Fourier Series

Lawrence C Paulson

Abstract

This development formalises the square integrable functions over the reals and the basics of Fourier series. It culminates with a proof that every well-behaved periodic function can be approximated by a Fourier series. The material is ported from HOL Light.[1]

Contents

[1]https://github.com/jrh13/hol-light/blob/master/100/fourier.ml

1 Shifting the origin for integration of periodic functions

theory *Periodic*
  imports *HOL−Analysis.Analysis*
begin

lemma *has-bochner-integral-null* [*intro*]:
  fixes f :: *'a::euclidean-space* ⇒ *'b::euclidean-space*
  assumes N ∈ *null-sets lebesgue*
  shows *has-bochner-integral* (*lebesgue-on* N) f *0*
  unfolding *has-bochner-integral-iff*
proof
  show *integrable* (*lebesgue-on* N) f
  proof (*subst integrable-restrict-space*)
    show N ∩ *space lebesgue* ∈ *sets lebesgue*
      using *assms* **by** *force*
    show *integrable lebesgue* (λ*x*. *indicat-real* N x $*_R$ f x)
    proof (*rule integrable-cong-AE-imp*)
      show *integrable lebesgue* (λ*x*. *0*)
        by *simp*
      show ∗: *AE* x *in lebesgue*. *0* = *indicat-real* N x $*_R$ f x
        using *assms*
        by (*simp add*: *indicator-def completion.null-sets-iff-AE eventually-mono*)
      show (λ*x*. *indicat-real* N x $*_R$ f x) ∈ *borel-measurable lebesgue*
        by (*auto intro*: *borel-measurable-AE* [*OF* - ∗])
    qed
  qed
  show *integral*^{L} (*lebesgue-on* N) f = *0*
  proof (*rule integral-eq-zero-AE*)
    show *AE* x *in lebesgue-on* N. f x = *0*

```
    by (rule AE-I' [where N=N]) (auto simp: assms null-setsD2 null-sets-restrict-space)
  qed
qed

lemma has-bochner-integral-null-eq[simp]:
  fixes f :: 'a::euclidean-space ⇒ 'b::euclidean-space
  assumes N ∈ null-sets lebesgue
  shows has-bochner-integral (lebesgue-on N) f i ⟷ i = 0
  using assms has-bochner-integral-eq by blast

lemma periodic-integer-multiple:
   (∀x. f(x + a) = f x) ⟷ (∀x. ∀n ∈ ℤ. f(x + n * a) = f x) (is ?lhs = ?rhs)
proof
  assume L [rule-format]: ?lhs
  have nat: f(x + of-nat n * a) = f x for x n
  proof (induction n)
    case (Suc n)
    with L [of x + of-nat n * a] show ?case
      by (simp add: algebra-simps)
  qed auto
  have f(x + of-int z * a) = f x for x z
  proof (cases z ≥ 0)
    case True
    then show ?thesis
      by (metis nat of-nat-nat)
  next
    case False
    then show ?thesis
      using nat [of x + of-int z * a nat (−z)] by auto
  qed
  then show ?rhs
    by (auto simp: Ints-def)
qed (use Ints-1 in fastforce)

lemma has-integral-offset:
  fixes f :: real ⇒ 'a::euclidean-space
  assumes has-bochner-integral (lebesgue-on {a..b}) f i
  shows has-bochner-integral (lebesgue-on {a−c..b−c}) (λx. f(x + c)) i
proof −
  have eq: indicat-real {a..b} (x + c) = indicat-real {a−c..b−c} x for x
    by (auto simp: indicator-def)
  have has-bochner-integral lebesgue (λx. indicator {a..b} x *R f x) i
    using assms  by (auto simp: has-bochner-integral-restrict-space)
   then have has-bochner-integral lebesgue (λx. indicat-real {a−c..b−c} x *R f(x
+ c)) i
    using has-bochner-integral-lebesgue-real-affine-iff [of 1 (λx. indicator {a..b} x
*R f x) i c]
    by (simp add: add-ac eq)
  then show ?thesis
```

