The Error Function

Manuel Eberl

March 17, 2025

Abstract

This entry provides the definitions and basic properties of the complex and real error function erf and the complementary error function erfc. Additionally, it gives their full asymptotic expansions.

Contents

1 The complex and real error function

theory *Error-Function*
imports *HOL−Complex-Analysis.Complex-Analysis HOL−Library.Landau-Symbols*
begin

1.1 Auxiliary Facts

lemma *tendsto-sandwich-mono*:
assumes $(\lambda n.\ f\ (real\ n)) \longrightarrow (c{::}real)$
assumes $eventually\ (\lambda x.\ \forall y\ z.\ x \le y \wedge y \le z \longrightarrow f\ y \le f\ z)\ at\text{-}top$
shows $(f \longrightarrow c)\ at\text{-}top$
proof (*rule tendsto-sandwich*)
from *assms*(*2*) **obtain** *C* **where** C: $\bigwedge x\ y.\ C \le x \Longrightarrow x \le y \Longrightarrow f\ x \le f\ y$
by (*auto simp*: *eventually-at-top-linorder*)
show $eventually\ (\lambda x.\ f\ (real\ (nat\ \lfloor x\rfloor)) \le f\ x)\ at\text{-}top$
using *eventually-gt-at-top*[*of 0::real*] *eventually-gt-at-top*[*of C+1::real*]
by *eventually-elim* (*rule C*, *linarith+*)
show $eventually\ (\lambda x.\ f\ (real\ (Suc\ (nat\ \lfloor x\rfloor))) \ge f\ x)\ at\text{-}top$
using *eventually-gt-at-top*[*of 0::real*] *eventually-gt-at-top*[*of C+1::real*]
by *eventually-elim* (*rule C*, *linarith+*)
qed (*auto intro*!: *filterlim-compose*[*OF assms*(*1*)]
filterlim-compose[*OF filterlim-nat-sequentially*]
filterlim-compose[*OF filterlim-Suc*] *filterlim-floor-sequentially*
simp del: *of-nat-Suc*)

lemma *tendsto-sandwich-antimono*:
assumes $(\lambda n.\ f\ (real\ n)) \longrightarrow (c{::}real)$
assumes $eventually\ (\lambda x.\ \forall y\ z.\ x \le y \wedge y \le z \longrightarrow f\ y \ge f\ z)\ at\text{-}top$
shows $(f \longrightarrow c)\ at\text{-}top$
proof (*rule tendsto-sandwich*)
from *assms*(*2*) **obtain** *C* **where** C: $\bigwedge x\ y.\ C \le x \Longrightarrow x \le y \Longrightarrow f\ x \ge f\ y$
by (*auto simp*: *eventually-at-top-linorder*)
show $eventually\ (\lambda x.\ f\ (real\ (nat\ \lfloor x\rfloor)) \ge f\ x)\ at\text{-}top$
using *eventually-gt-at-top*[*of 0::real*] *eventually-gt-at-top*[*of C+1::real*]
by *eventually-elim* (*rule C*, *linarith+*)
show $eventually\ (\lambda x.\ f\ (real\ (Suc\ (nat\ \lfloor x\rfloor))) \le f\ x)\ at\text{-}top$
using *eventually-gt-at-top*[*of 0::real*] *eventually-gt-at-top*[*of C+1::real*]
by *eventually-elim* (*rule C*, *linarith+*)
qed (*auto intro*!: *filterlim-compose*[*OF assms*(*1*)]
filterlim-compose[*OF filterlim-nat-sequentially*]
filterlim-compose[*OF filterlim-Suc*] *filterlim-floor-sequentially*
simp del: *of-nat-Suc*)

lemma *has-bochner-integral-completion* [*intro*]:
fixes $f :: 'a \Rightarrow 'b{::}\{banach,\ second\text{-}countable\text{-}topology\}$
shows $has\text{-}bochner\text{-}integral\ M\ f\ I \Longrightarrow has\text{-}bochner\text{-}integral\ (completion\ M)\ f\ I$
by (*auto simp*: *has-bochner-integral-iff integrable-completion integral-completion*
borel-measurable-integrable)

lemma *has-bochner-integral-imp-has-integral*:
has-bochner-integral lebesgue (λx. indicator S x $*_R$ f x) I ⟹
(f has-integral (I :: 'b :: euclidean-space)) S
using *has-integral-set-lebesgue*[*of S f*]
by (*simp add*: *has-bochner-integral-iff set-integrable-def set-lebesgue-integral-def*)

lemma *has-bochner-integral-imp-has-integral'*:
has-bochner-integral lborel (λx. indicator S x $*_R$ f x) I ⟹
(f has-integral (I :: 'b :: euclidean-space)) S
by (*intro has-bochner-integral-imp-has-integral has-bochner-integral-completion*)

lemma *has-bochner-integral-erf-aux*:
has-bochner-integral lborel (λx. indicator {0..} x $*_R$ exp (− x^2)) (sqrt pi / 2)
proof −
let ?pI = λf. (∫⁺s. f (s::real) * indicator {0..} s ∂lborel)
let ?gauss = λx. exp (− x^2)
let ?I = indicator {0<..} :: real ⇒ real
let ?ff= λx s. x * exp (− x^2 * (1 + s^2)) :: real
have *: ?pI ?gauss = (∫⁺x. ?gauss x * ?I x ∂lborel)
by (*intro nn-integral-cong-AE AE-I*[**where** *N*={*0*}]) (*auto split*: *split-indicator*)

have ?pI ?gauss * ?pI ?gauss = (∫⁺x. ∫⁺s. ?gauss x * ?gauss s * ?I s * ?I x ∂lborel ∂lborel)
by (*auto simp*: *nn-integral-cmult*[*symmetric*] *nn-integral-multc*[*symmetric*] *
ennreal-mult[*symmetric*] *intro*!: *nn-integral-cong split*: *split-indicator*)
also have ... = (∫⁺x. ∫⁺s. ?ff x s * ?I s * ?I x ∂lborel ∂lborel)
proof (*rule nn-integral-cong*, *cases*)
fix x :: real **assume** x ≠ 0
then show (∫⁺s. ?gauss x * ?gauss s * ?I s * ?I x ∂lborel) =
(∫⁺s. ?ff x s * ?I s * ?I x ∂lborel)
by (*subst nn-integral-real-affine*[**where** *t*=*0* **and** *c*=*x*])
(*auto simp*: *mult-exp-exp nn-integral-cmult*[*symmetric*] *field-simps zero-less-mult-iff ennreal-mult*[*symmetric*]
intro!: *nn-integral-cong split*: *split-indicator*)
qed *simp*
also have ... = ∫⁺s. ∫⁺x. ?ff x s * ?I s * ?I x ∂lborel ∂lborel
by (*rule lborel-pair.Fubini'*[*symmetric*]) *auto*
also have ... = ?pI (λs. ?pI (λx. ?ff x s))
by (*rule nn-integral-cong-AE*)
(*auto intro*!: *nn-integral-cong-AE AE-I*[**where** *N*={*0*}] *split*: *split-indicator-asm*)
also have ... = ?pI (λs. ennreal (1 / (2 * (1 + s^2))))
proof (*intro nn-integral-cong ennreal-mult-right-cong*)
fix s :: real **show** ?pI (λx. ?ff x s) = ennreal (1 / (2 * (1 + s^2)))
proof (*subst nn-integral-FTC-atLeast*)
have ((λa. − (exp (− ((1 + s^2) * a^2)) / (2 + 2 * s^2))) ⟶ (− (0 / (2 + 2 * s^2)))) at-top
by (*intro tendsto-intros filterlim-compose*[*OF exp-at-bot*]
filterlim-compose[*OF filterlim-uminus-at-bot-at-top*]

```
              filterlim-tendsto-pos-mult-at-top)
          (auto intro!: filterlim-at-top-mult-at-top[OF filterlim-ident filterlim-ident]
              simp: add-pos-nonneg  power2-eq-square add-nonneg-eq-0-iff)
    then show ((λa. − (exp (− a² − s² ∗ a²) / (2 + 2 ∗ s²))) ⟶ 0) at-top
      by (simp add: field-simps)
  qed (auto intro!: derivative-eq-intros simp: field-simps add-nonneg-eq-0-iff)
 qed
 also have ... = ennreal (pi / 4)
 proof (subst nn-integral-FTC-atLeast)
  show ((λa. arctan a / 2) ⟶ (pi / 2) / 2 ) at-top
    by (intro tendsto-intros) (simp-all add: tendsto-arctan-at-top)
 qed (auto intro!: derivative-eq-intros simp: add-nonneg-eq-0-iff field-simps power2-eq-square)
 finally have ?pI ?gauss^2 = pi / 4
  by (simp add: power2-eq-square)
 then have ?pI ?gauss = sqrt (pi / 4)
  using power-eq-iff-eq-base[of 2 enn2real (?pI ?gauss) sqrt (pi / 4)]
   by (cases ?pI ?gauss) (auto simp: power2-eq-square ennreal-mult[symmetric]
ennreal-top-mult)
 also have ?pI ?gauss = (∫⁺x. indicator {0..} x ∗R exp (− x²) ∂lborel)
  by (intro nn-integral-cong) (simp split: split-indicator)
 also have sqrt (pi / 4) = sqrt pi / 2
  by (simp add: real-sqrt-divide)
 finally show ?thesis
  by (rule has-bochner-integral-nn-integral[rotated 3])
    auto
qed

lemma has-integral-erf-aux: ((λt::real. exp (−t²)) has-integral (sqrt pi / 2)) {0..}
 by (intro has-bochner-integral-imp-has-integral′ has-bochner-integral-erf-aux)

lemma contour-integrable-on-linepath-neg-exp-squared [simp, intro]:
 (λt. exp (−(t^2))) contour-integrable-on linepath 0 z
 by (auto intro!: contour-integrable-continuous-linepath continuous-intros)

lemma holomorphic-on-chain:
 g holomorphic-on t ⟹ f holomorphic-on s ⟹ f ‘ s ⊆ t ⟹
  (λx. g (f x)) holomorphic-on s
 using holomorphic-on-compose-gen[of f s g t] by (simp add: o-def)

lemma holomorphic-on-chain-UNIV:
 g holomorphic-on UNIV ⟹ f holomorphic-on s⟹
  (λx. g (f x)) holomorphic-on s
 using holomorphic-on-compose-gen[of f s g UNIV] by (simp add: o-def)

lemmas holomorphic-on-exp′ [holomorphic-intros] =
 holomorphic-on-exp [THEN holomorphic-on-chain-UNIV]

lemma leibniz-rule-field-derivative-real:
 fixes f::′a::{real-normed-field, banach} ⇒ real ⇒ ′a
```

