Laplace Transform

Fabian Immler

March 17, 2025

Abstract

This entry formalizes the Laplace transform and concrete Laplace transforms for arithmetic functions, frequency shift, integration and (higher) differentiation in the time domain. It proves Lerch's lemma and uniqueness of the Laplace transform for continuous functions. In order to formalize the foundational assumptions, this entry contains a formalization of piecewise continuous functions and functions of exponential order.

Contents

theory *Laplace-Transform-Library*
  imports
    HOL−Analysis.Analysis
begin

1 References

Much of this formalization is based on Schiff's textbook [3]. Parts of this formalization are inspired by the HOL-Light formalization ([4], [1], [2]), but stated more generally for piecewise continuous (instead of piecewise continuously differentiable) functions.

2 Library Additions

2.1 Derivatives

lemma *DERIV-compose-FDERIV*:— TODO: generalize and move from HOL-ODE
  assumes *DERIV f* $(g\ x)$:> f'
  assumes $(g$ *has-derivative* $g')$ $(at\ x\ within\ s)$
  shows $((\lambda x.\ f\ (g\ x))$ *has-derivative* $(\lambda x.\ g'\ x * f'))$ $(at\ x\ within\ s)$
  using *assms has-derivative-compose*[*of g g' x s f* (∗) *f'*]
  by (*auto simp*: *has-field-derivative-def ac-simps*)

lemmas *has-derivative-sin*[*derivative-intros*] = *DERIV-sin*[*THEN DERIV-compose-FDERIV*]
 and *has-derivative-cos*[*derivative-intros*] = *DERIV-cos*[*THEN DERIV-compose-FDERIV*]
 and *has-derivative-exp*[*derivative-intros*] = *DERIV-exp*[*THEN DERIV-compose-FDERIV*]

2.2 Integrals

lemma *negligible-real-ivlI*:
  fixes *a b*::*real*
  assumes $a \geq b$
  shows *negligible* $\{a\ ..\ b\}$
proof −
  from *assms* **have** $\{a\ ..\ b\} = \{a\} \vee \{a\ ..\ b\} = \{\}$
    by *auto*
  then show *?thesis*
    by *auto*
qed

lemma *absolutely-integrable-on-combine*:
  fixes f :: *real* ⇒ *'a*::*euclidean-space*
  assumes *f absolutely-integrable-on* $\{a..c\}$
    and *f absolutely-integrable-on* $\{c..b\}$
    and $a \leq c$
    and $c \leq b$
  shows *f absolutely-integrable-on* $\{a..b\}$
  using *assms*
  unfolding *absolutely-integrable-on-def integrable-on-def*
  by (*auto intro*!: *has-integral-combine*)

lemma *dominated-convergence-at-top*:

```
  fixes f :: real ⇒ 'n::euclidean-space ⇒ 'm::euclidean-space
  assumes f: ⋀k. (f k) integrable-on s and h: h integrable-on s
    and le: ⋀k x. x ∈ s ⟹ norm (f k x) ≤ h x
    and conv: ∀x ∈ s. ((λk. f k x) ⟶ g x) at-top
  shows g integrable-on s ((λk. integral s (f k)) ⟶ integral s g) at-top
proof –
  have 3: set-integrable lebesgue s h
    unfolding absolutely-integrable-on-def
  proof
    show (λx. norm (h x)) integrable-on s
    proof (intro integrable-spike-finite[OF - - h, where S={}] ballI)
      fix x assume x ∈ s − {} then show norm (h x) = h x
        using order-trans[OF norm-ge-zero le[of x]] by auto
    qed auto
  qed fact
  have 2: set-borel-measurable lebesgue s (f k) for k
    using f[of k]
    using has-integral-implies-lebesgue-measurable[of f k]
    by (auto intro: simp: integrable-on-def set-borel-measurable-def)
  have conv': ∀x ∈ s. ((λk. f k x) ⟶ g x) sequentially
     using conv filterlim-filtermap filterlim-compose filterlim-real-sequentially by
blast
  from 2 have 1: set-borel-measurable lebesgue s g
    unfolding set-borel-measurable-def
   by (rule borel-measurable-LIMSEQ-metric) (use conv' in ‹auto split: split-indicator›)
  have 4: AE x in lebesgue. ((λi. indicator s x *R f i x) ⟶ indicator s x *R g
x) at-top
    ∀F i in at-top. AE x in lebesgue. norm (indicator s x *R f i x) ≤ indicator s x
*R h x
    using conv le by (auto intro!: always-eventually split: split-indicator)

  note 1 2 3 4
  note * = this[unfolded set-borel-measurable-def set-integrable-def]
  have g: g absolutely-integrable-on s
    unfolding set-integrable-def
    by (rule integrable-dominated-convergence-at-top[OF *])
  then show g integrable-on s
    by (auto simp: absolutely-integrable-on-def)
  have ((λk. (LINT x:s|lebesgue. f k x)) ⟶ (LINT x:s|lebesgue. g x)) at-top
    unfolding set-lebesgue-integral-def
    using *
    by (rule integral-dominated-convergence-at-top)
  then show ((λk. integral s (f k)) ⟶ integral s g) at-top
    using g absolutely-integrable-integrable-bound[OF le f h]
    by (subst (asm) (1 2) set-lebesgue-integral-eq-integral) auto
qed

lemma has-integral-dominated-convergence-at-top:
  fixes f :: real ⇒ 'n::euclidean-space ⇒ 'm::euclidean-space
```

assumes $\bigwedge k.$ (*f k has-integral y k*) *s h integrable-on s*
$\bigwedge k\ x.\ x{\in}s \Longrightarrow$ *norm* (*f k x*) $\leq$ *h x* $\forall x{\in}s.$ (($\lambda k.$ *f k x*) $\longrightarrow$ *g x*) *at-top*
and *x*: (*y* $\longrightarrow$ *x*) *at-top*
shows (*g has-integral x*) *s*
proof −
have *int-f*: $\bigwedge k.$ (*f k*) *integrable-on s*
using *assms* **by** (*auto simp*: *integrable-on-def*)
have (*g has-integral* (*integral s g*)) *s*
by (*intro integrable-integral dominated-convergence-at-top*[*OF int-f assms*(*2*)]) *fact+*
moreover have *integral s g* = *x*
proof (*rule tendsto-unique*)
show (($\lambda i.$ *integral s* (*f i*)) $\longrightarrow$ *x*) *at-top*
using *integral-unique*[*OF assms*(*1*)] *x* **by** *simp*
show (($\lambda i.$ *integral s* (*f i*)) $\longrightarrow$ *integral s g*) *at-top*
by (*intro dominated-convergence-at-top*[*OF int-f assms*(*2*)]) *fact+*
qed *simp*
ultimately show *?thesis*
by *simp*
qed

lemma *integral-indicator-eq-restriction*:
fixes *f*::*'a*::*euclidean-space* ⇒ *'b*::*banach*
assumes *f*: *f integrable-on R*
and $R \subseteq S$
shows *integral S* ($\lambda x.$ *indicator R x* $*_R$ *f x*) = *integral R f*
proof −
let *?f* = $\lambda x.$ *indicator R x* $*_R$ *f x*
have *?f integrable-on R*
using *f negligible-empty*
by (*rule integrable-spike*) *auto*
from *integrable-integral*[*OF this*]
have (*?f has-integral integral R ?f*) *S*
by (*rule has-integral-on-superset*) (*use* ‹$R \subseteq S$› **in** ‹*auto simp*: *indicator-def*›)
also have *integral R ?f* = *integral R f*
using *negligible-empty*
by (*rule integral-spike*) *auto*
finally show *?thesis*
by *blast*
qed

lemma
improper-integral-at-top:
fixes *f*::*real* ⇒ *'a*::*euclidean-space*
assumes *f absolutely-integrable-on* {*a..*}
shows (($\lambda x.$ *integral* {*a..x*} *f*) $\longrightarrow$ *integral* {*a..*} *f*) *at-top*
proof −
let *?f* = λ(*k*::*real*) (*t*::*real*). *indicator* {*a..k*} *t* $*_R$ *f t*
have *f*: *f integrable-on* {*a..k*} **for** *k*

using *set-lebesgue-integral-eq-integral*(*1*)[*OF assms*]
by (*rule integrable-on-subinterval*) *simp*
from *this negligible-empty* **have** *?f k integrable-on* {*a..k*} **for** *k*
by (*rule integrable-spike*) *auto*
from *this* **have** *?f k integrable-on* {*a..*} **for** *k*
by (*rule integrable-on-superset*) *auto*
moreover
have (λ*x*. *norm* (*f x*)) *integrable-on* {*a..*}
using *assms* **by** (*simp add*: *absolutely-integrable-on-def*)
moreover
note -
moreover
have $\forall_F$ *k in at-top*. *k* ≥ *x* **for** *x*::*real*
by (*simp add*: *eventually-ge-at-top*)
then have ∀ *x*∈{*a..*}. ((λ*k*. *?f k x*) ⟶ *f x*) *at-top*
by (*auto intro*!: *Lim-transform-eventually*[*OF tendsto-const*] *simp*: *indicator-def* *eventually-at-top-linorder*)
ultimately
have ((λ*k*. *integral* {*a..*} (*?f k*)) ⟶ *integral* {*a ..*} *f*) *at-top*
by (*rule dominated-convergence-at-top*) (*auto simp*: *indicator-def*)
also have (λ*k*. *integral* {*a..*} (*?f k*)) = (λ*k*. *integral* {*a..k*} *f*)
by (*auto intro*!: *ext integral-indicator-eq-restriction f*)
finally show *?thesis* **.**
qed

lemma *norm-integrable-onI*: (λ*x*. *norm* (*f x*)) *integrable-on S*
if *f absolutely-integrable-on S*
for *f*::*'a*::*euclidean-space*⇒*'b*::*euclidean-space*
using *that* **by** (*auto simp*: *absolutely-integrable-on-def*)

lemma
has-integral-improper-at-topI:
fixes *f*::*real* ⇒ *'a*::*banach*
assumes *I*: $\forall_F$ *k in at-top*. (*f has-integral I k*) {*a..k*}
assumes *J*: (*I* ⟶ *J*) *at-top*
shows (*f has-integral J*) {*a..*}
apply (*subst has-integral'*)
proof (*auto*, *goal-cases*)
case (*1 e*)
from *tendstoD*[*OF J* ‹*0* < *e*›]
have $\forall_F$ *x in at-top*. *dist* (*I x*) *J* < *e* **.**
moreover have $\forall_F$ *x in at-top*. (*x*::*real*) > *0* **by** *simp*
moreover have $\forall_F$ *x in at-top*. (*x*::*real*) > − *a*— TODO: this seems to be strange?
by *simp*
moreover note *I*
ultimately have $\forall_F$ *x in at-top*. *x* > *0* ∧ *x* > − *a* ∧ *dist* (*I x*) *J* < *e* ∧
(*f has-integral I x*) {*a..x*} **by** *eventually-elim auto*
then obtain *k* **where** *k*: ∀ *b*≥*k*. *norm* (*I b* − *J*) < *e k* > *0 k* > − *a*

```
    and I: ⋀c. c ≥ k ⟹ (f has-integral I c) {a..c}
    by (auto simp: eventually-at-top-linorder dist-norm)
  show ?case
    apply (rule exI[where x=k])
    apply (auto simp: ‹0 < k›)
    subgoal premises prems for b c
    proof -
     have ball-eq: ball 0 k = {−k <..< k} by (auto simp: abs-real-def split: if-splits)
      from prems ‹0 < k› have c ≥ 0 b ≤ 0
        by (auto simp: subset-iff)
      with prems ‹0 < k› have c ≥ k
        apply (auto simp: ball-eq)
        apply (auto simp: subset-iff)
        apply (drule spec[where x=(c + k)/2])
        apply (auto simp: algebra-split-simps not-less)
        using ‹0 ≤ c› by linarith
      then have norm (I c − J) < e using k by auto
      moreover
      from prems ‹0 < k› ‹c ≥ 0› ‹b ≤ 0› ‹c ≥ k› ‹k > − a› have a ≥ b
        apply (auto simp: ball-eq)
        apply (auto simp: subset-iff)
        by (meson ‹b ≤ 0› less-eq-real-def minus-less-iff not-le order-trans)
      have ((λx. if x ∈ cbox a c then f x else 0) has-integral I c) (cbox b c)
        apply (subst has-integral-restrict-closed-subintervals-eq)
        using I[of c] prems ‹a ≥ b› ‹k ≤ c›
        by (auto )
      from negligible-empty - this have ((λx. if a ≤ x then f x else 0) has-integral
I c) (cbox b c)
        by (rule has-integral-spike) auto
      ultimately
      show ?thesis
        by (intro exI[where x=I c]) auto
    qed
    done
qed

lemma has-integral-improperE:
  fixes f::real ⇒ 'a::euclidean-space
  assumes I: (f has-integral I) {a..}
  assumes ai: f absolutely-integrable-on {a..}
  obtains J where
    ⋀k. (f has-integral J k) {a..k}
    (J ⟶ I) at-top
proof -
  define J where J k = integral {a .. k} f for k
  have (f has-integral J k) {a..k} for k
    unfolding J-def
    by (force intro: integrable-on-subinterval has-integral-integrable[OF I])
  moreover
```

have *I-def*[*symmetric*]: *integral* {*a*..} *f* = *I*
using *I* **by** *auto*
from *improper-integral-at-top*[*OF ai*]
have (*J* ⟶ *I*) *at-top*
unfolding *J-def I-def* **.**
ultimately show *?thesis* **..**
qed

2.3 Miscellaneous

lemma *AE-BallI*: *AE x*∈*X in F*. *P x* **if** ∀ *x* ∈ *X*. *P x*
using *that* **by** (*intro always-eventually*) *auto*

lemma *bounded-le-Sup*:
assumes *bounded* (*f* ' *S*)
shows ∀ *x*∈*S*. *norm* (*f x*) ≤ *Sup* (*norm* ' *f* ' *S*)
by (*auto intro*!: *cSup-upper bounded-imp-bdd-above simp*: *bounded-norm-comp assms*)

end

3 Piecewise Continous Functions

theory *Piecewise-Continuous*
imports
Laplace-Transform-Library
begin

3.1 at within filters

lemma *at-within-self-singleton*[*simp*]: *at i within* {*i*} = *bot*
by (*auto intro*!: *antisym filter-leI simp*: *eventually-at-filter*)

lemma *at-within-t1-space-avoid*:
(*at x within X* − {*i*}) = (*at x within X*) **if** *x* ≠ *i* **for** *x i*::'*a*::*t1-space*
proof (*safe intro*!: *antisym filter-leI*)
fix *P*
assume *eventually P* (*at x within X* − {*i*})
moreover have *eventually* (λ*x*. *x* ≠ *i*) (*nhds x*)
by (*rule t1-space-nhds*) *fact*
ultimately
show *eventually P* (*at x within X*)
unfolding *eventually-at-filter*
by *eventually-elim auto*
qed (*simp add*: *eventually-mono order*.*order-iff-strict eventually-at-filter*)

lemma *at-within-t1-space-avoid-finite*:
(*at x within X* − *I*) = (*at x within X*) **if** *finite I x* ∉ *I* **for** *x*::'*a*::*t1-space*
using *that*

proof (*induction I*)
  case (*insert i I*)
  then show *?case*
    by *auto* (*metis Diff-insert at-within-t1-space-avoid*)
qed *simp*

lemma *at-within-interior*:
  NO-MATCH (*UNIV*::*'a set*) (*S*::*'a*::*topological-space set*) $\Longrightarrow$ *x* $\in$ *interior S* $\Longrightarrow$ *at x within S* = *at x*
  by (*rule at-within-interior*)

