* (*Suc n*))^*2* = *inverse* ((*real* (*Suc n*))^*2*) **by** (*simp add*: *field-simps*)
finally show *norm* (*D′* (*real n* + *y*)) ≤ (*1* / (*2* ∗ (*x*/*2*)^*2*)) ∗ *inverse* ((*real* (*Suc n*))^*2*) **.**
next
show *summable* (λ*n*. (*1* / (*2* ∗ (*x*/*2*)^*2*)) ∗ *inverse* ((*real* (*Suc n*))^*2*))
by (*subst summable-Suc-iff*, *intro summable-mult inverse-power-summable*) *simp-all*
qed
qed (*insert assms*, *simp-all add*: *interior-open*)
thus *?thesis* **by** (*rule DERIV-isCont*)
qed

qualified lemma *P-continuous-on* [*THEN continuous-on-subset*]: *continuous-on* {*0*<..} *P*
by (*intro continuous-at-imp-continuous-on ballI isCont-P*) *auto*

qualified lemma *P-integrable*:
assumes *a*: *a* > *0*
shows *P integrable-on* {*a*..}
proof −
define *f* **where** *f* = (λ*n x*. *if x* ∈ {*a*..*real n*} *then P x else 0*)
show *P integrable-on* {*a*..}
proof (*rule dominated-convergence*)
fix *n* :: *nat*
from *a* **have** *P integrable-on* {*a*..*real n*}
by (*intro integrable-continuous-real P-continuous-on*) *auto*

```
    hence f n integrable-on {a..real n}
      by (rule integrable-eq) (simp add: f-def)
    thus f n integrable-on {a..}
      by (rule integrable-on-superset) (auto simp: f-def)
  next
    fix n :: nat
    show norm (f n x) ≤ of-real (1/12) ∗ (1 / xˆ2) if x ∈ {a..} for x
      using a P-ge-0 P-upper-bound by (auto simp: f-def)
  next
    show (λx::real. of-real (1/12) ∗ (1 / xˆ2)) integrable-on {a..}
      using has-integral-inverse-power-to-inf[of 2 a] a
      by (intro integrable-on-cmult-left) auto
  next
    show (λn. f n x) ⟶ P x if x∈{a..} for x
    proof −
      have eventually (λn. real n ≥ x) at-top
        using filterlim-real-sequentially by (simp add: filterlim-at-top)
      with that not-frequently have eventually (λn. f n x = P x) at-top
        by (force simp: f-def)
      thus (λn. f n x) ⟶ P x by (simp add: tendsto-eventually)
    qed
  qed
qed
```

```
qualified definition c :: real where c = integral {1..} (λx. −P x) + g 1
```

We can now give bounds on g:

```
qualified lemma g-bounds:
  assumes x: x ≥ 1
  shows   g x ∈ {c..c + 1/(12∗x)}
proof −
  from assms have int-nonneg: integral {x..} P ≥ 0
    by (intro Henstock-Kurzweil-Integration.integral-nonneg P-integrable)
       (auto simp: P-ge-0)
  have int-upper-bound: integral {x..} P ≤ 1/(12∗x)
  proof (rule has-integral-le)
    from x show (P has-integral integral {x..} P) {x..}
      by (intro integrable-integral P-integrable) simp-all
     from x has-integral-mult-right[OF has-integral-inverse-power-to-inf[of 2 x], of
1/12]
        show ((λx. 1/(12∗xˆ2)) has-integral (1/(12∗x))) {x..} by (simp add:
field-simps)
  qed (insert P-upper-bound x, simp-all)

  note DERIV-g [THEN DERIV-chain2, derivative-intros]
  from assms have int1: ((λx. −P x) has-integral (g x − g 1)) {1..x}
    by (intro fundamental-theorem-of-calculus)
       (auto simp: has-real-derivative-iff-has-vector-derivative [symmetric]
             intro!: derivative-eq-intros)
```

from *x* **have** *int2*: ((λ*x*. −*P x*) *has-integral integral* {*x*..} (λ*x*. −*P x*)) {*x*..}
by (*intro integrable-integral integrable-neg P-integrable*) *simp-all*
from *has-integral-Un*[*OF int1 int2*] *x*
have ((λ*x*. − *P x*) *has-integral g x* − *g 1* − *integral* {*x*..} *P*) ({*1*..*x*} ∪ {*x*..})
by (*simp add*: *max-def*)
also from *x* **have** {*1*..*x*} ∪ {*x*..} = {*1*..} **by** *auto*
finally have ((λ*x*. −*P x*) *has-integral g x* − *g 1* − *integral* {*x*..} *P*) {*1*..} .
moreover have ((λ*x*. −*P x*) *has-integral integral* {*1*..} (λ*x*. −*P x*)) {*1*..}
by (*intro integrable-integral integrable-neg P-integrable*) *simp-all*
ultimately have *g x* − *g 1* − *integral* {*x*..} *P* = *integral* {*1*..} (λ*x*. −*P x*)
by (*simp add*: *has-integral-unique*)
hence *g x* = *c* + *integral* {*x*..} *P* **by** (*simp add*: *c-def algebra-simps*)
with *int-upper-bound int-nonneg* **show** *g x* ∈ {*c*..*c* + *1*/(*12*∗*x*)} **by** *simp*
qed

Finally, we have bounds on *ln-Gamma*, *Gamma*, and *fact*.

qualified lemma *ln-Gamma-bounds-aux*:
x ≥ *1* ⟹ *ln-Gamma x* ≥ *c* + (*x* − *1*/*2*) ∗ *ln x* − *x*
x ≥ *1* ⟹ *ln-Gamma x* ≤ *c* + (*x* − *1*/*2*) ∗ *ln x* − *x* + *1*/(*12*∗*x*)
using *g-bounds*[*of x*] **by** (*simp-all add*: *g-def*)

qualified lemma *Gamma-bounds-aux*:
assumes *x*: *x* ≥ *1*
shows *Gamma x* ≥ *exp c* ∗ *x powr* (*x* − *1*/*2*) / *exp x*
Gamma x ≤ *exp c* ∗ *x powr* (*x* − *1*/*2*) / *exp x* ∗ *exp* (*1*/(*12*∗*x*))
proof −
have *exp* (*ln-Gamma x*) ≥ *exp* (*c* + (*x* − *1*/*2*) ∗ *ln x* − *x*)
by (*subst exp-le-cancel-iff*, *rule ln-Gamma-bounds-aux*) (*simp add*: *x*)
with *x* **show** *Gamma x* ≥ *exp c* ∗ *x powr* (*x* − *1*/*2*) / *exp x*
by (*simp add*: *Gamma-real-pos-exp exp-add exp-diff powr-def del*: *exp-le-cancel-iff*)
next
have *exp* (*ln-Gamma x*) ≤ *exp* (*c* + (*x* − *1*/*2*) ∗ *ln x* − *x* + *1*/(*12*∗*x*))
by (*subst exp-le-cancel-iff*, *rule ln-Gamma-bounds-aux*) (*simp add*: *x*)
with *x* **show** *Gamma x* ≤ *exp c* ∗ *x powr* (*x* − *1*/*2*) / *exp x* ∗ *exp* (*1*/(*12*∗*x*))
by (*simp add*: *Gamma-real-pos-exp exp-add exp-diff powr-def del*: *exp-le-cancel-iff*)
qed

qualified lemma *Gamma-asymp-equiv-aux*:
Gamma ∼[*at-top*] (λ*x*. *exp c* ∗ *x powr* (*x* − *1*/*2*) / *exp x*)
proof (*rule asymp-equiv-sandwich*)
include *asymp-equiv-syntax*
show *eventually* (λ*x*. *exp c* ∗ *x powr* (*x* − *1*/*2*) / *exp x* ≤ *Gamma x*) *at-top*
eventually (λ*x*. *exp c* ∗ *x powr* (*x* − *1*/*2*) / *exp x* ∗ *exp* (*1*/(*12*∗*x*)) ≥ *Gamma x*) *at-top*
using *eventually-ge-at-top*[*of 1*::*real*]
by (*eventually-elim*; *use Gamma-bounds-aux* **in** *force*)+
have ((λ*x*::*real*. *exp* (*1* / (*12* ∗ *x*))) ⟶ *exp 0*) *at-top*
by *real-asymp*
hence (λ*x*. *exp* (*1* / (*12* ∗ *x*))) ∼ (λ*x*. *1* :: *real*)

by (*intro asymp-equivI'*) *simp-all*
hence (*λx. exp c ∗ x powr* (*x* − *1* / *2*) / *exp x* ∗ *1*) ∼
(*λx. exp c ∗ x powr* (*x* − *1* / *2*) / *exp x* ∗ *exp* (*1* / (*12* ∗ *x*)))
by (*intro asymp-equiv-mult asymp-equiv-refl*) (*simp add*: *asymp-equiv-sym*)
thus (*λx. exp c ∗ x powr* (*x* − *1* / *2*) / *exp x*) ∼
(*λx. exp c ∗ x powr* (*x* − *1* / *2*) / *exp x* ∗ *exp* (*1* / (*12* ∗ *x*))) **by** *simp*
qed *simp-all*

qualified lemma *exp-1-powr-real* [*simp*]: *exp* (*1*::*real*) *powr x* = *exp x*
by (*simp add*: *powr-def*)

qualified lemma *fact-asymp-equiv-aux*:
fact ∼[*at-top*] (*λx. exp c ∗ sqrt* (*real x*) ∗ (*real x* / *exp 1*) *powr real x*)
proof −
include *asymp-equiv-syntax*
have *fact* ∼ (*λn. Gamma* (*real* (*Suc n*))) **by** (*simp add*: *Gamma-fact*)
also have *eventually* (*λn. Gamma* (*real* (*Suc n*)) = *real n* ∗ *Gamma* (*real n*))
at-top
using *eventually-gt-at-top*[*of 0*::*nat*]
by *eventually-elim* (*insert Gamma-plus1*[*of real n* **for** *n*],
auto simp: *add-ac of-nat-in-nonpos-Ints-iff*)
also have (*λn. Gamma* (*real n*)) ∼ (*λn. exp c ∗ real n powr* (*real n* − *1/2*) /
exp (*real n*))
by (*rule asymp-equiv-compose'*[*OF Gamma-asymp-equiv-aux*] *filterlim-real-sequentially*)+
also have *eventually* (*λn. real n* ∗ (*exp c ∗ real n powr* (*real n* − *1* / *2*) / *exp*
(*real n*)) =
exp c ∗ sqrt (*real n*) ∗ (*real n* / *exp 1*) *powr real n*) *at-top*
using *eventually-gt-at-top*[*of 0*::*nat*]
proof *eventually-elim*
fix *n* :: *nat* **assume** *n*: *n* > *0*
thus *real n* ∗ (*exp c ∗ real n powr* (*real n* − *1* / *2*) / *exp* (*real n*)) =
exp c ∗ sqrt (*real n*) ∗ (*real n* / *exp 1*) *powr real n*
by (*subst powr-diff*) (*simp-all add*: *powr-divide powr-half-sqrt field-simps*)
qed
finally show *?thesis* **by** − (*simp-all add*: *asymp-equiv-mult*)
qed

We cal also bound *Digamma* above and below.

lemma *Digamma-plus-1-gt-ln*:
assumes *x*: *x* > (*0* :: *real*)
shows *Digamma* (*x* + *1*) > *ln x*
proof −
have *0* < (*17* :: *real*)
by *simp*
also have *17* ≤ *12* ∗ *x* ˆ *2* + *28* ∗ *x* + *17*
using *x* **by** *auto*
finally have *0* < (*12* ∗ *x* ˆ *2* + *28* ∗ *x* + *17*) / (*12* ∗ (*x* + *1*) ˆ *2* ∗ (*1* + *2* ∗
x))
using *x* **by** (*intro divide-pos-pos mult-pos-pos zero-less-power*) *auto*

also have . . . = *2* / (*2* ∗ *x* + *1*) − *1* / (*2* ∗ (*x* + *1*)) − *1* / (*12* ∗ (*x* + *1*) ^ *2*)
  using *x* **by** (*simp add*: *divide-simps*) (*auto simp*: *field-simps power2-eq-square add-eq-0-iff*)
also have *2* / (*2* ∗ *x* + *1*) ≤ *ln* (*x* + *1*) − *ln x*
  using *ln-inverse-approx-ge*[*of x x* + *1*] *x* **by** *simp*
also have . . . − *1* / (*2* ∗ (*x* + *1*)) − *1* / (*12* ∗ (*x* + *1*) ^ *2*) ≤
    ln (*x* + *1*) − *ln x* − *1* / (*2* ∗ (*x* + *1*)) − *P* (*x* + *1*)
  using *P-upper-bound*[*of x* + *1*] *x* **by** (*intro diff-mono*) *auto*
also have . . . = *Digamma* (*x* + *1*) − *ln x*
  by (*subst Digamma-approx*) (*use x* **in** *auto*)
finally show *Digamma* (*x* + *1*) > *ln x*
  by *simp*
qed

lemma *Digamma-less-ln*:
  assumes *x*: *x* > (*0* :: *real*)
  shows *Digamma x* < *ln x*
proof −
  have *Digamma x* − *ln x* = − (*1* / (*2* ∗ *x*)) − *P x*
    by (*subst Digamma-approx*) (*use x* **in** *auto*)
  also have . . . < *0* − *P x*
    using *x* **by** (*intro diff-strict-right-mono*) *auto*
  also have . . . ≤ *0*
    using *P-ge-0*[*of x*] *x* **by** *simp*
  finally show *Digamma x* < *ln x*
    by *simp*
qed

We still need to determine the constant term c, which we do using Wallis' product formula:

$$\prod_{n=1}^{\infty} \frac{4n^2}{4n^2-1} = \frac{\pi}{2}$$

qualified lemma *powr-mult-2*: (*x*::*real*) > *0* ⟹ *x powr* (*y* ∗ *2*) = (*x*^*2*) *powr y*
  by (*subst mult.commute*, *subst powr-powr* [*symmetric*]) (*simp add*: *powr-numeral*)

qualified lemma *exp-mult-2*: *exp* (*y* ∗ *2* :: *real*) = *exp y* ∗ *exp y*
  by (*subst exp-add* [*symmetric*]) *simp*

qualified lemma *exp-c*: *exp c* = *sqrt* (*2*∗*pi*)
proof −
  include *asymp-equiv-syntax*
  define *p* **where** *p* = (λ*n*. ∏ *k*=*1*..*n*. (*4*∗*real k*^*2*) / (*4*∗*real k*^*2* − *1*))

  have *p-0* [*simp*]: *p 0* = *1* **by** (*simp add*: *p-def*)
  have *p-Suc*: *p* (*Suc n*) = *p n* ∗ (*4* ∗ *real* (*Suc n*)^*2*) / (*4* ∗ *real* (*Suc n*)^*2* − *1*)
    for *n* **unfolding** *p-def* **by** (*subst prod.nat-ivl-Suc′*) *simp-all*
  have *p*: *p* = (λ*n*. *16* ^ *n* ∗ *fact n* ^ *4* / (*fact* (*2* ∗ *n*))^2 / (*2* ∗ *real n* + *1*))
  proof

fix *n* :: *nat*
have *p n* = (∏ *k*=*1*..*n*. (*2*∗*real k*)^*2* / (*2*∗*real k* − *1*) / (*2* ∗ *real k* + *1*))
unfolding *p-def* **by** (*intro prod.cong refl*) (*simp add*: *field-simps power2-eq-square*)
also have ... = (∏ *k*=*1*..*n*. (*2*∗*real k*)^*2* / (*2*∗*real k* − *1*)) / (∏ *k*=*1*..*n*. (*2* ∗ *real* (*Suc k*) − *1*))
by (*simp add*: *prod-dividef prod.distrib add-ac*)
also have (∏ *k*=*1*..*n*. (*2* ∗ *real* (*Suc k*) − *1*)) = (∏ *k*=*Suc 1*..*Suc n*. (*2* ∗ *real k* − *1*))
by (*subst prod.atLeast-Suc-atMost-Suc-shift*) *simp-all*
also have ... = (∏ *k*=*1*..*n*. (*2* ∗ *real k* − *1*)) ∗ (*2* ∗ *real n* + *1*)
by (*induction n*) (*simp-all add*: *prod.nat-ivl-Suc*′)
also have (∏ *k* = *1*..*n*. (*2* ∗ *real k*)^2 / (*2* ∗ *real k* − *1*)) / ... =
(∏ *k* = *1*..*n*. (*2* ∗ *real k*)^*2* / (*2* ∗ *real k* − *1*)^*2*) / (*2* ∗ *real n* + *1*)
unfolding *power2-eq-square* **by** (*simp add*: *prod.distrib prod-dividef*)
also have (∏ *k* = *1*..*n*. (*2* ∗ *real k*)^*2* / (*2* ∗ *real k* − *1*)^*2*) =
(∏ *k* = *1*..*n*. (*2* ∗ *real k*)^*4* / ((*2*∗*real k*)∗(*2* ∗ *real k* − *1*))^*2*)
by (*rule prod.cong*) (*simp-all add*: *power2-eq-square eval-nat-numeral*)
also have ... = *16*^*n* ∗ *fact n*^*4* / (∏ *k*=*1*..*n*. (*2*∗*real k*) ∗ (*2*∗*real k* − *1*))^*2*
by (*simp add*: *prod.distrib prod-dividef fact-prod*
prod-power-distrib [*symmetric*] *prod-constant*)
also have (∏ *k*=*1*..*n*. (*2*∗*real k*) ∗ (*2*∗*real k* − *1*)) = *fact* (*2*∗*n*)
by (*induction n*) (*simp-all add*: *algebra-simps prod.nat-ivl-Suc*′)
finally show *p n* = *16* ^ *n* ∗ *fact n* ^ *4* / (*fact* (*2* ∗ *n*))^2 / (*2* ∗ *real n* + *1*) **.**
qed

have *p* ∼ (λ*n*. *16* ^ *n* ∗ *fact n* ^ *4* / (*fact* (*2* ∗ *n*))^2 / (*2* ∗ *real n* + *1*))
by (*simp add*: *p*)
also have ... ∼ (λ*n*. *16*^*n* ∗ (*exp c* ∗ *sqrt* (*real n*) ∗ (*real n* / *exp 1*) *powr real n*)^*4* /
(*exp c* ∗ *sqrt* (*real* (*2*∗*n*)) ∗ (*real* (*2*∗*n*) / *exp 1*) *powr real* (*2*∗*n*))^*2* /
(*2* ∗ *real n* + *1*)) (**is** - ∼ *?f*)
by (*intro asymp-equiv-mult asymp-equiv-divide asymp-equiv-refl mult-nat-left-at-top*
fact-asymp-equiv-aux asymp-equiv-power asymp-equiv-compose′[*OF fact-asymp-equiv-aux*])
simp-all
also have *eventually* (λ*n*. ... *n* = *exp c* ^ *2* / (*4* + *2*/*n*)) *at-top*
using *eventually-gt-at-top*[*of 0*::*nat*]
proof *eventually-elim*
fix *n* :: *nat* **assume** *n*: *n* > *0*
have [*simp*]: *16*^*n* = *4*^*n* ∗ (*4*^*n* :: *real*) **by** (*simp add*: *power-mult-distrib* [*symmetric*])
from *n* **have** *?f n* = *exp c* ^ *2* ∗ (*n* / (*2*∗(*2*∗*n*+*1*)))
by (*simp add*: *power-mult-distrib divide-simps powr-mult real-sqrt-power-even*)
(*simp add*: *field-simps power2-eq-square eval-nat-numeral powr-mult-2*
exp-mult-2 powr-realpow)
also from *n* **have** ... = *exp c* ^ *2* / (*4* + *2*/*n*) **by** (*simp add*: *field-simps*)
finally show *?f n* = ... **.**
qed

also have $(\lambda x.\ 4 + 2 \ / \ real\ x) \sim (\lambda x.\ 4)$
  by (*subst asymp-equiv-add-right*) *auto*
finally have $p \longrightarrow exp\ c\ \hat{}\ 2 \ / \ 4$
  by (*rule asymp-equivD-const*) (*simp-all add*: *asymp-equiv-divide*)
moreover have $p \longrightarrow pi \ / \ 2$ **unfolding** *p-def* **by** (*rule wallis*)
ultimately have $exp\ c\ \hat{}\ 2 \ / \ 4 = pi \ / \ 2$ **by** (*rule LIMSEQ-unique*)
hence $2 * pi = exp\ c\ \hat{}\ 2$ **by** *simp*
also have $sqrt\ (exp\ c\ \hat{}\ 2) = exp\ c$ **by** *simp*
finally show $exp\ c = sqrt\ (2 * pi)$ **..**
qed

qualified lemma *c*: $c = ln\ (2{*}pi) \ / \ 2$
proof −
  note *exp-c* [*symmetric*]
  also have $ln\ (exp\ c) = c$ **by** *simp*
  finally show *?thesis* **by** (*simp add*: *ln-sqrt*)
qed