assumes *fx*: $\bigwedge x\ t.\ x \in U \Longrightarrow t \in \{a..b\} \Longrightarrow ((\lambda x.\ f\ x\ t)$ *has-field-derivative fx x t*) (*at x within U*)
assumes *integrable-f2*: $\bigwedge x.\ x \in U \Longrightarrow (f\ x)$ *integrable-on* $\{a..b\}$
assumes *cont-fx*: *continuous-on* $(U \times \{a..b\})\ (\lambda(x, t).\ fx\ x\ t)$
assumes *U*: *x0* $\in$ *U convex U*
shows $((\lambda x.$ *integral* $\{a..b\}\ (f\ x))$ *has-field-derivative integral* $\{a..b\}$ (*fx x0*)) (*at x0 within U*)
using *leibniz-rule-field-derivative*[*of U a b f fx x0*] *assms* **by** *simp*

lemma *has-vector-derivative-linepath-within* [*derivative-intros*]:
assumes [*derivative-intros*]:
(*f has-vector-derivative f′*) (*at x within S*) (*g has-vector-derivative g′*) (*at x within S*)
(*h has-real-derivative h′*) (*at x within S*)
shows $((\lambda x.$ *linepath* (*f x*) (*g x*) (*h x*)) *has-vector-derivative*
$(1 - h\ x) *_R f' + h\ x *_R g' - h' *_R (f\ x - g\ x))$ (*at x within S*)
unfolding *linepath-def* [*abs-def*]
by (*auto intro*!: *derivative-eq-intros simp*: *field-simps scaleR-diff-right*)

lemma *has-field-derivative-linepath-within* [*derivative-intros*]:
assumes [*derivative-intros*]:
(*f has-field-derivative f′*) (*at x within S*) (*g has-field-derivative g′*) (*at x within S*)
(*h has-real-derivative h′*) (*at x within S*)
shows $((\lambda x.$ *linepath* (*f x*) (*g x*) (*h x*)) *has-field-derivative*
$(1 - h\ x) *_R f' + h\ x *_R g' - h' *_R (f\ x - g\ x))$ (*at x within S*)
unfolding *linepath-def* [*abs-def*]
by (*auto intro*!: *derivative-eq-intros simp*: *field-simps scaleR-diff-right*)

lemma *continuous-on-linepath′* [*continuous-intros*]:
assumes [*continuous-intros*]: *continuous-on A f continuous-on A g continuous-on A h*
shows *continuous-on A* $(\lambda x.$ *linepath* (*f x*) (*g x*) (*h x*))
using *assms* **unfolding** *linepath-def* **by** (*auto intro*!: *continuous-intros*)

lemma *contour-integral-has-field-derivative*:
assumes *A*: *open A convex A* $a \in A\ z \in A$
assumes *integrable*: $\bigwedge z.\ z \in A \Longrightarrow$ *f contour-integrable-on linepath a z*
assumes *holo*: *f holomorphic-on A*
shows $((\lambda z.$ *contour-integral* (*linepath a z*) *f*) *has-field-derivative f z*) (*at z within B*)
proof −
have (*f has-field-derivative deriv f z*) (*at z*) **if** $z \in A$ **for** *z*
using *that assms* **by** (*auto intro*!: *holomorphic-derivI*)
note [*derivative-intros*] = *DERIV-chain2*[*OF this*]
note [*continuous-intros*] =
continuous-on-compose2[*OF holomorphic-on-imp-continuous-on* [*OF holo*]]
continuous-on-compose2[*OF holomorphic-on-imp-continuous-on* [*OF holomorphic-deriv*[*OF holo*]]]

```
  have [derivative-intros]:
      ((λx. linepath a x t) has-field-derivative of-real t) (at x within A) for t x
    by (auto simp: linepath-def scaleR-conv-of-real intro!: derivative-eq-intros)

  have *: linepath a b t ∈ A if a ∈ A b ∈ A t ∈ {0..1} for a b t
   using that linepath-in-convex-hull[of a A b t] ‹convex A› by (simp add: hull-same)

  have ((λz. integral {0..1} (λx. f (linepath a z x)) * (z − a)) has-field-derivative
      integral {0..1} (λt. deriv f (linepath a z t) * of-real t) * (z − a) +
      integral {0..1} (λx. f (linepath a z x))) (at z within A)
      (is (- has-field-derivative ?I) -)
    by (rule derivative-eq-intros leibniz-rule-field-derivative-real)+
       (insert assms,
        auto intro!: derivative-eq-intros leibniz-rule-field-derivative-real
                     integrable-continuous-real continuous-intros
             simp:   split-beta scaleR-conv-of-real *)
  also have (λz. integral {0..1} (λx. f (linepath a z x)) * (z − a)) =
               (λz. contour-integral (linepath a z) f)
    by (simp add: contour-integral-integral)
  also have ?I = integral {0..1} (λx. deriv f (linepath a z x) * of-real x * (z −
a) +
                     f (linepath a z x)) (is - = integral - ?g)
    by (subst integral-mult-left [symmetric], subst integral-add [symmetric])
       (insert assms, auto intro!: integrable-continuous-real continuous-intros simp:
*)
  also have (?g has-integral of-real 1 * f (linepath a z 1) − of-real 0 * f (linepath
a z 0)) {0..1}
    using * A
    by (intro fundamental-theorem-of-calculus)
       (auto intro!: derivative-eq-intros has-vector-derivative-real-field
             simp: linepath-def scaleR-conv-of-real)
  hence integral {0..1} ?g = f (linepath a z 1) by (simp add: has-integral-iff)
  also have linepath a z 1 = z by (simp add: linepath-def)
 also from ‹z ∈ A› and ‹open A› have at z within A = at z by (rule at-within-open)
  finally show ?thesis by (rule DERIV-subset) simp-all
qed
```