3.2 intervals

lemma *Compl-Icc*: $-$ {*a* .. *b*} = {..<*a*} $\cup$ {*b*<..} **for** *a b*::*'a*::*linorder*
  by *auto*

lemma *interior-Icc*[*simp*]: *interior* {*a*..*b*} = {*a*<..<*b*}
  for *a b*::*'a*::{*linorder-topology*, *dense-order*, *no-bot*, *no-top*}
  — TODO: is *no-bot* and *no-top* really required?
  by (*auto simp add*: *Compl-Icc interior-closure*)

lemma *closure-finite*[*simp*]: *closure X* = *X* **if** *finite X* **for** *X*::*'a*::*t1-space set*
  using *that*
  by (*induction X*) (*simp-all add*: *closure-insert*)

definition *piecewise-continuous-on* :: *'a*::*linorder-topology* $\Rightarrow$ *'a* $\Rightarrow$ *'a set* $\Rightarrow$ (*'a* $\Rightarrow$ *'b*::*topological-space*) $\Rightarrow$ *bool*
where *piecewise-continuous-on a b I f* $\longleftrightarrow$
  (*continuous-on* ({*a* .. *b*} $-$ *I*) *f* $\wedge$ *finite I* $\wedge$
    ($\forall$ *i*$\in$*I*. (*i* $\in$ {*a*<..*b*} $\longrightarrow$ ($\exists$ *l*. (*f* $\longrightarrow$ *l*) (*at-left i*))) $\wedge$
      (*i* $\in$ {*a*..<*b*} $\longrightarrow$ ($\exists$ *u*. (*f* $\longrightarrow$ *u*) (*at-right i*)))))

lemma *piecewise-continuous-on-subset*:
  piecewise-continuous-on a b I f $\Longrightarrow$ {*c* .. *d*} $\subseteq$ {*a* .. *b*} $\Longrightarrow$ *piecewise-continuous-on c d I f*
  by (*force simp add*: *piecewise-continuous-on-def intro*: *continuous-on-subset*)

lemma *piecewise-continuous-onE*:
  assumes *piecewise-continuous-on a b I f*
  obtains *l u*
  where *finite I*
    and *continuous-on* ({*a*..*b*} $-$ *I*) *f*
    and ($\bigwedge$*i*. *i* $\in$ *I* $\Longrightarrow$ *a* < *i* $\Longrightarrow$ *i* $\le$ *b* $\Longrightarrow$ (*f* $\longrightarrow$ *l i*) (*at-left i*))
    and ($\bigwedge$*i*. *i* $\in$ *I* $\Longrightarrow$ *a* $\le$ *i* $\Longrightarrow$ *i* < *b* $\Longrightarrow$ (*f* $\longrightarrow$ *u i*) (*at-right i*))
  using *assms*
  by (*auto simp*: *piecewise-continuous-on-def Ball-def*) *metis*

lemma *piecewise-continuous-onI*:
  assumes *finite I continuous-on* ({*a*..*b*} $-$ *I*) *f*

and $(\bigwedge i.\ i \in I \Longrightarrow a < i \Longrightarrow i \leq b \Longrightarrow (f \longrightarrow l\ i)\ (\textit{at-left}\ i))$
and $(\bigwedge i.\ i \in I \Longrightarrow a \leq i \Longrightarrow i < b \Longrightarrow (f \longrightarrow u\ i)\ (\textit{at-right}\ i))$
shows *piecewise-continuous-on a b I f*
using *assms*
by (*force simp*: *piecewise-continuous-on-def*)

lemma *piecewise-continuous-onI′*:
fixes *a b*::′*a*::{*linorder-topology*, *dense-order*, *no-bot*, *no-top*}
assumes *finite I* $\bigwedge x.\ a < x \Longrightarrow x < b \Longrightarrow \textit{isCont}\ f\ x$
and $a \notin I \Longrightarrow \textit{continuous}\ (\textit{at-right}\ a)\ f$
and $b \notin I \Longrightarrow \textit{continuous}\ (\textit{at-left}\ b)\ f$
and $(\bigwedge i.\ i \in I \Longrightarrow a < i \Longrightarrow i \leq b \Longrightarrow (f \longrightarrow l\ i)\ (\textit{at-left}\ i))$
and $(\bigwedge i.\ i \in I \Longrightarrow a \leq i \Longrightarrow i < b \Longrightarrow (f \longrightarrow u\ i)\ (\textit{at-right}\ i))$
shows *piecewise-continuous-on a b I f*
proof (*rule piecewise-continuous-onI*)
have $x \notin I \Longrightarrow a \leq x \Longrightarrow x \leq b \Longrightarrow (f \longrightarrow f\ x)\ (\textit{at}\ x\ \textit{within}\ \{a..b\})$ **for** x
using *assms*(*2*)[*of x*] *assms*(*3*,*4*)
by (*cases* $a = x$; *cases* $b = x$; *cases* $x \in \{a<..<b\}$)
(*auto simp*: *at-within-Icc-at-left at-within-Icc-at-right isCont-def continuous-within filterlim-at-split at-within-interior*)
then show *continuous-on* $(\{a\ ..\ b\} - I)\ f$
by (*auto simp*: *continuous-on-def* ‹*finite I*› *at-within-t1-space-avoid-finite*)
qed *fact*+

lemma *piecewise-continuous-onE′*:
fixes *a b*::′*a*::{*linorder-topology*, *dense-order*, *no-bot*, *no-top*}
assumes *piecewise-continuous-on a b I f*
obtains *l u*
where *finite I*
and $\bigwedge x.\ a < x \Longrightarrow x < b \Longrightarrow x \notin I \Longrightarrow \textit{isCont}\ f\ x$
and $(\bigwedge x.\ a < x \Longrightarrow x \leq b \Longrightarrow (f \longrightarrow l\ x)\ (\textit{at-left}\ x))$
and $(\bigwedge x.\ a \leq x \Longrightarrow x < b \Longrightarrow (f \longrightarrow u\ x)\ (\textit{at-right}\ x))$
and $\bigwedge x.\ a \leq x \Longrightarrow x \leq b \Longrightarrow x \notin I \Longrightarrow f\ x = l\ x$
and $\bigwedge x.\ a \leq x \Longrightarrow x \leq b \Longrightarrow x \notin I \Longrightarrow f\ x = u\ x$
proof −
from *piecewise-continuous-onE*[*OF assms*] **obtain** *l u*
where *finite I*
and *continuous*: *continuous-on* $(\{a..b\} - I)\ f$
and *left*: $(\bigwedge i.\ i \in I \Longrightarrow a < i \Longrightarrow i \leq b \Longrightarrow (f \longrightarrow l\ i)\ (\textit{at-left}\ i))$
and *right*: $(\bigwedge i.\ i \in I \Longrightarrow a \leq i \Longrightarrow i < b \Longrightarrow (f \longrightarrow u\ i)\ (\textit{at-right}\ i))$
by *metis*
define l' **where** $l'\ x = (\textit{if}\ x \in I\ \textit{then}\ l\ x\ \textit{else}\ f\ x)$ **for** x
define u' **where** $u'\ x = (\textit{if}\ x \in I\ \textit{then}\ u\ x\ \textit{else}\ f\ x)$ **for** x
note ‹*finite I*›
moreover from *continuous*
have $a < x \Longrightarrow x < b \Longrightarrow x \notin I \Longrightarrow \textit{isCont}\ f\ x$ **for** x
by (*rule continuous-on-interior*) (*auto simp*: *interior-diff* ‹*finite I*›)
moreover
from *continuous* **have** $a < x \Longrightarrow x \leq b \Longrightarrow x \notin I \Longrightarrow (f \longrightarrow f\ x)\ (\textit{at-left}\ x)$

for x
  by (*cases* $x = b$)
  (*auto simp*: *continuous-on-def at-within-t1-space-avoid-finite* ‹*finite I*›
    at-within-Icc-at-left at-within-interior filterlim-at-split
    dest!: *bspec*[**where** x=x])
then have $a < x \Longrightarrow x \le b \Longrightarrow (f \longrightarrow l'\ x)$ (*at-left* x) **for** x
  by (*auto simp*: l'-*def left*)
moreover
from *continuous* **have** $a \le x \Longrightarrow x < b \Longrightarrow x \notin I \Longrightarrow (f \longrightarrow f\ x)$ (*at-right* x) **for** x
  by (*cases* $x = a$)
  (*auto simp*: *continuous-on-def at-within-t1-space-avoid-finite* ‹*finite I*›
    at-within-Icc-at-right at-within-interior filterlim-at-split
    dest!: *bspec*[**where** x=x])
then have $a \le x \Longrightarrow x < b \Longrightarrow (f \longrightarrow u'\ x)$ (*at-right* x) **for** x
  by (*auto simp*: u'-*def right*)
moreover have $a \le x \Longrightarrow x \le b \Longrightarrow x \notin I \Longrightarrow f\,x = l'\ x$ **for** x **by** (*auto simp*: l'-*def*)
moreover have $a \le x \Longrightarrow x \le b \Longrightarrow x \notin I \Longrightarrow f\,x = u'\ x$ **for** x **by** (*auto simp*: u'-*def*)
ultimately show *?thesis* **..**
qed

lemma *tendsto-avoid-at-within*:
  $(f \longrightarrow l)$ (*at* x *within* X)
  if $(f \longrightarrow l)$ (*at* x *within* $X - \{x\}$)
  using *that*
  by (*auto simp*: *eventually-at-filter dest*!: *topological-tendstoD intro*!: *topological-tendstoI*)

lemma *tendsto-within-subset-eventuallyI*:
  $(f \longrightarrow fx)$ (*at* x *within* X)
  if g: $(g \longrightarrow gy)$ (*at* y *within* Y)
    and *ev*: $\forall_F\ x$ *in* (*at* y *within* Y). $f\,x = g\ x$
    and *xy*: $x = y$
    and *fxgy*: $fx = gy$
    and *XY*: $X - \{x\} \subseteq Y$
  apply (*rule tendsto-avoid-at-within*)
  apply (*rule tendsto-within-subset*[**where** $S = Y$])
  unfolding *xy*
   apply (*subst tendsto-cong*[*OF ev*])
   apply (*rule g*[*folded fxgy*])
  apply (*rule XY*[*unfolded xy*])
  done

lemma *piecewise-continuous-on-insertE*:
  assumes *piecewise-continuous-on* a b (*insert* i I) f
  assumes $i \in \{a\ ..\ b\}$
  obtains g h **where**

```
  piecewise-continuous-on a i I g
  piecewise-continuous-on i b I h
  ⋀x. a ≤ x ⟹ x < i ⟹ g x = f x
  ⋀x. i < x ⟹ x ≤ b ⟹ h x = f x
proof −
 from piecewise-continuous-onE[OF assms(1)] ‹i ∈ {a .. b}› obtain l u where
     finite: finite I
  and cf: continuous-on ({a..b} − insert i I) f
  and l: (⋀i. i ∈ I ⟹ a < i ⟹ i ≤ b ⟹ (f ⟶ l i) (at-left i)) i > a ⟹
(f ⟶ l i) (at-left i)
  and u: (⋀i. i ∈ I ⟹ a ≤ i ⟹ i < b ⟹ (f ⟶ u i) (at-right i)) i < b
⟹ (f ⟶ u i) (at-right i)
  by auto (metis (mono-tags))

 have fl: (f(i := x) ⟶ l j) (at-left j) if j ∈ I a < j j ≤ b for j x
  using l(1)
  by (rule tendsto-within-subset-eventuallyI)
   (auto simp: eventually-at-filter frequently-def t1-space-nhds that)
 have fr: (f(i := x) ⟶ u j) (at-right j) if j ∈ I a ≤ j j < b for j x
  using u(1)
  by (rule tendsto-within-subset-eventuallyI)
   (auto simp: eventually-at-filter frequently-def t1-space-nhds that)
 from cf have tendsto: (f ⟶ f x) (at x within {a..b} − insert i I)
  if x ∈ {a .. b} − insert i I for x using that
  by (auto simp: continuous-on-def)
 have continuous-on ({a..i} − I) (f(i:=l i))
  apply (cases a = i)
  subgoal by (auto simp: continuous-on-def Diff-triv)
  unfolding continuous-on-def
  apply safe
  subgoal for x
   apply (cases x = i)
   subgoal
    apply (rule tendsto-within-subset-eventuallyI)
      apply (rule l(2))
    by (auto simp: eventually-at-filter)
   subgoal
    apply (subst at-within-t1-space-avoid[symmetric], assumption)
    apply (rule tendsto-within-subset-eventuallyI[where y=x])
      apply (rule tendsto)
    using ‹i ∈ {a .. b}› by (auto simp: eventually-at-filter)
   done
  done
 then have piecewise-continuous-on a i I (f(i:=l i))
  using ‹i ∈ {a .. b}›
  by (auto intro!: piecewise-continuous-onI finite fl fr)

 moreover
 have continuous-on ({i..b} − I) (f(i:=u i))
```

```
    apply (cases b = i)
    subgoal by (auto simp: continuous-on-def Diff-triv)
    unfolding continuous-on-def
    apply safe
    subgoal for x
      apply (cases x = i)
      subgoal
        apply (rule tendsto-within-subset-eventuallyI)
            apply (rule u(2))
        by (auto simp: eventually-at-filter)
      subgoal
        apply (subst at-within-t1-space-avoid[symmetric], assumption)
        apply (rule tendsto-within-subset-eventuallyI[where y=x])
            apply (rule tendsto)
        using ‹i ∈ {a .. b}› by (auto simp: eventually-at-filter)
      done
    done
  then have piecewise-continuous-on i b I (f(i:=u i))
    using ‹i ∈ {a .. b}›
    by (auto intro!: piecewise-continuous-onI finite fl fr)
  moreover have (f(i:=l i)) x = f x if a ≤ x x < i for x
    using that by auto
  moreover have (f(i:=u i)) x = f x if i < x x ≤ b for x
    using that by auto
  ultimately show ?thesis ..
qed

lemma eventually-avoid-finite:
  ∀F x in at y within Y. x ∉ I if finite I for y::'a::t1-space
  using that
proof (induction)
  case empty
  then show ?case by simp
next
  case (insert x F)
  then show ?case
    apply (auto intro!: eventually-conj)
    apply (cases y = x)
    subgoal by (simp add: eventually-at-filter)
    subgoal by (rule tendsto-imp-eventually-ne) (rule tendsto-ident-at)
    done
qed

lemma eventually-at-left-linorder:— TODO: generalize ?b < ?a ⟹ ∀F x in at-left
?a. x ∈ {?b<..<?a}
  a > (b :: 'a :: linorder-topology) ⟹ eventually (λx. x ∈ {b<..<a}) (at-left a)
  unfolding eventually-at-left
  by auto
```

lemma *eventually-at-right-linorder*:— TODO: generalize $?a < ?b \Longrightarrow \forall_F\ x$ *in at-right* $?a.\ x \in \{?a<..<?b\}$
  $a > (b :: 'a :: linorder\text{-}topology) \Longrightarrow$ *eventually* $(\lambda x.\ x \in \{b<..<a\})$ (*at-right* b)
  unfolding *eventually-at-right*
  by *auto*

lemma *piecewise-continuous-on-congI*:
  piecewise-continuous-on $a\ b\ I\ g$
  if *piecewise-continuous-on* $a\ b\ I\ f$
    and *eq*: $\bigwedge x.\ x \in \{a\ ..\ b\} - I \Longrightarrow g\ x = f\ x$
proof −
  from *piecewise-continuous-onE*[*OF that*(*1*)]
  obtain *l u* **where** *finite*: *finite I*
    and ∗:
    continuous-on $(\{a..b\} - I)\ f$
    $(\bigwedge i.\ i \in I \Longrightarrow a < i \Longrightarrow i \le b \Longrightarrow (f \longrightarrow l\ i)$ (*at-left* i))
    $\bigwedge i.\ i \in I \Longrightarrow a \le i \Longrightarrow i < b \Longrightarrow (f \longrightarrow u\ i)$ (*at-right* i)
    by *blast*
  note *finite*
  moreover
  from ∗ **have** *continuous-on* $(\{a..b\} - I)\ g$
    using *that*(*2*)
    by (*auto simp*: *eq cong*: *continuous-on-cong*) (*subst continuous-on-cong*[*OF refl eq*]; *assumption*)
  moreover
  have $\forall_F\ x$ *in at-left* $i.\ f\ x = g\ x$ **if** $a < i\ i \le b$ **for** i
    using *eventually-avoid-finite*[*OF* ‹*finite I*›, *of i* {..<*i*}]
      eventually-at-left-linorder[*OF* ‹$a < i$›]
    by *eventually-elim* (*subst eq*, *use that* **in** *auto*)
  then have $i \in I \Longrightarrow a < i \Longrightarrow i \le b \Longrightarrow (g \longrightarrow l\ i)$ (*at-left* i) **for** i
    using ∗(*2*)
    by (*rule Lim-transform-eventually*[*rotated*]) *auto*
  moreover
  have $\forall_F\ x$ *in at-right* $i.\ f\ x = g\ x$ **if** $a \le i\ i < b$ **for** i
    using *eventually-avoid-finite*[*OF* ‹*finite I*›, *of i* {*i*<..}]
      eventually-at-right-linorder[*OF* ‹$i < b$›]
    by *eventually-elim* (*subst eq*, *use that* **in** *auto*)
  then have $i \in I \Longrightarrow a \le i \Longrightarrow i < b \Longrightarrow (g \longrightarrow u\ i)$ (*at-right* i) **for** i
    using ∗(*3*)
    by (*rule Lim-transform-eventually*[*rotated*]) *auto*
  ultimately
  show *?thesis*
    by (*rule piecewise-continuous-onI*) *auto*
qed

lemma *piecewise-continuous-on-cong*[*cong*]:
  piecewise-continuous-on $a\ b\ I\ f \longleftrightarrow$ *piecewise-continuous-on* $c\ d\ J\ g$
  if $a = c$
    $b = d$