This gives us the final bounds:

theorem *Gamma-bounds*:
  assumes $x \geq 1$
  shows $Gamma\ x \geq sqrt\ (2{*}pi/x) * (x \ / \ exp\ 1)\ powr\ x$ (**is** *?th1*)
    $Gamma\ x \leq sqrt\ (2{*}pi/x) * (x \ / \ exp\ 1)\ powr\ x * exp\ (1 \ / \ (12 * x))$ (**is**
?th2)
proof −
  from *assms* **have** $exp\ c * x\ powr\ (x - 1/2) \ / \ exp\ x = sqrt\ (2{*}pi/x) * (x \ / \ exp$
$1)\ powr\ x$
    by (*subst powr-diff*)
      (*simp add*: *exp-c real-sqrt-divide powr-divide powr-half-sqrt field-simps*)
  with *Gamma-bounds-aux*[*OF assms*] **show** *?th1 ?th2* **by** *simp-all*
qed

theorem *ln-Gamma-bounds*:
  assumes $x \geq 1$
  shows $ln\text{-}Gamma\ x \geq ln\ (2{*}pi/x) \ / \ 2 + x * ln\ x - x$ (**is** *?th1*)
    $ln\text{-}Gamma\ x \leq ln\ (2{*}pi/x) \ / \ 2 + x * ln\ x - x + 1/(12{*}x)$ (**is** *?th2*)
proof −
  from *ln-Gamma-bounds-aux*[*OF assms*] *assms* **show** *?th1 ?th2*
    by (*simp-all add*: *c field-simps ln-div*)
  from *assms* **have** $exp\ c * x\ powr\ (x - 1/2) \ / \ exp\ x = sqrt\ (2{*}pi/x) * (x \ / \ exp$
$1)\ powr\ x$
    by (*subst powr-diff*)
      (*simp add*: *exp-c real-sqrt-divide powr-divide powr-half-sqrt field-simps*)
qed

theorem *fact-bounds*:
  assumes $n > 0$
  shows $(fact\ n :: real) \geq sqrt\ (2{*}pi{*}n) * (n \ / \ exp\ 1)\ \hat{}\ n$ (**is** *?th1*)
    $(fact\ n :: real) \leq sqrt\ (2{*}pi{*}n) * (n \ / \ exp\ 1)\ \hat{}\ n * exp\ (1 \ / \ (12 * n))$ (**is**

```
?th2)
proof -
  from assms have n: real n ≥ 1 by simp
  from assms Gamma-plus1[of real n]
    have real n * Gamma (real n) = Gamma (real (Suc n))
    by (simp add: of-nat-in-nonpos-Ints-iff add-ac)
  also have Gamma (real (Suc n)) = fact n by (subst Gamma-fact [symmetric])
simp
  finally have *: fact n = real n * Gamma (real n) by simp

  have 2*pi/n = 2*pi*n / n^2 by (simp add: power2-eq-square)
  also have sqrt ... = sqrt (2*pi*n) / n by (subst real-sqrt-divide) simp-all
  also have real n * ... = sqrt (2*pi*n) by simp
  finally have **: real n * sqrt (2*pi/real n) = sqrt (2*pi*real n) .

  note *
  also note Gamma-bounds(2)[OF n]
  also have real n * (sqrt (2 * pi / real n) * (real n / exp 1) powr real n *
                exp (1 / (12 * real n))) =
             (real n * sqrt (2*pi/n)) * (n / exp 1) powr n * exp (1 / (12 * n))
    by (simp add: algebra-simps)
  also from n have (real n / exp 1) powr real n = (real n / exp 1) ^ n
    by (subst powr-realpow) simp-all
  also note **
  finally show ?th2 by - (insert assms, simp-all)

  have sqrt (2*pi*n) * (n / exp 1) powr n = n * (sqrt (2*pi/n) * (n / exp 1)
powr n)
    by (subst ** [symmetric]) (simp add: field-simps)
  also from assms have ... ≤ real n * Gamma (real n)
    by (intro mult-left-mono Gamma-bounds(1)) simp-all
  also from n have (real n / exp 1) powr real n = (real n / exp 1) ^ n
    by (subst powr-realpow) simp-all
  also note * [symmetric]
  finally show ?th1 .
qed

theorem ln-fact-bounds:
  assumes n > 0
  shows   ln (fact n :: real) ≥ ln (2*pi*n)/2 + n * ln n - n (is ?th1)
          ln (fact n :: real) ≤ ln (2*pi*n)/2 + n * ln n - n + 1/(12*real n) (is
?th2)
proof -
  have ln (fact n :: real) ≥ ln (sqrt (2*pi*real n)*(real n/exp 1)^n)
    using fact-bounds(1)[OF assms] assms by (subst ln-le-cancel-iff) auto
  also from assms have ln (sqrt (2*pi*real n)*(real n/exp 1)^n) = ln (2*pi*n)/2
+ n * ln n - n
    by (simp add: ln-mult ln-div ln-realpow ln-sqrt field-simps)
  finally show ?th1 .
```

next
  have *ln* (*fact n* :: *real*) ≤ *ln* (*sqrt* (*2*∗*pi*∗*real n*) ∗ (*real n*/*exp 1*)^*n* ∗ *exp* (*1*/(*12*∗*real n*)))
    using *fact-bounds*(*2*)[*OF assms*] *assms* **by** (*subst ln-le-cancel-iff*) *auto*
  also from *assms* **have** ... = *ln* (*2*∗*pi*∗*n*)/*2* + *n* ∗ *ln n* − *n* + *1*/(*12*∗*real n*)
    by (*simp add*: *ln-mult ln-div ln-realpow ln-sqrt field-simps*)
  finally show *?th2* **.**
qed

theorem *Gamma-asymp-equiv*:
  Gamma ∼[*at-top*] (λ*x*. *sqrt* (*2*∗*pi*/*x*) ∗ (*x* / *exp 1*) *powr x* :: *real*)
proof −
  note *Gamma-asymp-equiv-aux*
  also have *eventually* (λ*x*. *exp c* ∗ *x powr* (*x* − *1*/*2*) / *exp x* =
      sqrt (*2*∗*pi*/*x*) ∗ (*x* / *exp 1*) *powr x*) *at-top*
    using *eventually-gt-at-top*[*of 0*::*real*]
  proof *eventually-elim*
    fix *x* :: *real* **assume** *x*: *x* > *0*
    thus *exp c* ∗ *x powr* (*x* − *1*/*2*) / *exp x* = *sqrt* (*2*∗*pi*/*x*) ∗ (*x* / *exp 1*) *powr x*
      by (*subst powr-diff*)
        (*simp add*: *exp-c powr-half-sqrt powr-divide field-simps real-sqrt-divide*)
  qed
  finally show *?thesis* **.**
qed

theorem *fact-asymp-equiv*:
  fact ∼[*at-top*] (λ*n*. *sqrt* (*2*∗*pi*∗*n*) ∗ (*n* / *exp 1*) ^ *n* :: *real*)
proof −
  note *fact-asymp-equiv-aux*
  also have *eventually* (λ*n*. *exp c* ∗ *sqrt* (*real n*) = *sqrt* (*2* ∗ *pi* ∗ *real n*)) *at-top*
   using *eventually-gt-at-top*[*of 0*::*nat*] **by** *eventually-elim* (*simp add*: *exp-c real-sqrt-mult*)
  also have *eventually* (λ*n*. (*n* / *exp 1*) *powr n* = (*n* / *exp 1*) ^ *n*) *at-top*
   using *eventually-gt-at-top*[*of 0*::*nat*] **by** *eventually-elim* (*simp add*: *powr-realpow*)
  finally show *?thesis* **.**
qed

corollary *stirling-tendsto-sqrt-pi*:
  (λ*n*. *fact n* / (*sqrt* (*2* ∗ *n*) ∗ (*n* / *exp 1*) ^ *n*)) ⟶ *sqrt pi*
proof −
  have ∗: (λ*n*. *fact n* / (*sqrt* (*2* ∗ *pi* ∗ *n*) ∗ (*n* / *exp 1*) ^ *n*)) ⟶ *1*
    using *fact-asymp-equiv* **by** (*simp add*: *asymp-equiv-def*)
  have (λ*n*. *sqrt pi* ∗ *fact n* / (*sqrt* (*2* ∗ *pi* ∗ *real n*) ∗ (*real n* / *exp 1*) ^ *n*))
      = (λ*n*. *fact n* / (*sqrt* (*real* (*2* ∗ *n*)) ∗ (*real n* / *exp 1*) ^ *n*))
    by (*force simp add*: *divide-simps powr-realpow real-sqrt-mult*)
  with *tendsto-mult-left*[*OF* ∗, *of sqrt pi*] **show** *?thesis* **by** *simp*
qed

end

end

2 Complete asymptotics of the logarithmic Gamma function

theory *Gamma-Asymptotics*
imports
HOL−Complex-Analysis.Complex-Analysis
Bernoulli.Bernoulli-FPS
Bernoulli.Periodic-Bernpoly
Stirling-Formula
begin

2.1 Auxiliary Facts

lemma *stirling-limit-aux1*:
((λy. *Ln* (*1* + *z* ∗ *of-real y*) / *of-real y*) ⟶ *z*) (*at-right 0*) **for** *z* :: *complex*
proof (*cases z* = *0*)
case *True*
then show *?thesis* **by** *simp*
next
case *False*
have ((λy. *ln* (*1* + *z* ∗ *of-real y*)) *has-vector-derivative 1* ∗ *z*) (*at 0*)
by (*rule has-vector-derivative-real-field*) (*auto intro*!: *derivative-eq-intros*)
then have (λy. (*Ln* (*1* + *z* ∗ *of-real y*) − *of-real y* ∗ *z*) / *of-real* |y|) −*0*→ *0*
by (*auto simp add*: *has-vector-derivative-def has-derivative-def netlimit-at scaleR-conv-of-real field-simps*)
then have ((λy. (*Ln* (*1* + *z* ∗ *of-real y*) − *of-real y* ∗ *z*) / *of-real* |y|) ⟶ *0*) (*at-right 0*)
by (*rule filterlim-mono*[*OF - - at-le*]) *simp-all*
also have *?this* ⟷ ((λy. *Ln* (*1* + *z* ∗ *of-real y*) / (*of-real y*) − *z*) ⟶ *0*) (*at-right 0*)
using *eventually-at-right-less*[*of 0*::*real*]
by (*intro filterlim-cong refl*) (*auto elim*!: *eventually-mono simp*: *field-simps*)
finally show *?thesis* **by** (*simp only*: *LIM-zero-iff*)
qed

lemma *stirling-limit-aux2*:
((λy. *y* ∗ *Ln* (*1* + *z* / *of-real y*)) ⟶ *z*) *at-top* **for** *z* :: *complex*
using *stirling-limit-aux1*[*of z*] **by** (*subst filterlim-at-top-to-right*) (*simp add*: *field-simps*)

lemma *Union-atLeastAtMost*:
assumes *N* > *0*
shows (⋃*n*∈{*0*..<*N*}. {*real n*..*real* (*n* + *1*)}) = {*0*..*real N*}
proof (*intro equalityI subsetI*)
fix *x* **assume** *x*: *x* ∈ {*0*..*real N*}
thus *x* ∈ (⋃*n*∈{*0*..<*N*}. {*real n*..*real* (*n* + *1*)})
proof (*cases x* = *real N*)

case *True*
with *assms* **show** *?thesis* **by** (*auto intro*!: *bexI*[*of* - $N - 1$])
next
case *False*
with *x* **have** *x*: $x \geq 0$ $x < real\ N$ **by** *simp-all*
hence $x \geq real\ (nat\ \lfloor x \rfloor)$ $x \leq real\ (nat\ \lfloor x \rfloor + 1)$ **by** *linarith*+
moreover from *x* **have** $nat\ \lfloor x \rfloor < N$ **by** *linarith*
ultimately have $\exists\, n \in \{0..<N\}.\ x \in \{real\ n..real\ (n + 1)\}$
by (*intro bexI*[*of* - *nat* $\lfloor x \rfloor$]) *simp-all*
thus *?thesis* **by** *blast*
qed
qed *auto*

2.2 Cones in the complex plane

definition *complex-cone* :: *real* ⇒ *real* ⇒ *complex set* **where**
complex-cone a b = $\{z.\ \exists\, y \in \{a..b\}.\ z = rcis\ (norm\ z)\ y\}$

abbreviation *complex-cone′* :: *real* ⇒ *complex set* **where**
complex-cone′ a ≡ *complex-cone* $(-a)$ *a*

lemma *zero-in-complex-cone* [*simp*, *intro*]: $a \leq b \Longrightarrow 0 \in$ *complex-cone a b*
by (*auto simp*: *complex-cone-def*)

lemma *complex-coneE*:
assumes $z \in$ *complex-cone a b*
obtains *r* α **where** $r \geq 0$ $\alpha \in \{a..b\}$ $z = rcis\ r\ \alpha$
proof −
from *assms* **obtain** *y* **where** $y \in \{a..b\}$ $z = rcis\ (norm\ z)\ y$
unfolding *complex-cone-def* **by** *auto*
thus *?thesis* **using** *that*[*of norm z y*] **by** *auto*
qed

lemma *arg-cis* [*simp*]:
assumes $x \in \{-pi<..pi\}$
shows $Arg\ (cis\ x) = x$
using *assms* **by** (*intro cis-Arg-unique*) *auto*

lemma *arg-mult-of-real-left* [*simp*]:
assumes $r > 0$
shows $Arg\ (of\text{-}real\ r * z) = Arg\ z$
proof (*cases* $z = 0$)
case *False*
thus *?thesis*
using *Arg-bounded*[*of z*] *assms*
by (*intro cis-Arg-unique*) (*auto simp*: *sgn-mult sgn-of-real cis-Arg*)
qed *auto*

lemma *arg-mult-of-real-right* [*simp*]:

```
  assumes r > 0
  shows   Arg (z * of-real r) = Arg z
  by (subst mult.commute, subst arg-mult-of-real-left) (simp-all add: assms)

lemma arg-rcis [simp]:
  assumes x ∈ {−pi<..pi} r > 0
  shows   Arg (rcis r x) = x
  using assms by (simp add: rcis-def)

lemma rcis-in-complex-cone [intro]:
  assumes α ∈ {a..b} r ≥ 0
  shows   rcis r α ∈ complex-cone a b
  using assms by (auto simp: complex-cone-def)

lemma arg-imp-in-complex-cone:
  assumes Arg z ∈ {a..b}
  shows   z ∈ complex-cone a b
proof −
  have z = rcis (norm z) (Arg z)
    by (simp add: rcis-cmod-Arg)
  also have ... ∈ complex-cone a b
    using assms by auto
  finally show ?thesis .
qed

lemma complex-cone-altdef:
  assumes −pi < a a ≤ b b ≤ pi
  shows   complex-cone a b = insert 0 {z. Arg z ∈ {a..b}}
proof (intro equalityI subsetI)
  fix z assume z ∈ complex-cone a b
  then obtain r α where *: r ≥ 0 α ∈ {a..b} z = rcis r α
    by (auto elim: complex-coneE)
  have Arg z ∈ {a..b} if [simp]: z ≠ 0
  proof −
    have r > 0 using that * by (subst (asm) *) auto
    hence α ∈ {a..b}
      using *(1,2) assms by (auto simp: *(1))
    moreover from assms *(2) have α ∈ {−pi<..pi}
      by auto
    ultimately show ?thesis using *(3) ‹r > 0›
      by (subst *) auto
  qed
  thus z ∈ insert 0 {z. Arg z ∈ {a..b}}
    by auto
qed (use assms in ‹auto intro: arg-imp-in-complex-cone›)

lemma nonneg-of-real-in-complex-cone [simp, intro]:
  assumes x ≥ 0 a ≤ 0 0 ≤ b
  shows   of-real x ∈ complex-cone a b
```

proof −
  from *assms* **have** *rcis x 0* ∈ *complex-cone a b*
    by (*intro rcis-in-complex-cone*) *auto*
  thus *?thesis* **by** *simp*
qed

lemma *one-in-complex-cone* [*simp*, *intro*]: *a* ≤ *0* ⟹ *0* ≤ *b* ⟹ *1* ∈ *complex-cone a b*
  using *nonneg-of-real-in-complex-cone*[*of 1*] **by** (*simp del*: *nonneg-of-real-in-complex-cone*)

lemma *of-nat-in-complex-cone* [*simp*, *intro*]: *a* ≤ *0* ⟹ *0* ≤ *b* ⟹ *of-nat n* ∈ *complex-cone a b*
  using *nonneg-of-real-in-complex-cone*[*of real n*] **by** (*simp del*: *nonneg-of-real-in-complex-cone*)

2.3 Another integral representation of the Beta function

lemma *complex-cone-inter-nonpos-Reals*:
  assumes −*pi* < *a a* ≤ *b b* < *pi*
  shows *complex-cone a b* ∩ $\mathbb{R}_{\leq 0}$ = {*0*}
proof (*safe elim*!: *nonpos-Reals-cases*)
  fix *x* :: *real*
  assume *complex-of-real x* ∈ *complex-cone a b x* ≤ *0*
  hence ¬(*x* < *0*)
    using *assms* **by** (*intro notI*) (*auto simp*: *complex-cone-altdef*)
  with ‹*x* ≤ *0*› **show** *complex-of-real x* = *0* **by** *auto*
qed (*use assms* **in** *auto*)

theorem
  assumes *a*: *a* > *0* **and** *b*: *b* > (*0* :: *real*)
  shows *has-integral-Beta-real′*:
      ((λ*u. u powr* (*b* − *1*) / (*1* + *u*) *powr* (*a* + *b*)) *has-integral Beta a b*) {*0*<..}
    and *Beta-conv-nn-integral*:
      Beta a b = (∫⁺*u. ennreal* (*indicator* {*0*<..} *u* ∗ *u powr* (*b* − *1*) / (*1* + *u*) *powr* (*a* + *b*)) ∂*lborel*)
proof −
  define *I* **where**
    I = (∫⁺*u. ennreal* (*indicator* {*0*<..} *u* ∗ *u powr* (*b* − *1*) / (*1* + *u*) *powr* (*a* + *b*)) ∂*lborel*)
  have *Gamma* (*a* + *b*) > *0 Beta a b* > *0*
    using *assms* **by** (*simp-all add*: *add-pos-pos Beta-def*)
  from *a b* **have** *ennreal* (*Gamma a* ∗ *Gamma b*) =
    (∫⁺ *t. ennreal* (*indicator* {*0*..} *t* ∗ *t powr* (*a* − *1*) / *exp t*) ∂*lborel*) ∗
    (∫⁺ *t. ennreal* (*indicator* {*0*..} *t* ∗ *t powr* (*b* − *1*) / *exp t*) ∂*lborel*)
    by (*subst ennreal-mult′*) (*simp-all add*: *Gamma-conv-nn-integral-real*)
  also have ... = (∫⁺*t.* ∫⁺*u. ennreal* (*indicator* {*0*..} *t* ∗ *t powr* (*a* − *1*) / *exp t*) ∗
        ennreal (*indicator* {*0*..} *u* ∗ *u powr* (*b* − *1*) / *exp u*) ∂*lborel* ∂*lborel*)
    by (*simp add*: *nn-integral-cmult nn-integral-multc*)

also have ... = (∫+t. *indicator* {0<..} *t* ∗ (∫+u. *indicator* {0..} *u* ∗ *t powr* (*a* − 1) ∗ *u powr* (*b* − 1)
/ *exp* (*t* + *u*) ∂*lborel*) ∂*lborel*)
by (*intro nn-integral-cong-AE AE-I*[*of* - - {0}])
(*auto simp*: *indicator-def divide-ennreal ennreal-mult′* [*symmetric*] *exp-add mult-ac*)
also have ... = (∫+t. *indicator* {0<..} *t* ∗ (∫+u. *indicator* {0..} *u* ∗ *t powr* (*a* − 1) ∗ *u powr* (*b* − 1)
/ *exp* (*t* + *u*)
∂(*density* (*distr lborel borel* ((∗) *t*)) (λ*x*. *ennreal* |*t*|))) ∂*lborel*)
by (*intro nn-integral-cong mult-indicator-cong*, *subst lborel-distr-mult′* [*symmetric*]) *auto*
also have ... = (∫+(*t*::*real*). *indicator* {0<..} *t* ∗ (∫+u.
indicator {0..} (*u* ∗ *t*) ∗ *t powr a* ∗
(*u* ∗ *t*) *powr* (*b* − 1) / *exp* (*t* + *t* ∗ *u*) ∂*lborel*) ∂*lborel*)
by (*intro nn-integral-cong mult-indicator-cong*)
(*auto simp*: *nn-integral-density nn-integral-distr algebra-simps powr-diff*
simp flip: *ennreal-mult*)
also have ... = (∫+(*t*::*real*). ∫+u. *indicator* ({0<..}×{0..}) (*t*, *u*) ∗
t powr a ∗ (*u* ∗ *t*) *powr* (*b* − 1) / *exp* (*t* ∗ (*u* + 1)) ∂*lborel* ∂*lborel*)
by (*subst nn-integral-cmult* [*symmetric*], *simp*, *intro nn-integral-cong*)
(*auto simp*: *indicator-def zero-le-mult-iff algebra-simps*)
also have ... = (∫+(*t*::*real*). ∫+u. *indicator* ({0<..}×{0..}) (*t*, *u*) ∗
t powr (*a* + *b* − 1) ∗ *u powr* (*b* − 1) / *exp* (*t* ∗ (*u* + 1)) ∂*lborel* ∂*lborel*)
by (*intro nn-integral-cong*) (*auto simp*: *powr-add powr-diff indicator-def powr-mult field-simps*)
also have ... = (∫+(*u*::*real*). ∫+t. *indicator* ({0<..}×{0..}) (*t*, *u*) ∗
t powr (*a* + *b* − 1) ∗ *u powr* (*b* − 1) / *exp* (*t* ∗ (*u* + 1)) ∂*lborel* ∂*lborel*)
by (*rule lborel-pair.Fubini′*) *auto*
also have ... = (∫+(*u*::*real*). *indicator* {0..} *u* ∗ (∫+t. *indicator* {0<..} *t* ∗
t powr (*a* + *b* − 1) ∗ *u powr* (*b* − 1) / *exp* (*t* ∗ (*u* + 1)) ∂*lborel*) ∂*lborel*)
by (*intro nn-integral-cong mult-indicator-cong*) (*auto simp*: *indicator-def*)
also have ... = (∫+(*u*::*real*). *indicator* {0<..} *u* ∗ (∫+t. *indicator* {0<..} *t* ∗
t powr (*a* + *b* − 1) ∗ *u powr* (*b* − 1) / *exp* (*t* ∗ (*u* + 1)) ∂*lborel*) ∂*lborel*)
by (*intro nn-integral-cong-AE AE-I*[*of* - - {0}]) (*auto simp*: *indicator-def*)
also have ... = (∫+(*u*::*real*). *indicator* {0<..} *u* ∗ (∫+t. *indicator* {0<..} *t* ∗
t powr (*a* + *b* − 1) ∗ *u powr* (*b* − 1) / *exp* (*t* ∗ (*u* + 1))
∂(*density* (*distr lborel borel* ((∗) (1/(1+*u*)))) (λ*x*. *ennreal* |1/(1+*u*)|))) ∂*lborel*)
by (*intro nn-integral-cong mult-indicator-cong*, *subst lborel-distr-mult′* [*symmetric*]) *auto*
also have ... = (∫+(*u*::*real*). *indicator* {0<..} *u* ∗
(∫+t. *ennreal* (1 / (*u* + 1)) ∗ *ennreal* (*indicator* {0<..} (*t* / (*u* + 1)) ∗
(*t* / (1+*u*)) *powr* (*a* + *b* − 1) ∗ *u powr* (*b* − 1) / *exp t*)