1.2 Definition of the error function

```
definition erf-coeffs :: nat ⇒ real where
  erf-coeffs n =
     (if odd n then 2 / sqrt pi * (−1) ^ (n div 2) / (of-nat n * fact (n div 2))
      else 0)

lemma summable-erf:
  fixes z :: 'a :: {real-normed-div-algebra, banach}
  shows summable (λn. of-real (erf-coeffs n) * z ^ n)
proof −
  define b where b = (λn. 2 / sqrt pi * (if odd n then inverse (fact (n div 2))
```

else 0))
  show *?thesis*
  proof (*rule summable-comparison-test*[*OF exI*[*of - 1*]], *clarify*)
    fix *n* :: *nat* **assume** *n*: *n* ≥ *1*
    hence *norm* (*of-real* (*erf-coeffs n*) ∗ *z* ^ *n*) ≤ *b n* ∗ *norm z* ^ *n*
      unfolding *norm-mult norm-power erf-coeffs-def b-def*
      by (*intro mult-right-mono*) (*auto simp*: *field-simps norm-divide abs-mult*)
    thus *norm* (*of-real* (*erf-coeffs n*) ∗ *z* ^ *n*) ≤ *b n* ∗ *norm z* ^ *n*
      by (*simp add*: *mult-ac*)
  next
    have *summable* (λ*n*. (*norm z* ∗ *2* / *sqrt pi*) ∗ (*inverse* (*fact n*) ∗ *norm z* ^ (*2*∗*n*)))
    (**is** *summable ?c*) **unfolding** *power-mult* **by** (*intro summable-mult summable-exp*)
    also have *?c* = (λ*n*. *b* (*2*∗*n*+*1*) ∗ *norm z* ^ (*2*∗*n*+*1*))
      unfolding *b-def* **by** (*auto simp*: *fun-eq-iff b-def*)
    also have *summable* ... ⟷ *summable* (λ*n*. *b n* ∗ *norm z* ^ *n*)
      using *summable-mono-reindex* [*of* λ*n*. *2*∗*n*+*1*]
      by (*intro summable-mono-reindex* [*of* λ*n*. *2*∗*n*+*1*])
        (*auto elim*!: *oddE simp*: *strict-mono-def b-def*)
    finally show ... **.**
  qed
qed

definition *erf* :: (*'a* :: {*real-normed-field*, *banach*}) ⇒ *'a* **where**
  erf z = (∑ *n*. *of-real* (*erf-coeffs n*) ∗ *z* ^ *n*)

lemma *erf-converges*: (λ*n*. *of-real* (*erf-coeffs n*) ∗ *z* ^ *n*) *sums erf z*
  using *summable-erf* **by** (*simp add*: *sums-iff erf-def*)

lemma *erf-0* [*simp*]: *erf 0* = *0*
  unfolding *erf-def powser-zero* **by** (*simp add*: *erf-coeffs-def*)

lemma *erf-minus* [*simp*]: *erf* (−*z*) = − *erf z*
  unfolding *erf-def*
  by (*subst suminf-minus* [*OF summable-erf*, *symmetric*], *rule suminf-cong*)
    (*simp-all add*: *erf-coeffs-def*)

lemma *erf-of-real* [*simp*]: *erf* (*of-real x*) = *of-real* (*erf x*)
  unfolding *erf-def* **using** *summable-erf*[*of x*]
  by (*subst suminf-of-real*) (*simp-all add*: *summable-erf*)

lemma *of-real-erf-numeral* [*simp*]: *of-real* (*erf* (*numeral n*)) = *erf* (*numeral n*)
  by (*simp only*: *erf-of-real* [*symmetric*] *of-real-numeral*)

lemma *of-real-erf-1* [*simp*]: *of-real* (*erf 1*) = *erf 1*
  by (*simp only*: *erf-of-real* [*symmetric*] *of-real-1*)

lemma *erf-has-field-derivative*:

(*erf has-field-derivative of-real* (*2* / *sqrt pi*) ∗ *exp* (−(*z*^*2*))) (*at z within A*)
proof −
define *a'* **where** *a'* = (λ*n*. *2* / *sqrt pi* ∗
(*if even n then* (− *1*) ^ (*n div 2*) / *fact* (*n div 2*) *else 0*))

have (*erf has-field-derivative*
(∑ *n*. *diffs* (λ*n*. *of-real* (*erf-coeffs n*)) *n* ∗ *z* ^ *n*)) (*at z*)
using *summable-erf* **unfolding** *erf-def* **by** (*rule termdiffs-strong-converges-everywhere*)
also have *diffs* (λ*n*. *of-real* (*erf-coeffs n*)) = (λ*n*. *of-real* (*a' n*) :: *'a*)
by (*simp add*: *erf-coeffs-def a'-def diffs-def fun-eq-iff del*: *of-nat-Suc*)
hence (∑ *n*. *diffs* (λ*n*. *of-real* (*erf-coeffs n*)) *n* ∗ *z* ^ *n*) =
(∑ *n*. *of-real* (*a' n*) ∗ *z* ^ *n*) **by** *simp*
also have ... = (∑ *n*. *of-real* (*a'* (*2*∗*n*)) ∗ *z* ^ (*2*∗*n*))
by (*intro suminf-mono-reindex* [*symmetric*]) (*auto simp*: *strict-mono-def a'-def*
elim!: *evenE*)
also have (λ*n*. *of-real* (*a'* (*2*∗*n*)) ∗ *z* ^ (*2*∗*n*)) =
(λ*n*. *of-real* (*2* / *sqrt pi*) ∗ (*inverse* (*fact n*) ∗ (−(*z*^*2*))^*n*))
by (*simp add*: *fun-eq-iff power-mult* [*symmetric*] *a'-def field-simps power-minus'*)
also have *suminf* ... = *of-real* (*2* / *sqrt pi*) ∗ *exp* (−(*z*^*2*))
by (*subst suminf-mult*, *intro summable-exp*)
(*auto simp*: *field-simps scaleR-conv-of-real exp-def*)
finally show *?thesis* **by** (*rule DERIV-subset*) *simp-all*
qed

lemmas *erf-has-field-derivative'* [*derivative-intros*] =
erf-has-field-derivative [*THEN DERIV-chain2*]

lemma *erf-continuous-on*: *continuous-on A erf*
by (*rule DERIV-continuous-on erf-has-field-derivative*)+

lemma *continuous-on-compose2-UNIV*:
continuous-on UNIV g ⟹ *continuous-on s f* ⟹ *continuous-on s* (λ*x*. *g* (*f x*))
by (*rule continuous-on-compose2*[*of UNIV g s f*]) *simp-all*

lemmas *erf-continuous-on'* [*continuous-intros*] =
erf-continuous-on [*THEN continuous-on-compose2-UNIV*]

lemma *erf-continuous* [*continuous-intros*]: *continuous* (*at x within A*) *erf*
by (*rule continuous-within-subset*[*OF - subset-UNIV*])
(*insert erf-continuous-on*[*of UNIV*], *auto simp*: *continuous-on-eq-continuous-at*)

lemmas *erf-continuous'* [*continuous-intros*] =
continuous-within-compose2[*OF - erf-continuous*]

lemmas *tendsto-erf* [*tendsto-intros*] = *isCont-tendsto-compose*[*OF erf-continuous*]

lemma *erf-cnj* [*simp*]: *erf* (*cnj z*) = *cnj* (*erf z*)
proof −
interpret *bounded-linear cnj* **by** (*rule bounded-linear-cnj*)

from *suminf*[*OF summable-erf*] **show** *?thesis* **by** (*simp add*: *erf-def erf-coeffs-def*)
qed

lemma *integral-exp-minus-squared-real*:
assumes $a \leq b$
shows $((\lambda t.\ exp\ (-(t\hat{\ }2)))\ has\text{-}integral\ (sqrt\ pi\ /\ 2 * (erf\ b - erf\ a)))\ \{a..b\}$
proof −
have $((\lambda t.\ exp\ (-(t\hat{\ }2)))\ has\text{-}integral\ (sqrt\ pi\ /\ 2 * erf\ b - sqrt\ pi\ /\ 2 * erf\ a))$ $\{a..b\}$
using *assms*
by (*intro fundamental-theorem-of-calculus*)
(*auto intro*!: *derivative-eq-intros*
simp: *has-real-derivative-iff-has-vector-derivative* [*symmetric*])
thus *?thesis* **by** (*simp add*: *field-simps*)
qed