$I = J$
$\bigwedge x.\ c \le x \Longrightarrow x \le d \Longrightarrow x \notin J \Longrightarrow f\,x = g\,x$
using *that*
by (*auto intro*: *piecewise-continuous-on-congI*)

lemma *tendsto-at-left-continuous-on-avoidI*: $(f \longrightarrow g\ i)$ (*at-left i*)
if *g*: *continuous-on* $(\{a..i\} - I)$ *g*
and *gf*: $\bigwedge x.\ a < x \Longrightarrow x < i \Longrightarrow g\ x = f\ x$
$i \notin I$ *finite I* $a < i$
for *i*::′*a*::*linorder-topology*
proof (*rule Lim-transform-eventually*)
from *that* **have** $i \in \{a\ ..\ i\}$ **by** *auto*
from *g* **have** $(g \longrightarrow g\ i)$ (*at i within* $\{a..i\} - I$)
using ‹$i \notin I$› ‹$i \in \{a\ ..\ i\}$›
by (*auto elim*!: *piecewise-continuous-onE simp*: *continuous-on-def*)
then show $(g \longrightarrow g\ i)$ (*at-left i*)
by (*metis that at-within-Icc-at-left at-within-t1-space-avoid-finite*
greaterThanLessThan-iff)
show $\forall_F\ x$ *in at-left i*. $g\ x = f\ x$
using *eventually-at-left-linorder*[*OF* ‹$a < i$›]
by *eventually-elim* (*auto simp*: ‹$a < i$› *gf*)
qed

lemma *tendsto-at-right-continuous-on-avoidI*: $(f \longrightarrow g\ i)$ (*at-right i*)
if *g*: *continuous-on* $(\{i..b\} - I)$ *g*
and *gf*: $\bigwedge x.\ i < x \Longrightarrow x < b \Longrightarrow g\ x = f\ x$
$i \notin I$ *finite I* $i < b$
for *i*::′*a*::*linorder-topology*
proof (*rule Lim-transform-eventually*)
from *that* **have** $i \in \{i\ ..\ b\}$ **by** *auto*
from *g* **have** $(g \longrightarrow g\ i)$ (*at i within* $\{i..b\} - I$)
using ‹$i \notin I$› ‹$i \in \{i\ ..\ b\}$›
by (*auto elim*!: *piecewise-continuous-onE simp*: *continuous-on-def*)
then show $(g \longrightarrow g\ i)$ (*at-right i*)
by (*metis that at-within-Icc-at-right at-within-t1-space-avoid-finite*
greaterThanLessThan-iff)
show $\forall_F\ x$ *in at-right i*. $g\ x = f\ x$
using *eventually-at-right-linorder*[*OF* ‹$i < b$›]
by *eventually-elim* (*auto simp*: ‹$i < b$› *gf*)
qed

lemma *piecewise-continuous-on-insert-leftI*:
piecewise-continuous-on a b (*insert a I*) *f* **if** *piecewise-continuous-on a b I f*
apply (*cases* $a \in I$)
subgoal using *that* **by** (*auto dest*: *insert-absorb*)
subgoal
using *that*
apply (*rule piecewise-continuous-onE*)
subgoal for *l u*

apply (*rule piecewise-continuous-onI*[**where** *u*=*u*(*a*:=*f a*)])
apply (*auto intro*: *continuous-on-subset*)
apply (*rule tendsto-at-right-continuous-on-avoidI*, *assumption*)
apply *auto*
done
done
done

lemma *piecewise-continuous-on-insert-rightI*:
piecewise-continuous-on a b (*insert b I*) *f* **if** *piecewise-continuous-on a b I f*
apply (*cases* $b \in I$)
subgoal using *that* **by** (*auto dest*: *insert-absorb*)
subgoal
using *that*
apply (*rule piecewise-continuous-onE*)
subgoal for *l u*
apply (*rule piecewise-continuous-onI*[**where** *l*=*l*(*b*:=*f b*)])
apply (*auto intro*: *continuous-on-subset*)
apply (*rule tendsto-at-left-continuous-on-avoidI*, *assumption*)
apply *auto*
done
done
done

theorem *piecewise-continuous-on-induct*[*consumes 1*, *case-names empty combine weaken*]:
assumes *pc*: *piecewise-continuous-on a b I f*
assumes *1*: $\bigwedge a\ b\ f.$ *continuous-on* $\{a\ ..\ b\}$ *f* $\Longrightarrow$ *P a b* $\{\}$ *f*
assumes *2*: $\bigwedge a\ i\ b\ I\ f1\ f2\ f.\ a \le i \Longrightarrow i \le b \Longrightarrow i \notin I \Longrightarrow P\ a\ i\ I\ f1 \Longrightarrow P\ i\ b\ I\ f2 \Longrightarrow$
piecewise-continuous-on a i I f1 $\Longrightarrow$
piecewise-continuous-on i b I f2 $\Longrightarrow$
$(\bigwedge x.\ a \le x \Longrightarrow x < i \Longrightarrow f1\ x = f\ x) \Longrightarrow$
$(\bigwedge x.\ i < x \Longrightarrow x \le b \Longrightarrow f2\ x = f\ x) \Longrightarrow$
$(i > a \Longrightarrow (f \longrightarrow f1\ i)$ (*at-left i*)) $\Longrightarrow$
$(i < b \Longrightarrow (f \longrightarrow f2\ i)$ (*at-right i*)) $\Longrightarrow$
P a b (*insert i I*) *f*
assumes *3*: $\bigwedge a\ b\ i\ I\ f.\ P\ a\ b\ I\ f \Longrightarrow$ *finite I* $\Longrightarrow i \notin I \Longrightarrow$ *P a b* (*insert i I*) *f*
shows *P a b I f*
proof −
from *pc* **have** *finite I*
by (*auto simp*: *piecewise-continuous-on-def*)
then show *?thesis*
using *pc*
proof (*induction I arbitrary*: *a b f*)
case *empty*
then show *?case*
by (*auto simp*: *piecewise-continuous-on-def 1*)
next

```
    case (insert i I)
    show ?case
    proof (cases i ∈ {a .. b})
      case True
      from insert.prems[THEN piecewise-continuous-on-insertE, OF ‹i ∈ {a .. b}›]
      obtain g h
        where g: piecewise-continuous-on a i I g
          and h: piecewise-continuous-on i b I h
          and gf: ⋀x. a ≤ x ⟹ x < i ⟹ g x = f x
          and hf: ⋀x. i < x ⟹ x ≤ b ⟹ h x = f x
        by metis
      from g have pcg: piecewise-continuous-on a i I (f(i:=g i))
        by (rule piecewise-continuous-on-congI) (auto simp: gf)
      from h have pch: piecewise-continuous-on i b I (f(i:=h i))
        by (rule piecewise-continuous-on-congI) (auto simp: hf)

      have fg: (f ⟶ g i) (at-left i) if a < i
        apply (rule tendsto-at-left-continuous-on-avoidI[where a=a and I=I])
        using g ‹i ∉ I› ‹a < i›
        by (auto elim!: piecewise-continuous-onE simp: gf)
      have fh: (f ⟶ h i) (at-right i) if i < b
        apply (rule tendsto-at-right-continuous-on-avoidI[where b=b and I=I])
        using h ‹i ∉ I› ‹i < b›
        by (auto elim!: piecewise-continuous-onE simp: hf)
      show ?thesis
        apply (rule 2)
        using True apply force
        using True apply force
                apply (rule insert)
               apply (rule insert.IH, rule pcg)
              apply (rule insert.IH, rule pch)
             apply fact
            apply fact
        using 3
        by (auto simp: fg fh)
    next
      case False
      with insert.prems
      have piecewise-continuous-on a b I f
        by (auto simp: piecewise-continuous-on-def)
      from insert.IH[OF this] show ?thesis
        by (rule 3) fact+
    qed
  qed
qed

lemma continuous-on-imp-piecewise-continuous-on:
  continuous-on {a .. b} f ⟹ piecewise-continuous-on a b {} f
  by (auto simp: piecewise-continuous-on-def)
```

```
lemma piecewise-continuous-on-imp-absolutely-integrable:
  fixes a b::real and f::real ⇒ 'a::euclidean-space
  assumes piecewise-continuous-on a b I f
  shows f absolutely-integrable-on {a..b}
  using assms
proof (induction rule: piecewise-continuous-on-induct)
  case (empty a b f)
  show ?case
    by (auto intro!: absolutely-integrable-onI integrable-continuous-interval
          continuous-intros empty)
next
  case (combine a i b I f1 f2 f)
  from combine(10)
  have f absolutely-integrable-on {a..i}
    by (rule absolutely-integrable-spike[where S={i}]) (auto simp: combine)
  moreover
  from combine(11)
  have f absolutely-integrable-on {i..b}
    by (rule absolutely-integrable-spike[where S={i}]) (auto simp: combine)
  ultimately
  show ?case
    by (rule absolutely-integrable-on-combine) fact+
qed

lemma piecewise-continuous-on-integrable:
  fixes a b::real and f::real ⇒ 'a::euclidean-space
  assumes piecewise-continuous-on a b I f
  shows f integrable-on {a..b}
  using piecewise-continuous-on-imp-absolutely-integrable[OF assms]
  unfolding absolutely-integrable-on-def by auto

lemma piecewise-continuous-on-comp:
  assumes p: piecewise-continuous-on a b I f
  assumes c: ⋀x. isCont (λ(x, y). g x y) x
  shows piecewise-continuous-on a b I (λx. g x (f x))
proof −
  from piecewise-continuous-onE[OF p]
  obtain l u
    where I: finite I
      and cf: continuous-on ({a..b} − I) f
      and l: (⋀i. i ∈ I ⟹ a < i ⟹ i ≤ b ⟹ (f ⟶ l i) (at-left i))
      and u: (⋀i. i ∈ I ⟹ a ≤ i ⟹ i < b ⟹ (f ⟶ u i) (at-right i))
    by metis
  note ‹finite I›
  moreover
  from c have cg: continuous-on UNIV (λ(x, y). g x y)
   using c by (auto simp: continuous-on-def isCont-def intro: tendsto-within-subset)
  then have continuous-on ({a..b} − I) (λx. g x (f x))
```

by (*intro continuous-on-compose2*[*OF cg*, **where** f=λx. (x, f x), *simplified*])
(*auto intro*!: *continuous-intros cf*)
moreover
note *tendstcomp* = *tendsto-compose*[*OF c*[*unfolded isCont-def*], **where** f=λx. (x, f x), *simplified*, *THEN tendsto-eq-rhs*]
have ((λx. g x (f x)) $\longrightarrow$ g i (u i)) (*at-right* i) **if** $i \in I$ $a \le i$ $i < b$ **for** i
by (*rule tendstcomp*) (*auto intro*!: *tendsto-eq-intros* u[*OF* ‹$i \in I$›] *that*)
moreover
have ((λx. g x (f x)) $\longrightarrow$ g i (l i)) (*at-left* i) **if** $i \in I$ $a < i$ $i \le b$ **for** i
by (*rule tendstcomp*) (*auto intro*!: *tendsto-eq-intros* l[*OF* ‹$i \in I$›] *that*)
ultimately show *?thesis*
by (*intro piecewise-continuous-onI*)
qed

lemma *bounded-piecewise-continuous-image*:
bounded (f ‘ {a .. b})
if *piecewise-continuous-on* a b I f **for** a b::*real*
using *that*
proof (*induction rule*: *piecewise-continuous-on-induct*)
case (*empty* a b f)
then show *?case* **by** (*auto intro*!: *compact-imp-bounded compact-continuous-image*)
next
case (*combine* a i b I *f1 f2* f)
have (f ‘ {a..b}) $\subseteq$ (*insert* (f i) (*f1* ‘ {a..i} $\cup$ *f2* ‘ {i..b}))
using *combine*
by (*auto simp*: *image-iff*) (*metis antisym-conv atLeastAtMost-iff le-cases not-less*)
also have *bounded* ...
using *combine* **by** *auto*
finally (*bounded-subset*[*rotated*]) **show** *?case* **.**
qed

lemma *tendsto-within-eventually*:
(f $\longrightarrow$ l) (*at* x *within* X)
if
(f $\longrightarrow$ l) (*at* x *within* Y)
$\forall_F$ y *in at* x *within* X. $y \in Y$
using - *that*(*1*)
proof (*rule tendsto-mono*)
show *at* x *within* $X \le$ *at* x *within* Y
proof (*rule filter-leI*)
fix P
assume *eventually* P (*at* x *within* Y)
with *that*(*2*) **show** *eventually* P (*at* x *within* X)
unfolding *eventually-at-filter*
by *eventually-elim auto*
qed
qed

lemma *at-within-eq-bot-lemma*:

```
  at x within {b..c} = (if x < b ∨ b > c then bot else at x within {b..c})
  for x b c::'a::linorder-topology
  by (auto intro!: not-in-closure-trivial-limitI)

lemma at-within-eq-bot-lemma2:
  at x within {a..b} = (if x > b ∨ a > b then bot else at x within {a..b})
  for x a b::'a::linorder-topology
  by (auto intro!: not-in-closure-trivial-limitI)

lemma piecewise-continuous-on-combine:
  piecewise-continuous-on a c J f
  if piecewise-continuous-on a b J f piecewise-continuous-on b c J f
  using that
  apply (auto elim!: piecewise-continuous-onE)
  subgoal for l u l' u'
    apply (rule piecewise-continuous-onI[where
          l=λi. if i ≤ b then l i else l' i and
          u=λi. if i < b then u i else u' i])
    subgoal by force
    subgoal
      apply (rule continuous-on-subset[where s=({a .. b} ∪ {b .. c} − J)])
      apply (auto simp: continuous-on-def at-within-t1-space-avoid-finite)
       apply (rule Lim-Un)
      subgoal by auto
      subgoal by (subst at-within-eq-bot-lemma) auto
      apply (rule Lim-Un)
      subgoal by (subst at-within-eq-bot-lemma2) auto
      subgoal by auto
      done
    by auto
  done

lemma piecewise-continuous-on-finite-superset:
 piecewise-continuous-on a b I f ⟹ I ⊆ J ⟹ finite J ⟹ piecewise-continuous-on
a b J f
  for a b::'a::{linorder-topology, dense-order, no-bot, no-top}
  apply (auto simp add: piecewise-continuous-on-def)
    apply (rule continuous-on-subset, assumption, force)
  subgoal for i
    apply (cases i ∈ I)
     apply (auto simp: continuous-on-def at-within-t1-space-avoid-finite)
    apply (drule bspec[where x=i])
     apply (auto simp: at-within-t1-space-avoid)
    apply (cases i = b)
     apply (auto simp: at-within-Icc-at-left )
    apply (subst (asm) at-within-interior[where x=i])
    by (auto simp: filterlim-at-split)
  subgoal for i
    apply (cases i ∈ I)
```