∂lborel) *∂lborel*)
by (*intro nn-integral-cong mult-indicator-cong*)
(*auto simp*: *nn-integral-distr nn-integral-density add-ac*)
also have ... = ($\int^+$*u*. $\int^+$*t*. *indicator* ({*0*<..}×{*0*<..}) (*u*, *t*) ∗
1/(*u*+*1*) ∗ (*t* / (*u*+*1*)) *powr* (*a* + *b* − *1*) ∗ *u powr* (*b* − *1*) / *exp t*
∂lborel ∂lborel)
by (*subst nn-integral-cmult* [*symmetric*], *simp*, *intro nn-integral-cong*)
(*auto simp*: *indicator-def field-simps divide-ennreal simp flip*: *ennreal-mult ennreal-mult'*)
also have ... = ($\int^+$*u*. $\int^+$*t*. *ennreal* (*indicator* {*0*<..} *u* ∗ *u powr* (*b* − *1*) / (*1* + *u*) *powr* (*a* + *b*)) ∗
ennreal (*indicator* {*0*<..} *t* ∗ *t powr* (*a* + *b* − *1*) / *exp t*)
∂lborel ∂lborel)
by (*intro nn-integral-cong*)
(*auto simp*: *indicator-def powr-add powr-diff powr-divide powr-minus divide-simps add-ac*
simp flip: *ennreal-mult*)
also have ... = *I* ∗ ($\int^+$*t*. *indicator* {*0*<..} *t* ∗ *t powr* (*a* + *b* − *1*) / *exp t ∂lborel*)
by (*simp add*: *nn-integral-cmult nn-integral-multc I-def*)
also have ($\int^+$*t*. *indicator* {*0*<..} *t* ∗ *t powr* (*a* + *b* − *1*) / *exp t ∂lborel*) =
ennreal (*Gamma* (*a* + *b*))
using *assms*
by (*subst Gamma-conv-nn-integral-real*)
(*auto intro*!: *nn-integral-cong-AE*[*OF AE-I*[*of - - {0}*]]
simp: *indicator-def split*: *if-splits split-of-bool-asm*)
finally have *ennreal* (*Gamma a* ∗ *Gamma b*) = *I* ∗ *ennreal* (*Gamma* (*a* + *b*)) **.**
hence *ennreal* (*Gamma a* ∗ *Gamma b*) / *ennreal* (*Gamma* (*a* + *b*)) =
I ∗ *ennreal* (*Gamma* (*a* + *b*)) / *ennreal* (*Gamma* (*a* + *b*)) **by** *simp*
also have ... = *I*
using ‹*Gamma* (*a* + *b*) > *0*› **by** (*intro ennreal-mult-divide-eq*) *auto*
also have *ennreal* (*Gamma a* ∗ *Gamma b*) / *ennreal* (*Gamma* (*a* + *b*)) =
ennreal (*Gamma a* ∗ *Gamma b* / *Gamma* (*a* + *b*))
using *assms* **by** (*intro divide-ennreal*) *auto*
also have ... = *ennreal* (*Beta a b*)
by (*simp add*: *Beta-def*)
finally show ∗: *ennreal* (*Beta a b*) = *I* **.**

define *f* **where** *f* = (λ*u*. *u powr* (*b* − *1*) / (*1* + *u*) *powr* (*a* + *b*))
have ((λ*u*. *indicator* {*0*<..} *u* ∗ *f u*) *has-integral Beta a b*) *UNIV*
using ∗ ‹*Beta a b* > *0*›
by (*subst has-integral-iff-nn-integral-lebesgue*)
(*auto simp*: *f-def measurable-completion nn-integral-completion I-def mult-ac*)
also have (λ*u*. *indicator* {*0*<..} *u* ∗ *f u*) = (λ*u*. *if u* ∈ {*0*<..} *then f u else 0*)
by (*auto simp*: *fun-eq-iff*)
also have (... *has-integral Beta a b*) *UNIV* ⟷ (*f has-integral Beta a b*) {*0*<..}
by (*rule has-integral-restrict-UNIV*)
finally show ... **by** (*simp add*: *f-def*)
qed

lemma *has-integral-Beta2*:
fixes *a* :: *real*
assumes $a < -1/2$
shows $((\lambda x.\ (1 + x\ \hat{}\ 2)\ powr\ a)\ has\text{-}integral\ Beta\ (-\ a - 1\ /\ 2)\ (1\ /\ 2)\ /\ 2)\ \{0<..\}$
proof −
define *f* **where** $f = (\lambda u.\ u\ powr\ (-1/2)\ /\ (1 + u)\ powr\ (-a))$
define *C* **where** $C = Beta\ (-a-1/2)\ (1/2)$
have *I*: $(f\ has\text{-}integral\ C)\ \{0<..\}$
using *has-integral-Beta-real'*[*of* −*a*−*1/2 1/2*] *assms*
by (*simp-all add*: *diff-divide-distrib f-def C-def*)

define *g* **where** $g = (\lambda x.\ x\ \hat{}\ 2 :: real)$
have *bij*: *bij-betw g* {*0*<..} {*0*<..}
by (*intro bij-betwI*[*of - - - sqrt*]) (*auto simp*: *g-def*)

have (*f absolutely-integrable-on g* ‘ {*0*<..} ∧ *integral* (*g* ‘ {*0*<..}) *f* = *C*)
using *I bij* **by** (*simp add*: *bij-betw-def has-integral-iff absolutely-integrable-on-def f-def*)
also have *?this* ⟷ ((λ*x*. |*2* ∗ *x*| ∗ *f* (*g x*)) *absolutely-integrable-on* {*0*<..} ∧
integral {*0*<..} (λ*x*. |*2* ∗ *x*| ∗ *f* (*g x*)) = *C*)
using *bij* **by** (*intro has-absolute-integral-change-of-variables-1'* [*symmetric*])
(*auto intro*!: *derivative-eq-intros simp*: *g-def bij-betw-def*)
finally have ((λ*x*. |*2* ∗ *x*| ∗ *f* (*g x*)) *has-integral C*) {*0*<..}
by (*simp add*: *absolutely-integrable-on-def f-def has-integral-iff*)
also have *?this* ⟷ $((\lambda x::real.\ 2 * (1 + x^2)\ powr\ a)\ has\text{-}integral\ C)\ \{0<..\}$
by (*intro has-integral-cong*) (*auto simp*: *f-def g-def powr-def exp-minus ln-realpow field-simps*)
finally have $((\lambda x::real.\ 1/2 * (2 * (1 + x^2)\ powr\ a))\ has\text{-}integral\ 1/2 * C)\ \{0<..\}$
by (*intro has-integral-mult-right*)
thus *?thesis* **by** (*simp add*: *C-def*)
qed

lemma *has-integral-Beta3*:
fixes *a b* :: *real*
assumes $a < -1/2$ **and** $b > 0$
shows $((\lambda x.\ (b + x\ \hat{}\ 2)\ powr\ a)\ has\text{-}integral$
$Beta\ (-a - 1/2)\ (1/2)\ /\ 2 * b\ powr\ (a + 1/2))\ \{0<..\}$
proof −
define *C* **where** $C = Beta\ (-\ a - 1\ /\ 2)\ (1\ /\ 2)\ /\ 2$
have *int*: *nn-integral lborel* $(\lambda x.\ indicator\ \{0<..\}\ x * (1 + x\ \hat{}\ 2)\ powr\ a) = C$
using *nn-integral-has-integral-lebesgue*[*OF - has-integral-Beta2*[*OF assms*(*1*)]]
by (*auto simp*: *C-def*)
have *nn-integral lborel* $(\lambda x.\ indicator\ \{0<..\}\ x * (b + x\ \hat{}\ 2)\ powr\ a) =$
$(\int^{+} x.\ ennreal\ (indicat\text{-}real\ \{0<..\}\ (x * sqrt\ b) * (b + (x * sqrt\ b)^2)\ powr\ a * sqrt\ b)\ \partial lborel)$
using *assms*

by (*subst lborel-distr-mult′[of sqrt b]*)
(*auto simp*: *nn-integral-density nn-integral-distr mult-ac simp flip*: *ennreal-mult*)
also have ... = (∫ +*x*. *ennreal* (*indicat-real* {*0*<..} *x* ∗ (*b* ∗ (*1* + *x* ^ *2*)) *powr a* ∗ *sqrt b*) ∂*lborel*)
using *assms*
by (*intro nn-integral-cong*) (*auto simp*: *indicator-def field-simps zero-less-mult-iff*)
also have ... = (∫ +*x*. *ennreal* (*indicat-real* {*0*<..} *x* ∗ *b powr* (*a* + *1*/*2*) ∗ (*1* + *x* ^ *2*) *powr a*) ∂*lborel*)
using *assms*
by (*intro nn-integral-cong*) (*auto simp*: *indicator-def powr-add powr-half-sqrt powr-mult*)
also have ... = *b powr* (*a* + *1*/*2*) ∗ (∫ +*x*. *ennreal* (*indicat-real* {*0*<..} *x* ∗ (*1* + *x* ^ *2*) *powr a*) ∂*lborel*)
using *assms* **by** (*subst nn-integral-cmult* [*symmetric*]) (*simp-all add*: *mult-ac flip*: *ennreal-mult*)
also have (∫ +*x*. *ennreal* (*indicat-real* {*0*<..} *x* ∗ (*1* + *x* ^ *2*) *powr a*) ∂*lborel*) = *C*
using *int* **by** *simp*
also have *ennreal* (*b powr* (*a* + *1*/*2*)) ∗ *ennreal C* = *ennreal* (*C* ∗ *b powr* (*a* + *1*/*2*))
using *assms* **by** (*subst ennreal-mult*) (*auto simp*: *C-def mult-ac Beta-def*)
finally have ∗: (∫ + *x*. *ennreal* (*indicat-real* {*0*<..} *x* ∗ (*b* + x^2) *powr a*) ∂*lborel*) = ...
hence ((λ*x*. *indicator* {*0*<..} *x* ∗ (*b* + *x*^*2*) *powr a*) *has-integral C* ∗ *b powr* (*a* + *1*/*2*)) *UNIV*
using *assms*
by (*subst has-integral-iff-nn-integral-lebesgue*)
(*auto simp*: *C-def measurable-completion nn-integral-completion Beta-def*)
also have (λ*x*. *indicator* {*0*<..} *x* ∗ (*b* + *x*^*2*) *powr a*) =
(λ*x*. *if x* ∈ {*0*<..} *then* (*b* + *x*^*2*) *powr a else 0*)
by (*auto simp*: *fun-eq-iff*)
finally show *?thesis*
by (*subst* (*asm*) *has-integral-restrict-UNIV*) (*auto simp*: *C-def*)
qed

2.4 Asymptotics of the real $\log \Gamma$ function and its derivatives

This is the error term that occurs in the expansion of *ln-Gamma*. It can be shown to be of order $O(s^{-n})$.

definition *stirling-integral* :: *nat* ⇒ ′*a* :: {*real-normed-div-algebra*, *banach*} ⇒ ′*a*
where
stirling-integral n s =
lim (λ*N*. *integral* {*0*..*N*} (λ*x*. *of-real* (*pbernpoly n x*) / (*of-real x* + *s*) ^ *n*))

context
fixes *s* :: *complex* **assumes** *s*: *s* ∉ $\mathbb{R}_{\leq 0}$
fixes *approx* :: *nat* ⇒ *complex*
defines *approx* ≡ (λ*N*.
(∑ *n* = *1*..<*N*. *s* / *of-nat n* − *ln* (*1* + *s* / *of-nat n*)) − (*euler-mascheroni* ∗ *s*

+ *ln s*) − — ⟶ *ln-Gamma s*
(*ln-Gamma* (*of-nat N*) − *ln* (*2* ∗ *pi* / *of-nat N*) / *2* − *of-nat N* ∗ *ln* (*of-nat N*) + *of-nat N*) − — ⟶ *0*
s ∗ (*harm* (*N* − *1*) − *ln* (*of-nat* (*N* − *1*)) − *euler-mascheroni*) + — ⟶ *0*
s ∗ (*ln* (*of-nat N* + *s*) − *ln* (*of-nat* (*N* − *1*))) − — ⟶ *0*
(*1/2*) ∗ (*ln* (*of-nat N* + *s*) − *ln* (*of-nat N*)) + — ⟶ *0*
of-nat N ∗ (*ln* (*of-nat N* + *s*) − *ln* (*of-nat N*)) − — ⟶ *s*
(*s* − *1/2*) ∗ *ln s* − *ln* (*2* ∗ *pi*) / *2*)
begin

qualified lemma
assumes *N*: *N* > *0*
shows *integrable-pbernpoly-1*:
(λ*x*. *of-real* (−*pbernpoly 1 x*) / (*of-real x* + *s*)) *integrable-on* {*0*..*real N*}
and *integral-pbernpoly-1-aux*:
integral {*0*..*real N*} (λ*x*. −*of-real* (*pbernpoly 1 x*) / (*of-real x* + *s*)) = *approx N*
and *has-integral-pbernpoly-1*:
((λ*x*. *pbernpoly 1 x* /(*x* + *s*)) *has-integral*
(∑*m*<*N*. (*of-nat m* + *1* / *2* + *s*) ∗ (*ln* (*of-nat m* + *s*) −
ln (*of-nat m* + *1* + *s*)) + *1*)) {*0*..*real N*}
proof −
let *?A* = (λ*n*. {*of-nat n*..*of-nat* (*n*+*1*)}) ' {*0*..<*N*}
have *has-integral*:
((λ*x*. −*pbernpoly 1 x* / (*x* + *s*)) *has-integral*
(*of-nat n* + *1/2* + *s*) ∗ (*ln* (*of-nat* (*n* + *1*) + *s*) − *ln* (*of-nat n* + *s*)) − *1*)
{*of-nat n*..*of-nat* (*n* + *1*)} **for** *n*
proof (*rule has-integral-spike*)
have ((λ*x*. (*of-nat n* + *1/2* + *s*) ∗ (*1* / (*of-real x* + *s*)) − *1*) *has-integral*
(*of-nat n* + *1/2* + *s*) ∗ (*ln* (*of-real* (*real* (*n* + *1*)) + *s*) − *ln* (*of-real* (*real n*) + *s*)) − *1*)
{*of-nat n*..*of-nat* (*n* + *1*)}
using *s has-integral-const-real*[*of 1 of-nat n of-nat* (*n* + *1*)]
by (*intro has-integral-diff has-integral-mult-right fundamental-theorem-of-calculus*)
(*auto intro*!: *derivative-eq-intros has-vector-derivative-real-field*
simp: *has-real-derivative-iff-has-vector-derivative* [*symmetric*] *field-simps*
complex-nonpos-Reals-iff)
thus ((λ*x*. (*of-nat n* + *1/2* + *s*) ∗ (*1* / (*of-real x* + *s*)) − *1*) *has-integral*
(*of-nat n* + *1/2* + *s*) ∗ (*ln* (*of-nat* (*n* + *1*) + *s*) − *ln* (*of-nat n* + *s*)) − *1*)
{*of-nat n*..*of-nat* (*n* + *1*)} **by** *simp*

show −*pbernpoly 1 x* / (*x* + *s*) = (*of-nat n* + *1/2* + *s*) ∗ (*1* / (*x* + *s*)) − *1*
if *x* ∈ {*of-nat n*..*of-nat* (*n* + *1*)} − {*of-nat* (*n* + *1*)} **for** *x*
proof −
have *x*: *x* ≥ *real n x* < *real* (*n* + *1*) **using** *that* **by** *simp-all*
hence *floor x* = *int n* **by** *linarith*
moreover from *s x* **have** *complex-of-real x* ≠ −*s*

by (*auto simp add*: *complex-eq-iff complex-nonpos-Reals-iff simp del*: *of-nat-Suc*)
ultimately show −*pbernpoly 1 x* / (*x* + *s*) = (*of-nat n* + *1*/*2* + *s*) ∗ (*1* / (*x* + *s*)) − *1*
by (*auto simp*: *pbernpoly-def bernpoly-def frac-def divide-simps add-eq-0-iff2*)
qed
qed *simp-all*
hence ∗: ⋀*I*. *I*∈*?A* ⟹ ((λ*x*. −*pbernpoly 1 x* / (*x* + *s*)) *has-integral*
(*Inf I* + *1*/*2* + *s*) ∗ (*ln* (*Inf I* + *1* + *s*) − *ln* (*Inf I* + *s*)) − *1*) *I*
by (*auto simp*: *add-ac*)
have ((λ*x*. − *pbernpoly 1 x* / (*x* + *s*)) *has-integral*
(∑*I*∈*?A*. (*Inf I* + *1* / *2* + *s*) ∗ (*ln* (*Inf I* + *1* + *s*) − *ln* (*Inf I* + *s*)) − *1*))
(⋃*n*∈{*0*..<*N*}. {*real n*..*real* (*n* + *1*)}) (**is** (- *has-integral ?i*) -)
apply (*intro has-integral-Union* ∗ *finite-imageI*)
apply (*force intro*!: *negligible-atLeastAtMostI pairwiseI*)+
done
hence *has-integral*: ((λ*x*. − *pbernpoly 1 x* / (*x* + *s*)) *has-integral ?i*) {*0*..*real N*}
by (*subst has-integral-spike-set-eq*)
(*use Union-atLeastAtMost assms* **in** ‹*auto simp*: *intro*!: *empty-imp-negligible*›)
hence (λ*x*. − *pbernpoly 1 x* / (*x* + *s*)) *integrable-on* {*0*..*real N*}
and *integral*: *integral* {*0*..*real N*} (λ*x*. − *pbernpoly 1 x* / (*x* + *s*)) = *?i*
by (*simp-all add*: *has-integral-iff*)
show (λ*x*. − *pbernpoly 1 x* / (*x* + *s*)) *integrable-on* {*0*..*real N*} **by** *fact*

note *has-integral-neg*[*OF has-integral*]
also have −*?i* = (∑*x*<*N*. (*of-nat x* + *1* / *2* + *s*) ∗ (*ln* (*of-nat x* + *s*) − *ln* (*of-nat x* + *1* + *s*)) + *1*)
by (*subst sum.reindex*)
(*simp-all add*: *inj-on-def atLeast0LessThan algebra-simps sum-negf* [*symmetric*])
finally show *has-integral*:
((λ*x*. *of-real* (*pbernpoly 1 x*) / (*of-real x* + *s*)) *has-integral* ...) {*0*..*real N*} **by** *simp*

note *integral*
also have *?i* = (∑*n*<*N*. (*of-nat n* + *1* / *2* + *s*) ∗
(*ln* (*of-nat n* + *1* + *s*) − *ln* (*of-nat n* + *s*))) − *N* (**is** - = *?S* − -)
by (*subst sum.reindex*) (*simp-all add*: *inj-on-def sum-subtractf atLeast0LessThan*)
also have *?S* = (∑*n*<*N*. *of-nat n* ∗ (*ln* (*of-nat n* + *1* + *s*) − *ln* (*of-nat n* + *s*))) +
(*s* + *1* / *2*) ∗ (∑*n*<*N*. *ln* (*of-nat* (*Suc n*) + *s*) − *ln* (*of-nat n* + *s*))
(**is** - = *?S1* + - ∗ *?S2*) **by** (*simp add*: *algebra-simps sum.distrib sum-subtractf sum-distrib-left*)
also have *?S2* = *ln* (*of-nat N* + *s*) − *ln s* **by** (*subst sum-lessThan-telescope*) *simp*
also have *?S1* = (∑*n*=*1*..<*N*. *of-nat n* ∗ (*ln* (*of-nat n* + *1* + *s*) − *ln* (*of-nat n* + *s*)))
by (*intro sum.mono-neutral-right*) *auto*
also have ... = (∑*n*=*1*..<*N*. *of-nat n* ∗ *ln* (*of-nat n* + *1* + *s*)) − (∑*n*=*1*..<*N*.

of-nat n ∗ ln (of-nat n + s))
  by (simp add: algebra-simps sum-subtractf)
 also have (∑ n=1..<N. of-nat n ∗ ln (of-nat n + 1 + s)) =
    (∑ n=1..<N. (of-nat n − 1) ∗ ln (of-nat n + s)) + (N − 1) ∗ ln (of-nat N + s)
  by (induction N) (simp-all add: add-ac of-nat-diff)
 also have ... − (∑ n = 1..<N. of-nat n ∗ ln (of-nat n + s)) =
    −(∑ n=1..<N. ln (of-nat n + s)) + (N − 1) ∗ ln (of-nat N + s)
  by (induction N) (simp-all add: algebra-simps)
 also from s **have** neq: s + of-nat x ≠ 0 **for** x
  by (auto simp: complex-nonpos-Reals-iff complex-eq-iff)
 hence (∑ n=1..<N. ln (of-nat n + s)) = (∑ n=1..<N. ln (of-nat n) + ln (1 + s/n))
  by (intro sum.cong refl, subst Ln-times-of-nat [symmetric]) (auto simp: divide-simps add-ac)
 also have ... = ln (fact (N − 1)) + (∑ n=1..<N. ln (1 + s/n))
  by (induction N) (simp-all add: Ln-times-of-nat fact-reduce add-ac)
 also have (∑ n=1..<N. ln (1 + s/n)) = −(∑ n=1..<N. s / n − ln (1 + s/n)) + s ∗ (∑ n=1..<N. 1 / of-nat n)
  by (simp add: sum-distrib-left sum-subtractf)
 also from N **have** ln (fact (N − 1)) = ln-Gamma (of-nat N :: complex)
  by (simp add: ln-Gamma-complex-conv-fact)
 also have {1..<N} = {1..N − 1} **by** auto
 hence (∑ n = 1..<N. 1 / of-nat n) = (harm (N − 1) :: complex)
  by (simp add: harm-def divide-simps)
 also have − (ln-Gamma (of-nat N) + (− (∑ n = 1..<N. s / of-nat n − ln (1 + s / of-nat n)) +
    s ∗ harm (N − 1))) + of-nat (N − 1) ∗ ln (of-nat N + s) +
    (s + 1 / 2) ∗ (ln (of-nat N + s) − ln s) − of-nat N = approx N
  using N **by** (simp add: field-simps of-nat-diff ln-div approx-def Ln-of-nat
    ln-Gamma-complex-of-real [symmetric])
 finally show integral {0..of-nat N} (λx. −of-real (pbernpoly 1 x) / (of-real x + s)) = ...
  by simp
qed

lemma integrable-ln-Gamma-aux:
 shows (λx. of-real (pbernpoly n x) / (of-real x + s) ^ n) integrable-on {0..real N}
proof (cases n = 1)
 case True
 with s **show** ?thesis **using** integrable-neg[OF integrable-pbernpoly-1[of N]]
  by (cases N = 0) (simp-all add: integrable-negligible)
next
 case False
 from s **have** of-real x + s ≠ 0 **if** x ≥ 0 **for** x **using** that
  by (auto simp: complex-eq-iff add-eq-0-iff2 complex-nonpos-Reals-iff)
 with False s **show** ?thesis
  by (auto intro!: integrable-continuous-real continuous-intros)

qed

This following proof is based on "Rudiments of the theory of the gamma function" by Bruce Berndt [1].