lemma *erf-real-altdef-nonneg*:
$x \geq 0 \Longrightarrow erf\ (x{::}real) = 2\ /\ sqrt\ pi * integral\ \{0..x\}\ (\lambda t.\ exp\ (-(t\hat{\ }2)))$
using *integral-exp-minus-squared-real*[*of 0 x*]
by (*simp add*: *has-integral-iff field-simps*)

lemma *erf-real-altdef-nonpos*:
$x \leq 0 \Longrightarrow erf\ (x{::}real) = -2\ /\ sqrt\ pi * integral\ \{0..-x\}\ (\lambda t.\ exp\ (-(t\hat{\ }2)))$
using *erf-real-altdef-nonneg*[*of* $-x$] **by** *simp*

lemma *less-imp-erf-real-less*:
assumes $a < (b{::}real)$
shows $erf\ a < erf\ b$
proof −
from *assms* **have** $\exists z.\ z > a \wedge z < b \wedge erf\ b - erf\ a = (b - a) * (2\ /\ sqrt\ pi * exp\ (-\ z^2))$
by (*intro MVT2*) (*auto intro*!: *derivative-eq-intros simp*: *field-simps*)
then obtain *z* **where** *z*: $a < z\ z < b$
and *erf*: $erf\ b - erf\ a = (b - a) * (2\ /\ sqrt\ pi * exp\ (-\ z^2))$
by *blast*
note *erf*
also from *assms* **have** $(b - a) * (2\ /\ sqrt\ pi * exp\ (-\ z^2)) > 0$
by (*intro mult-pos-pos divide-pos-pos*) *simp-all*
finally show *?thesis* **by** *simp*
qed

lemma *le-imp-erf-real-le*: $a \leq (b{::}real) \Longrightarrow erf\ a \leq erf\ b$
by (*cases* $a < b$) (*auto dest*: *less-imp-erf-real-less*)

lemma *erf-real-less-cancel* [*simp*]: $(erf\ (a :: real) < erf\ b) \longleftrightarrow a < b$
using *less-imp-erf-real-less*[*of a b*] *less-imp-erf-real-less*[*of b a*]
by (*cases a b rule*: *linorder-cases*) *simp-all*

lemma *erf-real-eq-iff* [*simp*]: *erf* (*a*::*real*) = *erf b* ⟷ *a* = *b*
by (*cases a b rule*: *linorder-cases*) (*auto dest*: *less-imp-erf-real-less*)

lemma *erf-real-le-cancel* [*simp*]: (*erf* (*a* :: *real*) ≤ *erf b*) ⟷ *a* ≤ *b*
by (*cases a b rule*: *linorder-cases*) (*auto simp*: *less-eq-real-def*)

lemma *inj-on-erf-real* [*intro*]: *inj-on* (*erf* :: *real* ⇒ *real*) *A*
by (*auto simp*: *inj-on-def*)

lemma *strict-mono-erf-real* [*intro*]: *strict-mono* (*erf* :: *real* ⇒ *real*)
by (*auto simp*: *strict-mono-def*)

lemma *mono-erf-real* [*intro*]: *mono* (*erf* :: *real* ⇒ *real*)
by (*auto simp*: *mono-def*)

lemma *erf-real-ge-0-iff* [*simp*]: *erf* (*x*::*real*) ≥ *0* ⟷ *x* ≥ *0*
using *erf-real-le-cancel*[*of 0 x*] **unfolding** *erf-0* **.**

lemma *erf-real-le-0-iff* [*simp*]: *erf* (*x*::*real*) ≤ *0* ⟷ *x* ≤ *0*
using *erf-real-le-cancel*[*of x 0*] **unfolding** *erf-0* **.**

lemma *erf-real-gt-0-iff* [*simp*]: *erf* (*x*::*real*) > *0* ⟷ *x* > *0*
using *erf-real-less-cancel*[*of 0 x*] **unfolding** *erf-0* **.**

lemma *erf-real-less-0-iff* [*simp*]: *erf* (*x*::*real*) < *0* ⟷ *x* < *0*
using *erf-real-less-cancel*[*of x 0*] **unfolding** *erf-0* **.**

lemma *erf-at-top* [*tendsto-intros*]: ((*erf* :: *real* ⇒ *real*) ⟶ *1*) *at-top*
proof −
have ∗: ($\bigcup$*n*. {*0*..*real n*}) = {*0*..} **by** (*auto intro*!: *real-nat-ceiling-ge*)
let *?f* = λ*t*::*real*. *exp* ($-t^2$)
have (λ*n*. *set-lebesgue-integral lborel* {*0*..*real n*} *?f*)
⟶ *set-lebesgue-integral lborel* ($\bigcup$*n*. {*0*..*real n*}) *?f*
using *has-bochner-integral-erf-aux*
by (*intro set-integral-cont-up*)
(*insert* ∗, *auto simp*: *incseq-def has-bochner-integral-iff set-integrable-def*)
also note ∗
also have (λ*n*. *set-lebesgue-integral lborel* {*0*..*real n*} *?f*) = (λ*n*. *integral* {*0*..*real n*} *?f*)
proof −
have $\bigwedge$*n*. *set-integrable lborel* {*0*..*real n*} (λ*x*. *exp* (− x^2))
unfolding *set-integrable-def*
by (*intro borel-integrable-compact*) (*auto intro*!: *continuous-intros*)
then show *?thesis*
by (*intro set-borel-integral-eq-integral ext*)
qed
also have . . . = (λ*n*. *sqrt pi* / *2* ∗ *erf* (*real n*)) **by** (*simp add*: *erf-real-altdef-nonneg*)

also have *set-lebesgue-integral lborel* {*0* ..} *?f* = *sqrt pi* / *2*
using *has-bochner-integral-erf-aux* **by** (*simp add*: *has-bochner-integral-iff set-lebesgue-integral-def*)
finally have (λ*n*. *2* / *sqrt pi* ∗ (*sqrt pi* / *2* ∗ *erf* (*real n*))) ⟶
(*2* / *sqrt pi*) ∗ (*sqrt pi* / *2*) **by** (*intro tendsto-intros*)
hence (λ*n*. *erf* (*real n*)) ⟶ *1* **by** *simp*
thus *?thesis* **by** (*rule tendsto-sandwich-mono*) *auto*
qed

lemma *erf-at-bot* [*tendsto-intros*]: ((*erf* :: *real* ⇒ *real*) ⟶ −*1*) *at-bot*
by (*simp add*: *filterlim-at-bot-mirror tendsto-minus-cancel-left erf-at-top*)

lemmas *tendsto-erf-at-top* [*tendsto-intros*] = *filterlim-compose*[*OF erf-at-top*]
lemmas *tendsto-erf-at-bot* [*tendsto-intros*] = *filterlim-compose*[*OF erf-at-bot*]