apply (*auto simp*: *continuous-on-def at-within-t1-space-avoid-finite*)
apply (*drule bspec*[**where** $x=i$])
apply (*auto simp*: *at-within-t1-space-avoid*)
apply (*cases* $i = a$)
apply (*auto simp*: *at-within-Icc-at-right*)
apply (*subst* (*asm*) *at-within-interior*[**where** $x=i$])
subgoal by (*simp add*: *interior-Icc*)
by (*auto simp*: *filterlim-at-split*)
done

lemma *piecewise-continuous-on-splitI*:
piecewise-continuous-on a c K f
if
piecewise-continuous-on a b I f
piecewise-continuous-on b c J f
$I \subseteq K$ $J \subseteq K$ *finite K*
for *a b*::′*a*::{*linorder-topology*, *dense-order*, *no-bot*, *no-top*}
apply (*rule piecewise-continuous-on-combine*[**where** $b=b$])
subgoal
by (*rule piecewise-continuous-on-finite-superset*, *fact*)
(*use that* **in** ‹*auto elim*!: *piecewise-continuous-onE*›)
subgoal
by (*rule piecewise-continuous-on-finite-superset*, *fact*)
(*use that* **in** ‹*auto elim*!: *piecewise-continuous-onE*›)
done

end

4 Existence

theory *Existence* **imports**
Piecewise-Continuous
begin

4.1 Definition

definition *has-laplace* :: (*real* ⇒ *complex*) ⇒ *complex* ⇒ *complex* ⇒ *bool*
(**infixr** ‹*has*′*-laplace*› *46*)
where (*f has-laplace L*) *s* ⟷ (($\lambda t.$ *exp* ($t *_R - s$) $*$ *f t*) *has-integral L*) {*0*..}

lemma *has-laplaceI*:
assumes (($\lambda t.$ *exp* ($t *_R - s$) $*$ *f t*) *has-integral L*) {*0*..}
shows (*f has-laplace L*) *s*
using *assms*
by (*auto simp*: *has-laplace-def*)

lemma *has-laplaceD*:
assumes (*f has-laplace L*) *s*
shows (($\lambda t.$ *exp* ($t *_R - s$) $*$ *f t*) *has-integral L*) {*0*..}

using *assms*
by (*auto simp*: *has-laplace-def*)

lemma *has-laplace-unique*:
$L = M$ **if**
(*f has-laplace L*) *s*
(*f has-laplace M*) *s*
using *that*
by (*auto simp*: *has-laplace-def has-integral-unique*)

4.2 Condition for Existence: Exponential Order

definition *exponential-order M c f* $\longleftrightarrow$ *0* $<$ *M* $\wedge$ ($\forall_F$ *t in at-top. norm* (*f t*) $\leq$ *M* $*$ *exp* (*c* $*$ *t*))

lemma *exponential-orderI*:
assumes *0* $<$ *M* **and** *eo*: $\forall_F$ *t in at-top. norm* (*f t*) $\leq$ *M* $*$ *exp* (*c* $*$ *t*)
shows *exponential-order M c f*
by (*auto intro*!: *assms simp*: *exponential-order-def*)

lemma *exponential-orderD*:
assumes *exponential-order M c f*
shows *0* $<$ *M* $\forall_F$ *t in at-top. norm* (*f t*) $\leq$ *M* $*$ *exp* (*c* $*$ *t*)
using *assms* **by** (*auto simp*: *exponential-order-def*)

context
fixes *f*::*real* $\Rightarrow$ *complex*
begin

definition *laplace-integrand*::*complex* $\Rightarrow$ *real* $\Rightarrow$ *complex*
where *laplace-integrand s t* = *exp* (*t* $*_R$ $-$ *s*) $*$ *f t*

lemma *laplace-integrand-absolutely-integrable-on-Icc*:
laplace-integrand s absolutely-integrable-on {*a*..*b*}
if *AE x*∈{*a*..*b*} *in lebesgue. cmod* (*f x*) $\leq$ *B f integrable-on* {*a*..*b*}
apply (*cases b* $\leq$ *a*)
subgoal by (*auto intro*!: *absolutely-integrable-onI integrable-negligible*[*OF negligible-real-ivlI*])
proof *goal-cases*
case *1*
have *compact* ((λ*x. exp* ($-$ (*x* $*_R$ *s*))) ‘ {*a* .. *b*})
by (*rule compact-continuous-image*) (*auto intro*!: *continuous-intros*)
then obtain *C* **where** *C*: *0* $\leq$ *C a* $\leq$ *x* $\Longrightarrow$ *x* $\leq$ *b* $\Longrightarrow$ *cmod* (*exp* ($-$ (*x* $*_R$ *s*))) $\leq$ *C* **for** *x*
using *1*
apply (*auto simp*: *bounded-iff dest*!: *compact-imp-bounded*)
by (*metis atLeastAtMost-iff exp-ge-zero order-refl order-trans scaleR-complex.sel*(*1*))

have *m*: (λ*x. indicator* {*a*..*b*} *x* $*_R$ *f x*) ∈ *borel-measurable lebesgue*

```
    apply (rule has-integral-implies-lebesgue-measurable)
    apply (rule integrable-integral)
    apply (rule that)
    done
  have complex-set-integrable lebesgue {a..b} (λx. exp (− (x *R s)) * (indicator {a
.. b} x *R f x))
    unfolding set-integrable-def
    apply (rule integrableI-bounded-set-indicator[where B=C * B])
       apply (simp; fail)
      apply (rule borel-measurable-times)
       apply measurable
        apply (simp add: measurable-completion)
       apply (simp add: measurable-completion)
      apply (rule m)
    apply (simp add: emeasure-lborel-Icc-eq)
    using that(1)
    apply eventually-elim
    apply (auto simp: norm-mult)
    apply (rule mult-mono)
    using C
    by auto
  then show ?case
    unfolding set-integrable-def
    by (simp add: laplace-integrand-def[abs-def] indicator-inter-arith[symmetric])
qed

lemma laplace-integrand-integrable-on-Icc:
  laplace-integrand s integrable-on {a..b}
  if AE x∈{a..b} in lebesgue. cmod (f x) ≤ B f integrable-on {a..b}
  using laplace-integrand-absolutely-integrable-on-Icc[OF that]
  using set-lebesgue-integral-eq-integral(1) by blast

lemma eventually-laplace-integrand-le:
  ∀F t in at-top. cmod (laplace-integrand s t) ≤ M * exp (− (Re s − c) * t)
  if exponential-order M c f
  using exponential-orderD(2)[OF that]
proof (eventually-elim)
  case (elim t)
  show ?case
    unfolding laplace-integrand-def
    apply (rule norm-mult-ineq[THEN order-trans])
    apply (auto intro!: mult-left-mono[THEN order-trans, OF elim])
    apply (auto simp: exp-minus divide-simps algebra-simps exp-add[symmetric])
    done
qed

lemma
  assumes eo: exponential-order M c f
    and cs: c < Re s
```

shows *laplace-integrand-integrable-on-Ici-iff*:
laplace-integrand s integrable-on {*a*..} ⟷
(∀ *k*>*a*. *laplace-integrand s integrable-on* {*a*..*k*})
(**is** *?th1*)
and *laplace-integrand-absolutely-integrable-on-Ici-iff*:
laplace-integrand s absolutely-integrable-on {*a*..} ⟷
(∀ *k*>*a*. *laplace-integrand s absolutely-integrable-on* {*a*..*k*})
(**is** *?th2*)
proof −
have ∀$_F$ *t in at-top*. *a* < (*t*::*real*)
using *eventually-gt-at-top* **by** *blast*
then have ∀$_F$ *t in at-top*. *t* > *a* ∧ *cmod* (*laplace-integrand s t*) ≤ *M* ∗ *exp* (− (*Re s* − *c*) ∗ *t*)
using *eventually-laplace-integrand-le*[*OF eo*]
by *eventually-elim* (*auto*)
then obtain *A* **where** *A*: *A* > *a* **and** *le*: *t* ≥ *A* ⟹ *cmod* (*laplace-integrand s t*) ≤ *M* ∗ *exp* (− (*Re s* − *c*) ∗ *t*) **for** *t*
unfolding *eventually-at-top-linorder*
by *blast*

let *?f* = λ(*k*::*real*) (*t*::*real*). *indicat-real* {*A*..*k*} *t* ∗$_R$ *laplace-integrand s t*

from *exponential-orderD*[*OF eo*] **have** *M* ≠ *0* **by** *simp*
have *2*: (λ*t*. *M* ∗ *exp* (− (*Re s* − *c*) ∗ *t*)) *integrable-on* {*A*..}
unfolding *integrable-on-cmult-iff*[*OF* ‹*M* ≠ *0*›] *norm-exp-eq-Re*
by (*rule integrable-on-exp-minus-to-infinity*) (*simp add*: *cs*)

have *3*: *t*∈{*A*..} ⟹ *cmod* (*?f k t*) ≤ *M* ∗ *exp* (− (*Re s* − *c*) ∗ *t*)
(**is** *t*∈*-*⟹ *?lhs t* ≤ *?rhs t*)
for *t k*
proof *safe*
fix *t* **assume** *A* ≤ *t*
have *?lhs t* ≤ *cmod* (*laplace-integrand s t*)
by (*auto simp*: *indicator-def*)
also have ... ≤ *?rhs t* **using** ‹*A* ≤ *t*› *le* **by** (*simp add*: *laplace-integrand-def*)
finally show *?lhs t* ≤ *?rhs t* **.**
qed

have *4*: ∀ *t*∈{*A*..}. ((λ*k*. *?f k t*) ⟶ *laplace-integrand s t*) *at-top*
proof *safe*
fix *t* **assume** *t*: *t* ≥ *A*
have ∀$_F$ *k in at-top*. *k* ≥ *t*
by (*simp add*: *eventually-ge-at-top*)
then have ∀$_F$ *k in at-top*. *laplace-integrand s t* = *?f k t*
by *eventually-elim* (*use t* **in** ‹*auto simp*: *indicator-def*›)
then show ((λ*k*. *?f k t*) ⟶ *laplace-integrand s t*) *at-top* **using** *tendsto-const*
by (*rule Lim-transform-eventually*[*rotated*])
qed

```
show th1: ?th1
proof safe
  assume ∀k>a. laplace-integrand s integrable-on {a..k}
  note li = this[rule-format]
  have liA: laplace-integrand s integrable-on {A..k} for k
  proof cases
    assume k ≤ A
    then have {A..k} = (if A = k then {k} else {}) by auto
    then show ?thesis by (auto intro!: integrable-negligible)
  next
    assume n: ¬ k ≤ A
    show ?thesis
      by (rule integrable-on-subinterval[OF li[of k]]) (use A n in auto)
  qed
  have ?f k integrable-on {A..k} for k
    using liA[of k] negligible-empty
    by (rule integrable-spike) auto
  then have 1: ?f k integrable-on {A..} for k
    by (rule integrable-on-superset) auto
  note 1 2 3 4
  note * = this[unfolded set-integrable-def]
  from li[of A] dominated-convergence-at-top(1)[OF *]
  show laplace-integrand s integrable-on {a..}
    by (rule integrable-Un') (use ‹a < A› in ‹auto simp: max-def li›)
qed (rule integrable-on-subinterval, assumption, auto)

show ?th2
proof safe
  assume ai: ∀k>a. laplace-integrand s absolutely-integrable-on {a..k}
  then have laplace-integrand s absolutely-integrable-on {a..A}
    using A by auto
  moreover
  from ai have ∀k>a. laplace-integrand s integrable-on {a..k}
    using set-lebesgue-integral-eq-integral(1) by blast
  with th1 have i: laplace-integrand s integrable-on {a..} by auto
  have 1: ?f k integrable-on {A..} for k
    apply (rule integrable-on-superset[where S={A..k}])
    using - negligible-empty
      apply (rule integrable-spike[where f=laplace-integrand s])
      apply (rule integrable-on-subinterval)
       apply (rule i)
    by (use ‹a < A› in auto)
  have laplace-integrand s absolutely-integrable-on {A..}
    using - dominated-convergence-at-top(1)[OF 1 2 3 4] 2
    by (rule absolutely-integrable-integrable-bound) (use le in auto)
  ultimately
  have laplace-integrand s absolutely-integrable-on ({a..A} ∪ {A..})
    by (rule set-integrable-Un) auto
  also have {a..A} ∪ {A..} = {a..} using ‹a < A› by auto
```

finally show *local.laplace-integrand s absolutely-integrable-on* {*a*..} **.**
qed (*rule set-integrable-subset*, *assumption*, *auto*)
qed

theorem *laplace-exists-laplace-integrandI*:
assumes *laplace-integrand s integrable-on* {*0*..}
obtains *F* **where** (*f has-laplace F*) *s*
proof −
from *assms*
have (*f has-laplace integral* {*0*..} (*laplace-integrand s*)) *s*
unfolding *has-laplace-def laplace-integrand-def* **by** *blast*
thus *?thesis* **..**
qed

lemma
assumes *eo*: *exponential-order M c f*
and *pc*: ⋀*k*. *AE x*∈{*0*..*k*} *in lebesgue*. *cmod* (*f x*) ≤ *B k* ⋀*k*. *f integrable-on* {*0*..*k*}
and *s*: *Re s* > *c*
shows *laplace-integrand-integrable*: *laplace-integrand s integrable-on* {*0*..} (**is** *?th1*)
and *laplace-integrand-absolutely-integrable*:
laplace-integrand s absolutely-integrable-on {*0*..} (**is** *?th2*)
using *eo laplace-integrand-absolutely-integrable-on-Icc*[*OF pc*] *s*
by (*auto simp*: *laplace-integrand-integrable-on-Ici-iff*
laplace-integrand-absolutely-integrable-on-Ici-iff
set-lebesgue-integral-eq-integral)

lemma *piecewise-continuous-on-AE-boundedE*:
assumes *pc*: ⋀*k*. *piecewise-continuous-on a k* (*I k*) *f*
obtains *B* **where** ⋀*k*. *AE x*∈{*a*..*k*} *in lebesgue*. *cmod* (*f x*) ≤ *B k*
apply *atomize-elim*
apply (*rule choice*)
apply (*rule allI*)
subgoal for *k*
using *bounded-piecewise-continuous-image*[*OF pc*[*of k*]]
by (*force simp*: *bounded-iff*)
done

theorem *piecewise-continuous-on-has-laplace*:
assumes *eo*: *exponential-order M c f*
and *pc*: ⋀*k*. *piecewise-continuous-on 0 k* (*I k*) *f*
and *s*: *Re s* > *c*
obtains *F* **where** (*f has-laplace F*) *s*
proof −
from *piecewise-continuous-on-AE-boundedE*[*OF pc*]
obtain *B* **where** *AE*: *AE x*∈{*0*..*k*} *in lebesgue*. *cmod* (*f x*) ≤ *B k* **for** *k* **by** *force*
have *int*: *f integrable-on* {*0*..*k*} **for** *k*
using *pc*

by (*rule piecewise-continuous-on-integrable*)
show *?thesis*
using *pc*
apply (*rule piecewise-continuous-on-AE-boundedE*)
apply (*rule laplace-exists-laplace-integrandI*)
apply (*rule laplace-integrand-integrable*)
apply (*rule eo*)
apply *assumption*
apply (*rule int*)
apply (*rule s*)
by (*rule that*)
qed