lemma *tendsto-of-real-0-I*:
$(f \longrightarrow 0)\ G \Longrightarrow ((\lambda x.\ (\textit{of-real}\ (f\ x))) \longrightarrow (0 :: 'a::\textit{real-normed-div-algebra}))\ G$
using *tendsto-of-real-iff* **by** *force*

qualified lemma *integral-pbernpoly-1*:
$(\lambda N.\ \textit{integral}\ \{0..\textit{real}\ N\}\ (\lambda x.\ \textit{pbernpoly}\ 1\ x\ /\ (x + s)))$
$\longrightarrow -\textit{ln-Gamma}\ s - s + (s - 1\ /\ 2) * \textit{ln}\ s + \textit{ln}\ (2 * \textit{pi})\ /\ 2$
proof −
have *neq*: $s + \textit{of-real}\ x \neq 0$ **if** $x \geq 0$ **for** $x :: \textit{real}$
using *that s* **by** (*auto simp*: *complex-eq-iff complex-nonpos-Reals-iff*)
have $(\textit{approx} \longrightarrow \textit{ln-Gamma}\ s - 0 - 0 + 0 - 0 + s - (s - 1/2) * \textit{ln}\ s - \textit{ln}\ (2 * \textit{pi})\ /\ 2)\ \textit{at-top}$
unfolding *approx-def*
proof (*intro tendsto-add tendsto-diff*)
from *s* **have** *s'*: $s \notin \mathbb{Z}_{\leq 0}$ **by** (*auto simp*: *complex-nonpos-Reals-iff elim!*: *nonpos-Ints-cases*)
have $(\lambda n.\ \sum i=1..<n.\ s\ /\ \textit{of-nat}\ i - \textit{ln}\ (1 + s\ /\ \textit{of-nat}\ i)) \longrightarrow$
$\textit{ln-Gamma}\ s + \textit{euler-mascheroni} * s + \textit{ln}\ s$ (**is** $?f \longrightarrow$ -)
using *ln-Gamma-series'-aux*[*OF s'*] **unfolding** *sums-def*
by (*subst filterlim-sequentially-Suc* [*symmetric*], *subst* (*asm*) *sum.atLeast1-atMost-eq* [*symmetric*])
(*simp add*: *atLeastLessThanSuc-atLeastAtMost*)
thus $((\lambda n.\ ?f\ n - (\textit{euler-mascheroni} * s + \textit{ln}\ s)) \longrightarrow \textit{ln-Gamma}\ s)\ \textit{at-top}$
by (*auto intro*: *tendsto-eq-intros*)
next
show $(\lambda x.\ \textit{complex-of-real}\ (\textit{ln-Gamma}\ (\textit{real}\ x) - \textit{ln}\ (2 * \textit{pi}\ /\ \textit{real}\ x)\ /\ 2 -$
$\textit{real}\ x * \textit{ln}\ (\textit{real}\ x) + \textit{real}\ x)) \longrightarrow 0$
proof (*intro tendsto-of-real-0-I*
filterlim-compose[*OF tendsto-sandwich filterlim-real-sequentially*])
show $\textit{eventually}\ (\lambda x::\textit{real}.\ \textit{ln-Gamma}\ x - \textit{ln}\ (2 * \textit{pi}\ /\ x)\ /\ 2 - x * \textit{ln}\ x + x \geq 0)\ \textit{at-top}$
using *eventually-ge-at-top*[*of 1::real*]
by *eventually-elim* (*insert ln-Gamma-bounds*(*1*), *simp add*: *algebra-simps*)
show $\textit{eventually}\ (\lambda x::\textit{real}.\ \textit{ln-Gamma}\ x - \textit{ln}\ (2 * \textit{pi}\ /\ x)\ /\ 2 - x * \textit{ln}\ x + x \leq$
$1\ /\ 12 * \textit{inverse}\ x)\ \textit{at-top}$
using *eventually-ge-at-top*[*of 1::real*]
by *eventually-elim* (*insert ln-Gamma-bounds*(*2*), *simp add*: *field-simps*)
show $((\lambda x::\textit{real}.\ 1\ /\ 12 * \textit{inverse}\ x) \longrightarrow 0)\ \textit{at-top}$
by *real-asymp*
qed *simp-all*
next
have $(\lambda x.\ s * \textit{of-real}\ (\textit{harm}\ (x - 1) - \textit{ln}\ (\textit{real}\ (x - 1)) - \textit{euler-mascheroni})) \longrightarrow$

s ∗ *of-real* (*euler-mascheroni* − *euler-mascheroni*)
by (*subst filterlim-sequentially-Suc* [*symmetric*], *intro tendsto-intros*)
(*insert euler-mascheroni-LIMSEQ*, *simp-all*)
also have *?this* ⟷ (λ*x*. *s* ∗ (*harm* (*x* − *1*) − *ln* (*of-nat* (*x* − *1*)) − *euler-mascheroni*)) ⟶ *0*
by (*intro filterlim-cong refl eventually-mono*[*OF eventually-gt-at-top*[*of 1*::*nat*]])
(*auto simp*: *of-real-harm simp del*: *of-nat-diff*)
finally show (λ*x*. *s* ∗ (*harm* (*x* − *1*) − *ln* (*of-nat* (*x* − *1*)) − *euler-mascheroni*)) ⟶ *0* **.**
next
have ((λ*x*. *ln* (*1* + (*s* + *1*) / *of-real x*)) ⟶ *ln* (*1* + *0*)) *at-top* (**is** *?P*)
by (*intro tendsto-intros tendsto-divide-0*[*OF tendsto-const*])
(*simp-all add*: *filterlim-ident filterlim-at-infinity-conv-norm-at-top filterlim-abs-real*)
also have *ln* (*of-real* (*x* + *1*) + *s*) − *ln* (*complex-of-real x*) = *ln* (*1* + (*s* + *1*) / *of-real x*)
if *x* > *1* **for** *x* **using** *that s*
using *Ln-divide-of-real*[*of x of-real* (*x* + *1*) + *s*, *symmetric*] *neq*[*of x*+*1*]
by (*simp add*: *field-simps Ln-of-real*)
hence *?P* ⟷ ((λ*x*. *ln* (*of-real* (*x* + *1*) + *s*) − *ln* (*of-real x*)) ⟶ *0*) *at-top*
by (*intro filterlim-cong refl*)
(*auto intro*: *eventually-mono*[*OF eventually-gt-at-top*[*of 1*::*real*]])
finally have ((λ*n*. *ln* (*of-real* (*real n* + *1*) + *s*) − *ln* (*of-real* (*real n*))) ⟶ *0*) *at-top*
by (*rule filterlim-compose*[*OF - filterlim-real-sequentially*])
hence ((λ*n*. *ln* (*of-nat n* + *s*) − *ln* (*of-nat* (*n* − *1*))) ⟶ *0*) *at-top*
by (*subst filterlim-sequentially-Suc* [*symmetric*]) (*simp add*: *add-ac*)
thus (λ*x*. *s* ∗ (*ln* (*of-nat x* + *s*) − *ln* (*of-nat* (*x* − *1*)))) ⟶ *0*
by (*rule tendsto-mult-right-zero*)
next
have ((λ*x*. *ln* (*1* + *s* / *of-real x*)) ⟶ *ln* (*1* + *0*)) *at-top* (**is** *?P*)
by (*intro tendsto-intros tendsto-divide-0*[*OF tendsto-const*])
(*simp-all add*: *filterlim-ident filterlim-at-infinity-conv-norm-at-top filterlim-abs-real*)
also have *ln* (*of-real x* + *s*) − *ln* (*of-real x*) = *ln* (*1* + *s* / *of-real x*) **if** *x* > *0* **for** *x*
using *Ln-divide-of-real*[*of x of-real x* + *s*] *neq*[*of x*] *that*
by (*auto simp*: *field-simps Ln-of-real*)
hence *?P* ⟷ ((λ*x*. *ln* (*of-real x* + *s*) − *ln* (*of-real x*)) ⟶ *0*) *at-top*
using *s* **by** (*intro filterlim-cong refl*)
(*auto intro*: *eventually-mono* [*OF eventually-gt-at-top*[*of 1*::*real*]])
finally have (λ*x*. (*1*/*2*) ∗ (*ln* (*of-real* (*real x*) + *s*) − *ln* (*of-real* (*real x*)))) ⟶ *0*
by (*rule tendsto-mult-right-zero*[*OF filterlim-compose*[*OF - filterlim-real-sequentially*]])
thus (λ*x*. (*1*/*2*) ∗ (*ln* (*of-nat x* + *s*) − *ln* (*of-nat x*))) ⟶ *0* **by** *simp*
next
have ((λ*x*. *x* ∗ (*ln* (*1* + *s* / *of-real x*))) ⟶ *s*) *at-top* (**is** *?P*)
by (*rule stirling-limit-aux2*)
also have *ln* (*1* + *s* / *of-real x*) = *ln* (*of-real x* + *s*) − *ln* (*of-real x*) **if** *x* > *1*

for *x*
      using *that s Ln-divide-of-real* [*of x of-real x* + *s*, *symmetric*] *neq*[*of x*]
      by (*auto simp*: *Ln-of-real field-simps*)
    hence *?P* ⟷ ((λ*x*. *of-real x* ∗ (*ln* (*of-real x* + *s*) − *ln* (*of-real x*))) ⟶ *s*) *at-top*
      by (*intro filterlim-cong refl*)
        (*auto intro*: *eventually-mono*[*OF eventually-gt-at-top*[*of 1*::*real*]])
    finally have (λ*n*. *of-real* (*real n*) ∗ (*ln* (*of-real* (*real n*) + *s*) − *ln* (*of-real* (*real n*)))) ⟶ *s*
      by (*rule filterlim-compose*[*OF* - *filterlim-real-sequentially*])
    thus (λ*n*. *of-nat n* ∗ (*ln* (*of-nat n* + *s*) − *ln* (*of-nat n*))) ⟶ *s* **by** *simp*
  qed *simp-all*
  also have *?this* ⟷ ((λ*N*. *integral* {*0*..*real N*} (λ*x*. −*pbernpoly 1 x* / (*x* + *s*))) ⟶
                        ln-Gamma s + *s* − (*s* − *1*/*2*) ∗ *ln s* − *ln* (*2* ∗ *pi*) / *2*) *at-top*
    using *integral-pbernpoly-1-aux*
    by (*intro filterlim-cong refl*)
      (*auto intro*: *eventually-mono*[*OF eventually-gt-at-top*[*of 0*::*nat*]])
  also have (λ*N*. *integral* {*0*..*real N*} (λ*x*. −*pbernpoly 1 x* / (*x* + *s*))) =
             (λ*N*. −*integral* {*0*..*real N*} (λ*x*. *pbernpoly 1 x* / (*x* + *s*)))
    by (*simp add*: *fun-eq-iff*)
  finally show *?thesis* **by** (*simp add*: *tendsto-minus-cancel-left* [*symmetric*] *algebra-simps*)
qed

qualified lemma *pbernpoly-integral-conv-pbernpoly-integral-Suc*:
  assumes *n* ≥ *1*
  shows *integral* {*0*..*real N*} (λ*x*. *pbernpoly n x* / (*x* + *s*) ^ *n*) =
           of-real (*pbernpoly* (*Suc n*) (*real N*)) / (*of-nat* (*Suc n*) ∗ (*s* + *of-nat N*) ^ *n*) −
          of-real (*bernoulli* (*Suc n*)) / (*of-nat* (*Suc n*) ∗ *s* ^ *n*) + *of-nat n* / *of-nat* (*Suc n*) ∗
            integral {*0*..*real N*} (λ*x*. *of-real* (*pbernpoly* (*Suc n*) *x*) / (*of-real x* + *s*) ^ *Suc n*)
proof −
  note [*derivative-intros*] = *has-field-derivative-pbernpoly-Suc′*
  define *I* **where** *I* = −*of-real* (*pbernpoly* (*Suc n*) (*of-nat N*)) / (*of-nat* (*Suc n*) ∗ (*of-nat N* + *s*) ^ *n*) +
          of-real (*bernoulli* (*Suc n*) / *real* (*Suc n*)) / *s* ^ *n* +
          integral {*0*..*real N*} (λ*x*. *of-real* (*pbernpoly n x*) / (*of-real x* + *s*) ^ *n*)
  have ((λ*x*. (−*of-nat n* ∗ *inverse* (*of-real x* + *s*) ^ *Suc n*) ∗
        (*of-real* (*pbernpoly* (*Suc n*) *x*) / (*of-nat* (*Suc n*))))
        has-integral −*I*) {*0*..*real N*}
  proof (*rule integration-by-parts-interior-strong*[*OF bounded-bilinear-mult*])
    fix *x* :: *real* **assume** *x*: *x* ∈ {*0*<..<*real N*} − *real* ‘ {*0*..*N*}
    have *x* ∉ ℤ
    proof
      assume *x* ∈ ℤ

then obtain n **where** $x = \mathit{of\text{-}int}\ n$ **by** (*auto elim*!: *Ints-cases*)
with x **have** x': $x = \mathit{of\text{-}nat}\ (\mathit{nat}\ n)$ **by** *simp*
from x **show** *False* **by** (*auto simp*: x')
qed
hence ((λ*x*. *of-real* (*pbernpoly* (*Suc n*) *x* / *of-nat* (*Suc n*))) *has-vector-derivative*
complex-of-real (*pbernpoly n x*)) (*at x*)
by (*intro has-vector-derivative-of-real*) (*auto intro*!: *derivative-eq-intros*)
thus ((λ*x*. *of-real* (*pbernpoly* (*Suc n*) *x*) / *of-nat* (*Suc n*)) *has-vector-derivative*
complex-of-real (*pbernpoly n x*)) (*at x*) **by** *simp*
from *x s* **have** *complex-of-real x* + *s* ≠ *0*
by (*auto simp*: *complex-eq-iff complex-nonpos-Reals-iff*)
thus ((λ*x*. *inverse* (*of-real x* + *s*) ^ *n*) *has-vector-derivative*
− *of-nat n* ∗ *inverse* (*of-real x* + *s*) ^ *Suc n*) (*at x*) **using** *x s assms*
by (*auto intro*!: *derivative-eq-intros has-vector-derivative-real-field simp*: *divide-simps power-add* [*symmetric*]
simp del: *power-Suc*)
next
have *complex-of-real x* + *s* ≠ *0* **if** *x* ≥ *0* **for** *x*
using *that s* **by** (*auto simp*: *complex-eq-iff complex-nonpos-Reals-iff*)
thus *continuous-on* {*0..real N*} (λ*x*. *inverse* (*of-real x* + *s*) ^ *n*)
continuous-on {*0..real N*} (λ*x*. *complex-of-real* (*pbernpoly* (*Suc n*) *x*) / *of-nat* (*Suc n*))
using *assms s* **by** (*auto intro*!: *continuous-intros simp del*: *of-nat-Suc*)
next
have ((λ*x*. *inverse* (*of-real x* + *s*) ^ *n* ∗ *of-real* (*pbernpoly n x*)) *has-integral*
pbernpoly (*Suc n*) (*of-nat N*) / (*of-nat* (*Suc n*) ∗ (*of-nat N* + *s*) ^ *n*) −
of-real (*bernoulli* (*Suc n*) / *real* (*Suc n*)) / *s* ^ *n* − −*I*) {*0..real N*}
using *integrable-ln-Gamma-aux*[*of n N*] *assms*
by (*auto simp*: *I-def has-integral-integral divide-simps*)
thus ((λ*x*. *inverse* (*of-real x* + *s*) ^ *n* ∗ *of-real* (*pbernpoly n x*)) *has-integral*
inverse (*of-real* (*real N*) + *s*) ^ *n* ∗ (*of-real* (*pbernpoly* (*Suc n*) (*real N*)) /
of-nat (*Suc n*)) −
inverse (*of-real 0* + *s*) ^ *n* ∗ (*of-real* (*pbernpoly* (*Suc n*) *0*) / *of-nat* (*Suc n*)) − − *I*)
{*0..real N*} **by** (*simp-all add*: *field-simps*)
qed *simp-all*
also have (λ*x*. − *of-nat n* ∗ *inverse* (*of-real x* + *s*) ^ *Suc n* ∗ (*of-real* (*pbernpoly* (*Suc n*) *x*) /
of-nat (*Suc n*))) =
(λ*x*. − (*of-nat n* / *of-nat* (*Suc n*) ∗ *of-real* (*pbernpoly* (*Suc n*) *x*) /
(*of-real x* + *s*) ^ *Suc n*))
by (*simp add*: *divide-simps fun-eq-iff*)
finally have ((λ*x*. − (*of-nat n* / *of-nat* (*Suc n*) ∗ *of-real* (*pbernpoly* (*Suc n*) *x*) /
(*of-real x* + *s*) ^ *Suc n*)) *has-integral* − *I*) {*0..real N*} **.**
from *has-integral-neg*[*OF this*] **show** *?thesis*
by (*auto simp add*: *I-def has-integral-iff algebra-simps integral-mult-right* [*symmetric*]
simp del: *power-Suc of-nat-Suc*)

qed

lemma *pbernpoly-over-power-tendsto-0*:
assumes *n* > *0*
shows (λ*x*. *of-real* (*pbernpoly* (*Suc n*) (*real x*)) / (*of-nat* (*Suc n*) ∗ (*s* + *of-nat x*) ^ *n*)) ⟶ *0*
proof −
from *s* **have** *neq*: *s* + *of-nat n* ≠ *0* **for** *n*
by (*auto simp*: *complex-eq-iff complex-nonpos-Reals-iff*)
obtain *c* **where** *c*: ⋀*x*. *norm* (*pbernpoly* (*Suc n*) *x*) ≤ *c*
using *bounded-pbernpoly* **by** *auto*
have *eventually* (λ*x*. *real x* + *Re s* > *0*) *at-top*
by *real-asymp*
hence *eventually* (λ*x*. *norm* (*of-real* (*pbernpoly* (*Suc n*) (*real x*)) /
(*of-nat* (*Suc n*) ∗ (*s* + *of-nat x*) ^ *n*)) ≤
(*c* / *real* (*Suc n*)) / (*real x* + *Re s*) ^ *n*) *at-top*
using *eventually-gt-at-top*[*of 0*::*nat*]
proof *eventually-elim*
case (*elim x*)
have *norm* (*of-real* (*pbernpoly* (*Suc n*) (*real x*)) /
(*of-nat* (*Suc n*) ∗ (*s* + *of-nat x*) ^ *n*)) ≤
(*c* / *real* (*Suc n*)) / *norm* (*s* + *of-nat x*) ^ *n* (**is** - ≤ *?rhs*) **using** *c*[*of x*]
by (*auto simp*: *norm-divide norm-mult norm-power neq field-simps simp del*: *of-nat-Suc*)
also have (*real x* + *Re s*) ≤ *cmod* (*s* + *of-nat x*)
using *complex-Re-le-cmod*[*of s* + *of-nat x*] *s* **by** (*auto simp add*: *complex-nonpos-Reals-iff*)
hence *?rhs* ≤ (*c* / *real* (*Suc n*)) / (*real x* + *Re s*) ^ *n* **using** *s elim c*[*of 0*] *neq*[*of x*]
by (*intro divide-left-mono power-mono mult-pos-pos divide-nonneg-pos zero-less-power*) *auto*
finally show *?case* .
qed
moreover have (λ*x*. (*c* / *real* (*Suc n*)) / (*real x* + *Re s*) ^ *n*) ⟶ *0*
using ‹*n* > *0*› **by** *real-asymp*
ultimately show *?thesis* **by** (*rule Lim-null-comparison*)
qed

lemma *convergent-stirling-integral*:
assumes *n* > *0*
shows *convergent* (λ*N*. *integral* {*0*..*real N*}
(λ*x*. *of-real* (*pbernpoly n x*) / (*of-real x* + *s*) ^ *n*)) (**is** *convergent* (*?f n*))
proof −
have *convergent* (*?f* (*Suc n*)) **for** *n*
proof (*induction n*)
case *0*
thus *?case* **using** *integral-pbernpoly-1* **by** (*auto intro*!: *convergentI*)
next
case (*Suc n*)

have *convergent* (λ*N*. *?f* (*Suc n*) *N* −
of-real (*pbernpoly* (*Suc* (*Suc n*)) (*real N*)) /
(*of-nat* (*Suc* (*Suc n*)) ∗ (*s* + *of-nat N*) ^ *Suc n*) +
of-real (*bernoulli* (*Suc* (*Suc n*)) / (*real* (*Suc* (*Suc n*)))) / *s* ^ *Suc n*)
(**is** *convergent ?g*)
by (*intro convergent-add convergent-diff Suc*
convergent-const convergentI[*OF pbernpoly-over-power-tendsto-0*]) *simp-all*
also have *?g* = (λ*N*. *of-nat* (*Suc n*) / *of-nat* (*Suc* (*Suc n*)) ∗ *?f* (*Suc* (*Suc n*))
N) **using** *s*
by (*subst pbernpoly-integral-conv-pbernpoly-integral-Suc*)
(*auto simp*: *fun-eq-iff field-simps simp del*: *of-nat-Suc power-Suc*)
also have *convergent* . . . ⟷ *convergent* (*?f* (*Suc* (*Suc n*)))
by (*intro convergent-mult-const-iff*) (*simp-all del*: *of-nat-Suc*)
finally show *?case* **.**
qed
from *this*[*of n* − *1*] *assms* **show** *?thesis* **by** *simp*
qed