1.3 The complimentary error function

definition *erfc* **where** *erfc z* = *1* − *erf z*

lemma *erf-conv-erfc*: *erf z* = *1* − *erfc z* **by** (*simp add*: *erfc-def*)

lemma *erfc-0* [*simp*]: *erfc 0* = *1*
by (*simp add*: *erfc-def*)

lemma *erfc-minus*: *erfc* (−*z*) = *2* − *erfc z*
by (*simp add*: *erfc-def*)

lemma *erfc-of-real* [*simp*]: *erfc* (*of-real x*) = *of-real* (*erfc x*)
by (*simp add*: *erfc-def*)

lemma *of-real-erfc-numeral* [*simp*]: *of-real* (*erfc* (*numeral n*)) = *erfc* (*numeral n*)
by (*simp add*: *erfc-def*)

lemma *of-real-erfc-1* [*simp*]: *of-real* (*erfc 1*) = *erfc 1*
by (*simp add*: *erfc-def*)

lemma *less-imp-erfc-real-less*: *a* < (*b*::*real*) ⟹ *erfc a* > *erfc b*
by (*simp add*: *erfc-def*)

lemma *le-imp-erfc-real-le*: *a* ≤ (*b*::*real*) ⟹ *erfc a* ≥ *erfc b*
by (*simp add*: *erfc-def*)

lemma *erfc-real-less-cancel* [*simp*]: (*erfc* (*a* :: *real*) < *erfc b*) ⟷ *a* > *b*
by (*simp add*: *erfc-def*)

lemma *erfc-real-eq-iff* [*simp*]: *erfc* (*a*::*real*) = *erfc b* ⟷ *a* = *b*
by (*simp add*: *erfc-def*)

lemma *erfc-real-le-cancel* [*simp*]: (*erfc* (*a* :: *real*) ≤ *erfc b*) ⟷ *a* ≥ *b*

by (*simp add*: *erfc-def*)

lemma *inj-on-erfc-real* [*intro*]: *inj-on* (*erfc* :: *real* ⇒ *real*) *A*
by (*auto simp*: *inj-on-def*)

lemma *antimono-erfc-real* [*intro*]: *antimono* (*erfc* :: *real* ⇒ *real*)
by (*auto simp*: *antimono-def*)

lemma *erfc-real-ge-0-iff* [*simp*]: *erfc* (*x*::*real*) ≥ *1* ⟷ *x* ≤ *0*
by (*simp add*: *erfc-def*)

lemma *erfc-real-le-0-iff* [*simp*]: *erfc* (*x*::*real*) ≤ *1* ⟷ *x* ≥ *0*
by (*simp add*: *erfc-def*)

lemma *erfc-real-gt-0-iff* [*simp*]: *erfc* (*x*::*real*) > *1* ⟷ *x* < *0*
by (*simp add*: *erfc-def*)

lemma *erfc-real-less-0-iff* [*simp*]: *erfc* (*x*::*real*) < *1* ⟷ *x* > *0*
by (*simp add*: *erfc-def*)

lemma *erfc-has-field-derivative*:
(*erfc has-field-derivative* −*of-real* (*2* / *sqrt pi*) ∗ *exp* (−(*z*^*2*))) (*at z within A*)
unfolding *erfc-def* [*abs-def*] **by** (*auto intro*!: *derivative-eq-intros*)

lemmas *erfc-has-field-derivative′* [*derivative-intros*] =
erfc-has-field-derivative [*THEN DERIV-chain2*]

lemma *erfc-continuous-on*: *continuous-on A erfc*
by (*rule DERIV-continuous-on erfc-has-field-derivative*)+

lemmas *erfc-continuous-on′* [*continuous-intros*] =
erfc-continuous-on [*THEN continuous-on-compose2-UNIV*]

lemma *erfc-continuous* [*continuous-intros*]: *continuous* (*at x within A*) *erfc*
by (*rule continuous-within-subset*[*OF* - *subset-UNIV*])
(*insert erfc-continuous-on*[*of UNIV*], *auto simp*: *continuous-on-eq-continuous-at*)

lemmas *erfc-continuous′* [*continuous-intros*] =
continuous-within-compose2[*OF* - *erfc-continuous*]

lemmas *tendsto-erfc* [*tendsto-intros*] = *isCont-tendsto-compose*[*OF erfc-continuous*]

lemma *erfc-at-top* [*tendsto-intros*]: ((*erfc* :: *real* ⇒ *real*) ⟶ *0*) *at-top*
unfolding *erfc-def* [*abs-def*] **by** (*auto intro*!: *tendsto-eq-intros*)

lemma *erfc-at-bot* [*tendsto-intros*]: ((*erfc* :: *real* ⇒ *real*) ⟶ *2*) *at-bot*
unfolding *erfc-def* [*abs-def*] **by** (*auto intro*!: *tendsto-eq-intros*)

```
lemmas tendsto-erfc-at-top [tendsto-intros] = filterlim-compose[OF erfc-at-top]
lemmas tendsto-erfc-at-bot [tendsto-intros] = filterlim-compose[OF erfc-at-bot]

lemma integrable-exp-minus-squared:
  assumes A ⊆ {0..} A ∈ sets lborel
  shows   set-integrable lborel A (λt::real. exp (−t^2)) (is ?thesis1)
    and   (λt::real. exp (−t^2)) integrable-on A (is ?thesis2)
proof −
  show ?thesis1
    by (rule set-integrable-subset[of - {0..}])
      (insert assms has-bochner-integral-erf-aux, auto simp: has-bochner-integral-iff
set-integrable-def)
  thus ?thesis2 by (rule set-borel-integral-eq-integral)
qed

lemma
  assumes x ≥ 0
  shows   erfc-real-altdef-nonneg: erfc x = 2 / sqrt pi ∗ integral {x..} (λt. exp
(−t^2))
    and   has-integral-erfc:      ((λt. exp (−t^2)) has-integral (sqrt pi / 2 ∗ erfc x))
{x..}
proof −
  let ?f = λt::real. exp (−t^2)
  have int: set-integrable lborel {0..} ?f
   using has-bochner-integral-erf-aux by (simp add: has-bochner-integral-iff set-integrable-def)
  from assms have ∗: {(0::real)..} = {0..x} ∪ {x..} by auto
  have set-lebesgue-integral lborel ({0..x} ∪ {x..}) ?f =
            set-lebesgue-integral lborel {0..x} ?f + set-lebesgue-integral lborel {x..}
?f
    by (subst set-integral-Un-AE; (rule set-integrable-subset[OF int])?)
      (insert assms AE-lborel-singleton[of x], auto elim!: eventually-mono)
  also note ∗ [symmetric]
  also have set-lebesgue-integral lborel {0..} ?f = sqrt pi / 2
   using has-bochner-integral-erf-aux by (simp add: has-bochner-integral-iff set-lebesgue-integral-def)
  also have set-lebesgue-integral lborel {0..x} ?f = sqrt pi / 2 ∗ erf x
    by (subst set-borel-integral-eq-integral(2)[OF set-integrable-subset[OF int]])
      (insert assms, auto simp: erf-real-altdef-nonneg)
  also have set-lebesgue-integral lborel {x..} ?f = integral {x..} ?f
    by (subst set-borel-integral-eq-integral(2)[OF set-integrable-subset[OF int]])
      (insert assms, auto)
  finally show erfc x = 2 / sqrt pi ∗ integral {x..} ?f by (simp add: field-simps
erfc-def)
  with integrable-exp-minus-squared(2)[of {x..}] assms
  show (?f has-integral (sqrt pi / 2 ∗ erfc x)) {x..}
    by (simp add: has-integral-iff)
qed
```

lemma *erfc-real-gt-0* [*simp*, *intro*]: *erfc* (*x*::*real*) > *0*
proof (*cases x* ≥ *0*)
  case *True*
  have *0* < *integral* {*x*..*x*+*1*} (λ*t*. *exp* (−(*x*+*1*)^2)) **by** *simp*
  also from *True* **have** ... ≤ *integral* {*x*..*x*+*1*} (λ*t*. *exp* (−*t*^{2}))
    by (*intro integral-le*)
      (*auto intro*!: *integrable-continuous-real continuous-intros power-mono*)
  also have ... ≤ *sqrt pi* / *2* ∗ *erfc x*
    by (*rule has-integral-subset-le*[*OF* - *integrable-integral has-integral-erfc*])
      (*auto intro*!: *integrable-continuous-real continuous-intros True*)
  finally have *sqrt pi* / *2* ∗ *erfc x* > *0* **.**
  hence ... ∗ (*2* / *sqrt pi*) > *0* **by** (*rule mult-pos-pos*) *simp-all*
  thus *erfc x* > *0* **by** *simp*
next
  case *False*
  have *0* ≤ (*1*::*real*) **by** *simp*
  also from *False* **have** ... < *erfc x* **by** *simp*
  finally show *?thesis* **.**
qed

lemma *erfc-real-less-2* [*intro*]: *erfc* (*x*::*real*) < *2*
  using *erfc-real-gt-0*[*of* −*x*] **unfolding** *erfc-minus* **by** *simp*

lemma *erf-real-gt-neg1* [*intro*]: *erf* (*x*::*real*) > −*1*
  using *erfc-real-less-2*[*of x*] **unfolding** *erfc-def* **by** *simp*

lemma *erf-real-less-1* [*intro*]: *erf* (*x*::*real*) < *1*
  using *erfc-real-gt-0*[*of x*] **unfolding** *erfc-def* **by** *simp*

lemma *erfc-cnj* [*simp*]: *erfc* (*cnj z*) = *cnj* (*erfc z*)
  by (*simp add*: *erfc-def*)