end

4.3 Concrete Laplace Transforms

lemma *exp-scaleR-has-vector-derivative-left'*[*derivative-intros*]:
$((\lambda t.\ exp\ (t *_R A))\ has\text{-}vector\text{-}derivative\ A * exp\ (t *_R A))\ (at\ t\ within\ S)$
by (*metis exp-scaleR-has-vector-derivative-right exp-times-scaleR-commute*)

lemma
fixes *a*::*complex*— TODO: generalize
assumes *a*: $0 < Re\ a$
shows *integrable-on-cexp-minus-to-infinity*: $(\lambda x.\ exp\ (x *_R - a))$ *integrable-on* $\{c..\}$
and *integral-cexp-minus-to-infinity*: *integral* $\{c..\}$ $(\lambda x.\ exp\ (x *_R - a)) = exp\ (c *_R - a) / a$
proof −
from *a* **have** $a \neq 0$ **by** *auto*
define *f* **where** $f = (\lambda k\ x.\ if\ x \in \{c..real\ k\}\ then\ exp\ (x *_R -a)\ else\ 0)$
{
fix *k* :: *nat* **assume** *k*: *of-nat* $k \geq c$
from ‹$a \neq 0$› *k*
have $((\lambda x.\ exp\ (x *_R -a))\ has\text{-}integral\ (-exp\ (k *_R -a)/a - (-exp\ (c *_R -a)/a)))\ \{c..real\ k\}$
by (*intro fundamental-theorem-of-calculus*)
(*auto intro*!: *derivative-eq-intros exp-scaleR-has-vector-derivative-left*
simp: *divide-inverse-commute*
simp del: *scaleR-minus-left scaleR-minus-right*)
hence $(f\ k\ has\text{-}integral\ (exp\ (c *_R -a)/a - exp\ (k *_R -a)/a))\ \{c..\}$ **unfolding** *f-def*
by (*subst has-integral-restrict*) *simp-all*
} **note** *has-integral-f* = *this*

have *integrable-fk*: *f k integrable-on* $\{c..\}$ **for** *k*
proof −
have $(\lambda x.\ exp\ (x *_R -a))$ *integrable-on* $\{c..of\text{-}real\ k\}$ (**is** *?P*)
unfolding *f-def* **by** (*auto intro*!: *continuous-intros integrable-continuous-real*)

then have *int*: (*f k*) *integrable-on* {*c..of-real k*}
by (*rule integrable-eq*) (*simp add*: *f-def*)
show *?thesis*
by (*rule integrable-on-superset*[*OF int*]) (*auto simp*: *f-def*)
qed
have *limseq*: $\bigwedge x.\ x \in$ {*c..*} $\Longrightarrow$ ($\lambda k.$ *f k x*) $\longrightarrow$ *exp* ($x *_R - a$)
apply (*auto intro*!: *Lim-transform-eventually*[*OF tendsto-const*] *simp*: *f-def*)
by (*meson eventually-sequentiallyI nat-ceiling-le-eq*)
have *bnd*: $\bigwedge x.\ x \in$ {*c..*} $\Longrightarrow$ *cmod* (*f k x*) $\le$ *exp* ($-$ *Re a* $*$ *x*) **for** *k*
by (*auto simp*: *f-def*)

have [*simp*]: *f k* = (λ-. *0*) **if** *of-nat k* $<$ *c* **for** *k* **using** *that* **by** (*auto simp*: *fun-eq-iff f-def*)
have *integral-f*: *integral* {*c..*} (*f k*) =
(*if real k* $\ge$ *c then exp* ($c *_R -a$)/*a* $-$ *exp* ($k *_R -a$)/*a else 0*)
for *k* **using** *integral-unique*[*OF has-integral-f*[*of k*]] **by** *simp*

have ($\lambda k.$ *exp* ($c *_R -a$)/*a* $-$ *exp* ($k *_R -a$)/*a*) $\longrightarrow$ *exp* ($c*_R-a$)/*a* $-$ *0*/*a*
apply (*intro tendsto-intros filterlim-compose*[*OF exp-at-bot*]
filterlim-tendsto-neg-mult-at-bot[*OF tendsto-const*] *filterlim-real-sequentially*)+
apply (*rule tendsto-norm-zero-cancel*)
by (*auto intro*!: *assms* ‹*a* $\neq$ *0*› *filterlim-real-sequentially*
filterlim-compose[*OF exp-at-bot*] *filterlim-compose*[*OF filterlim-uminus-at-bot-at-top*]
filterlim-at-top-mult-tendsto-pos[*OF tendsto-const*])
moreover
note *A* = *dominated-convergence*[**where** *g*=$\lambda x.$ *exp* ($x *_R -a$),
OF integrable-fk integrable-on-exp-minus-to-infinity[**where** *a*=*Re a* **and** *c*=*c*, *OF* ‹*0* $<$ *Re a*›]
bnd limseq]
from *A*(*1*) **show** ($\lambda x.$ *exp* ($x *_R - a$)) *integrable-on* {*c..*} **.**
from *eventually-gt-at-top*[*of nat* ⌈*c*⌉] **have** *eventually* ($\lambda k.$ *of-nat k* $>$ *c*) *sequentially*
by *eventually-elim linarith*
hence *eventually* ($\lambda k.$ *exp* ($c *_R -a$)/*a* $-$ *exp* ($k *_R -a$)/*a* = *integral* {*c..*} (*f k*)) *sequentially*
by *eventually-elim* (*simp add*: *integral-f*)
ultimately have ($\lambda k.$ *integral* {*c..*} (*f k*)) $\longrightarrow$ *exp* ($c *_R -a$)/*a* $-$ *0*/*a*
by (*rule Lim-transform-eventually*)
from *LIMSEQ-unique*[*OF A*(*2*) *this*]
show *integral* {*c..*} ($\lambda x.$ *exp* ($x *_R -a$)) = *exp* ($c *_R -a$)/*a* **by** *simp*
qed

lemma *has-integral-cexp-minus-to-infinity*:
fixes *a*::*complex*— TODO: generalize
assumes *a*: *0* $<$ *Re a*
shows (($\lambda x.$ *exp* ($x *_R - a$)) *has-integral exp* ($c *_R - a$) / *a*) {*c..*}
using *integral-cexp-minus-to-infinity*[*OF assms*]
integrable-on-cexp-minus-to-infinity[*OF assms*]
using *has-integral-integrable-integral* **by** *blast*

lemma *has-laplace-one*:
  ((λ-. *1*) *has-laplace inverse s*) *s* **if** *Re s* > *0*
proof (*safe intro*!: *has-laplaceI*)
  from *that* **have** ((λ*t*. *exp* (*t* $*_R$ − *s*)) *has-integral inverse s*) {*0*..}
    by (*rule has-integral-cexp-minus-to-infinity*[*THEN has-integral-eq-rhs*])
      (*auto simp*: *inverse-eq-divide*)
  then show ((λ*t*. *exp* (*t* $*_R$ − *s*) ∗ *1*) *has-integral inverse s*) {*0*..} **by** *simp*
qed

lemma *has-laplace-add*:
  assumes *f*: (*f has-laplace F*) *S*
  assumes *g*: (*g has-laplace G*) *S*
  shows ((λ*x*. *f x* + *g x*) *has-laplace F* + *G*) *S*
  apply (*rule has-laplaceI*)
  using *has-integral-add*[*OF has-laplaceD*[*OF f*] *has-laplaceD*[*OF g*]]
  by (*auto simp*: *algebra-simps*)

lemma *has-laplace-cmul*:
  assumes (*f has-laplace F*) *S*
  shows ((λ*x*. *r* $*_R$ *f x*) *has-laplace r* $*_R$ *F*) *S*
  apply (*rule has-laplaceI*)
  using *has-laplaceD*[*OF assms*, *THEN has-integral-cmul*[**where** *c*=*r*]]
  by *auto*

lemma *has-laplace-uminus*:
  assumes (*f has-laplace F*) *S*
  shows ((λ*x*. − *f x*) *has-laplace* − *F*) *S*
  using *has-laplace-cmul*[*OF assms*, *of* −*1*]
  by *auto*

lemma *has-laplace-minus*:
  assumes *f*: (*f has-laplace F*) *S*
  assumes *g*: (*g has-laplace G*) *S*
  shows ((λ*x*. *f x* − *g x*) *has-laplace F* − *G*) *S*
  using *has-laplace-add*[*OF f has-laplace-uminus*[*OF g*]]
  by *simp*

lemma *has-laplace-spike*:
  (*f has-laplace L*) *s*
  if *L*: (*g has-laplace L*) *s*
    and *negligible T*
    and ⋀*t*. *t* ∉ *T* ⟹ *t* ≥ *0* ⟹ *f t* = *g t*
  by (*auto intro*!: *has-laplaceI has-integral-spike*[**where** *S*=*T*, *OF* - - *has-laplaceD*[*OF L*]] *that*)

lemma *has-laplace-frequency-shift*:— First Translation Theorem in Schiff
  ((λ*t*. *exp* (*t* $*_R$ *b*) ∗ *f t*) *has-laplace L*) *s*

if (*f has-laplace L*) (*s* − *b*)
using *that*
by (*auto intro*!: *has-laplaceI dest*!: *has-laplaceD*
simp: *mult-exp-exp algebra-simps*)

theorem *has-laplace-derivative-time-domain*:
(f' *has-laplace* $s * L - f0$) *s*
if *L*: (*f has-laplace L*) *s*
and f': $\bigwedge t.\ t > 0 \Longrightarrow$ (*f has-vector-derivative* f' *t*) (*at t*)
and *f0*: ($f \longrightarrow f0$) (*at-right 0*)
and *eo*: *exponential-order M c f*
and *cs*: $c <$ *Re s*
— Proof and statement follow "The Laplace Transform: Theory and Applications" by Joel L. Schiff.
proof (*rule has-laplaceI*)
have *ce*: *continuous-on S* ($\lambda t.$ *exp* ($t *_R - s$)) **for** *S*
by (*auto intro*!: *continuous-intros*)
have *de*: (($\lambda t.$ *exp* ($t *_R - s$)) *has-vector-derivative* ($- s *$ *exp* ($-$ ($t *_R s$)))) (*at t*) **for** *t*
by (*auto simp*: *has-vector-derivative-def intro*!: *derivative-eq-intros ext*)
have (($\lambda x. -s *$ (*f x* $*$ *exp* ($-$ ($x *_R s$)))) *has-integral* $- s * L$) {*0*..}
apply (*rule has-integral-mult-right*)
using *has-laplaceD*[*OF L*]
by (*auto simp*: *ac-simps*)

define *g* **where** *g x* = (*if* $x \leq 0$ *then f0 else f x*) **for** *x*

have *eog*: *exponential-order M c g*
proof −
from *exponential-orderD*[*OF eo*] **have** $0 < M$
and *ev*: $\forall_F$ *t in at-top. cmod* (*f t*) $\leq M *$ *exp* ($c * t$) **.**
have $\forall_F$ *t*::*real in at-top.* $t > 0$ **by** *simp*
with *ev* **have** $\forall_F$ *t in at-top. cmod* (*g t*) $\leq M *$ *exp* ($c * t$)
by *eventually-elim* (*auto simp*: *g-def*)
with ‹$0 < M$› **show** *?thesis*
by (*rule exponential-orderI*)
qed
have *Lg*: (*g has-laplace L*) *s*
using *L*
by (*rule has-laplace-spike*[**where** *T*={*0*}]) (*auto simp*: *g-def*)
have g': $\bigwedge t.\ 0 < t \Longrightarrow$ (*g has-vector-derivative* f' *t*) (*at t*)
using f'
by (*rule has-vector-derivative-transform-within-open*[**where** *S*={*0*<..}]) (*auto simp*: *g-def*)
have *cg*: *continuous-on* {*0*..*k*} *g* **for** *k*
apply (*auto simp*: *g-def continuous-on-def*)
apply (*rule filterlim-at-within-If*)
subgoal by (*rule tendsto-intros*)
subgoal

```
      apply (rule tendsto-within-subset)
       apply (rule f0)
      by auto
    subgoal premises prems for x
    proof –
      from prems have 0 < x by auto
      from order-tendstoD[OF tendsto-ident-at this]
      have eventually ((<) 0) (at x within {0..k}) by auto
      then have ∀F x in at x within {0..k}. f x = (if x ≤ 0 then f0 else f x)
        by eventually-elim auto
      moreover
      note [simp] = at-within-open[where S={0<..}]
      have continuous-on {0<..} f
        by (rule continuous-on-vector-derivative)
          (auto simp add: intro!: f')
      then have (f ⟶ f x) (at x within {0..k})
        using ‹0 < x›
        by (auto simp: continuous-on-def intro: Lim-at-imp-Lim-at-within)
      ultimately show ?thesis
        by (rule Lim-transform-eventually[rotated])
    qed
    done
  then have pcg: piecewise-continuous-on 0 k {} g for k
    by (auto simp: piecewise-continuous-on-def)
  from piecewise-continuous-on-AE-boundedE[OF this]
  obtain B where B: AE x∈{0..k} in lebesgue. cmod (g x) ≤ B k for k by auto
  have 1: laplace-integrand g s absolutely-integrable-on {0..}
    apply (rule laplace-integrand-absolutely-integrable[OF eog])
      apply (rule B)
     apply (rule piecewise-continuous-on-integrable)
     apply (rule pcg)
    apply (rule cs)
    done
  then have csi: complex-set-integrable lebesgue {0..} (λx. exp (x *R − s) * g x)
    by (auto simp: laplace-integrand-def[abs-def])
  from has-laplaceD[OF Lg, THEN has-integral-improperE, OF csi]
  obtain J where J: ⋀k. ((λt. exp (t *R − s) * g t) has-integral J k) {0..k}
    and [tendsto-intros]: (J ⟶ L) at-top
    by auto
  have ((λx. −s * (exp (x *R − s) * g x)) has-integral −s * J k) {0..k} for k
    by (rule has-integral-mult-right) (rule J)
  then have *: ((λx. g x * (− s * exp (− (x *R s)))) has-integral −s * J k) {0..k}
for k
    by (auto simp: algebra-simps)
  have ∀F k::real in at-top. k ≥ 0
    using eventually-ge-at-top by blast
  then have evI: ∀F k in at-top. ((λt. exp (t *R − s) * f' t) has-integral
    g k * exp (k *R − s) + s * J k − g 0) {0..k}
  proof eventually-elim
```

```
    case (elim k)
    show ?case
      apply (subst mult.commute)
      apply (rule integration-by-parts-interior[OF bounded-bilinear-mult], fact)
      apply (rule cg) apply (rule ce) apply (rule g') apply force apply (rule de)
      apply (rule has-integral-eq-rhs)
       apply (rule *)
      by auto
  qed
  have t1: ((λx. g x * exp (x *R − s)) ⟶ 0) at-top
    apply (subst mult.commute)
    unfolding laplace-integrand-def[symmetric]
    apply (rule Lim-null-comparison)
    apply (rule eventually-laplace-integrand-le[OF eog])
    apply (rule tendsto-mult-right-zero)
    apply (rule filterlim-compose[OF exp-at-bot])
    apply (rule filterlim-tendsto-neg-mult-at-bot)
      apply (rule tendsto-intros)
    using cs apply simp
    apply (rule filterlim-ident)
    done
  show ((λt. exp (t *R − s) * f' t) has-integral s * L − f0) {0..}
    apply (rule has-integral-improper-at-topI[OF evI])
    subgoal
      apply (rule tendsto-eq-intros)
       apply (rule tendsto-intros)
        apply (rule t1)
       apply (rule tendsto-intros)
        apply (rule tendsto-intros)
       apply (rule tendsto-intros)
       apply (rule tendsto-intros)
      by (simp add: g-def)
    done
qed

lemma exp-times-has-integral:
  ((λt. exp (c * t)) has-integral (if c = 0 then t else exp (c * t) / c) − (if c = 0
then t0 else exp (c * t0) / c)) {t0 .. t}
  if t0 ≤ t
  for c t::real
  apply (cases c = 0)
  subgoal
    using that
    apply auto
    apply (rule has-integral-eq-rhs)
     apply (rule has-integral-const-real)
    by auto
  subgoal
    apply (rule fundamental-theorem-of-calculus)
```