using *assms* **by** (*auto simp*: *has-bochner-integral-restrict-space*)
qed

lemma *has-integral-periodic-offset-lemma*:
fixes *f* :: *real* ⇒ *'a*::*euclidean-space*
assumes *periodic*: $\bigwedge x.\ f(x + (b-a)) = f\ x$ **and** *f*: *has-bochner-integral* (*lebesgue-on* {*a*..*a*+*c*}) *f i*
shows *has-bochner-integral* (*lebesgue-on* {*b*..*b*+*c*}) *f i*
proof −
have $f(x + a-b) = f\ x$ **for** *x*
using *periodic* [*of* $x + (a-b)$] **by** (*simp add*: *algebra-simps*)
then show *?thesis*
using *has-integral-offset* [*OF f*, *of* $a-b$]
by (*auto simp*: *algebra-simps*)
qed

lemma *has-integral-periodic-offset-pos*:
fixes *f* :: *real* ⇒ *real*
assumes *f*: *has-bochner-integral* (*lebesgue-on* {*a*..*b*}) *f i* **and** *periodic*: $\bigwedge x.\ f(x + (b-a)) = f\ x$
and *c*: $c \geq 0$ $a + c \leq b$
shows *has-bochner-integral* (*lebesgue-on* {*a*..*b*}) ($\lambda x.\ f(x + c)$) *i*
proof −
have {*a*..*a* + *c*} ⊆ {*a*..*b*}
by (*simp add*: *assms*(*4*))
then have *1*: *has-bochner-integral* (*lebesgue-on* {*a*..*a*+*c*}) *f*(*integral*^{L} (*lebesgue-on* {*a*..*a*+*c*}) *f*)
by (*meson integrable-subinterval f has-bochner-integral-iff*)
then have *3*: *has-bochner-integral* (*lebesgue-on* {*b*..*b*+*c*}) *f*(*integral*^{L} (*lebesgue-on* {*a*..*a*+*c*}) *f*)
using *has-integral-periodic-offset-lemma periodic* **by** *blast*
have {*a* + *c*..*b*} ⊆ {*a*..*b*}
by (*simp add*: *c*)
then have *2*: *has-bochner-integral* (*lebesgue-on* {*a*+*c*..*b*}) *f*(*integral*^{L} (*lebesgue-on* {*a*+*c*..*b*}) *f*)
by (*meson integrable-subinterval f has-bochner-integral-integrable integrable.intros*)
have *integral*^{L} (*lebesgue-on* {*a* + *c*..*b*}) *f* + *integral*^{L} (*lebesgue-on* {*a*..*a* + *c*}) *f* = *i*
by (*metis integral-combine add.commute c f has-bochner-integral-iff le-add-same-cancel1*)
then have *has-bochner-integral* (*lebesgue-on* {*a*+*c*..*b*+*c*}) *f i*
using *has-bochner-integral-combine* [*OF* - - *2 3*] *1 2* **by** (*simp add*: *c*)
then show *?thesis*
by (*metis add-diff-cancel-right' has-integral-offset*)
qed

lemma *has-integral-periodic-offset-weak*:

fixes f :: *real* ⇒ *real*
assumes f: *has-bochner-integral* (*lebesgue-on* $\{a..b\}$) f i **and** *periodic*: $\bigwedge x.\ f(x + (b-a)) = f\ x$ **and** c: $|c| \leq b-a$
shows *has-bochner-integral* (*lebesgue-on* $\{a..b\}$) $(\lambda x.\ f(x + c))$ i
proof (*cases* $c \geq 0$)
case *True*
then show *?thesis*
using *c f has-integral-periodic-offset-pos periodic* **by** *auto*
next
case *False*
have *per'*: $f(-\ (x + (-\ a - -\ b))) = f(-\ x)$ **for** x
using *periodic* [*of* $(a-b)-x$] **by** *simp*
have f': *has-bochner-integral* (*lebesgue-on* $\{-\ b..-\ a\}$) $(\lambda x.\ f(-\ x))$ i
using f **by** *blast*
show *?thesis*
using *has-integral-periodic-offset-pos* [*of* $-b$ $-a$ $\lambda x.\ f(-x)$ i $-c$, *OF* f' *per'*] c *False*
by (*simp flip*: *has-bochner-integral-reflect-real* [*of* b a])
qed