1.4 Specific facts about the complex case

lemma *erf-complex-altdef*:
  erf z = *of-real* (*2* / *sqrt pi*) ∗ *contour-integral* (*linepath 0 z*) (λ*t*. *exp* (−(*t*^*2*)))
proof −
  define *A* **where** *A* = (λ*z*. *contour-integral* (*linepath 0 z*) (λ*t*. *exp* (−(*t*^*2*))))
  have [*derivative-intros*]: (*A has-field-derivative exp* (−(*z*^*2*))) (*at z*) **for** *z* :: *complex*
    unfolding *A-def*
    by (*rule contour-integral-has-field-derivative*[**where** *A* = *UNIV*])
      (*auto intro*!: *holomorphic-intros*)
  have *erf z* − *2* / *sqrt pi* ∗ *A z* = *erf 0* − *2* / *sqrt pi* ∗ *A 0*
    by (*rule has-derivative-zero-unique* [**where** *f* = λ*z*. *erf z* − *2* / *sqrt pi* ∗ *A z* **and** *s* = *UNIV*])
      (*auto intro*!: *has-field-derivative-imp-has-derivative derivative-eq-intros*)
  also have *A 0* = *0* **by** (*simp only*: *A-def contour-integral-trivial*)
  finally show *?thesis* **unfolding** *A-def* **by** (*simp add*: *algebra-simps*)

qed

lemma *erf-holomorphic-on*: *erf holomorphic-on A*
by (*auto simp*: *holomorphic-on-def field-differentiable-def intro*!: *erf-has-field-derivative*)

lemmas *erf-holomorphic-on′* [*holomorphic-intros*] =
erf-holomorphic-on [*THEN holomorphic-on-chain-UNIV*]

lemma *erf-analytic-on*: *erf analytic-on A*
by (*auto simp*: *analytic-on-def*) (*auto intro*!: *exI*[*of - 1*] *holomorphic-intros*)

lemma *erf-analytic-on′* [*analytic-intros*]:
assumes *f analytic-on A*
shows (λx. *erf* (*f x*)) *analytic-on A*
proof −
from *assms* **and** *erf-analytic-on* **have** *erf* ∘ *f analytic-on A*
by (*rule analytic-on-compose-gen*) *auto*
thus *?thesis* **by** (*simp add*: *o-def*)
qed

lemma *erfc-holomorphic-on*: *erfc holomorphic-on A*
by (*auto simp*: *holomorphic-on-def field-differentiable-def intro*!: *erfc-has-field-derivative*)

lemmas *erfc-holomorphic-on′* [*holomorphic-intros*] =
erfc-holomorphic-on [*THEN holomorphic-on-chain-UNIV*]

lemma *erfc-analytic-on*: *erfc analytic-on A*
by (*auto simp*: *analytic-on-def*) (*auto intro*!: *exI*[*of - 1*] *holomorphic-intros*)

lemma *erfc-analytic-on′* [*analytic-intros*]:
assumes *f analytic-on A*
shows (λx. *erfc* (*f x*)) *analytic-on A*
proof −
from *assms* **and** *erfc-analytic-on* **have** *erfc* ∘ *f analytic-on A*
by (*rule analytic-on-compose-gen*) *auto*
thus *?thesis* **by** (*simp add*: *o-def*)
qed

end

1.5 Asymptotics

theory *Error-Function-Asymptotics*
imports *Error-Function Landau-Symbols.Landau-More*
begin

lemma *real-powr-eq-powerI*:
$x > 0 \Longrightarrow y = real\ y' \Longrightarrow x\ powr\ y = x\ \hat{}\ y'$
by (*simp add*: *powr-realpow*)

definition *erf-remainder-integral* **where**
erf-remainder-integral n x =
lim (λ*m. integral* {*x..x + real m*} (λ*t. exp* (−(*t*^*2*)) / *t* ^ (*2*∗*n*)))

The following is the remainder term in the asymptotic expansion of *erfc*.

definition *erf-remainder* **where**
erf-remainder n x =
((−*1*)^*n* ∗ *2* ∗ *fact* (*2*∗*n*)) / (*sqrt pi* ∗ *4* ^ *n* ∗ *fact n*) ∗
erf-remainder-integral n x

lemma *erf-remainder-integral-aux-nonneg*:
x > *0* ⟹ *integral* {*x..x + real m*} (λ*t. exp* (−(*t*^*2*)) / *t* ^ (*2*∗*n*)) ≥ *0*
by (*intro integral-nonneg integrable-continuous-real*) (*auto intro*!: *continuous-intros*)

lemma *erf-remainder-integral-aux-bound*:
assumes *x* > *0*
shows *norm* (*integral* {*x..x + real m*} (λ*t. exp* ($-t^2$) / *t* ^ (*2*∗*n*))) ≤ *exp* ($-x^2$) / *x* ^ (*2*∗*n*+*1*)
and *integral* {*x..x + real m*} (λ*t. exp* ($-t^2$) / *t* ^ (*2*∗*n*)) ≤ *exp* ($-x^2$) / *x* ^ (*2*∗*n*+*1*)
proof −
have *norm* (*integral* {*x..x + real m*} (λ*t. exp* ($-t^2$) / *t* ^ (*2*∗*n*))) ≤
integral {*x..x + real m*} (λ*t. exp* (−*x*∗*t*) / *x* ^ (*2*∗*n*))
proof (*intro integral-norm-bound-integral ballI*)
fix *t* **assume** *t*: *t* ∈ {*x..x + real m*}
from *t* **have** *norm* (*exp* ($-t^2$) / *t* ^ (*2*∗*n*)) = *exp* ($-t^2$) / *t* ^ (*2*∗*n*) **by** *simp*
also have . . . ≤ *exp* (−*x*∗*t*) / *x* ^ (*2*∗*n*) **using** *t assms*
by (*intro frac-le*) (*simp-all add*: *self-le-power power2-eq-square power-mono*)
finally show *norm* (*exp* ($-t^2$) / *t* ^ (*2*∗*n*)) ≤ . . . **by** *simp*
qed (*insert assms*, *auto intro*!: *continuous-intros integrable-continuous-real*)
also have . . . = −*exp* (−*x*∗(*x + real m*)) / *x* ^ (*2*∗*n*+*1*) − (−*exp* (−*x*∗*x*) / *x* ^ (*2*∗*n*+*1*))
using *assms*
by (*intro integral-unique fundamental-theorem-of-calculus*)
(*auto simp*: *has-real-derivative-iff-has-vector-derivative* [*symmetric*]
intro!: *derivative-eq-intros*)
also have . . . ≤ *exp* ($-x^2$) / *x* ^ (*2*∗*n*+*1*) **using** *assms* **by** (*simp add*: *power2-eq-square*)
finally show ∗: *norm* (*integral* {*x..x + real m*} (λ*t. exp* ($-t^2$) / *t* ^ (*2*∗*n*))) ≤
exp ($-x^2$) / *x* ^ (*2*∗*n*+*1*) **.**
have *integral* {*x..x + real m*} (λ*t. exp* ($-t^2$) / *t* ^ (*2*∗*n*)) ≤
norm (*integral* {*x..x + real m*} (λ*t. exp* ($-t^2$) / *t* ^ (*2*∗*n*))) **by** *simp*
also note ∗
finally show *integral* {*x..x + real m*} (λ*t. exp* ($-t^2$) / *t* ^ (*2*∗*n*)) ≤ *exp* ($-x^2$) / *x* ^ (*2*∗*n*+*1*) **.**
qed

lemma *convergent-erf-remainder-integral*:

assumes *x* > *0*
shows *convergent* (λ*m. integral* {*x..x* + *real m*} (λ*t. exp* (−(*t*^*2*)) / *t* ^ (*2*∗*n*)))
proof (*intro Bseq-mono-convergent BseqI′*; *clarify?*)
fix *m* :: *nat*
show *norm* (*integral* {*x..x* + *real m*} (λ*t. exp* (−t^2) / *t* ^ (*2*∗*n*))) ≤ *exp* (−x^2) / *x* ^ (*2*∗*n*+*1*)
using *assms* **by** (*rule erf-remainder-integral-aux-bound*)
qed (*insert assms*, *auto intro*!: *integral-subset-le integrable-continuous-real continuous-intros*)

lemma *LIMSEQ-erf-remainder-integral*:
x > *0* ⟹ (λ*m. integral* {*x..x* + *real m*} (λ*t. exp* (−(*t*^*2*)) / *t* ^ (*2*∗*n*))) ⟶ *erf-remainder-integral n x*
using *convergent-erf-remainder-integral*[*of x*]
by (*simp add*: *convergent-LIMSEQ-iff erf-remainder-integral-def*)