using *that*
by (*auto simp*: *has-vector-derivative-def intro*!: *derivative-eq-intros*)
done

lemma *integral-exp-times*:
integral {*t0* .. *t*} (λ*t*. *exp* (*c* ∗ *t*)) = (*if c* = *0 then t* − *t0 else exp* (*c* ∗ *t*) / *c* − *exp* (*c* ∗ *t0*) / *c*)
if *t0* ≤ *t*
for *c t*::*real*
using *exp-times-has-integral*[*OF that*, *of c*] *that*
by (*auto split*: *if-splits*)

lemma *filtermap-times-pos-at-top*: *filtermap* ((∗) *e*) *at-top* = *at-top*
if *e* > *0*
for *e*::*real*
apply (*rule filtermap-fun-inverse*[*of* (∗) (*inverse e*)])
apply (*rule filterlim-tendsto-pos-mult-at-top*)
apply (*rule tendsto-intros*)
subgoal using *that* **by** *simp*
apply (*rule filterlim-ident*)
apply (*rule filterlim-tendsto-pos-mult-at-top*)
apply (*rule tendsto-intros*)
subgoal using *that* **by** *simp*
apply (*rule filterlim-ident*)
using *that* **by** *auto*

lemma *exponential-order-additiveI*:
assumes *0* < *M* **and** *eo*: $\forall_F$ *t in at-top*. *norm* (*f t*) ≤ *K* + *M* ∗ *exp* (*c* ∗ *t*) **and** *c* ≥ *0*
obtains *M'* **where** *exponential-order M' c f*
proof −
consider *c* = *0* | *c* > *0* **using** ‹*c* ≥ *0*› **by** *arith*
then show *?thesis*
proof *cases*
assume *c* = *0*
have *exponential-order* (*max K 0* + *M*) *c f*
using *eo*
apply (*auto intro*!: *exponential-orderI add-nonneg-pos* ‹*0* < *M*› *simp*: ‹*c* = *0*›)
apply (*auto simp*: *max-def*)
using *eventually-elim2* **by** *force*
then show *?thesis* **..**
next
assume *c* > *0*
have $\forall_F$ *t in at-top*. *norm* (*f t*) ≤ *K* + *M* ∗ *exp* (*c* ∗ *t*)
by *fact*
moreover
have $\forall_F$ *t in* (*filtermap exp* (*filtermap* ((∗) *c*) *at-top*)). *K* < *t*
by (*simp add*: *filtermap-times-pos-at-top* ‹*c* > *0*› *filtermap-exp-at-top*)

```
    then have ∀F t in at-top. K < exp (c * t)
      by (simp add: eventually-filtermap)
    ultimately
    have ∀F t in at-top. norm (f t) ≤ (1 + M) * exp (c * t)
      by eventually-elim (auto simp: algebra-simps)
    with add-nonneg-pos[OF zero-le-one ‹0 < M›]
    have exponential-order (1 + M) c f
      by (rule exponential-orderI)
    then show ?thesis ..
  qed
qed

lemma exponential-order-integral:
  fixes f::real ⇒ 'a::banach
  assumes I: ⋀t. t ≥ a ⟹ (f has-integral I t) {a .. t}
    and eo: exponential-order M c f
    and c > 0
  obtains M' where exponential-order M' c I
proof −
  from exponential-orderD[OF eo] have 0 < M
    and bound: ∀F t in at-top. norm (f t) ≤ M * exp (c * t)
    by auto
  have ∀F t in at-top. t > a
    by simp
  from bound this
  have ∀F t in at-top. norm (f t) ≤ M * exp (c * t) ∧ t > a
    by eventually-elim auto
  then obtain t0 where t0: ⋀t. t ≥ t0 ⟹ norm (f t) ≤ M * exp (c * t) t0 > a
    by (auto simp: eventually-at-top-linorder)
  have ∀F t in at-top. t > t0 by simp
  then have ∀F t in at-top. norm (I t) ≤ norm (integral {a..t0} f) − M * exp (c
* t0) / c + (M / c) * exp (c * t)
  proof eventually-elim
    case (elim t) then have that: t ≥ t0 by simp
    from t0 have a ≤ t0 by simp
    have f integrable-on {a .. t0} f integrable-on {t0 .. t}
      subgoal by (rule has-integral-integrable[OF I[OF ‹a ≤ t0›]])
      subgoal
       apply (rule integrable-on-subinterval[OF has-integral-integrable[OF I[where
t=t]]])
        using ‹t0 > a› that by auto
      done
    have I t = integral {a .. t0} f + integral {t0 .. t} f
     by (metis Henstock-Kurzweil-Integration.integral-combine I ‹a ≤ t0› dual-order.strict-trans
          has-integral-integrable-integral less-eq-real-def that)
     also have norm ... ≤ norm (integral {a .. t0} f) + norm (integral {t0 .. t}
f) by norm
    also
    have norm (integral {t0 .. t} f) ≤ integral {t0 .. t} (λt. M * exp (c * t))
```

```
      apply (rule integral-norm-bound-integral)
        apply fact
      by (auto intro!: integrable-continuous-interval continuous-intros t0)
    also have ... = M * integral {t0 .. t} (λt. exp (c * t))
      by simp
    also have integral {t0 .. t} (λt. exp (c * t)) = exp (c * t) / c − exp (c * t0)
/ c
      using ‹c > 0› ‹t0 ≤ t›
      by (subst integral-exp-times) auto
    finally show ?case
      using ‹c > 0›
      by (auto simp: algebra-simps)
  qed
  from exponential-order-additiveI[OF divide-pos-pos[OF ‹0 < M› ‹0 < c›] this
less-imp-le[OF ‹0 < c›]]
  obtain M′ where exponential-order M′ c I .
  then show ?thesis ..
qed

lemma integral-has-vector-derivative-piecewise-continuous:
  fixes f :: real ⇒ ′a::euclidean-space— TODO: generalize?
  assumes piecewise-continuous-on a b D f
  shows ⋀x. x ∈ {a .. b} − D ⟹
    ((λu. integral {a..u} f) has-vector-derivative f(x)) (at x within {a..b} − D)
  using assms
proof (induction a b D f rule: piecewise-continuous-on-induct)
  case (empty a b f)
  then show ?case
    by (auto intro: integral-has-vector-derivative)
next
  case (combine a i b I f1 f2 f)
  then consider x < i | i < x by auto arith

  then show ?case
  proof cases— TODO: this is very explicit...
    case 1
    have evless: ∀F xa in nhds x. xa < i
      apply (rule order-tendstoD[OF - ‹x < i›])
      by (simp add: filterlim-ident)
    have eq: at x within {a..b} − insert i I = at x within {a .. i} − I
      unfolding filter-eq-iff
    proof safe
      fix P
      assume eventually P (at x within {a..i} − I)
      with evless show eventually P (at x within {a..b} − insert i I)
        unfolding eventually-at-filter
        by eventually-elim auto
    next
      fix P
```

assume *eventually P* (*at x within* {*a..b*} − *insert i I*)
with *evless* **show** *eventually P* (*at x within* {*a..i*} − *I*)
unfolding *eventually-at-filter*
apply *eventually-elim*
using *1 combine*
by *auto*
qed
have *f x* = *f1 x* **using** *combine 1* **by** *auto*
have *i-eq*: *integral* {*a..y*} *f* = *integral* {*a..y*} *f1* **if** *y* < *i* **for** *y*
using *negligible-empty*
apply (*rule integral-spike*)
using *combine 1 that*
by *auto*
from *evless* **have** *ev-eq*: ∀$_F$ *x in nhds x. x* ∈ {*a..i*} − *I* ⟶ *integral* {*a..x*} *f* = *integral* {*a..x*} *f1*
by *eventually-elim* (*auto simp*: *i-eq*)
show *?thesis* **unfolding** *eq* ‹*f x* = *f1 x*›
apply (*subst has-vector-derivative-cong-ev*[*OF ev-eq*])
using *combine.IH*[*of x*]
using *combine.hyps combine.prems 1*
by (*auto simp*: *i-eq*)
next
case *2*
have *evless*: ∀$_F$ *xa in nhds x. xa* > *i*
apply (*rule order-tendstoD*[*OF -* ‹*x* > *i*›])
by (*simp add*: *filterlim-ident*)
have *eq*: *at x within* {*a..b*} − *insert i I* = *at x within* {*i .. b*} − *I*
unfolding *filter-eq-iff*
proof *safe*
fix *P*
assume *eventually P* (*at x within* {*i..b*} − *I*)
with *evless* **show** *eventually P* (*at x within* {*a..b*} − *insert i I*)
unfolding *eventually-at-filter*
by *eventually-elim auto*
next
fix *P*
assume *eventually P* (*at x within* {*a..b*} − *insert i I*)
with *evless* **show** *eventually P* (*at x within* {*i..b*} − *I*)
unfolding *eventually-at-filter*
apply *eventually-elim*
using *2 combine*
by *auto*
qed
have *f x* = *f2 x* **using** *combine 2* **by** *auto*
have *i-eq*: *integral* {*a..y*} *f* = *integral* {*a..i*} *f* + *integral* {*i..y*} *f2* **if** *i* < *y y* ≤ *b* **for** *y*
proof −
have *integral* {*a..y*} *f* = *integral* {*a..i*} *f* + *integral* {*i..y*} *f*
apply (*cases i* = *y*)

subgoal by *auto*
subgoal
apply (*rule Henstock-Kurzweil-Integration.integral-combine*[*symmetric*])
using *combine that* **apply** *auto*
apply (*rule integrable-Un′*[**where** *A*={*a* .. *i*} **and** *B*={*i*..*y*}])
subgoal
by (*rule integrable-spike*[**where** *S*={*i*} **and** *f*=*f1*])
(*auto intro*: *piecewise-continuous-on-integrable*)
subgoal
apply (*rule integrable-on-subinterval*[**where** *S*={*i*..*b*}])
by (*rule integrable-spike*[**where** *S*={*i*} **and** *f*=*f2*])
(*auto intro*: *piecewise-continuous-on-integrable*)
subgoal by (*auto simp*: *max-def min-def*)
subgoal by *auto*
done
done
also have *integral* {*i*..*y*} *f* = *integral* {*i*..*y*} *f2*
apply (*rule integral-spike*[**where** *S*={*i*}])
using *combine 2 that*
by *auto*
finally show *?thesis* **.**
qed
from *evless* **have** *ev-eq*: $\forall_F$ *y in nhds x*. *y* ∈ {*i*..*b*} − *I* ⟶ *integral* {*a*..*y*} *f* = *integral* {*a*..*i*} *f* + *integral* {*i*..*y*} *f2*
by *eventually-elim* (*auto simp*: *i-eq*)
show *?thesis* **unfolding** *eq*
apply (*subst has-vector-derivative-cong-ev*[*OF ev-eq*])
using *combine.IH*[*of x*] *combine.prems combine.hyps 2*
by (*auto simp*: *i-eq intro*!: *derivative-eq-intros*)
qed
qed (*auto intro*: *has-vector-derivative-within-subset*)

lemma *has-derivative-at-split*:
(*f has-derivative f′*) (*at x*) ⟷ (*f has-derivative f′*) (*at-left x*) ∧ (*f has-derivative f′*) (*at-right x*)
for *x*::′*a*::{*linorder-topology*, *real-normed-vector*}
by (*auto simp*: *has-derivative-at-within filterlim-at-split*)

lemma *has-vector-derivative-at-split*:
(*f has-vector-derivative f′*) (*at x*) ⟷
(*f has-vector-derivative f′*) (*at-left x*) ∧
(*f has-vector-derivative f′*) (*at-right x*)
using *has-derivative-at-split*[*of f* λ*h*. *h* $*_R$ *f′ x*]
by (*simp add*: *has-vector-derivative-def*)

lemmas *differentiableI-vector*[*intro*]

lemma *differentiable-at-splitD*:
f differentiable at-left x

f differentiable at-right x
if *f differentiable* (*at x*)
for *x*::*real*
using *that*[*unfolded vector-derivative-works has-vector-derivative-at-split*]
by *auto*

lemma *integral-differentiable*:
fixes *f* :: *real* ⇒ *'a*::*banach*
assumes *continuous-on* {*a*..*b*} *f*
and *x* ∈ {*a*..*b*}
shows (*λu. integral* {*a*..*u*} *f*) *differentiable at x within* {*a*..*b*}
using *integral-has-vector-derivative*[*OF assms*]
by *blast*

theorem *integral-has-vector-derivative-piecewise-continuous'*:
fixes *f* :: *real* ⇒ *'a*::*euclidean-space*— TODO: generalize?
assumes *piecewise-continuous-on a b D f a* < *b*
shows
(∀ *x*. *a* < *x* ⟶ *x* < *b* ⟶ *x* ∉ *D* ⟶ (*λu. integral* {*a*..*u*} *f*) *differentiable at x*) ∧
(∀ *x*. *a* ≤ *x* ⟶ *x* < *b* ⟶ (*λt. integral* {*a*..*t*} *f*) *differentiable at-right x*) ∧
(∀ *x*. *a* < *x* ⟶ *x* ≤ *b* ⟶ (*λt. integral* {*a*..*t*} *f*) *differentiable at-left x*)
using *assms*
proof (*induction a b D f rule*: *piecewise-continuous-on-induct*)
case (*empty a b f*)
have *a* < *x* ⟹ *x* < *b* ⟹ (*λu. integral* {*a*..*u*} *f*) *differentiable* (*at x*) **for** *x*
using *integral-differentiable*[*OF empty*(*1*), *of x*]
by (*auto simp*: *at-within-interior*)
then show *?case*
using *integral-differentiable*[*OF empty*(*1*), *of a*]
integral-differentiable[*OF empty*(*1*), *of b*]
‹*a* < *b*›
by (*auto simp*: *at-within-Icc-at-right at-within-Icc-at-left le-less*
intro: *differentiable-at-withinI*)
next
case (*combine a i b I f1 f2 f*)
from ‹*piecewise-continuous-on a i I f1*› **have** *finite I*
by (*auto elim*!: *piecewise-continuous-onE*)

from *combine*(*4*) **have** *piecewise-continuous-on a i* (*insert i I*) *f1*
by (*rule piecewise-continuous-on-insert-rightI*)
then have *piecewise-continuous-on a i* (*insert i I*) *f*
by (*rule piecewise-continuous-on-congI*) (*auto simp*: *combine*)
moreover
from *combine*(*5*) **have** *piecewise-continuous-on i b* (*insert i I*) *f2*
by (*rule piecewise-continuous-on-insert-leftI*)
then have *piecewise-continuous-on i b* (*insert i I*) *f*
by (*rule piecewise-continuous-on-congI*) (*auto simp*: *combine*)
ultimately have *piecewise-continuous-on a b* (*insert i I*) *f*

by (*rule piecewise-continuous-on-combine*)
then have *f-int*: *f integrable-on* {*a* .. *b*}
by (*rule piecewise-continuous-on-integrable*)

from *combine.IH*
have *f1*: $x>a \Longrightarrow x < i \Longrightarrow x \notin I \Longrightarrow$ (λu. *integral* {*a..u*} *f1*) *differentiable* (*at* *x*)
$x \geq a \Longrightarrow x < i \Longrightarrow$ (λt. *integral* {*a..t*} *f1*) *differentiable* (*at-right x*)
$x>a \Longrightarrow x \leq i \Longrightarrow$ (λt. *integral* {*a..t*} *f1*) *differentiable* (*at-left x*)
and *f2*: $x>i \Longrightarrow x < b \Longrightarrow x \notin I \Longrightarrow$ (λu. *integral* {*i..u*} *f2*) *differentiable* (*at* *x*)
$x \geq i \Longrightarrow x < b \Longrightarrow$ (λt. *integral* {*i..t*} *f2*) *differentiable* (*at-right x*)
$x>i \Longrightarrow x \leq b \Longrightarrow$ (λt. *integral* {*i..t*} *f2*) *differentiable* (*at-left x*)
for *x*
by *auto*