lemma *stirling-integral-conv-stirling-integral-Suc*:
assumes *n* > *0*
shows *stirling-integral n s* =
of-nat n / *of-nat* (*Suc n*) ∗ *stirling-integral* (*Suc n*) *s* −
of-real (*bernoulli* (*Suc n*)) / (*of-nat* (*Suc n*) ∗ *s* ^ *n*)
proof −
have (λ*N*. *of-real* (*pbernpoly* (*Suc n*) (*real N*)) / (*of-nat* (*Suc n*) ∗ (*s* + *of-nat*
N) ^ *n*) −
of-real (*bernoulli* (*Suc n*)) / (*real* (*Suc n*) ∗ *s* ^ *n*) +
integral {*0*..*real N*} (λ*x*. *of-nat n* / *of-nat* (*Suc n*) ∗
(*of-real* (*pbernpoly* (*Suc n*) *x*) / (*of-real x* + *s*) ^ *Suc n*)))
⟶ *0* − *of-real* (*bernoulli* (*Suc n*)) / (*of-nat* (*Suc n*) ∗ *s* ^ *n*) +
of-nat n / *of-nat* (*Suc n*) ∗ *stirling-integral* (*Suc n*) *s* (**is** *?f* ⟶
-)
unfolding *stirling-integral-def integral-mult-right*
using *convergent-stirling-integral*[*of Suc n*] *assms s*
by (*intro tendsto-intros pbernpoly-over-power-tendsto-0*)
(*auto simp*: *convergent-LIMSEQ-iff simp del*: *of-nat-Suc*)
also have *?this* ⟷ (λ*N*. *integral* {*0*..*real N*}
(λ*x*. *of-real* (*pbernpoly n x*) / (*of-real x* + *s*) ^ *n*)) ⟶
of-nat n / *of-nat* (*Suc n*) ∗ *stirling-integral* (*Suc n*) *s* −
of-real (*bernoulli* (*Suc n*)) / (*of-nat* (*Suc n*) ∗ *s* ^ *n*)
using *eventually-gt-at-top*[*of 0*::*nat*] *pbernpoly-integral-conv-pbernpoly-integral-Suc*[*of*
n]
assms **unfolding** *integral-mult-right*
by (*intro filterlim-cong refl*) (*auto elim*!: *eventually-mono simp del*: *power-Suc*)
finally show *?thesis* **unfolding** *stirling-integral-def*[*of n*] **by** (*rule limI*)
qed

lemma *stirling-integral-1-unfold*:
assumes *m* > *0*

shows *stirling-integral 1 s* = *stirling-integral m s* / *of-nat m* −
($\sum$*k*=*1*..<*m*. *of-real* (*bernoulli* (*Suc k*)) / (*of-nat k* ∗ *of-nat* (*Suc k*) ∗ *s* ^ *k*))
proof −
have *stirling-integral 1 s* = *stirling-integral* (*Suc m*) *s* / *of-nat* (*Suc m*) −
($\sum$*k*=*1*..<*Suc m*. *of-real* (*bernoulli* (*Suc k*)) / (*of-nat k* ∗ *of-nat* (*Suc k*) ∗ *s* ^ *k*)) **for** *m*
proof (*induction m*)
case (*Suc m*)
let *?C* = ($\sum$*k* = *1*..<*Suc m*. *of-real* (*bernoulli* (*Suc k*)) / (*of-nat k* ∗ *of-nat* (*Suc k*) ∗ *s* ^ *k*))
note *Suc.IH*
also have *stirling-integral* (*Suc m*) *s* / *of-nat* (*Suc m*) =
stirling-integral (*Suc* (*Suc m*)) *s* / *of-nat* (*Suc* (*Suc m*)) −
of-real (*bernoulli* (*Suc* (*Suc m*))) /
(*of-nat* (*Suc m*) ∗ *of-nat* (*Suc* (*Suc m*)) ∗ *s* ^ *Suc m*)
(**is** - = *?A* − *?B*) **by** (*subst stirling-integral-conv-stirling-integral-Suc*)
(*simp-all del*: *of-nat-Suc power-Suc add*: *divide-simps*)
also have *?A* − *?B* − *?C* = *?A* − (*?B* + *?C*) **by** (*rule diff-diff-eq*)
also have *?B* + *?C* = ($\sum$*k* = *1*..<*Suc* (*Suc m*). *of-real* (*bernoulli* (*Suc k*)) /
(*of-nat k* ∗ *of-nat* (*Suc k*) ∗ *s* ^ *k*))
using *s* **by** (*simp add*: *divide-simps*)
finally show *?case* **.**
qed *simp-all*
note *this*[*of m* − *1*]
also from *assms* **have** *Suc* (*m* − *1*) = *m* **by** *simp*
finally show *?thesis* **.**
qed

lemma *ln-Gamma-stirling-complex*:
assumes *m* > *0*
shows *ln-Gamma s* = (*s* − *1* / *2*) ∗ *ln s* − *s* + *ln* (*2* ∗ *pi*) / *2* +
($\sum$*k*=*1*..<*m*. *of-real* (*bernoulli* (*Suc k*)) / (*of-nat k* ∗ *of-nat* (*Suc k*) ∗ *s* ^ *k*)) −
stirling-integral m s / *of-nat m*
proof −
have *ln-Gamma s* = (*s* − *1* / *2*) ∗ *ln s* − *s* + *ln* (*2* ∗ *pi*) / *2* − *stirling-integral 1 s*
using *limI*[*OF integral-pbernpoly-1*] **by** (*simp add*: *stirling-integral-def algebra-simps*)
also have *stirling-integral 1 s* = *stirling-integral m s* / *of-nat m* −
($\sum$*k* = *1*..<*m*. *of-real* (*bernoulli* (*Suc k*)) / (*of-nat k* ∗ *of-nat* (*Suc k*) ∗ *s* ^ *k*))
using *assms* **by** (*rule stirling-integral-1-unfold*)
finally show *?thesis* **by** *simp*
qed

lemma *LIMSEQ-stirling-integral*:
n > *0* ⟹ (λ*x*. *integral* {*0*..*real x*} (λ*x*. *of-real* (*pbernpoly n x*) / (*of-real x* + *s*)

⌢ *n*))
   ⟶ *stirling-integral n s* **unfolding** *stirling-integral-def*
  using *convergent-stirling-integral*[*of n*] **by** (*simp only*: *convergent-LIMSEQ-iff*)

end

lemmas *has-integral-of-real* = *has-integral-linear*[*OF - bounded-linear-of-real*, *unfolded o-def*]
lemmas *integral-of-real* = *integral-linear*[*OF - bounded-linear-of-real*, *unfolded o-def*]

lemma *integrable-ln-Gamma-aux-real*:
  assumes *0* < *s*
  shows (λ*x*. *pbernpoly n x* / (*x* + *s*) ⌢ *n*) *integrable-on* {*0*..*real N*}
proof −
  have (λ*x*. *complex-of-real* (*pbernpoly n x* / (*x* + *s*) ⌢ *n*)) *integrable-on* {*0*..*real N*}
    using *integrable-ln-Gamma-aux*[*of of-real s n N*] *assms* **by** *simp*
  from *integrable-linear*[*OF this bounded-linear-Re*] **show** *?thesis*
    by (*simp only*: *o-def Re-complex-of-real*)
qed

lemma
  assumes *x* > *0 n* > *0*
  shows *stirling-integral-complex-of-real*:
    stirling-integral n (*complex-of-real x*) = *of-real* (*stirling-integral n x*)
  and *LIMSEQ-stirling-integral-real*:
    (λ*N*. *integral* {*0*..*real N*} (λ*t*. *pbernpoly n t* / (*t* + *x*) ⌢ *n*))
    ⟶ *stirling-integral n x*
  and *stirling-integral-real-convergent*:
    convergent (λ*N*. *integral* {*0*..*real N*} (λ*t*. *pbernpoly n t* / (*t* + *x*) ⌢ *n*))
proof −
  have (λ*N*. *integral* {*0*..*real N*} (λ*t*. *of-real* (*pbernpoly n t* / (*t* + *x*) ⌢ *n*)))
    ⟶ *stirling-integral n* (*complex-of-real x*)
    using *LIMSEQ-stirling-integral*[*of complex-of-real x n*] *assms* **by** *simp*
  hence (λ*N*. *of-real* (*integral* {*0*..*real N*} (λ*t*. *pbernpoly n t* / (*t* + *x*) ⌢ *n*)))
    ⟶ *stirling-integral n* (*complex-of-real x*)
    using *integrable-ln-Gamma-aux-real*[*OF assms*(*1*), *of n*]
    by (*subst* (*asm*) *integral-of-real*) *simp*
  from *tendsto-Re*[*OF this*]
    have (λ*N*. *integral* {*0*..*real N*} (λ*t*. *pbernpoly n t* / (*t* + *x*) ⌢ *n*))
    ⟶ *Re* (*stirling-integral n* (*complex-of-real x*)) **by** *simp*
  thus *convergent* (λ*N*. *integral* {*0*..*real N*} (λ*t*. *pbernpoly n t* / (*t* + *x*) ⌢ *n*))
    by (*rule convergentI*)
  thus (λ*N*. *integral* {*0*..*real N*} (λ*t*. *pbernpoly n t* / (*t* + *x*) ⌢ *n*))
    ⟶ *stirling-integral n x* **unfolding** *stirling-integral-def*
    by (*simp add*: *convergent-LIMSEQ-iff*)
  from *tendsto-of-real*[*OF this*, **where** ′*a* = *complex*]
    integrable-ln-Gamma-aux-real[*OF assms*(*1*), *of n*]
    have (λ*xa*. *integral* {*0*..*real xa*}

(*λxa. complex-of-real* (*pbernpoly n xa*) / (*complex-of-real xa* + *x*) ^ *n*))
$\longrightarrow$ *complex-of-real* (*stirling-integral n x*)
by (*subst* (*asm*) *integral-of-real* [*symmetric*]) *simp-all*
from *LIMSEQ-unique*[*OF this LIMSEQ-stirling-integral*[*of complex-of-real x n*]] *assms*
show *stirling-integral n* (*complex-of-real x*) = *of-real* (*stirling-integral n x*) **by** *simp*
qed

lemma *ln-Gamma-stirling-real*:
assumes *x* > (*0* :: *real*) *m* > (*0*::*nat*)
shows *ln-Gamma x* = (*x* − *1* / *2*) ∗ *ln x* − *x* + *ln* (*2* ∗ *pi*) / *2* +
($\sum$ *k* = *1*..<*m*. *bernoulli* (*Suc k*) / (*of-nat k* ∗ *of-nat* (*Suc k*) ∗ *x* ^ *k*)) −
stirling-integral m x / *of-nat m*
proof −
from *assms* **have** *complex-of-real* (*ln-Gamma x*) = *ln-Gamma* (*complex-of-real x*)
by (*simp add*: *ln-Gamma-complex-of-real*)
also have *ln-Gamma* (*complex-of-real x*) = *complex-of-real* (
(*x* − *1* / *2*) ∗ *ln x* − *x* + *ln* (*2* ∗ *pi*) / *2* +
($\sum$ *k* = *1*..<*m*. *bernoulli* (*Suc k*) / (*of-nat k* ∗ *of-nat* (*Suc k*) ∗ *x* ^ *k*)) −
stirling-integral m x / *of-nat m*) **using** *assms*
by (*subst ln-Gamma-stirling-complex*[*of - m*])
(*simp-all add*: *Ln-of-real stirling-integral-complex-of-real*)
finally show *?thesis* **by** (*subst* (*asm*) *of-real-eq-iff*)
qed

lemma *stirling-integral-bound-aux*:
assumes *n*: *n* > (*1*::*nat*)
obtains *c* **where** $\bigwedge$*s*. *Re s* > *0* ⟹ *norm* (*stirling-integral n s*) ≤ *c* / *Re s* ^ (*n* − *1*)
proof −
obtain *c* **where** *c*: *norm* (*pbernpoly n x*) ≤ *c* **for** *x* **by** (*rule bounded-pbernpoly*[*of n*]) *blast*
have *c'*: *pbernpoly n x* ≤ *c* **for** *x* **using** *c*[*of x*] **by** (*simp add*: *abs-real-def split*: *if-splits*)
from *c*[*of 0*] **have** *c-nonneg*: *c* ≥ *0* **by** *simp*
have *norm* (*stirling-integral n s*) ≤ *c* / (*real n* − *1*) / *Re s* ^ (*n* − *1*) **if** *s*: *Re s* > *0* **for** *s*
proof (*rule Lim-norm-ubound*[*OF - LIMSEQ-stirling-integral*])
have *pos*: *x* + *norm s* > *0* **if** *x* ≥ *0* **for** *x* **using** *s that* **by** (*intro add-nonneg-pos*) *auto*
have *nz*: *of-real x* + *s* ≠ *0* **if** *x* ≥ *0* **for** *x* **using** *s that* **by** (*auto simp*: *complex-eq-iff*)
let *?bound* = *λN*. *c* / (*Re s* ^ (*n* − *1*) ∗ (*real n* − *1*)) −

```
                c / ((real N + Re s) ^ (n − 1) ∗ (real n − 1))
  show eventually (λN. norm (integral {0..real N}
          (λx. of-real (pbernpoly n x) / (of-real x + s) ^ n)) ≤
        c / (real n − 1) / Re s ^ (n − 1)) at-top
    using eventually-gt-at-top[of 0::nat]
  proof eventually-elim
    case (elim N)
    let ?F = λx. −c / ((x + Re s) ^ (n − 1) ∗ (real n − 1))
    from n s have ((λx. c / (x + Re s) ^ n) has-integral (?F (real N) − ?F 0))
{0..real N}
      by (intro fundamental-theorem-of-calculus)
        (auto intro!: derivative-eq-intros simp: divide-simps power-diff add-eq-0-iff2
              has-real-derivative-iff-has-vector-derivative [symmetric])
    also have ?F (real N) − ?F 0 = ?bound N by simp
    finally have ∗: ((λx. c / (x + Re s) ^ n) has-integral ?bound N) {0..real N}
.
    have norm (integral {0..real N} (λx. of-real (pbernpoly n x) / (of-real x + s)
^ n)) ≤
          integral {0..real N} (λx. c / (x + Re s) ^ n)
    proof (intro integral-norm-bound-integral integrable-ln-Gamma-aux s ballI)
      fix x assume x: x ∈ {0..real N}
     have norm (of-real (pbernpoly n x) / (of-real x + s) ^ n) ≤ c / norm (of-real
x + s) ^ n
        unfolding norm-divide norm-power using c by (intro divide-right-mono)
simp-all
      also have ... ≤ c / (x + Re s) ^ n
        using x c c-nonneg s nz[of x] complex-Re-le-cmod[of of-real x + s]
           by (intro divide-left-mono power-mono mult-pos-pos zero-less-power
add-nonneg-pos) auto
      finally show norm (of-real (pbernpoly n x) / (of-real x + s) ^ n) ≤ ... .
    qed (insert n s ∗ pos nz c, auto simp: complex-nonpos-Reals-iff)
    also have ... = ?bound N using ∗ by (simp add: has-integral-iff)
    also have ... ≤ c / (Re s ^ (n − 1) ∗ (real n − 1)) using c-nonneg elim s
n by simp
    also have ... = c / (real n − 1) / (Re s ^ (n − 1)) by simp
    finally show norm (integral {0..real N} (λx. of-real (pbernpoly n x) /
              (of-real x + s) ^ n)) ≤ c / (real n − 1) / Re s ^ (n − 1) .
  qed
 qed (insert s n, simp-all add: complex-nonpos-Reals-iff)
 thus ?thesis by (rule that)
qed

lemma stirling-integral-bound-aux-integral1:
 fixes a b c :: real and n :: nat
 assumes a ≥ 0 b > 0 c ≥ 0 n > 1 l < a − b r > a + b
 shows ((λx. c / max b |x − a| ^ n) has-integral
     2∗c∗(n / (n − 1))/b^(n−1) − c/(n−1) ∗ (1/(a−l)^(n−1) + 1/(r−a)^(n−1)))
{l..r}
proof −
```

define *x1 x2* **where** *x1* = *a* − *b* **and** *x2* = *a* + *b*
define *F1* **where** *F1* = (λ*x*::*real*. *c* / (*a* − *x*) ^ (*n* − *1*) / (*n* − *1*))
define *F3* **where** *F3* = (λ*x*::*real*. −*c* / (*x* − *a*) ^ (*n* − *1*) / (*n* − *1*))
have *deriv*: (*F1 has-vector-derivative* (*c* / (*a* − *x*) ^ *n*)) (*at x within A*)
(*F3 has-vector-derivative* (*c* / (*x* − *a*) ^ *n*)) (*at x within A*)
if *x* ≠ *a* **for** *x* :: *real* **and** *A*
unfolding *F1-def F3-def* **using** *assms that*
by (*auto intro*!: *derivative-eq-intros simp*: *divide-simps power-diff add-eq-0-iff2*
simp flip: *has-real-derivative-iff-has-vector-derivative*)

from *assms* **have** ((λ*x*. *c* / (*a* − *x*) ^ *n*) *has-integral* (*F1 x1* − *F1 l*)) {*l*..*x1*}
by (*intro fundamental-theorem-of-calculus deriv*) (*auto simp*: *x1-def max-def split*: *if-splits*)
also have *?this* ⟷ ((λ*x*. *c* / *max b* |*x* − *a*| ^ *n*) *has-integral* (*F1 x1* − *F1 l*)) {*l*..*x1*}
using *assms*
by (*intro has-integral-spike-finite-eq*[*of* {*l*}]) (*auto simp*: *x1-def max-def split*: *if-splits*)
finally have *I1*: ((λ*x*. *c* / *max b* |*x* − *a*| ^ *n*) *has-integral* (*F1 x1* − *F1 l*)) {*l*..*x1*} **.**

have ((λ*x*. *c* / *b* ^ *n*) *has-integral* (*x2* − *x1*) ∗ *c* / *b* ^ *n*) {*x1* ..*x2*}
using *has-integral-const-real*[*of c* / *b* ^ *n x1 x2*] *assms* **by** (*simp add*: *x1-def x2-def*)
also have *?this* ⟷ ((λ*x*. *c* / *max b* |*x* − *a*| ^ *n*) *has-integral* ((*x2* − *x1*) ∗ *c* / *b* ^ *n*)) {*x1* ..*x2*}
by (*intro has-integral-spike-finite-eq*[*of* {*x1*, *x2*}])
(*auto simp*: *x1-def x2-def split*: *if-splits*)
finally have *I2*: ((λ*x*. *c* / *max b* |*x* − *a*| ^ *n*) *has-integral* ((*x2* − *x1*) ∗ *c* / *b* ^ *n*)) {*x1* ..*x2*} **.**

from *assms* **have** *I3*: ((λ*x*. *c* / (*x* − *a*) ^ *n*) *has-integral* (*F3 r* − *F3 x2*)) {*x2* ..*r*}
by (*intro fundamental-theorem-of-calculus deriv*) (*auto simp*: *x2-def min-def split*: *if-splits*)
also have *?this* ⟷ ((λ*x*. *c* / *max b* |*x* − *a*| ^ *n*) *has-integral* (*F3 r* − *F3 x2*)) {*x2* ..*r*}
using *assms*
by (*intro has-integral-spike-finite-eq*[*of* {*r*}]) (*auto simp*: *x2-def min-def split*: *if-splits*)
finally have *I3*: ((λ*x*. *c* / *max b* |*x* − *a*| ^ *n*) *has-integral* (*F3 r* − *F3 x2*)) {*x2* ..*r*} **.**

have ((λ*x*. *c* / *max b* |*x* − *a*| ^ *n*) *has-integral* (*F1 x1* − *F1 l*) + ((*x2* − *x1*) ∗ *c* / *b* ^ *n*) + (*F3 r* − *F3 x2*)) {*l*..*r*}
using *assms*
by (*intro has-integral-combine*[*OF* - - *has-integral-combine*[*OF* - - *I1 I2*] *I3*])
(*auto simp*: *x1-def x2-def*)
also have (*F1 x1* − *F1 l*) + ((*x2* − *x1*) ∗ *c* / *b* ^ *n*) + (*F3 r* − *F3 x2*) =
F1 x1 − *F1 l* + *F3 r* − *F3 x2* + (*x2* − *x1*) ∗ *c* / *b* ^ *n*

```
    by (simp add: algebra-simps)
  also have x2 − x1 = 2 ∗ b
    using assms by (simp add: x2-def x1-def min-def max-def)
  also have 2 ∗ b ∗ c / b ^ n = 2 ∗ c / b ^ (n − 1)
    using assms by (simp add: power-diff field-simps)
  also have F1 x1 − F1 l + F3 r − F3 x2 =
              c/(n−1) ∗ (2/b^(n−1) − 1/(a−l)^(n−1) − 1/(r−a)^(n−1))
   using assms by (simp add: x1-def x2-def F1-def F3-def field-simps del: of-nat-diff)
  also have . . . + 2 ∗ c / b ^ (n − 1) =
                2∗c∗(1 + 1/(n−1))/b^(n−1) − c/(n−1) ∗ (1/(a−l)^(n−1) +
1/(r−a)^(n−1))
    using assms by (simp add: field-simps del: of-nat-diff)
  also have 1 + 1 / (n − 1) = n / (n − 1)
    using assms by (simp add: field-simps)
  finally show ?thesis .
qed

lemma stirling-integral-bound-aux-integral2:
  fixes a b c :: real and n :: nat
  assumes a ≥ 0 b > 0 c ≥ 0 n > 1
  obtains I where ((λx. c / max b |x − a| ^ n) has-integral I) {l..r}
              I ≤ 2 ∗ c ∗ (n / (n − 1)) / b ^(n−1)
proof −
  define l′ where l′ = min l (a − b − 1)
  define r′ where r′ = max r (a + b + 1)

  define A where A = 2 ∗ c ∗ (n / (n − 1)) / b ^(n − 1)
  define B where B = c / real (n − 1) ∗ (1 / (a − l′) ^(n − 1) + 1 / (r′ −
a) ^(n − 1))

  have has-int: ((λx. c / max b |x − a| ^ n) has-integral (A − B)) {l′..r′}
    using assms unfolding A-def B-def
    by (intro stirling-integral-bound-aux-integral1) (auto simp: l′-def r′-def)
  have (λx. c / max b |x − a| ^ n) integrable-on {l..r}
    by (rule integrable-on-subinterval[OF has-integral-integrable[OF has-int]])
       (auto simp: l′-def r′-def)
  then obtain I where has-int′: ((λx. c / max b |x − a| ^ n) has-integral I) {l..r}
    by (auto simp: integrable-on-def)

  from assms have I ≤ A − B
    by (intro has-integral-subset-le[OF - has-int′ has-int]) (auto simp: l′-def r′-def)
  also have . . . ≤ A
    using assms by (simp add: B-def l′-def r′-def)
  finally show ?thesis using that[of I] has-int′ unfolding A-def by blast
qed

lemma stirling-integral-bound-aux′:
  assumes n: n > (1::nat) and α: α ∈ {0<..<pi}
  obtains c where ⋀s::complex. s ∈ complex-cone′ α − {0} ⟹
```