We show some bounds on the remainder term.

lemma
assumes *x* > *0*
shows *erf-remainder-integral-nonneg*: *erf-remainder-integral n x* ≥ *0*
and *erf-remainder-integral-bound*: *erf-remainder-integral n x* ≤ *exp* (−x^2) / *x* ^ (*2*∗*n*+*1*)
proof −
note ∗ = *LIMSEQ-erf-remainder-integral*[*OF assms*]
show *erf-remainder-integral n x* ≥ *0*
by (*intro tendsto-le*[*OF* - ∗ *tendsto-const*] *always-eventually erf-remainder-integral-aux-nonneg allI assms sequentially-bot*)
show *erf-remainder-integral n x* ≤ *exp* (−x^2) / *x* ^ (*2*∗*n*+*1*)
by (*intro tendsto-le*[*OF* - *tendsto-const* ∗] *always-eventually erf-remainder-integral-aux-bound allI assms sequentially-bot*)
qed

lemma *erf-remainder-integral-bigo*:
erf-remainder-integral n ∈ *O*(λ*x. exp* (−x^2) / *x* ^ (*2*∗*n*+*1*))
using *erf-remainder-integral-nonneg erf-remainder-integral-bound*
by (*auto intro*!: *bigoI*[*of* - *1*] *eventually-mono* [*OF eventually-gt-at-top*[*of 0*::*real*]])

theorem *erf-remainder-bigo*: *erf-remainder n* ∈ *O*(λ*x. exp* (−x^2) / *x* ^ (*2*∗*n*+*1*))
using *erf-remainder-integral-bigo*[*of n*] **by** (*simp add*: *erf-remainder-def* [*abs-def*])

Next, we unroll the remainder term to develop the asymptotic expansion.

lemma *erf-remainder-integral-0-conv-erfc*:
assumes (*x*::*real*) > *0*
shows *erf-remainder-integral 0 x* = *sqrt pi* / *2* ∗ *erfc x*
proof −
have (λ*m. sqrt pi* / *2* ∗ (*erf* (*x* + *real m*) − *erf x*)) ⟶ *sqrt pi* / *2* ∗ *erfc x*
(**is** *filterlim ?f* - -) **unfolding** *erfc-def*
by (*intro tendsto-intros filterlim-tendsto-add-at-top*[*OF tendsto-const filterlim-real-sequentially*])

also have *?f* = (λ*m*. *integral* {*x*..*x*+*real m*} (λ*t*. *exp* ($-t^2$)))
by (*auto simp*: *fun-eq-iff integral-unique*[*OF integral-exp-minus-squared-real*])
finally have (λ*m*. *integral* {*x*..*x* + *real m*} (λ*t*. *exp* (− t^2))) ⟶ *sqrt pi* / *2* ∗ *erfc x* **.**
moreover have (λ*m*. *integral* {*x*..*x* + *real m*} (λ*t*. *exp* (− t^2))) ⟶ *erf-remainder-integral 0 x*
using *LIMSEQ-erf-remainder-integral*[*of x 0*] *assms* **by** *simp*
ultimately show *erf-remainder-integral 0 x* = *sqrt pi* / *2* ∗ *erfc x*
by (*intro LIMSEQ-unique*)
qed

The first remainder is the *erfc* function itself.

lemma *erf-remainder-0-conv-erfc*: *x* > *0* ⟹ *erf-remainder 0 x* = *erfc x*
by (*simp add*: *erf-remainder-def erf-remainder-integral-0-conv-erfc*)

Also, the following recurrence allows us to get the next term of the asymptotic expansion.

lemma *erf-remainder-integral-conv-Suc*:
assumes *x* > *0*
shows *erf-remainder-integral n x* = *exp* ($-x^2$) / (*2*∗*x*^(*2*∗*n*+*1*)) −
real (*2*∗*n*+*1*) / *2* ∗ *erf-remainder-integral* (*Suc n*) *x*
proof −
let *?A* = λ*m*. {*x*..*x* + *real m*}
let *?J* = λ*m n*. *integral* {*x*..*x*+*real m*} (λ*t*. *exp*($-t^2$) / *t* ^ (*2*∗*n*))
define *I* **where**
I = (λ*m*. *exp* (− (x + *real m*)^2) / (− *2* ∗ (*x* + *real m*) ^ (*2* ∗ *n* + *1*)) −
exp (− x^2) ∗ *inverse* (− *2* ∗ *x* ^ (*2* ∗ *n* + *1*)) − *real* (*2*∗*n*+*1*)/*2* ∗ *?J m* (*Suc n*))

have *I-eq*: *I* = (λ*m*. *integral* (*?A m*) (λ*t*. *exp* (− t^2) / *t* ^ (*2* ∗ *n*)))
proof
fix *m* :: *nat*
have ((λ*t*. (−*2*∗*t*∗*exp* (−(*t*^*2*))) ∗ *inverse* (−*2*∗*t*^(*2*∗*n*+*1*))) *has-integral I m*) (*?A m*)
proof (*rule integration-by-parts*[*OF bounded-bilinear-mult*])
fix *t* **assume** *t*: *t* ∈ *?A m*
with *assms* **show** ((λ*t*. *exp* (−(*t*^*2*))) *has-vector-derivative* −*2*∗*t*∗*exp* (−(*t*^*2*))) (*at t*)
by (*auto simp*: *has-real-derivative-iff-has-vector-derivative* [*symmetric*]
field-simps intro!: *derivative-eq-intros*)
from *assms t* **have** ((λ*t*. −(*1*/*2*) ∗ *t powr* (−*2*∗*n*−*1*)) *has-field-derivative*
(*2*∗*n*+*1*)/*2* ∗ *t powr* (−*2*∗*n*−*2*)) (*at t*)
by (*auto intro*!: *derivative-eq-intros simp*: *field-simps powr-numeral power2-eq-square*
powr-minus powr-divide [*symmetric*] *powr-add*)
also have *?this* ⟷ ((λ*t*. *inverse* (−*2*∗*t*^(*2*∗*n*+*1*))) *has-field-derivative*
(*2*∗*n*+*1*)/*2* / *t* ^ (*2*∗*Suc n*)) (*at t*) **using** *t*
using *eventually-nhds-in-open*[*of* {*0*<..} *t*] *assms*
by (*intro DERIV-cong-ev refl*)
(*auto elim*!: *eventually-mono simp*: *powr-minus field-simps powr-diff*