have (λu. *integral* {*a..u*} *f*) *differentiable at x* **if** $a < x$ $x < b$ $x \neq i$ $x \notin I$ **for** *x*
proof −
from *that* **consider** $x < i$ |$i < x$ **by** *arith*
then show *?thesis*
proof *cases*
case *1*
have *at*: *at x within* {*a*<..<*i*} − *I* = *at x*
using *that 1*
by (*intro at-within-open*) (*auto intro*!: *open-Diff finite-imp-closed* ‹*finite I*›)
then have (λu. *integral* {*a..u*} *f1*) *differentiable at x within* {*a*<..<*i*} − *I*
using *that 1 f1* **by** *auto*
then have (λu. *integral* {*a..u*} *f*) *differentiable at x within* {*a*<..<*i*} − *I*
apply (*rule differentiable-transform-within*[*OF* - *zero-less-one*])
using *that combine.hyps 1* **by** (*auto intro*!: *integral-cong*)
then show *?thesis* **by** (*simp add*: *at*)
next
case *2*
have *at*: *at x within* {*i*<..<*b*} − *I* = *at x*
using *that 2*
by (*intro at-within-open*) (*auto intro*!: *open-Diff finite-imp-closed* ‹*finite I*›)
then have (λu. *integral* {*a..i*} *f* + *integral* {*i..u*} *f2*) *differentiable at x within* {*i*<..<*b*} − *I*
using *that 2 f2* **by** *auto*
then have (λu. *integral* {*a..i*} *f* + *integral* {*i..u*} *f*) *differentiable at x within* {*i*<..<*b*} − *I*
apply (*rule differentiable-transform-within*[*OF* - *zero-less-one*])
using *that combine.hyps 2* **by** (*auto intro*!: *integral-spike*[**where** *S*={*i*,*x*}])
then have (λu. *integral* {*a..u*} *f*) *differentiable at x within* {*i*<..<*b*} − *I*
apply (*rule differentiable-transform-within*[*OF* - *zero-less-one*])
subgoal using *that 2* **by** *auto*
apply *auto*
apply (*subst Henstock-Kurzweil-Integration.integral-combine*)
using *that 2* ‹$a \leq i$›

apply *auto*
by (*auto intro*: *integrable-on-subinterval f-int*)
then show *?thesis* **by** (*simp add*: *at*)
qed
qed
moreover
have (λt. *integral* $\{a..t\}$ *f*) *differentiable at-right x* **if** $a \leq x$ $x < b$ **for** *x*
proof −
from *that* **consider** $x < i$ |$i \leq x$ **by** *arith*
then show *?thesis*
proof *cases*
case *1*
have *at*: *at x within* $\{x..i\}$ = *at-right x*
using ‹$x < i$› **by** (*rule at-within-Icc-at-right*)
then have (λu. *integral* $\{a..u\}$ *f1*) *differentiable at x within* $\{x..i\}$
using *that 1 f1* **by** *auto*
then have (λu. *integral* $\{a..u\}$ *f*) *differentiable at x within* $\{x..i\}$
apply (*rule differentiable-transform-within*[*OF - zero-less-one*])
using *that combine.hyps 1* **by** (*auto intro*!: *integral-spike*[**where** S=$\{i,x\}$])
then show *?thesis* **by** (*simp add*: *at*)
next
case *2*
have *at*: *at x within* $\{x..b\}$ = *at-right x*
using ‹$x < b$› **by** (*rule at-within-Icc-at-right*)
then have (λu. *integral* $\{a..i\}$ *f* + *integral* $\{i..u\}$ *f2*) *differentiable at x within* $\{x..b\}$
using *that 2 f2* **by** *auto*
then have (λu. *integral* $\{a..i\}$ *f* + *integral* $\{i..u\}$ *f*) *differentiable at x within* $\{x..b\}$
apply (*rule differentiable-transform-within*[*OF - zero-less-one*])
using *that combine.hyps 2* **by** (*auto intro*!: *integral-spike*[**where** S=$\{i,x\}$])
then have (λu. *integral* $\{a..u\}$ *f*) *differentiable at x within* $\{x..b\}$
apply (*rule differentiable-transform-within*[*OF - zero-less-one*])
subgoal using *that 2* **by** *auto*
apply *auto*
apply (*subst Henstock-Kurzweil-Integration.integral-combine*)
using *that 2* ‹$a \leq i$›
apply *auto*
by (*auto intro*: *integrable-on-subinterval f-int*)
then show *?thesis* **by** (*simp add*: *at*)
qed
qed
moreover
have (λt. *integral* $\{a..t\}$ *f*) *differentiable at-left x* **if** $a < x$ $x \leq b$ **for** *x*
proof −
from *that* **consider** $x \leq i$ |$i < x$ **by** *arith*
then show *?thesis*
proof *cases*
case *1*

```
      have at: at x within {a..x} = at-left x
        using ‹a < x› by (rule at-within-Icc-at-left)
      then have (λu. integral {a..u} f1) differentiable at x within {a..x}
        using that 1 f1 by auto
      then have (λu. integral {a..u} f) differentiable at x within {a..x}
        apply (rule differentiable-transform-within[OF - zero-less-one])
        using that combine.hyps 1 by (auto intro!: integral-spike[where S={i,x}])
      then show ?thesis by (simp add: at)
    next
      case 2
      have at: at x within {i..x} = at-left x
        using ‹i < x› by (rule at-within-Icc-at-left)
     then have (λu. integral {a..i} f + integral {i..u} f2) differentiable at x within
{i..x}
        using that 2 f2 by auto
     then have (λu. integral {a..i} f + integral {i..u} f) differentiable at x within
{i..x}
        apply (rule differentiable-transform-within[OF - zero-less-one])
        using that combine.hyps 2 by (auto intro!: integral-spike[where S={i,x}])
      then have (λu. integral {a..u} f) differentiable at x within {i..x}
        apply (rule differentiable-transform-within[OF - zero-less-one])
        subgoal using that 2 by auto
        apply auto
        apply (subst Henstock-Kurzweil-Integration.integral-combine)
        using that 2 ‹a ≤ i›
        apply auto
        by (auto intro: integrable-on-subinterval f-int)
      then show ?thesis by (simp add: at)
    qed
  qed
  ultimately
  show ?case
    by auto
next
  case (weaken a b i I f)
  from weaken.IH[OF ‹a < b›]
  obtain l u where IH:
    ⋀x. a < x ⟹ x < b ⟹ x ∉ I ⟹ (λu. integral {a..u} f) differentiable (at x)
    ⋀x. a ≤ x ⟹ x < b ⟹ (λt. integral {a..t} f) differentiable (at-right x)
    ⋀x. a < x ⟹ x ≤ b ⟹ (λt. integral {a..t} f) differentiable (at-left x)
    by metis
  then show ?case by auto
qed

lemma closure (−S) ∩ closure S = frontier S
  by (auto simp add: frontier-def closure-complement)

theorem integral-time-domain-has-laplace:
  ((λt. integral {0 .. t} f) has-laplace L / s) s
```

```
  if pc: ⋀k. piecewise-continuous-on 0 k D f
    and eo: exponential-order M c f
    and L: (f has-laplace L) s
    and s: Re s > c
    and c: c > 0
    and TODO: D = {} — TODO: generalize to actual piecewise-continuous-on
  for f::real ⇒ complex
proof −
  define I where I = (λt. integral {0 .. t} f)
  have I': (I has-vector-derivative f t) (at t within {0..x} − D)
    if t ∈ {0 .. x} − D
    for x t
    unfolding I-def
    by (rule integral-has-vector-derivative-piecewise-continuous; fact)
  have fi: f integrable-on {0..t} for t
    by (rule piecewise-continuous-on-integrable) fact
  have Ic: continuous-on {0 .. t} I for t
    unfolding I-def using fi
    by (rule indefinite-integral-continuous-1)
  have Ipc: piecewise-continuous-on 0 t {} I for t
    by (rule piecewise-continuous-onI) (auto intro!: Ic)
  have I: (f has-integral I t) {0 .. t} for t
    unfolding I-def
    using fi
    by (rule integrable-integral)
  from exponential-order-integral[OF I eo ‹0 < c›] obtain M'
    where Ieo: exponential-order M' c I .
  have Ili: laplace-integrand I s integrable-on {0..}
    using Ipc
    apply (rule piecewise-continuous-on-AE-boundedE)
    apply (rule laplace-integrand-integrable)
    apply (rule Ieo)
      apply assumption
     apply (rule integrable-continuous-interval)
     apply (rule Ic)
    apply (rule s)
    done
  then obtain LI where LI: (I has-laplace LI) s
    by (rule laplace-exists-laplace-integrandI)

  from piecewise-continuous-onE[OF pc] have ‹finite D› by auto
  have I'2: (I has-vector-derivative f t) (at t) if t > 0 t ∉ D for t
    apply (subst at-within-open[symmetric, where S={0<..<t+1} − D])
    subgoal using that by auto
    subgoal by (auto intro!:open-Diff finite-imp-closed ‹finite D›)
    subgoal using I'[where x=t + 1]
      apply (rule has-vector-derivative-within-subset)
      using that
      by auto
```

done
have *I-tndsto*: $(I \longrightarrow 0)$ (*at-right 0*)
apply (*rule tendsto-eq-rhs*)
apply (*rule continuous-on-Icc-at-rightD*)
apply (*rule Ic*)
apply (*rule zero-less-one*)
by (*auto simp*: *I-def*)
have (*f has-laplace* $s * LI - 0$) *s*
by (*rule has-laplace-derivative-time-domain*[*OF LI I'2 I-tndsto Ieo s*])
(*auto simp*: *TODO*)
from *has-laplace-unique*[*OF this L*] **have** $LI = L / s$
using *s c* **by** *auto*
with *LI* **show** (*I has-laplace* L / s) *s* **by** *simp*
qed

4.4 higher derivatives

definition *nderiv i f X* = $((\lambda f.\ (\lambda x.\ \textit{vector-derivative}\ f\ (\textit{at}\ x\ \textit{within}\ X)))$^^*i*) *f*

definition *ndiff n f X* $\longleftrightarrow$ ($\forall i<n.\ \forall x \in X.$ *nderiv i f X differentiable at x within X*)

lemma *nderiv-zero*[*simp*]: *nderiv 0 f X* = *f*
by (*auto simp*: *nderiv-def*)

lemma *nderiv-Suc*[*simp*]:
nderiv (*Suc i*) *f X x* = *vector-derivative* (*nderiv i f X*) (*at x within X*)
by (*auto simp*: *nderiv-def*)

lemma *ndiff-zero*[*simp*]: *ndiff 0 f X*
by (*auto simp*: *ndiff-def*)

lemma *ndiff-Sucs*[*simp*]:
ndiff (*Suc i*) *f X* $\longleftrightarrow$
(*ndiff i f X*) $\wedge$
($\forall x \in X.$ (*nderiv i f X*) *differentiable* (*at x within X*))
apply (*auto simp*: *ndiff-def*)
using *less-antisym* **by** *blast*

theorem *has-laplace-vector-derivative*:
(($\lambda t.$ *vector-derivative f* (*at t*)) *has-laplace* $s * L - f0$) *s*
if *L*: (*f has-laplace L*) *s*
and *f'*: $\bigwedge t.\ t > 0 \Longrightarrow$ *f differentiable* (*at t*)
and *f0*: $(f \longrightarrow f0)$ (*at-right 0*)
and *eo*: *exponential-order M c f*
and *cs*: $c < Re\ s$
proof −
have *f'*: ($\bigwedge t.\ 0 < t \Longrightarrow$ (*f has-vector-derivative vector-derivative f* (*at t*)) (*at t*))
using *f'*

```
    by (subst vector-derivative-works[symmetric])
  show ?thesis
    by (rule has-laplace-derivative-time-domain[OF L f' f0 eo cs])
qed

lemma has-laplace-nderiv:
  (nderiv n f {0<..} has-laplace s^n * L − (∑ i<n. s^(n − Suc i) * f0 i)) s
  if L: (f has-laplace L) s
    and f': ndiff n f {0<..}
    and f0: ⋀i. i < n ⟹ (nderiv i f {0<..} ⟶ f0 i) (at-right 0)
    and eo: ⋀i. i < n ⟹ exponential-order M c (nderiv i f {0<..})
    and cs: c < Re s
  using f' f0 eo
proof (induction n)
  case 0
  then show ?case
    by (auto simp: L)
next
  case (Suc n)
  have awo: at t within {0<..} = at t if t > 0 for t::real
    using that
    by (subst at-within-open) auto
  have ((λa. vector-derivative (nderiv n f {0<..}) (at a)) has-laplace
    s * ( s ^ n * L − (∑ i<n. s^(n − Suc i) * f0 i)) − f0 n) s
    (is (- has-laplace ?L) -)
    apply (rule has-laplace-vector-derivative)
        apply (rule Suc.IH)
    subgoal using Suc.prems by auto
    subgoal using Suc.prems by auto
    subgoal using Suc.prems by auto
    subgoal using Suc.prems by (auto simp: awo)
    subgoal using Suc.prems by auto
     apply (rule Suc.prems; force)
    apply (rule cs)
    done
  also have ?L = s ^ Suc n * L − (∑ i<Suc n. s ^ (Suc n − Suc i) * f0 i)
    by (auto simp: algebra-simps sum-distrib-left diff-Suc Suc-diff-le
        split: nat.splits
        intro!: sum.cong)
  finally show ?case
    by (rule has-laplace-spike[where T={0}]) (auto simp: awo)
qed

end
```