norm (*stirling-integral n s*) ≤ *c* / *norm s* ^ (*n* − *1*)

proof −

obtain *c* **where** *c*: *norm* (*pbernpoly n x*) ≤ *c* **for** *x* **by** (*rule bounded-pbernpoly*[*of n*]) *blast*

have *c'*: *pbernpoly n x* ≤ *c* **for** *x* **using** *c*[*of x*] **by** (*simp add*: *abs-real-def split*: *if-splits*)

from *c*[*of 0*] **have** *c-nonneg*: *c* ≥ *0* **by** *simp*

define *D* **where** *D* = *c* ∗ *Beta* (− (*real-of-int* (− *int n*) / *2*) − *1* / *2*) (*1* / *2*) / *2*

define *C* **where** *C* = *max D* (*2*∗*c*∗(*n*/(*n*−*1*))/*sin* α^(*n*−*1*))

have ∗: *norm* (*stirling-integral n s*) ≤ *C* / *norm s* ^ (*n* − *1*)
if *s*: *s* ∈ *complex-cone'* α − {*0*} **for** *s* :: *complex*
proof (*rule Lim-norm-ubound*[*OF - LIMSEQ-stirling-integral*])
from *s* α **have** *Arg*: |*Arg s*| ≤ α **by** (*auto simp*: *complex-cone-altdef*)
have *s'*: *s* ∉ $\mathbb{R}_{\leq 0}$
using *complex-cone-inter-nonpos-Reals*[*of* −α α] α *s* **by** *auto*
from *s* **have** [*simp*]: *s* ≠ *0* **by** *auto*

show *eventually* (λ*N*. *norm* (*integral* {*0*..*real N*}
(λ*x*. *of-real* (*pbernpoly n x*) / (*of-real x* + *s*) ^ *n*)) ≤
C / *norm s* ^ (*n* − *1*)) *at-top*
using *eventually-gt-at-top*[*of 0*::*nat*]
proof *eventually-elim*
case (*elim N*)
show *?case*
proof (*cases Re s* > *0*)
case *True*
have *int*: ((λ*x*. *c* ∗ (*x*^*2* + *norm s*^*2*) *powr* (−*n* / *2*)) *has-integral*
D ∗ (*norm s* ^ *2*) *powr* (−*n* / *2* + *1* / *2*)) {*0*<..}
using *has-integral-mult-left*[*OF has-integral-Beta3*[*of* −*n*/*2 norm s* ^ *2*], *of c*] *assms True*
unfolding *D-def* **by** (*simp add*: *algebra-simps*)
hence *int'*: ((λ*x*. *c* ∗ (*x*^*2* + *norm s*^*2*) *powr* (−*n* / *2*)) *has-integral*
D ∗ (*norm s* ^ *2*) *powr* (−*n* / *2* + *1* / *2*)) {*0*..}
by (*subst has-integral-interior* [*symmetric*]) *simp-all*
hence *integrable*: (λ*x*. *c* ∗ (*x*^*2* + *norm s*^*2*) *powr* (−*n* / *2*)) *integrable-on* {*0*..}
by (*simp add*: *has-integral-iff*)

have *norm* (*integral* {*0*..*real N*} (λ*x*. *of-real* (*pbernpoly n x*) / (*of-real x* + *s*) ^ *n*)) ≤
integral {*0*..*real N*} (λ*x*. *c* ∗ (*x*^*2* + *norm s*^*2*) *powr* (−*n* / *2*))
proof (*intro integral-norm-bound-integral s ballI integrable-ln-Gamma-aux*)
have [*simp*]: {*0*<..} − {*0*::*real*..} = {} {*0*..} − {*0*<..} = {*0*::*real*}
by *auto*
have (λ*x*. *c* ∗ (x^2 + (*cmod s*)^2) *powr* (*real-of-int* (− *int n*) / *2*)) *integrable-on* {*0*<..}

using *int* **by** (*simp add*: *has-integral-iff*)
also have *?this* ⟷ (λ*x*. *c* ∗ (x^2 + (*cmod s*)^2) *powr* (*real-of-int* (− *int n*) / *2*)) *integrable-on* {*0*..}
by (*intro integrable-spike-set-eq*) *auto*
finally show (λ*x*. *c* ∗ (x^2 + (*cmod s*)^2) *powr* (*real-of-int* (− *int n*) / *2*)) *integrable-on*
{*0*..*real N*} **by** (*rule integrable-on-subinterval*) *auto*
next
fix *x* **assume** *x*: *x* ∈ {*0*..*real N*}
have *nz*: *complex-of-real x* + *s* ≠ *0*
using *True x* **by** (*auto simp*: *complex-eq-iff*)
have *norm* (*of-real* (*pbernpoly n x*) / (*of-real x* + *s*) ^ *n*) ≤ *c* / *norm* (*of-real x* + *s*) ^ *n*
unfolding *norm-divide norm-power* **using** *c* **by** (*intro divide-right-mono*) *simp-all*
also have ... ≤ *c* / *sqrt* (*x* ^ *2* + *norm s* ^ *2*) ^ *n*
proof (*intro divide-left-mono mult-pos-pos zero-less-power power-mono*)
show *sqrt* (x^2 + (*cmod s*)^2) ≤ *cmod* (*complex-of-real x* + *s*)
using *x True* **by** (*simp add*: *cmod-def algebra-simps power2-eq-square*)
qed (*use x True c-nonneg assms nz* **in** ‹*auto simp*: *add-nonneg-pos*›)
also have *sqrt* (*x* ^ *2* + *norm s* ^ *2*) ^ *n* = (*x* ^ *2* + *norm s* ^ *2*) *powr* (*1*/*2* ∗ *n*)
by (*subst powr-powr* [*symmetric*], *subst powr-realpow*)
(*auto simp*: *powr-half-sqrt add-nonneg-pos*)
also have *c* / ... = *c* ∗ (*x*^*2* + *norm s*^*2*) *powr* (−*n* / *2*)
by (*simp add*: *powr-minus field-simps*)
finally show *norm* (*complex-of-real* (*pbernpoly n x*) / (*complex-of-real x* + *s*) ^ *n*) ≤ ... **.**
qed *fact*+
also have ... ≤ *integral* {*0*..} (λ*x*. *c* ∗ (*x*^*2* + *norm s*^*2*) *powr* (−*n* / *2*))
using *c-nonneg*
by (*intro integral-subset-le integrable integrable-on-subinterval*[*OF integrable*]) *auto*
also have ... = *D* ∗ (*norm s* ^ *2*) *powr* (−*n* / *2* + *1* / *2*)
using *int′* **by** (*simp add*: *has-integral-iff*)
also have (*norm s* ^ *2*) *powr* (−*n* / *2* + *1* / *2*) = *norm s powr* (*2* ∗ (−*n* / *2* + *1* / *2*))
by (*subst powr-powr* [*symmetric*]) *auto*
also have ... = *norm s powr* (−*real* (*n* − *1*))
using *assms* **by** (*simp add*: *of-nat-diff*)
also have *D* ∗ ... = *D* / *norm s* ^(*n* − *1*)
by (*auto simp*: *powr-minus powr-realpow field-simps*)
also have ... ≤ *C* / *norm s* ^(*n* − *1*)
by (*intro divide-right-mono*) (*auto simp*: *C-def*)
finally show *norm* (*integral* {*0*..*real N*} (λ*x*. *of-real* (*pbernpoly n x*) / (*of-real x* + *s*) ^ *n*)) ≤ ... **.**

next

case *False*
have *cos* |*Arg s*| = *cos* (*Arg s*)
  by (*simp add*: *abs-if*)
also have *cos* (*Arg s*) = *Re* (*rcis* (*norm s*) (*Arg s*)) / *norm s*
  by (*subst Re-rcis*) *auto*
also have . . . = *Re s* / *norm s*
  by (*subst rcis-cmod-Arg*) *auto*
also have . . . ≤ *cos* (*pi* / *2*)
  using *False* **by** (*auto simp*: *field-simps*)
finally have |*Arg s*| ≥ *pi* / *2*
  using *Arg α* **by** (*subst* (*asm*) *cos-mono-le-eq*) *auto*

have *sin α* ∗ *norm s* = *sin* (*pi* − *α*) ∗ *norm s*
  by *simp*
also have . . . ≤ *sin* (*pi* − |*Arg s*|) ∗ *norm s*
  using *α Arg* ‹|*Arg s*| ≥ *pi* / *2*›
  by (*intro mult-right-mono sin-monotone-2pi-le*) *auto*
also have *sin* |*Arg s*| ≥ *0*
  using *Arg-bounded*[*of s*] **by** (*intro sin-ge-zero*) *auto*
hence *sin* (*pi* − |*Arg s*|) = |*sin* |*Arg s*||
  by *simp*
also have . . . = |*sin* (*Arg s*)|
  by (*simp add*: *abs-if*)
also have . . . ∗ *norm s* = |*Im* (*rcis* (*norm s*) (*Arg s*))|
  by (*simp add*: *abs-mult*)
also have . . . = |*Im s*|
  by (*subst rcis-cmod-Arg*) *auto*
finally have *abs-Im-ge*: |*Im s*| ≥ *sin α* ∗ *norm s* **.**

have [*simp*]: *Im s* ≠ *0 s* ≠ *0*
  using *s* ‹*s* ∉ $\mathbb{R}_{\leq 0}$› *False*
  by (*auto simp*: *cmod-def zero-le-mult-iff complex-nonpos-Reals-iff*)
have *sin α* > *0*
  using *assms* **by** (*intro sin-gt-zero*) *auto*

obtain *I* **where** *I*: ((*λx. c* / *max* |*Im s*| |*x* + *Re s*| ^ *n*) *has-integral I*) {*0*..*real N*}
    I ≤ *2*∗*c*∗(*n*/(*n*−*1*)) / |*Im s*| ^(*n* − *1*)
  using *s c-nonneg assms False*
    stirling-integral-bound-aux-integral2[*of* −*Re s* |*Im s*| *c n 0 real N*] **by** *auto*

have *norm* (*integral* {*0*..*real N*} (*λx. of-real* (*pbernpoly n x*) / (*of-real x* + *s*) ^ *n*)) ≤
    integral {*0*..*real N*} (*λx. c* / *max* |*Im s*| |*x* + *Re s*| ^ *n*)
proof (*intro integral-norm-bound-integral integrable-ln-Gamma-aux s ballI*)
  show (*λx. c* / *max* |*Im s*| |*x* + *Re s*| ^ *n*) *integrable-on* {*0*..*real N*}
    using *I*(*1*) **by** (*simp add*: *has-integral-iff*)
next

fix *x* **assume** *x*: $x \in \{0..real\ N\}$
have *nz*: *complex-of-real x + s* $\neq$ *0*
by (*auto simp*: *complex-eq-iff*)
have *norm* (*complex-of-real* (*pbernpoly n x*) / (*complex-of-real x + s*) ^ *n*) $\leq$
c / *norm* (*complex-of-real x + s*) ^ *n*
unfolding *norm-divide norm-power* **using** *c*[*of x*] **by** (*intro divide-right-mono*) *simp-all*
also have ... $\leq$ *c* / *max* |*Im s*| |*x + Re s*| ^ *n*
using *c-nonneg nz abs-Re-le-cmod*[*of of-real x + s*] *abs-Im-le-cmod*[*of of-real x + s*]
by (*intro divide-left-mono power-mono mult-pos-pos zero-less-power*)
(*auto simp*: *less-max-iff-disj*)
finally show *norm* (*complex-of-real* (*pbernpoly n x*) / (*complex-of-real x + s*) ^ *n*) $\leq$ ... **.**
qed (*auto simp*: *complex-nonpos-Reals-iff*)
also have ... $\leq$ *2∗c∗*(*n*/(*n−1*)) / |*Im s*| ^(*n − 1*)
using *I* **by** (*simp add*: *has-integral-iff*)
also have ... $\leq$ *2∗c∗*(*n*/(*n−1*)) / (*sin* α ∗ *norm s*) ^(*n − 1*)
using ‹*sin* α > *0*› *s c-nonneg abs-Im-ge*
by (*intro divide-left-mono mult-pos-pos zero-less-power power-mono mult-nonneg-nonneg*) *auto*
also have ... = *2∗c∗*(*n*/(*n−1*))/*sin* α^(*n−1*) / *norm s* ^(*n − 1*)
by (*simp add*: *field-simps*)
also have ... $\leq$ *C* / *norm s* ^(*n − 1*)
by (*intro divide-right-mono*) (*auto simp*: *C-def*)
finally show *?thesis* **.**
qed
qed
qed (*use that assms complex-cone-inter-nonpos-Reals*[*of* $-\alpha$ α] α **in** *auto*)
thus *?thesis* **by** (*rule that*)
qed

lemma *stirling-integral-bound*:
assumes *n* > *0*
obtains *c* **where**
$\bigwedge$*s. Re s* > *0* $\Longrightarrow$ *norm* (*stirling-integral n s*) $\leq$ *c* / *Re s* ^ *n*
proof −
let *?f* = λ*s. of-nat n* / *of-nat* (*Suc n*) ∗ *stirling-integral* (*Suc n*) *s* −
of-real (*bernoulli* (*Suc n*)) / (*of-nat* (*Suc n*) ∗ *s* ^ *n*)
from *stirling-integral-bound-aux*[*of Suc n*] *assms* **obtain** *c* **where**
c: $\bigwedge$*s. Re s* > *0* $\Longrightarrow$ *norm* (*stirling-integral* (*Suc n*) *s*) $\leq$ *c* / *Re s* ^ *n* **by** *auto*
define *c1* **where** *c1* = *real n* / *real* (*Suc n*) ∗ *c*
define *c2* **where** *c2* = |*bernoulli* (*Suc n*)| / *real* (*Suc n*)
have *c2-nonneg*: *c2* $\geq$ *0* **by** (*simp add*: *c2-def*)
show *?thesis*
proof (*rule that*)
fix *s* :: *complex* **assume** *s*: *Re s* > *0*
hence *s'*: $s \notin \mathbb{R}_{\leq 0}$ **by** (*auto simp*: *complex-nonpos-Reals-iff*)

have *stirling-integral n s* = *?f s* **using** *s′ assms*
by (*rule stirling-integral-conv-stirling-integral-Suc*)
also have *norm* … ≤ *norm* (*of-nat n* / *of-nat* (*Suc n*) ∗ *stirling-integral* (*Suc n*) *s*) +
norm (*of-real* (*bernoulli* (*Suc n*)) / (*of-nat* (*Suc n*) ∗ *s* ^ *n*))
by (*rule norm-triangle-ineq4*)
also have … = *real n* / *real* (*Suc n*) ∗ *norm* (*stirling-integral* (*Suc n*) *s*) +
c2 / *norm s* ^ *n* (**is** - = *?A* + *?B*)
by (*simp add*: *norm-divide norm-mult norm-power c2-def field-simps del*: *of-nat-Suc*)
also have *?A* ≤ *real n* / *real* (*Suc n*) ∗ (*c* / *Re s* ^ *n*)
by (*intro mult-left-mono c s*) *simp-all*
also have … = *c1* / *Re s* ^ *n* **by** (*simp add*: *c1-def*)
also have *c2* / *norm s* ^ *n* ≤ *c2* / *Re s* ^ *n* **using** *s c2-nonneg*
by (*intro divide-left-mono power-mono complex-Re-le-cmod mult-pos-pos zero-less-power*) *auto*
also have *c1* / *Re s* ^ *n* + *c2* / *Re s* ^ *n* = (*c1* + *c2*) / *Re s* ^ *n*
using *s* **by** (*simp add*: *field-simps*)
finally show *norm* (*stirling-integral n s*) ≤ (*c1* + *c2*) / *Re s* ^ *n* **by** − *simp-all*
qed
qed

lemma *stirling-integral-bound′*:
assumes *n* > *0* **and** α ∈ {*0*<..<*pi*}
obtains *c* **where**
⋀*s*::*complex*. *s* ∈ *complex-cone′* α − {*0*} ⟹ *norm* (*stirling-integral n s*) ≤ *c* / *norm s* ^ *n*
proof −
let *?f* = λ*s*. *of-nat n* / *of-nat* (*Suc n*) ∗ *stirling-integral* (*Suc n*) *s* −
of-real (*bernoulli* (*Suc n*)) / (*of-nat* (*Suc n*) ∗ *s* ^ *n*)
from *stirling-integral-bound-aux′*[*of Suc n*] *assms* **obtain** *c* **where**
c: ⋀*s*::*complex*. *s* ∈ *complex-cone′* α − {*0*} ⟹
norm (*stirling-integral* (*Suc n*) *s*) ≤ *c* / *norm s* ^ *n* **by** *auto*
define *c1* **where** *c1* = *real n* / *real* (*Suc n*) ∗ *c*
define *c2* **where** *c2* = |*bernoulli* (*Suc n*)| / *real* (*Suc n*)
have *c2-nonneg*: *c2* ≥ *0* **by** (*simp add*: *c2-def*)
show *?thesis*
proof (*rule that*)
fix *s* :: *complex* **assume** *s*: *s* ∈ *complex-cone′* α − {*0*}
have *s′*: *s* ∉ $\mathbb{R}_{\leq 0}$
using *complex-cone-inter-nonpos-Reals*[*of* −α α] *assms s* **by** *auto*

have *stirling-integral n s* = *?f s* **using** *s′ assms*
by (*intro stirling-integral-conv-stirling-integral-Suc*) *auto*
also have *norm* … ≤ *norm* (*of-nat n* / *of-nat* (*Suc n*) ∗ *stirling-integral* (*Suc n*) *s*) +
norm (*of-real* (*bernoulli* (*Suc n*)) / (*of-nat* (*Suc n*) ∗ *s* ^ *n*))
by (*rule norm-triangle-ineq4*)
also have … = *real n* / *real* (*Suc n*) ∗ *norm* (*stirling-integral* (*Suc n*) *s*) +

c2 / *norm s* ^ *n* (**is** - = *?A* + *?B*)
by (*simp add*: *norm-divide norm-mult norm-power c2-def field-simps del*: *of-nat-Suc*)
also have *?A* ≤ *real n* / *real* (*Suc n*) ∗ (*c* / *norm s* ^ *n*)
by (*intro mult-left-mono c s*) *simp-all*
also have . . . = *c1* / *norm s* ^ *n* **by** (*simp add*: *c1-def*)
also have *c1* / *norm s* ^ *n* + *c2* / *norm s* ^ *n* = (*c1* + *c2*) / *norm s* ^ *n*
using *s* **by** (*simp add*: *divide-simps*)
finally show *norm* (*stirling-integral n s*) ≤ (*c1* + *c2*) / *norm s* ^ *n* **by** − *simp-all*
qed
qed

lemma *stirling-integral-holomorphic* [*holomorphic-intros*]:
assumes *m*: *m* > *0* **and** *A* ∩ $\mathbb{R}_{\leq 0}$ = {}
shows *stirling-integral m holomorphic-on A*
proof −
from *assms* **have** [*simp*]: *z* ∉ $\mathbb{R}_{\leq 0}$ **if** *z* ∈ *A* **for** *z*
using *that* **by** *auto*
let *?f* = λ*s*::*complex*. *of-nat m* ∗ ((*s* − *1* / *2*) ∗ *Ln s* − *s* + *of-real* (*ln* (*2* ∗ *pi*) / *2*) +
(∑ *k*=*1*..<*m*. *of-real* (*bernoulli* (*Suc k*)) / (*of-nat k* ∗ *of-nat* (*Suc k*) ∗ *s* ^ *k*)) −
ln-Gamma s)
have *?f holomorphic-on A* **using** *assms*
by (*auto intro*!: *holomorphic-intros simp del*: *of-nat-Suc elim*!: *nonpos-Reals-cases*)
also have *?this* ⟷ *stirling-integral m holomorphic-on A*
using *assms* **by** (*intro holomorphic-cong refl*)
(*simp-all add*: *field-simps ln-Gamma-stirling-complex*)
finally show *stirling-integral m holomorphic-on A* **.**
qed

lemma *stirling-integral-continuous-on-complex* [*continuous-intros*]:
assumes *m*: *m* > *0* **and** *A* ∩ $\mathbb{R}_{\leq 0}$ = {}
shows *continuous-on A* (*stirling-integral m* :: - ⇒ *complex*)
by (*intro holomorphic-on-imp-continuous-on stirling-integral-holomorphic assms*)

lemma *has-field-derivative-stirling-integral-complex*:
fixes *x* :: *complex*
assumes *x* ∉ $\mathbb{R}_{\leq 0}$ *n* > *0*
shows (*stirling-integral n has-field-derivative deriv* (*stirling-integral n*) *x*) (*at x*)
using *assms*
by (*intro holomorphic-derivI*[*OF stirling-integral-holomorphic*, *of n* −$\mathbb{R}_{\leq 0}$]) *auto*

lemma

assumes *n*: *n* > *0* **and** *x* > *0*
shows *deriv-stirling-integral-complex-of-real*:
(*deriv* ^^ *j*) (*stirling-integral n*) (*complex-of-real x*) =
complex-of-real ((*deriv* ^^ *j*) (*stirling-integral n*) *x*) (**is** *?lhs x* = *?rhs x*)
and *differentiable-stirling-integral-real*:
(*deriv* ^^ *j*) (*stirling-integral n*) *field-differentiable at x* (**is** *?thesis2*)
proof −
let *?A* = {*s*. *Re s* > *0*}
let *?f* = λ*j x*. (*deriv* ^^ *j*) (*stirling-integral n*) (*complex-of-real x*)
let *?f'* = λ*j x*. *complex-of-real* ((*deriv* ^^ *j*) (*stirling-integral n*) *x*)

have [*simp*]: *open ?A* **by** (*simp add*: *open-halfspace-Re-gt*)