powr-realpow power2-eq-square intro!: *real-powr-eq-powerI*)
finally show ((λ*t*. *inverse* (−*2*∗*t*⌢(*2*∗*n*+*1*))) *has-vector-derivative*
(*2*∗*n*+*1*)/*2* / *t* ⌢(*2*∗*Suc n*)) (*at t*)
by (*simp add*: *has-real-derivative-iff-has-vector-derivative*)
next
have ((λ*t*. *real* (*2*∗*n*+*1*)/*2* ∗ (*exp* (− t^2) / *t* ⌢(*2* ∗ *Suc n*))) *has-integral*
real (*2*∗*n*+*1*)/*2* ∗ *?J m* (*Suc n*)) (*?A m*) (**is** (*?f has-integral ?a*) -)
using *assms*
by (*intro has-integral-mult-right integrable-integral integrable-continuous-real*)
(*auto intro*!: *continuous-intros*)
also have *?f* = (λ*t*. *exp* (− t^2) ∗ (*real* (*2* ∗ *n* + *1*) / *2* / *t* ⌢(*2* ∗ *Suc n*)))
by (*simp add*: *fun-eq-iff field-simps*)
also have *?a* = *exp* (− (*x* + *real m*)^2) ∗ *inverse* (− *2* ∗ (*x* + *real m*) ⌢(*2* ∗
n + *1*)) −
exp (− x^2) ∗ *inverse* (− *2* ∗ *x* ⌢(*2* ∗ *n* + *1*)) − *I m* **using**
assms
by (*simp add*: *I-def algebra-simps inverse-eq-divide*)
finally show ((λ*t*. *exp* (− t^2) ∗ (*real* (*2* ∗ *n* + *1*) / *2* / *t* ⌢(*2* ∗ *Suc n*)))
has-integral . . .)
{*x*..*x* + *real m*} **.**
qed (*insert assms*, *auto intro*!: *continuous-intros*)
hence *I m* = *integral* {*x*..*x* + *real m*} (λ*t*. − *2*∗*t*∗*exp* (− t^2) ∗ *inverse* (−*2*∗*t*⌢(*2*
∗ *n* + *1*)))
by (*simp add*: *has-integral-iff*)
also have . . . = *integral* {*x*..*x* + *real m*} (λ*t*. *exp* (− t^2) / *t* ⌢(*2*∗*n*))
using *assms* **by** (*intro integral-cong*) (*simp-all add*: *field-simps*)
finally show *I m* = . . . **.**
qed

have *filterlim* (λ*m*. (−*exp* (− (*x* + *real m*)^2)) / (*2* ∗ (*x* + *real m*) ⌢(*2* ∗ *n* +
1)))
(*nhds 0*) *at-top*
by (*rule real-tendsto-divide-at-top filterlim-real-sequentially tendsto-minus*
filterlim-compose[*OF exp-at-bot*] *filterlim-compose*[*OF filterlim-uminus-at-bot-at-top*]
filterlim-pow-at-top filterlim-tendsto-add-at-top tendsto-const filterlim-ident
filterlim-tendsto-pos-mult-at-top | *simp*)+
hence ∗: *filterlim* (λ*m*. (*exp* (− (*x* + *real m*)^2)) / (−*2* ∗ (*x* + *real m*) ⌢(*2* ∗ *n*
+ *1*)))
(*nhds 0*) *at-top* **by** (*simp add*: *add-ac*)
have *I* ⟶ *0* − *exp* (− x^2) ∗ *inverse* (− *2* ∗ *x* ⌢(*2* ∗ *n* + *1*)) −
real (*2* ∗ *n* + *1*) / *2* ∗ *erf-remainder-integral* (*Suc n*) *x*
unfolding *I-def*
by (*intro tendsto-diff* ∗ *tendsto-const tendsto-mult LIMSEQ-erf-remainder-integral*
assms)
moreover from *LIMSEQ-erf-remainder-integral*[*OF assms*, *of n*] *I-eq*
have *I* ⟶ *erf-remainder-integral n x* **by** *simp*
ultimately have *0* − *exp* (− x^2) ∗ *inverse* (− *2* ∗ *x* ⌢(*2* ∗ *n* + *1*)) − *real* (*2*
∗ *n* + *1*) / *2* ∗
erf-remainder-integral (*Suc n*) *x* = *erf-remainder-integral n x*

```
    by (rule LIMSEQ-unique)
  thus ?thesis by (simp add: field-simps)
qed

lemma erf-remainder-conv-Suc:
  assumes x > 0
  shows   erf-remainder n x = (− 1) ^ n ∗ fact (2 ∗ n) / (sqrt pi ∗ 4 ^ n ∗ fact n) ∗
                 exp (− x²) / (x ^ (2 ∗ n + 1)) + erf-remainder (Suc n) x
proof −
  have erf-remainder n x =
          (− 1) ^ n ∗ 2 ∗ fact (2 ∗ n) / (sqrt pi ∗ 4 ^ n ∗ fact n) ∗
            exp (− x²) / (2 ∗ x ^ (2 ∗ n + 1)) + −(
          (− 1) ^ n ∗ 2 ∗ fact (2 ∗ n) / (sqrt pi ∗ 4 ^ n ∗ fact n) ∗
            real (2 ∗ n + 1) / 2 ∗ erf-remainder-integral (Suc n) x) (is - = ?A + ?B)
    unfolding erf-remainder-def using assms
    by (subst erf-remainder-integral-conv-Suc)
       (auto simp: assms algebra-simps simp del: power-Suc)
  also have ?B = erf-remainder (Suc n) x
    by (simp add: divide-simps erf-remainder-def)
  also have ?A = (− 1) ^ n ∗ fact (2 ∗ n) / (sqrt pi ∗ 4 ^ n ∗ fact n) ∗
                 exp (− x²) / (x ^ (2 ∗ n + 1))
    by (simp add: divide-simps)
  finally show ?thesis .
qed
```

Finally, this gives us the full asymptotic expansion for *erfc*:

```
theorem erfc-unroll:
  assumes x > 0
  shows   erfc x = exp (−x²) / sqrt pi ∗
           (∑ i<n. (−1)^i ∗ fact (2∗i) / (4^i∗fact i) / x^(2∗i+1)) + erf-remainder n x
proof (induction n)
  case (Suc n)
  note Suc.IH
  also note erf-remainder-conv-Suc[OF assms, of n]
  also have exp (− x²) / sqrt pi ∗
               (∑ i<n. (− 1) ^ i ∗ fact (2 ∗ i) / (4 ^ i ∗ fact i) / x ^ (2∗i+1)) +
                 ((− 1) ^ n ∗ fact (2∗n) / (sqrt pi ∗ 4 ^ n ∗ fact n) ∗ exp (− x²) / x^(2∗n+1) +
                 erf-remainder (Suc n) x) =
               exp (− x²) / sqrt pi ∗
                 (∑ i<Suc n. (− 1) ^ i ∗ fact (2 ∗ i) / (4 ^ i ∗ fact i) / x ^ (2 ∗ i + 1)) +
                 erf-remainder (Suc n) x
    by (subst sum.lessThan-Suc) (simp add: algebra-simps)
  finally show ?case .
qed (auto simp: assms erf-remainder-0-conv-erfc)
```

For convenience, we define another auxiliary function that is more suitable for use in an automated expansion framework, since it has a simple asymptotic expansion in powers of x.

definition *erfc-aux* **where** *erfc-aux x = exp* (x^2) ∗ *sqrt pi* ∗ *erfc x*
definition *erf-remainder′* **where** *erf-remainder′ n x = exp* (x^2) ∗ *sqrt pi* ∗ *erf-remainder n x*

lemma *erfc-aux-unroll*:
x > 0 ⟹
erfc-aux x = (∑ *i<n*. (−*1*)^*i* ∗ *fact* (*2*∗*i*) / (*4*^*i*∗*fact i*) / *x*^(*2*∗*i*+*1*)) + *erf-remainder′ n x*
using *erfc-unroll*[*of x n*]
by (*simp add*: *erfc-aux-def erf-remainder′-def exp-minus field-simps del*: *of-nat-Suc*)

lemma *erf-remainder′-bigo*: *erf-remainder′ n* ∈ *O*(λ*x*. *1* / *x* ^ (*2*∗*n*+*1*))
proof −
have (λ*x*. *exp* (x^2) ∗ *erf-remainder n x*) ∈ *O*(λ*x*. *exp* (x^2) ∗ (*exp* $(-x^2)$ / *x* ^ (*2*∗*n*+*1*)))
by (*intro landau-o.big.mult erf-remainder-bigo*) *simp-all*
thus *?thesis* **by** (*simp add*: *exp-minus erf-remainder′-def* [*abs-def*])
qed

lemma *has-field-derivative-erfc-aux*:
(*erfc-aux has-field-derivative* (*2* ∗ *x* ∗ *erfc-aux x* − *2*)) (*at x*)
by (*auto simp*: *erfc-aux-def* [*abs-def*] *exp-minus field-simps intro*!: *derivative-eq-intros*)

end