5 Lerch Lemma

```
theory Lerch-Lemma
  imports
```

```
    HOL−Analysis.Analysis
begin
```

The main tool to prove uniqueness of the Laplace transform.

```
lemma lerch-lemma-real:
  fixes h::real ⇒ real
  assumes h-cont[continuous-intros]: continuous-on {0 .. 1} h
  assumes int-0: ⋀n. ((λu. u ^ n * h u) has-integral 0) {0 .. 1}
  assumes u: 0 ≤ u u ≤ 1
  shows h u = 0
proof −
  from Stone-Weierstrass-uniform-limit[OF compact-Icc h-cont]
  obtain g where g: uniform-limit {0..1} g h sequentially polynomial-function (g
n) for n
    by blast
  then have rpf-g: real-polynomial-function (g n) for n
    by (simp add: real-polynomial-function-eq)

  let ?P = λn x. h x * g n x
  have continuous-on-g[continuous-intros]: continuous-on s (g n) for s n
    by (rule continuous-on-polymonial-function) fact
  have P-cont: continuous-on {0 .. 1} (?P n) for n
    by (auto intro!: continuous-intros)
  have uniform-limit {0 .. 1} (λn x. h x * g n x) (λx. h x * h x) sequentially
   by (auto intro!: uniform-limit-intros g assms compact-imp-bounded compact-continuous-image)

  from uniform-limit-integral[OF this P-cont]
  obtain I J where
    I: (⋀n. (?P n has-integral I n) {0..1})
    and J: ((λx. h x * h x) has-integral J) {0..1}
    and IJ: I ⟶ J
    by auto

  have (?P n has-integral 0) {0..1} for n
  proof −
    from real-polynomial-function-imp-sum[OF rpf-g]
    obtain gn ga where g n = (λx. ∑ i≤gn. ga i * x ^ i) by metis
    then have ?P n x = (∑ i≤gn. x ^ i * h x * ga i) for x
      by (auto simp: sum-distrib-left algebra-simps)
    moreover have ((λx. ... x) has-integral 0) {0 .. 1}
     by (auto intro!: has-integral-sum[THEN has-integral-eq-rhs] has-integral-mult-left
assms)
    ultimately show ?thesis by simp
  qed
  with I have I n = 0 for n
    using has-integral-unique by blast
  with IJ J have ((λx. h x * h x) has-integral 0) (cbox 0 1)
   by (metis (full-types) LIMSEQ-le-const LIMSEQ-le-const2 box-real(2) dual-order.antisym
order-refl)
```

with - - **have** *h u* ∗ *h u* = *0*
by (*rule has-integral-0-cbox-imp-0*) (*auto intro*!: *continuous-intros u*)
then show *h u* = *0*
by *simp*
qed

lemma *lerch-lemma*:
fixes *h*::*real* ⇒ ′*a*::*euclidean-space*
assumes [*continuous-intros*]: *continuous-on* {*0* .. *1*} *h*
assumes *int-0*: ⋀*n*. ((λ*u*. *u* ^ *n* $*_R$ *h u*) *has-integral 0*) {*0* .. *1*}
assumes *u*: *0* ≤ *u u* ≤ *1*
shows *h u* = *0*
proof (*rule euclidean-eqI*)
fix *b*::′*a* **assume** *b* ∈ *Basis*
have *continuous-on* {*0* .. *1*} (λ*x*. *h x* · *b*)
by (*auto intro*!: *continuous-intros*)
moreover
from ‹*b* ∈ *Basis*› **have** ((λ*u*. *u* ^ *n* ∗ (*h u* · *b*)) *has-integral 0*) {*0* .. *1*} **for** *n*
using *int-0*[*of n*] *has-integral-componentwise-iff*[*of* λ*u*. *u* ^ *n* $*_R$ *h u 0* {*0* .. *1*}]
by *auto*
moreover note *u*
ultimately show *h u* · *b* = *0* · *b*
unfolding *inner-zero-left*
by (*rule lerch-lemma-real*)
qed

end

6 Uniqueness of Laplace Transform

theory *Uniqueness*
imports
Existence
Lerch-Lemma
begin

We show uniqueness of the Laplace transform for continuous functions.

lemma *laplace-transform-zero*:— should also work for piecewise continuous
assumes *cont-f*: *continuous-on* {*0*..} *f*
assumes *eo*: *exponential-order M a f*
assumes *laplace*: ⋀*s*. *Re s* > *a* ⟹ (*f has-laplace 0*) *s*
assumes *t* ≥ *0*
shows *f t* = *0*
proof −
define *I* **where** *I* ≡ λ*s k*. *integral* {*0*..*k*} (*laplace-integrand f s*)
have *bounded-image*: *bounded* (*f* ‘ {*0*..*b*}) **for** *b*
by (*auto intro*!: *compact-imp-bounded compact-continuous-image cont-f intro*: *continuous-on-subset*)

obtain *B* **where** *B*: ∀ *x*∈{*0*..*b*}. *cmod* (*f x*) ≤ *B b* **for** *b*
  apply *atomize-elim*
  apply (*rule choice*)
  using *bounded-image*[*unfolded bounded-iff*]
  by *auto*
have *fi*: *f integrable-on* {*0*..*b*} **for** *b*
 by (*auto intro*!: *integrable-continuous-interval intro*: *continuous-on-subset cont-f*)
have *aint*: *complex-set-integrable lebesgue* {*0*..*b*} (*laplace-integrand f s*) **for** *b s*
  by (*rule laplace-integrand-absolutely-integrable-on-Icc*[*OF*
    AE-BallI[*OF bounded-le-Sup*[*OF bounded-image*]] *fi*])
have *int*: ((λ*t*. *exp* (*t* $*_R$ − *s*) ∗ *f t*) *has-integral I s b*) {*0* .. *b*} **for** *s b*
  using *aint*[*of b s*]
  unfolding *laplace-integrand-def*[*symmetric*] *I-def absolutely-integrable-on-def*
  by *blast*
have *I-integral*: *Re s* > *a* ⟹ (*I s* ⟶ *integral* {*0*..} (*laplace-integrand f s*)) *at-top* **for** *s*
  unfolding *I-def*
 by (*metis aint eo improper-integral-at-top laplace-integrand-absolutely-integrable-on-Ici-iff*)
have *imp*: (*I s* ⟶ *0*) *at-top* **if** *s*: *Re s* > *a* **for** *s*
  using *I-integral*[*of s*] *laplace*[*unfolded has-laplace-def*, *rule-format*, *OF s*] *s*
  unfolding *has-laplace-def I-def laplace-integrand-def*
  by (*simp add*: *integral-unique*)

define *s0* **where** *s0* = *a* + *1*
then have *s0* > *a* **by** *auto*
have ∀ $_F$ *x in at-right* (*0*::*real*). *0* < *x* ∧ *x* < *1*
  by (*auto intro*!: *eventually-at-rightI*)
moreover
from *exponential-orderD*(*2*)[*OF eo*]
have ∀ $_F$ *t in at-right 0*. *cmod* (*f* (− *ln t*)) ≤ *M* ∗ *exp* (*a* ∗ (− *ln t*))
  unfolding *at-top-mirror filtermap-ln-at-right*[*symmetric*] *eventually-filtermap* **.**
ultimately have ∀ $_F$ *x in at-right 0*. *cmod* ((*x powr s0*) ∗ *f* (− *ln x*)) ≤ *M* ∗ *x powr* (*s0* − *a*)
  (**is** ∀ $_F$ *x in* -. *?l x* ≤ *?r x*)
proof *eventually-elim*
  case *x*: (*elim x*)
  then have *cmod* ((*x powr s0*) ∗ *f* (− *ln x*)) ≤ *x powr s0* ∗ (*M* ∗ *exp* (*a* ∗ (− *ln x*)))
    by (*intro norm-mult-ineq*[*THEN order-trans*]) (*auto intro*!: *x*(*2*)[*THEN order-trans*])
  also have . . . = *M* ∗ *x powr* (*s0* − *a*)
    by (*simp add*: *exp-minus ln-inverse divide-simps powr-def mult-exp-exp algebra-simps*)
  finally show *?case* **.**
qed
then have ((λ*x*. *x powr s0* ∗ *f* (− *ln x*)) ⟶ *0*) (*at-right 0*)
  by (*rule Lim-null-comparison*)
    (*auto intro*!: *tendsto-eq-intros* ‹*a* < *s0*› *eventually-at-rightI zero-less-one*)
moreover have ∀ $_F$ *x in at x*. *ln x* ≤ *0* **if** *0* < *x x* < *1* **for** *x*::*real*

using *order-tendstoD*(*1*)[*OF tendsto-ident-at* ‹*0* < *x*›, *of UNIV*]
order-tendstoD(*2*)[*OF tendsto-ident-at* ‹*x* < *1* ›, *of UNIV*]
by *eventually-elim simp*
ultimately have [*continuous-intros*]:
continuous-on {*0*..*1*} (λ*x*. *x powr s0* ∗ *f* (− *ln x*))
by (*intro continuous-on-IccI*;
force intro!: *continuous-on-tendsto-compose*[*OF cont-f*] *tendsto-eq-intros* *eventually-at-leftI*
zero-less-one)
{
fix *n*::*nat*
let *?i* = (λ*u*. *u* ^ *n* $*_R$ (*u powr s0* ∗ *f* (− *ln u*)))
let *?I* = λ*n b*. *integral* {*exp* (− *b*).. *1*} *?i*
have $\forall_F$ (*b*::*real*) *in at-top*. *b* > *0*
by (*simp add*: *eventually-gt-at-top*)
then have $\forall_F$ *b in at-top*. *I* (*s0* + *Suc n*) *b* = *?I n b*
proof *eventually-elim*
case (*elim b*)
have *eq*: *exp* (*t* $*_R$ − *complex-of-real* (*s0* + *real* (*Suc n*))) ∗ *f t* =
complex-of-real (*exp* (− (*real n* ∗ *t*)) ∗ *exp* (− *t*) ∗ *exp* (− (*s0* ∗ *t*))) ∗ *f t*
for *t*
by (*auto simp*: *Euler mult-exp-exp algebra-simps simp del*: *of-real-mult*)
from *int*[*of s0* + *Suc n b*]
have *int′*: ((λ*t*. *exp* (− (*n* ∗ *t*)) ∗ *exp* (−*t*) ∗ *exp* (− (*s0* ∗ *t*)) ∗ *f t*)
has-integral I (*s0* + *Suc n*) *b*) {*0*..*b*}
(**is** (*?fe has-integral* -) -)
unfolding *eq* **.**
have ((λ*x*. − *exp* (− *x*) $*_R$ *exp* (− *x*) ^ *n* $*_R$ (*exp* (− *x*) *powr s0* ∗ *f* (− *ln* (*exp* (− *x*)))))
has-integral
integral {*exp* (− *0*)..*exp* (− *b*)} *?i* − *integral* {*exp* (− *b*)..*exp* (− *0*)} *?i*)
{*0*..*b*}
by (*rule has-integral-substitution-general*[*of* {} *0 b* λ*t*. *exp*(−*t*) *0 1 ?i* λ*x*. −*exp*(−*x*)])
(*auto intro*!: *less-imp-le*[*OF* ‹*b* > *0*›] *continuous-intros integrable-continuous-real*
derivative-eq-intros)
then have (*?fe has-integral ?I n b*) {*0*..*b*}
using ‹*b* > *0*›
by (*auto simp*: *algebra-simps mult-exp-exp exp-of-nat-mult*[*symmetric*] *scaleR-conv-of-real*
exp-add powr-def of-real-exp has-integral-neg-iff)
with *int′* **show** *?case*
by (*rule has-integral-unique*)
qed
moreover have (*I* (*s0* + *Suc n*) ⟶ *0*) *at-top*
by (*rule imp*) (*use* ‹*s0* > *a*› **in** *auto*)
ultimately have (*?I n* ⟶ *0*) *at-top*
by (*rule Lim-transform-eventually*[*rotated*])
then have *1*: ((λ*x*. *integral* {*exp* (*ln x*)..*1*} *?i*) ⟶ *0*) (*at-right 0*)

```
      unfolding at-top-mirror filtermap-ln-at-right[symmetric] filtermap-filtermap
filterlim-filtermap
      by simp
    have ∀F x in at-right 0. x > 0
      by (simp add: eventually-at-filter)
    then have ∀F x in at-right 0. integral {exp (ln x)..1} ?i = integral {x .. 1} ?i
      by eventually-elim (auto simp:)
    from Lim-transform-eventually[OF 1 this]
    have ((λx. integral {x..1} ?i) ⟶ 0) (at-right 0)
      by simp
    moreover
    have ?i integrable-on {0..1}
      by (force intro: continuous-intros integrable-continuous-real)
    from continuous-on-Icc-at-rightD[OF indefinite-integral-continuous-1'[OF this]
zero-less-one]
    have ((λx. integral {x..1} ?i) ⟶ integral {0 .. 1} ?i) (at-right 0)
      by simp
    ultimately have integral {0 .. 1} ?i = 0
      by (rule tendsto-unique[symmetric, rotated]) simp
    then have (?i has-integral 0) {0 .. 1}
      using integrable-integral ‹?i integrable-on {0..1}›
      by (metis (full-types))
  } from lerch-lemma[OF - this, of exp (− t)]
  show f t = 0 using ‹t ≥ 0›
    by (auto intro!: continuous-intros)
qed

lemma exponential-order-eventually-eq: exponential-order M a f
  if exponential-order M a g ⋀t. t ≥ k ⟹ f t = g t
proof −
  have ∀F t in at-top. f t = g t
    using that
    unfolding eventually-at-top-linorder
    by blast
  with exponential-orderD(2)[OF that(1)]
  have (∀F t in at-top. norm (f t) ≤ M ∗ exp (a ∗ t))
    by eventually-elim auto
  with exponential-orderD(1)[OF that(1)]
  show ?thesis
    by (rule exponential-orderI)
qed

lemma exponential-order-mono:
  assumes eo: exponential-order M a f
  assumes a ≤ b M ≤ N
  shows exponential-order N b f
proof (rule exponential-orderI)
  from exponential-orderD[OF eo] assms(3)
  show 0 < N by simp
```

have $\forall_F$ *t in at-top.* $(t::real) > 0$
  by (*simp add*: *eventually-gt-at-top*)
then have $\forall_F$ *t in at-top.* $M * exp\ (a * t) \leq N * exp\ (b * t)$
  by *eventually-elim*
    (*use* ‹*0* < *N*› **in** ‹*force intro*: *mult-mono assms*›)
with *exponential-orderD*(*2*)[*OF eo*]
show $\forall_F$ *t in at-top. norm* $(f\ t) \leq N * exp\ (b * t)$
  by (*eventually-elim*) *simp*
qed

lemma *exponential-order-uminus-iff*:
  exponential-order M a $(\lambda x. - f\ x)$ = *exponential-order M a f*
  by (*auto simp*: *exponential-order-def*)

lemma *exponential-order-add*:
  assumes *exponential-order M a f exponential-order M a g*
  shows *exponential-order* $(2 * M)$ *a* $(\lambda x. f\ x + g\ x)$
  using *assms*
  apply (*auto simp*: *exponential-order-def*)
  subgoal premises *prems*
    using *prems*(*1*,*3*)
    apply (*eventually-elim*)
    apply (*rule norm-triangle-le*)
    by *linarith*
  done

theorem *laplace-transform-unique*:
  assumes *f*: $\bigwedge s.\ Re\ s > a \Longrightarrow (f\ has\text{-}laplace\ F)\ s$
  assumes *g*: $\bigwedge s.\ Re\ s > b \Longrightarrow (g\ has\text{-}laplace\ F)\ s$
  assumes [*continuous-intros*]: *continuous-on* {*0*..} *f*
  assumes [*continuous-intros*]: *continuous-on* {*0*..} *g*
  assumes *eof*: *exponential-order M a f*
  assumes *eog*: *exponential-order N b g*
  assumes $t \geq 0$
  shows $f\ t = g\ t$
proof −
  define *c* **where** *c* = *max a b*
  define *L* **where** *L* = *max M N*
  from *eof* **have** *eof*: *exponential-order L c f*
    by (*rule exponential-order-mono*) (*auto simp*: *L-def c-def*)
  from *eog* **have** *eog*: *exponential-order L c* $(\lambda x. - g\ x)$
    unfolding *exponential-order-uminus-iff*
    by (*rule exponential-order-mono*) (*auto simp*: *L-def c-def*)
  from *exponential-order-add*[*OF eof eog*]
  have *eom*: *exponential-order* $(2 * L)$ *c* $(\lambda x. f\ x - g\ x)$
    by *simp*
  have *l0*: $((\lambda x. f\ x - g\ x)\ has\text{-}laplace\ 0)\ s$ **if** $Re\ s > c$ **for** *s*
    using *has-laplace-minus*[*OF f g*, *of s*] *that* **by** (*simp add*: *c-def max-def split*: *if-splits*)

have *f t* − *g t* = *0*
by (*rule laplace-transform-zero*[*OF* - *eom l0* ‹*t* ≥ *0*›])
(*auto intro*!: *continuous-intros*)
then show *?thesis* **by** *simp*
qed

end
theory *Laplace-Transform*
imports
Existence
Uniqueness
begin

end

References

[1] A. Rashid and O. Hasan. Formalization of transform methods using HOLăLight. In H. Geuvers, M. England, O. Hasan, F. Rabe, and O. Teschke, editors, *Intelligent Computer Mathematics*, pages 319–332, Cham, 2017. Springer International Publishing.

[2] A. Rashid and O. Hasan. Formalization of Lerch's theorem using HOL Light. *FLAP*, 5(8):1623–1652, 2018.

[3] J. L. Schiff. *The Laplace transform: theory and applications*. Springer New York, 1999.

[4] S. H. Taqdees and O. Hasan. Formalization of Laplace transform using the multivariable calculus theory of HOL-Light. In K. McMillan, A. Middeldorp, and A. Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 744–758, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.