have *?lhs x* = *?rhs x* ∧ (*deriv* ^^ *j*) (*stirling-integral n*) *field-differentiable at x*
if *x* > *0* **for** *x* **using** *that*
proof (*induction j arbitrary*: *x*)
case *0*
have ((λ*x*. *Re* (*stirling-integral n* (*of-real x*))) *has-field-derivative*
Re (*deriv* (λ*x*. *stirling-integral n x*) (*of-real x*))) (*at x*) **using** *0 n*
by (*auto intro*!: *derivative-intros has-vector-derivative-real-field*
field-differentiable-derivI holomorphic-on-imp-differentiable-at[*of* - *?A*]
stirling-integral-holomorphic simp: *complex-nonpos-Reals-iff*)
also have *?this* ⟷ (*stirling-integral n has-field-derivative*
Re (*deriv* (λ*x*. *stirling-integral n x*) (*of-real x*))) (*at x*)
using *eventually-nhds-in-open*[*of* {*0*<..} *x*] *0 n*
by (*intro has-field-derivative-cong-ev refl*)
(*auto elim*!: *eventually-mono simp*: *stirling-integral-complex-of-real*)
finally have *stirling-integral n field-differentiable at x*
by (*auto simp*: *field-differentiable-def*)
with *0 n* **show** *?case* **by** (*auto simp*: *stirling-integral-complex-of-real*)
next
case (*Suc j x*)
note *IH* = *conjunct1*[*OF Suc.IH*] *conjunct2*[*OF Suc.IH*]
have ∗: (*deriv* ^^ *Suc j*) (*stirling-integral n*) (*complex-of-real x*) =
of-real ((*deriv* ^^ *Suc j*) (*stirling-integral n*) *x*) **if** *x*: *x* > *0* **for** *x*
proof −
have *deriv* ((*deriv* ^^ *j*) (*stirling-integral n*)) (*complex-of-real x*) =
vector-derivative (λ*x*. (*deriv* ^^ *j*) (*stirling-integral n*) (*of-real x*)) (*at x*)
using *n x*
by (*intro vector-derivative-of-real-right* [*symmetric*]
holomorphic-on-imp-differentiable-at[*of* - *?A*] *holomorphic-higher-deriv*
stirling-integral-holomorphic) (*auto simp*: *complex-nonpos-Reals-iff*)
also have ... = *vector-derivative* (λ*x*. *of-real* ((*deriv* ^^ *j*) (*stirling-integral n*) *x*)) (*at x*)
using *eventually-nhds-in-open*[*of* {*0*<..} *x*] *x*
by (*intro vector-derivative-cong-eq*) (*auto elim*!: *eventually-mono simp*: *IH*(*1*))
also have ... = *of-real* (*deriv* ((*deriv* ^^ *j*) (*stirling-integral n*)) *x*)
by (*intro vector-derivative-of-real-left holomorphic-on-imp-differentiable-at*[*of*

```
- ?A]
          field-differentiable-imp-differentiable IH(2) x)
    finally show ?thesis by simp
   qed
  have ((λx. Re ((deriv ^^ Suc j) (stirling-integral n) (of-real x))) has-field-derivative

          Re (deriv ((deriv ^^ Suc j) (stirling-integral n)) (of-real x))) (at x)
    using Suc.prems n
   by (intro derivative-intros has-vector-derivative-real-field field-differentiable-derivI
         holomorphic-on-imp-differentiable-at[of - ?A] stirling-integral-holomorphic
           holomorphic-higher-deriv) (auto simp: complex-nonpos-Reals-iff)
   also have ?this ⟷ ((deriv ^^ Suc j) (stirling-integral n) has-field-derivative
            Re (deriv ((deriv ^^ Suc j) (stirling-integral n)) (of-real x))) (at x)
    using eventually-nhds-in-open[of {0<..} x] Suc.prems *
    by (intro has-field-derivative-cong-ev refl) (auto elim!: eventually-mono)
   finally have (deriv ^^ Suc j) (stirling-integral n) field-differentiable at x
    by (auto simp: field-differentiable-def)
   with *[OF Suc.prems] show ?case by blast
  qed
  from this[OF assms(2)] show ?lhs x = ?rhs x ?thesis2 by blast+
qed
```

Unfortunately, asymptotic power series cannot, in general, be differentiated. However, since *ln-Gamma* is holomorphic on the entire positive real half-space, we can differentiate its asymptotic expansion after all.

To do this, we use an ad-hoc version of the more general approach outlined in Erdelyi's "Asymptotic Expansions" for holomorphic functions: We bound the value of the j-th derivative of the remainder term at some value x by applying Cauchy's integral formula along a circle centred at x with radius $\frac{1}{2}x$.

```
lemma deriv-stirling-integral-real-bound:
  assumes m: m > 0
  shows  (deriv ^^ j) (stirling-integral m) ∈ O(λx::real. 1 / x ^ (m + j))
proof -
  obtain c where c: ⋀s. 0 < Re s ⟹ cmod (stirling-integral m s) ≤ c / Re s ^
m
   using stirling-integral-bound[OF m] by auto
  have 0 ≤ cmod (stirling-integral m 1) by simp
  also have ... ≤ c using c[of 1] by simp
  finally have c-nonneg: c ≥ 0 .
  define B where B = c * 2 ^ (m + Suc j)
  define B' where B' = B * fact j / 2

  have eventually (λx::real. norm ((deriv ^^ j) (stirling-integral m) x) ≤
        B' * norm (1 / x ^ (m+ j))) at-top
   using eventually-gt-at-top[of 0::real]
  proof eventually-elim
   case (elim x)
```

have *s* ∉ $\mathbb{R}_{\leq 0}$ **if** *s* ∈ *cball* (*of-real x*) (*x/2*) **for** *s* :: *complex*
proof −
have *x* − *Re s* ≤ *norm* (*of-real x* − *s*) **using** *complex-Re-le-cmod*[*of of-real x* − *s*] **by** *simp*
also from *that* **have** ... ≤ *x/2* **by** (*simp add*: *dist-complex-def*)
finally show *?thesis* **using** *elim* **by** (*auto simp*: *complex-nonpos-Reals-iff*)
qed
hence ((λ*u*. *stirling-integral m u* / (*u* − *of-real x*) ^ *Suc j*) *has-contour-integral*
complex-of-real (*2* ∗ *pi*) ∗ i / *fact j* ∗
(*deriv* ^^ *j*) (*stirling-integral m*) (*of-real x*)) (*circlepath* (*of-real x*) (*x/2*))
using *m elim*
by (*intro Cauchy-has-contour-integral-higher-derivative-circlepath*
stirling-integral-continuous-on-complex stirling-integral-holomorphic) *auto*
hence *norm* (*of-real* (*2* ∗ *pi*) ∗ i / *fact j* ∗ (*deriv* ^^ *j*) (*stirling-integral m*) (*of-real x*)) ≤
B / *x* ^ (*m* + *Suc j*) ∗ (*2* ∗ *pi* ∗ (*x* / *2*))
proof (*rule has-contour-integral-bound-circlepath*)
fix *u* :: *complex* **assume** *dist*: *norm* (*u* − *of-real x*) = *x* / *2*
have *Re* (*of-real x* − *u*) ≤ *norm* (*of-real x* − *u*) **by** (*rule complex-Re-le-cmod*)
also have ... = *x* / *2* **using** *dist* **by** (*simp add*: *norm-minus-commute*)
finally have *Re-u*: *Re u* ≥ *x/2* **using** *elim* **by** *simp*
have *norm* (*stirling-integral m u* / (*u* − *of-real x*) ^ *Suc j*) ≤
c / *Re u* ^ *m* / (*x* / *2*) ^ *Suc j* **using** *Re-u elim*
unfolding *norm-divide norm-power dist*
by (*intro divide-right-mono zero-le-power c*) *simp-all*
also have ... ≤ *c* / (*x/2*) ^ *m* / (*x* / *2*) ^ *Suc j* **using** *c-nonneg elim Re-u*
by (*intro divide-right-mono divide-left-mono power-mono*) *simp-all*
also have ... = *B* / *x* ^ (*m* + *Suc j*) **using** *elim* **by** (*simp add*: *B-def field-simps power-add*)
finally show *norm* (*stirling-integral m u* / (*u* − *of-real x*) ^ *Suc j*) ≤ *B* / *x* ^ (*m* + *Suc j*) **.**
qed (*insert elim c-nonneg*, *auto simp*: *B-def simp del*: *power-Suc*)
hence *cmod* ((*deriv* ^^ *j*) (*stirling-integral m*) (*of-real x*)) ≤ *B'* / *x* ^ (*j* + *m*)
using *elim* **by** (*simp add*: *field-simps norm-divide norm-mult norm-power B'-def*)
with *elim m* **show** *?case* **by** (*simp-all add*: *add-ac deriv-stirling-integral-complex-of-real*)
qed
thus *?thesis* **by** (*rule bigoI*)
qed

definition *stirling-sum* **where**
stirling-sum j m x =
(−*1*) ^ *j* ∗ (∑*k* = *1*..<*m*. (*of-real* (*bernoulli* (*Suc k*)) ∗ *pochhammer* (*of-nat k*) *j* / (*of-nat k* ∗
of-nat (*Suc k*))) ∗ *inverse x* ^ (*k* + *j*))

definition *stirling-sum'* **where**
stirling-sum' j m x =

(−1) ^ (Suc j) ∗ (∑ k≤m. (of-real (bernoulli′ k) ∗
pochhammer (of-nat (Suc k)) (j − 1) ∗ inverse x ^ (k + j)))

lemma *stirling-sum-complex-of-real*:
stirling-sum j m (complex-of-real x) = complex-of-real (stirling-sum j m x)
by (*simp add*: *stirling-sum-def pochhammer-of-real* [*symmetric*] *del*: *of-nat-Suc*)

lemma *stirling-sum′-complex-of-real*:
stirling-sum′ j m (complex-of-real x) = complex-of-real (stirling-sum′ j m x)
by (*simp add*: *stirling-sum′-def pochhammer-of-real* [*symmetric*] *del*: *of-nat-Suc*)

lemma *has-field-derivative-stirling-sum-complex* [*derivative-intros*]:
Re x > 0 ⟹ (stirling-sum j m has-field-derivative stirling-sum (Suc j) m x) (at x)
unfolding *stirling-sum-def* [*abs-def*] *sum-distrib-left*
by (*rule DERIV-sum*) (*auto intro*!: *derivative-eq-intros simp del*: *of-nat-Suc*
simp: *pochhammer-Suc power-diff*)

lemma *has-field-derivative-stirling-sum-real* [*derivative-intros*]:
x > (0::real) ⟹ (stirling-sum j m has-field-derivative stirling-sum (Suc j) m x) (at x)
unfolding *stirling-sum-def* [*abs-def*] *sum-distrib-left*
by (*rule DERIV-sum*) (*auto intro*!: *derivative-eq-intros simp del*: *of-nat-Suc*
simp: *pochhammer-Suc power-diff*)

lemma *has-field-derivative-stirling-sum′-complex* [*derivative-intros*]:
assumes *j > 0 Re x > 0*
shows (*stirling-sum′ j m has-field-derivative stirling-sum′ (Suc j) m x) (at x)*
proof (*cases j*)
case (*Suc j′*)
from *assms* **have** [*simp*]: *x ≠ 0* **by** *auto*
define *c* **where** *c = (λn. (−1) ^ Suc j ∗ complex-of-real (bernoulli′ n) ∗
pochhammer (of-nat (Suc n)) j′)*
define *T* **where** *T = (λn x. c n ∗ inverse x ^ (j + n))*
define *T′* **where** *T′ = (λn x. − (of-nat (j + n)) ∗ c n ∗ inverse x ^ (Suc (j + n)))*
have *((λx. ∑ k≤m. T k x) has-field-derivative (∑ k≤m. T′ k x)) (at x)* **using** *assms Suc*
by (*intro DERIV-sum*)
(*auto simp*: *T-def T′-def intro*!: *derivative-eq-intros*
simp: *field-simps power-add* [*symmetric*] *simp del*: *of-nat-Suc power-Suc of-nat-add*)
also have *(λx. (∑ k≤m. T k x)) = stirling-sum′ j m*
by (*simp add*: *Suc T-def c-def stirling-sum′-def fun-eq-iff add-ac mult.assoc sum-distrib-left*)
also have *(∑ k≤m. T′ k x) = stirling-sum′ (Suc j) m x*
by (*simp add*: *T′-def c-def Suc stirling-sum′-def sum-distrib-left
sum-distrib-right algebra-simps pochhammer-Suc*)
finally show *?thesis* **.**

qed (*insert assms*, *simp-all*)

lemma *has-field-derivative-stirling-sum′-real* [*derivative-intros*]:
  assumes $j > 0$ $x > (0::real)$
  shows (*stirling-sum′ j m has-field-derivative stirling-sum′* (*Suc j*) *m x*) (*at x*)
proof (*cases j*)
  case (*Suc j′*)
  from *assms* **have** [*simp*]: $x \neq 0$ **by** *auto*
  define *c* **where** $c = (\lambda n.\ (-1)\ \hat{}\ Suc\ j * (bernoulli'\ n) * pochhammer\ (of\text{-}nat\ (Suc\ n))\ j')$
  define *T* **where** $T = (\lambda n\ x.\ c\ n * inverse\ x\ \hat{}\ (j + n))$
  define *T′* **where** $T' = (\lambda n\ x.\ -\ (of\text{-}nat\ (j + n)) * c\ n * inverse\ x\ \hat{}\ (Suc\ (j + n)))$
  have $((\lambda x.\ \sum k{\le}m.\ T\ k\ x)$ *has-field-derivative* $(\sum k{\le}m.\ T'\ k\ x))$ (*at x*) **using** *assms Suc*
    by (*intro DERIV-sum*)
      (*auto simp*: *T-def T′-def intro*!: *derivative-eq-intros*
        simp: *field-simps power-add* [*symmetric*] *simp del*: *of-nat-Suc power-Suc of-nat-add*)
  also have $(\lambda x.\ (\sum k{\le}m.\ T\ k\ x))$ = *stirling-sum′ j m*
    by (*simp add*: *Suc T-def c-def stirling-sum′-def fun-eq-iff add-ac mult.assoc sum-distrib-left*)
  also have $(\sum k{\le}m.\ T'\ k\ x)$ = *stirling-sum′* (*Suc j*) *m x*
    by (*simp add*: *T′-def c-def Suc stirling-sum′-def sum-distrib-left*
      sum-distrib-right algebra-simps pochhammer-Suc)
  finally show *?thesis* **.**
qed (*insert assms*, *simp-all*)

lemma *higher-deriv-stirling-sum-complex*:
  $Re\ x > 0 \Longrightarrow$ (*deriv* ^^ *i*) (*stirling-sum j m*) *x* = *stirling-sum* (*i* + *j*) *m x*
proof (*induction i arbitrary*: *x*)
  case (*Suc i*)
  have *deriv* ((*deriv* ^^ *i*) (*stirling-sum j m*)) *x* = *deriv* (*stirling-sum* (*i* + *j*) *m*) *x*
    using *eventually-nhds-in-open*[*of* {*x*. $Re\ x > 0$} *x*] *Suc.prems*
    by (*intro deriv-cong-ev refl*) (*auto elim*!: *eventually-mono simp*: *open-halfspace-Re-gt Suc.IH*)
  also from *Suc.prems* **have** . . . = *stirling-sum* (*Suc* (*i* + *j*)) *m x*
    by (*intro DERIV-imp-deriv has-field-derivative-stirling-sum-complex*)
  finally show *?case* **by** *simp*
qed *simp-all*

definition *Polygamma-approx* :: *nat* ⇒ *nat* ⇒ *′a* ⇒ *′a* :: {*real-normed-field*, *ln*}
where
  Polygamma-approx j m =
    (*deriv* ^^ *j*) $(\lambda x.\ (x - 1\ /\ 2) * ln\ x - x + of\text{-}real\ (ln\ (2 * pi))\ /\ 2 + stirling\text{-}sum\ 0\ m\ x)$

lemma *Polygamma-approx-Suc*: *Polygamma-approx* (*Suc j*) *m* = *deriv* (*Polygamma-approx*

j m)
by (*simp add*: *Polygamma-approx-def*)

lemma *Polygamma-approx-0*:
Polygamma-approx 0 m x = (*x* − *1/2*) ∗ *ln x* − *x* + *of-real* (*ln* (*2*∗*pi*)) / *2* + *stirling-sum 0 m x*
by (*simp add*: *Polygamma-approx-def*)

lemma *Polygamma-approx-1-complex*:
Re x > *0* ⟹
Polygamma-approx (*Suc 0*) *m x* = *ln x* − *1* / (*2*∗*x*) + *stirling-sum* (*Suc 0*) *m x*
unfolding *Polygamma-approx-Suc Polygamma-approx-0*
by (*intro DERIV-imp-deriv*)
(*auto intro*!: *derivative-eq-intros elim*!: *nonpos-Reals-cases simp*: *field-simps*)

lemma *Polygamma-approx-1-real*:
x > (*0* :: *real*) ⟹
Polygamma-approx (*Suc 0*) *m x* = *ln x* − *1* / (*2*∗*x*) + *stirling-sum* (*Suc 0*) *m x*
unfolding *Polygamma-approx-Suc Polygamma-approx-0*
by (*intro DERIV-imp-deriv*)
(*auto intro*!: *derivative-eq-intros elim*!: *nonpos-Reals-cases simp*: *field-simps*)

lemma *stirling-sum-2-conv-stirling-sum′-1*:
fixes *x* :: *′a* :: {*real-div-algebra*, *field-char-0*}
assumes *m* > *0 x* ≠ *0*
shows *stirling-sum′ 1 m x* = *1* / *x* + *1* / (*2* ∗ *x*^*2*) + *stirling-sum 2 m x*
proof −
have *pochhammer-2*: *pochhammer* (*of-nat k*) *2* = *of-nat k* ∗ *of-nat* (*Suc k*) **for** *k*
by (*simp add*: *pochhammer-Suc eval-nat-numeral add-ac*)
have *stirling-sum 2 m x* =
(∑*k* = *Suc 0*..<*m*. *of-real* (*bernoulli′* (*Suc k*)) ∗ *inverse x* ^ *Suc* (*Suc k*))
unfolding *stirling-sum-def pochhammer-2 power2-minus power-one mult-1-left*
by (*intro sum.cong refl*)
(*simp-all add*: *stirling-sum-def pochhammer-2 power2-eq-square divide-simps bernoulli′-def*
del: *of-nat-Suc power-Suc*)
also have *1* / (*2* ∗ *x*^*2*) + ... =
(∑*k*=*0*..<*m*. *of-real* (*bernoulli′* (*Suc k*)) ∗ *inverse x* ^ *Suc* (*Suc k*))
using *assms*
by (*subst* (*2*) *sum.atLeast-Suc-lessThan*) (*simp-all add*: *power2-eq-square field-simps*)
also have *1* / *x* + ... = (∑*k*=*0*..<*Suc m*. *of-real* (*bernoulli′ k*) ∗ *inverse x* ^ *Suc k*)
by (*subst sum.atLeast0-lessThan-Suc-shift*) (*simp-all add*: *bernoulli′-def divide-simps*)
also have ... = (∑*k*≤*m*. *of-real* (*bernoulli′ k*) ∗ *inverse x* ^ *Suc k*)
by (*intro sum.cong*) *auto*

also have . . . = *stirling-sum′ 1 m x* **by** (*simp add*: *stirling-sum′-def*)
finally show *?thesis* **by** (*simp add*: *add-ac*)
qed

lemma *Polygamma-approx-2-real*:
assumes *x* > (*0*::*real*) *m* > *0*
shows *Polygamma-approx* (*Suc* (*Suc 0*)) *m x* = *stirling-sum′ 1 m x*
proof −
have *Polygamma-approx* (*Suc* (*Suc 0*)) *m x* = *deriv* (*Polygamma-approx* (*Suc 0*) *m*) *x*
by (*simp add*: *Polygamma-approx-Suc*)
also have . . . = *deriv* (*λx*. *ln x* − *1* / (*2*∗*x*) + *stirling-sum* (*Suc 0*) *m x*) *x*
using *eventually-nhds-in-open*[*of* {*0*<..} *x*] *assms*
by (*intro deriv-cong-ev*) (*auto elim*!: *eventually-mono simp*: *Polygamma-approx-1-real*)
also have . . . = *1* / *x* + *1* / (*2*∗*x*^*2*) + *stirling-sum* (*Suc* (*Suc 0*)) *m x* **using** *assms*
by (*intro DERIV-imp-deriv*) (*auto intro*!: *derivative-eq-intros*
elim!: *nonpos-Reals-cases simp*: *field-simps power2-eq-square*)
also have . . . = *stirling-sum′ 1 m x* **using** *stirling-sum-2-conv-stirling-sum′-1*[*of m x*] *assms*
by (*simp add*: *eval-nat-numeral*)
finally show *?thesis* **.**
qed

lemma *Polygamma-approx-2-complex*:
assumes *Re x* > *0 m* > *0*
shows *Polygamma-approx* (*Suc* (*Suc 0*)) *m x* = *stirling-sum′ 1 m x*
proof −
have *Polygamma-approx* (*Suc* (*Suc 0*)) *m x* = *deriv* (*Polygamma-approx* (*Suc 0*) *m*) *x*
by (*simp add*: *Polygamma-approx-Suc*)
also have . . . = *deriv* (*λx*. *ln x* − *1* / (*2*∗*x*) + *stirling-sum* (*Suc 0*) *m x*) *x*
using *eventually-nhds-in-open*[*of* {*s*. *Re s* > *0*} *x*] *assms*
by (*intro deriv-cong-ev*)
(*auto simp*: *open-halfspace-Re-gt elim*!: *eventually-mono simp*: *Polygamma-approx-1-complex*)
also have . . . = *1* / *x* + *1* / (*2*∗*x*^*2*) + *stirling-sum* (*Suc* (*Suc 0*)) *m x* **using** *assms*
by (*intro DERIV-imp-deriv*) (*auto intro*!: *derivative-eq-intros*
elim!: *nonpos-Reals-cases simp*: *field-simps power2-eq-square*)
also have . . . = *stirling-sum′ 1 m x* **using** *stirling-sum-2-conv-stirling-sum′-1*[*of m x*] *assms*
by (*subst stirling-sum-2-conv-stirling-sum′-1*) (*auto simp*: *eval-nat-numeral*)
finally show *?thesis* **.**
qed

lemma *Polygamma-approx-ge-2-real*:
assumes *x* > (*0*::*real*) *m* > *0*
shows *Polygamma-approx* (*Suc* (*Suc j*)) *m x* = *stirling-sum′* (*Suc j*) *m x*
using *assms*(*1*)

proof (*induction j arbitrary*: *x*)
 case (*0 x*)
 with *assms* **show** *?case* **by** (*simp add*: *Polygamma-approx-2-real*)
next
 case (*Suc j x*)
 have *Polygamma-approx* (*Suc* (*Suc* (*Suc j*))) *m x* = *deriv* (*Polygamma-approx* (*Suc* (*Suc j*)) *m*) *x*
  by (*simp add*: *Polygamma-approx-Suc*)
 also have ... = *deriv* (*stirling-sum′* (*Suc j*) *m*) *x*
  using *eventually-nhds-in-open*[*of* {*0*<..} *x*] *Suc.prems*
  by (*intro deriv-cong-ev refl*) (*auto elim*!: *eventually-mono simp*: *Suc.IH*)
 also have ... = *stirling-sum′* (*Suc* (*Suc j*)) *m x* **using** *Suc.prems*
  by (*intro DERIV-imp-deriv derivative-intros*) *simp-all*
 finally show *?case* **.**
qed

lemma *Polygamma-approx-ge-2-complex*:
 assumes *Re x* > *0 m* > *0*
 shows *Polygamma-approx* (*Suc* (*Suc j*)) *m x* = *stirling-sum′* (*Suc j*) *m x*
using *assms*(*1*)
proof (*induction j arbitrary*: *x*)
 case (*0 x*)
 with *assms* **show** *?case* **by** (*simp add*: *Polygamma-approx-2-complex*)
next
 case (*Suc j x*)
 have *Polygamma-approx* (*Suc* (*Suc* (*Suc j*))) *m x* = *deriv* (*Polygamma-approx* (*Suc* (*Suc j*)) *m*) *x*
  by (*simp add*: *Polygamma-approx-Suc*)
 also have ... = *deriv* (*stirling-sum′* (*Suc j*) *m*) *x*
  using *eventually-nhds-in-open*[*of* {*x*. *Re x* > *0*} *x*] *Suc.prems*
  by (*intro deriv-cong-ev refl*) (*auto elim*!: *eventually-mono simp*: *Suc.IH open-halfspace-Re-gt*)
 also have ... = *stirling-sum′* (*Suc* (*Suc j*)) *m x* **using** *Suc.prems*
  by (*intro DERIV-imp-deriv derivative-intros*) *simp-all*
 finally show *?case* **.**
qed

lemma *Polygamma-approx-complex-of-real*:
 assumes *x* > *0 m* > *0*
 shows *Polygamma-approx j m* (*complex-of-real x*) = *of-real* (*Polygamma-approx j m x*)
proof (*cases j*)
 case *0*
 with *assms* **show** *?thesis* **by** (*simp add*: *Polygamma-approx-0 Ln-of-real stirling-sum-complex-of-real*)
next
 case [*simp*]: (*Suc j′*)
 thus *?thesis*
 proof (*cases j′*)
  case *0*

with *assms* **show** *?thesis*
by (*simp add*: *Polygamma-approx-1-complex*
Polygamma-approx-1-real stirling-sum-complex-of-real Ln-of-real)
next
case (*Suc j''*)
with *assms* **show** *?thesis*
by (*simp add*: *Polygamma-approx-ge-2-complex Polygamma-approx-ge-2-real*
stirling-sum'-complex-of-real)
qed
qed

lemma *higher-deriv-Polygamma-approx* [*simp*]:
(*deriv* ^^ *j*) (*Polygamma-approx i m*) = *Polygamma-approx* (*j* + *i*) *m*
by (*simp add*: *Polygamma-approx-def funpow-add*)

lemma *stirling-sum-holomorphic* [*holomorphic-intros*]:
0 ∉ *A* ⟹ *stirling-sum j m holomorphic-on A*
unfolding *stirling-sum-def* **by** (*intro holomorphic-intros*) *auto*

lemma *Polygamma-approx-holomorphic* [*holomorphic-intros*]:
Polygamma-approx j m holomorphic-on {*s*. *Re s* > *0*}
unfolding *Polygamma-approx-def*
by (*intro holomorphic-intros*) (*auto simp*: *open-halfspace-Re-gt elim*!: *nonpos-Reals-cases*)

lemma *higher-deriv-lnGamma-stirling*:
assumes *m*: *m* > *0*
shows (λ*x*::*real*. (*deriv* ^^ *j*) *ln-Gamma x* − *Polygamma-approx j m x*) ∈ *O*(λ*x*.
1 / *x* ^ (*m* + *j*))
proof −
have *eventually* (λ*x*. |(*deriv* ^^ *j*) *ln-Gamma x* − *Polygamma-approx j m x*| =
inverse (*real m*) ∗ |(*deriv* ^^ *j*) (*stirling-integral m*) *x*|) *at-top*
using *eventually-gt-at-top*[*of 0*::*real*]
proof *eventually-elim*
case (*elim x*)
note *x* = *this*
have ∀$_F$ *y in nhds* (*complex-of-real x*). *y* ∈ − $\mathbb{R}_{\leq 0}$
using *elim* **by** (*intro eventually-nhds-in-open*) *auto*
hence (*deriv* ^^ *j*) (λ*x*. *ln-Gamma x* − *Polygamma-approx 0 m x*) (*complex-of-real*
x) =
(*deriv* ^^ *j*) (λ*x*. (−*inverse* (*of-nat m*)) ∗ *stirling-integral m x*)
(*complex-of-real x*)
using *x m*
by (*intro higher-deriv-cong-ev refl*)
(*auto elim*!: *eventually-mono simp*: *ln-Gamma-stirling-complex Polygamma-approx-def*

field-simps open-halfspace-Re-gt stirling-sum-def)
also have . . . = − *inverse* (*of-nat m*) ∗ (*deriv* ^^ *j*) (*stirling-integral m*) (*of-real*
x) **using** *x m*
by (*intro higher-deriv-cmult*[*of - −*$\mathbb{R}_{\leq 0}$] *stirling-integral-holomorphic*)

(*auto simp*: *open-halfspace-Re-gt*)
also have (*deriv* ^^ *j*) (λ*x*. *ln-Gamma x* − *Polygamma-approx 0 m x*) (*complex-of-real x*) =
(*deriv* ^^ *j*) *ln-Gamma* (*of-real x*) − (*deriv* ^^ *j*) (*Polygamma-approx 0 m*) (*of-real x*)
using *x*
by (*intro higher-deriv-diff*[*of* - {*s*. *Re s* > *0*}])
(*auto intro*!: *holomorphic-intros elim*!: *nonpos-Reals-cases simp*: *open-halfspace-Re-gt*)
also have (*deriv* ^^ *j*) (*Polygamma-approx 0 m*) (*complex-of-real x*) =
of-real (*Polygamma-approx j m x*) **using** *x m*
by (*simp add*: *Polygamma-approx-complex-of-real*)
also have *norm* (− *inverse* (*of-nat m*) ∗ (*deriv* ^^ *j*) (*stirling-integral m*) (*complex-of-real x*)) =
inverse (*real m*) ∗ |(*deriv* ^^ *j*) (*stirling-integral m*) *x*|
using *x m* **by** (*simp add*: *norm-mult norm-inverse deriv-stirling-integral-complex-of-real*)
also have (*deriv* ^^ *j*) *ln-Gamma* (*complex-of-real x*) = *of-real* ((*deriv* ^^ *j*) *ln-Gamma x*) **using** *x*
by (*simp add*: *higher-deriv-ln-Gamma-complex-of-real*)
also have *norm* (. . . − *of-real* (*Polygamma-approx j m x*)) =
|(*deriv* ^^ *j*) *ln-Gamma x* − *Polygamma-approx j m x*|
by (*simp only*: *of-real-diff* [*symmetric*] *norm-of-real*)
finally show *?case* **.**
qed
from *bigthetaI-cong*[*OF this*] *m*
have (λ*x*::*real*. (*deriv* ^^ *j*) *ln-Gamma x* − *Polygamma-approx j m x*) ∈
Θ(λ*x*. (*deriv* ^^ *j*) (*stirling-integral m*) *x*) **by** *simp*
also have (λ*x*::*real*. (*deriv* ^^ *j*) (*stirling-integral m*) *x*) ∈ *O*(λ*x*. *1* / *x* ^ (*m* + *j*)) **using** *m*
by (*rule deriv-stirling-integral-real-bound*)
finally show *?thesis* **.**
qed

lemma *Polygamma-approx-1-real'*:
assumes *x*: (*x*::*real*) > *0* **and** *m*: *m* > *0*
shows *Polygamma-approx 1 m x* = *ln x* − (∑*k* = *Suc 0*..*m*. *bernoulli' k* ∗ *inverse x* ^ *k* / *real k*)
proof −
have *Polygamma-approx 1 m x* = *ln x* − (*1* / (*2* ∗ *x*) +
(∑*k*=*Suc 0*..<*m*. *bernoulli* (*Suc k*) ∗ *inverse x* ^ *Suc k* / *real* (*Suc k*)))
(**is** - = - − (- + *?S*)) **using** *x* **by** (*simp add*: *Polygamma-approx-1-real stirling-sum-def*)
also have *?S* = (∑*k*=*Suc 0*..<*m*. *bernoulli'* (*Suc k*) ∗ *inverse x* ^ *Suc k* / *real* (*Suc k*))
by (*intro sum.cong refl*) (*simp-all add*: *bernoulli'-def*)
also have *1* / (*2* ∗ *x*) + . . . =
(∑*k*=*0*..<*m*. *bernoulli'* (*Suc k*) ∗ *inverse x* ^ *Suc k* / *real* (*Suc k*))
using *m*
by (*subst* (*2*) *sum.atLeast-Suc-lessThan*) (*simp-all add*: *field-simps*)
also have . . . = (∑*k* = *Suc 0*..*m*. *bernoulli' k* ∗ *inverse x* ^ *k* / *real k*) **using**

```
assms
  by (subst sum.shift-bounds-Suc-ivl [symmetric]) (simp add: atLeastLessThanSuc-atLeastAtMost)
 finally show ?thesis .
qed

theorem
 assumes m: m > 0
 shows   ln-Gamma-real-asymptotics:
          (λx. ln-Gamma x − ((x − 1 / 2) ∗ ln x − x + ln (2 ∗ pi) / 2 +
                (∑k = 1..<m. bernoulli (Suc k) / (real k ∗ real (Suc k)) / x^k)))
             ∈ O(λx. 1 / x ^ m) (is ?th1)
   and   Digamma-real-asymptotics:
          (λx. Digamma x − (ln x − (∑k=1..m. bernoulli' k / real k / x ^ k)))
             ∈ O(λx. 1 / (x ^ Suc m)) (is ?th2)
   and   Polygamma-real-asymptotics: j > 0 ⟹
          (λx. Polygamma j x − (− 1) ^ Suc j ∗ (∑k≤m. bernoulli' k ∗
                 pochhammer (real (Suc k)) (j − 1) / x ^ (k + j)))
             ∈ O(λx. 1 / x ^ (m+j+1)) (is - ⟹ ?th3)
proof −
 define G :: nat ⇒ real ⇒ real where
   G = (λm. if m = 0 then ln-Gamma else Polygamma (m − 1))
 have ∗: (λx. G j x − h x) ∈ O(λx. 1 / x ^ (m + j))
   if ⋀x::real. x > 0 ⟹ Polygamma-approx j m x = h x for j h
 proof −
   have (λx. G j x − h x) ∈
            Θ(λx. (deriv ^^ j) ln-Gamma x − Polygamma-approx j m x) (is - ∈
Θ(?f))
     using that
    by (intro bigthetaI-cong) (auto intro: eventually-mono[OF eventually-gt-at-top[of
0::real]]
           simp del: funpow.simps simp: higher-deriv-ln-Gamma-real G-def)
   also have ?f ∈ O(λx::real. 1 / x ^ (m + j)) using m
     by (rule higher-deriv-lnGamma-stirling)
   finally show ?thesis .
 qed

 note [[simproc del: simplify-landau-sum]]
 from ∗[OF Polygamma-approx-0] assms show ?th1
   by (simp add: G-def Polygamma-approx-0 stirling-sum-def field-simps)
  from ∗[OF Polygamma-approx-1-real'] assms show ?th2 by (simp add: G-def
field-simps)

 assume j: j > 0
 from ∗[OF Polygamma-approx-ge-2-real, of j − 1] assms j show ?th3
   by (simp add: G-def stirling-sum'-def power-add power-diff field-simps)
qed
```

2.5 Asymptotics of the complex Gamma function

The m-th order remainder of Stirling's formula for $\log\Gamma$ is $O(s^{-m})$ uniformly over any complex cone $\mathrm{Arg}(z) \le \alpha$, $z \ne 0$ for any angle $\alpha \in (0, \pi)$. This means that there is bounded by cz^{-m} for some constant c for all z in this cone.

```
context
  fixes F and α
  assumes α: α ∈ {0<..<pi}
  defines F ≡ principal (complex-cone′ α − {0})
begin

lemma stirling-integral-bigo:
  fixes m :: nat
  assumes m: m > 0
  shows   stirling-integral m ∈ O[F](λs. 1 / s ^ m)
proof −
  obtain c where c: ⋀s. s ∈ complex-cone′ α − {0} ⟹ norm (stirling-integral
m s) ≤ c / norm s ^ m
    using stirling-integral-bound′[OF ‹m > 0› α] by blast
  have 0 ≤ norm (stirling-integral m 1 :: complex)
    by simp
  also have ... ≤ c
    using c[of 1] α by simp
  finally have c ≥ 0 .

  have eventually (λs. s ∈ complex-cone′ α − {0}) F
    unfolding F-def by (auto simp: eventually-principal)
  hence eventually (λs. norm (stirling-integral m s) ≤
                    c ∗ norm (1 / s ^ m)) F
    by eventually-elim (use c in ‹simp add: norm-divide norm-power›)
  thus stirling-integral m ∈ O[F](λs. 1 / s ^ m)
    by (intro bigoI[of - c]) auto
qed

end
```

The following is a more explicit statement of this:

```
theorem ln-Gamma-complex-asymptotics-explicit:
  fixes m :: nat and α :: real
  assumes m > 0 and α ∈ {0<..<pi}
  obtains C :: real and R :: complex ⇒ complex
  where ∀ s::complex. s ∉ ℝ≤0 ⟶
              ln-Gamma s = (s − 1/2) ∗ ln s − s + ln (2 ∗ pi) / 2 +
                    (∑ k=1..<m. bernoulli (k+1) / (k ∗ (k+1) ∗ s ^ k)) − R s
    and ∀ s. s ≠ 0 ∧ |Arg s| ≤ α ⟶ norm (R s) ≤ C / norm s ^ m
proof −
  obtain c where c: ⋀s. s ∈ complex-cone′ α − {0} ⟹ norm (stirling-integral
m s) ≤ c / norm s ^ m
```

```
    using stirling-integral-bound'[OF assms] by blast
  have 0 ≤ norm (stirling-integral m 1 :: complex)
    by simp
  also have ... ≤ c
    using c[of 1] assms by simp
  finally have c ≥ 0 .
  define R where R = (λs::complex. stirling-integral m s / of-nat m)
  show ?thesis
  proof (rule that)
    from ln-Gamma-stirling-complex[of - m] assms show
          ∀s::complex. s ∉ ℝ≤0 ⟶
            ln-Gamma s = (s − 1 / 2) ∗ ln s − s + ln (2 ∗ pi) / 2 +
            (∑k=1..<m. bernoulli (k+1) / (k ∗ (k+1) ∗ s ^ k)) − R s
      by (auto simp add: R-def algebra-simps)
    show ∀s. s ≠ 0 ∧ |Arg s| ≤ α ⟶ cmod (R s) ≤ c / real m / cmod s ^ m
    proof (safe, goal-cases)
      case (1 s)
      show ?case
        using 1 c[of s] assms
        by (auto simp: complex-cone-altdef abs-le-iff R-def norm-divide field-simps)
    qed
  qed
qed
```

Lastly, we can also derive the asymptotics of Γ itself:

$$\Gamma(z) \sim \sqrt{2\pi/z}\left(\frac{z}{e}\right)^z$$

uniformly for $|z| \to \infty$ within the cone $\mathrm{Arg}(z) \leq \alpha$ for $\alpha \in (0, \pi)$:

```
context
  fixes F and α
  assumes α: α ∈ {0<..<pi}
  defines F ≡ inf at-infinity (principal (complex-cone' α))
begin

lemma Gamma-complex-asymp-equiv:
  Gamma ∼[F] (λs. sqrt (2 ∗ pi) ∗ (s / exp 1) powr s / s powr (1 / 2))
proof −
  define I :: complex ⇒ complex where I = stirling-integral 1
  have eventually (λs. s ∈ complex-cone' α) F
    by (auto simp: eventually-inf-principal F-def)
  moreover have eventually (λs. s ≠ 0) F
    unfolding F-def eventually-inf-principal
    using eventually-not-equal-at-infinity by eventually-elim auto
  ultimately have eventually (λs. Gamma s =
                  sqrt (2 ∗ pi) ∗ (s / exp 1) powr s / s powr (1 / 2) / exp (I s)) F
  proof eventually-elim
    case (elim s)
    from elim have s': s ∉ ℝ≤0
```

using *complex-cone-inter-nonpos-Reals*[*of* −α α] α **by** *auto*
from *elim* **have** [*simp*]: *s* ≠ *0* **by** *auto*
from *s′* **have** *Gamma s* = *exp* (*ln-Gamma s*)
unfolding *Gamma-complex-altdef* **using** *nonpos-Ints-subset-nonpos-Reals* **by** *auto*
also from *s′* **have** *ln-Gamma s* = (*s*−*1*/*2*) ∗ *Ln s* − *s* + *complex-of-real* (*ln* (*2* ∗ *pi*) / *2*) − *I s*
by (*subst ln-Gamma-stirling-complex*[*of* - *1*]) (*simp-all add*: *exp-add exp-diff I-def*)
also have *exp* ... = *exp* ((*s* − *1* / *2*) ∗ *Ln s*) / *exp s* ∗
exp (*complex-of-real* (*ln* (*2* ∗ *pi*) / *2*)) / *exp* (*I s*)
unfolding *exp-diff exp-add* **by** (*simp add*: *exp-diff exp-add*)
also have *exp* ((*s* − *1* / *2*) ∗ *Ln s*) = *s powr* (*s* − *1* / *2*)
by (*simp add*: *powr-def*)
also have *exp* (*complex-of-real* (*ln* (*2* ∗ *pi*) / *2*)) = *sqrt* (*2* ∗ *pi*)
by (*subst exp-of-real*) (*auto simp*: *powr-def simp flip*: *powr-half-sqrt*)
also have *exp s* = *exp 1 powr s*
by (*simp add*: *powr-def*)
also have *s powr* (*s* − *1* / *2*) / *exp 1 powr s* = (*s powr s* / *exp 1 powr s*) / *s powr* (*1*/*2*)
by (*subst powr-diff*) *auto*
also have ∗: *Ln* (*s* / *exp 1*) = *Ln s* − *1*
using *Ln-divide-of-real*[*of exp 1 s*] **by** (*simp flip*: *exp-of-real*)
hence *s powr s* / *exp 1 powr s* = (*s* / *exp 1*) *powr s*
unfolding *powr-def* **by** (*subst* ∗) (*auto simp*: *exp-diff field-simps*)
finally show *Gamma s* = *sqrt* (*2* ∗ *pi*) ∗ (*s* / *exp 1*) *powr s* / *s powr* (*1* / *2*) / *exp* (*I s*)
by (*simp add*: *algebra-simps*)
qed
hence *Gamma* ∼[*F*] (λ*s*. *sqrt* (*2* ∗ *pi*) ∗ (*s* / *exp 1*) *powr s* / *s powr* (*1* / *2*) / *exp* (*I s*))
by (*rule asymp-equiv-refl-ev*)
also have ... ∼[*F*] (λ*s*. *sqrt* (*2* ∗ *pi*) ∗ (*s* / *exp 1*) *powr s* / *s powr* (*1* / *2*) / *1*)
proof (*intro asymp-equiv-intros*)
have *F* ≤ *principal* (*complex-cone′* α − {*0*})
unfolding *le-principal F-def eventually-inf-principal*
using *eventually-not-equal-at-infinity* **by** *eventually-elim auto*
moreover have *I* ∈ *O*[*principal* (*complex-cone′* α − {*0*})](λ*s*. *1* / *s*)
using *stirling-integral-bigo*[*of* α *1*] α **unfolding** *F-def* **by** (*simp add*: *I-def*)
ultimately have *I* ∈ *O*[*F*](λ*s*. *1* / *s*)
by (*rule landau-o.big.filter-mono*)
also have (λ*s*. *1* / *s*) ∈ *o*[*F*](λ*s*. *1*)
proof (*rule landau-o.smallI*)
fix *c* :: *real*
assume *c*: *c* > *0*
hence *eventually* (λ*z*::*complex*. *norm z* ≥ *1* / *c*) *at-infinity*
by (*auto simp*: *eventually-at-infinity*)
moreover have *eventually* (λ*z*::*complex*. *z* ≠ *0*) *at-infinity*
by (*rule eventually-not-equal-at-infinity*)

ultimately show *eventually* (λz::*complex*. *norm* (*1* / *z*) ≤ *c* ∗ *norm* (*1* :: *complex*)) *F*
unfolding *F-def eventually-inf-principal*
by *eventually-elim* (*use* ‹*c* > *0*› **in** ‹*auto simp*: *norm-divide field-simps*›)
qed
finally have *I* ∈ *o*[*F*](λs. *1*) **.**
from *smalloD-tendsto*[*OF this*] **have** [*tendsto-intros*]: (*I* ⟶ *0*) *F*
by *simp*
show (λx. *exp* (*I x*)) ∼[*F*] (λx. *1*)
by (*rule asymp-equivI′ tendsto-eq-intros refl* | *simp*)+
qed
finally show *?thesis* **by** *simp*
qed

end

end

References

[1] B. Berndt. *Rudiments of the Theory of the Gamma Function.* University of Chicago, 1976.

[2] A. Erdélyi. *Asymptotic Expansions.* Dover Publications, 1956.

[3] G. J. O. Jameson. A simple proof of Stirling's formula for the Gamma function. *The Mathematical Gazette*, 99:68–74, 3 2015.