lemma *has-integral-periodic-offset*:
fixes f :: *real* ⇒ *real*
assumes f: *has-bochner-integral* (*lebesgue-on* $\{a..b\}$) f i **and** *periodic*: $\bigwedge x.\ f(x + (b-a)) = f\ x$
shows *has-bochner-integral* (*lebesgue-on* $\{a..b\}$) $(\lambda x.\ f(x + c))$ i
proof −
consider $b \leq a \mid a < b$ **by** *linarith*
then show *?thesis*
proof *cases*
case *1*
then have $\{a..b\} \in$ *null-sets lebesgue*
using *less-eq-real-def* **by** *auto*
with f **show** *?thesis*
by *auto*
next
case *2*
define *fba* **where** $fba \equiv \lambda x.\ f(x + (b-a) * frac(c\ /\ (b-a)))$
have *eq*: $fba\ x = f(x + c)$
if $x \in \{a..b\}$ **for** x
proof −
have $f(x + n * (b-a)) = f\ x$ **if** $n \in \mathbb{Z}$ **for** $n\ x$
using *periodic periodic-integer-multiple that* **by** *blast*
then have $f((x + c) + -\ floor\ (c\ /\ (b - a)) * (b - a)) = f(x + c)$
using *Ints-of-int* **by** *blast*
moreover have $((x + c) + -floor\ (c\ /\ (b - a)) * (b - a)) = (x + (b - a) * frac\ (c\ /\ (b - a)))$
using *2* **by** (*simp add*: *field-simps frac-def*)
ultimately show *?thesis*
unfolding *fba-def* **by** *metis*

qed
have ∗: *has-bochner-integral* (*lebesgue-on* {*a*..*b*}) *fba i*
unfolding *fba-def*
proof (*rule has-integral-periodic-offset-weak* [*OF f*])
show *f*(*x* + (*b* − *a*)) = *f x* **for** *x*
by *fact*
have |*frac* (*c* / (*b* − *a*))| ≤ *1*
using *frac-unique-iff less-eq-real-def* **by** *auto*
then show |(*b* − *a*) ∗ *frac* (*c* / (*b* − *a*))| ≤ *b* − *a*
using *2* **by** *auto*
qed
then show *?thesis*
proof (*rule has-bochner-integralI-AE* [*OF - - AE-I2*])
have *fba* ∈ *borel-measurable* (*lebesgue-on* {*a*..*b*})
using ∗ *borel-measurable-has-bochner-integral* **by** *blast*
then show (λ*x*. *f*(*x* + *c*)) ∈ *borel-measurable* (*lebesgue-on* {*a*..*b*})
by (*subst measurable-lebesgue-cong* [*OF eq*, *symmetric*])
qed (*auto simp*: *eq*)
qed
qed

lemma *integrable-periodic-offset*:
fixes *f* :: *real* ⇒ *real*
assumes *f*: *integrable* (*lebesgue-on* {*a*..*b*}) *f* **and** *periodic*: ⋀*x*. *f*(*x* + (*b*−*a*)) = *f x*
shows *integrable* (*lebesgue-on* {*a*..*b*}) (λ*x*. *f*(*x* + *c*))
using *f has-integral-periodic-offset periodic*
by (*simp add*: *has-bochner-integral-iff*)

lemma *absolutely-integrable-periodic-offset*:
fixes *f* :: *real* ⇒ *real*
assumes *f*: *f absolutely-integrable-on* {*a*..*b*} **and** *periodic*: ⋀*x*. *f*(*x* + (*b*−*a*)) = *f x*
shows (λ*x*. *f*(*x* + *c*)) *absolutely-integrable-on* {*a*..*b*} (λ*x*. *f*(*c* + *x*)) *absolutely-integrable-on* {*a*..*b*}
using *assms integrable-periodic-offset* [*of a b f*]
by (*auto simp*: *integrable-restrict-space set-integrable-def add*.*commute* [*of c*])

lemma *integral-periodic-offset*:
fixes *f* :: *real* ⇒ *real*
assumes *periodic*: ⋀*x*. *f*(*x* + (*b*−*a*)) = *f x*
shows $integral^L$ (*lebesgue-on* {*a*..*b*}) (λ*x*. *f*(*x* + *c*)) = $integral^L$ (*lebesgue-on* {*a*..*b*}) *f*
proof (*cases integrable* (*lebesgue-on* {*a*..*b*}) *f*)
case *True*
then show *?thesis*
using *has-integral-periodic-offset periodic*
by (*simp add*: *has-bochner-integral-iff*)
next

case *False*
then have ¬ *integrable* (*lebesgue-on* {*a*..*b*}) (λ*x*. *f*(*x* + *c*))
using *periodic*[*of* -+*c*]
by (*auto simp*: *algebra-simps intro*: *dest*: *integrable-periodic-offset* [**where** *c* = −*c*])
then have $integral^L$ (*lebesgue-on* {*a*..*b*}) *f* = *0* $integral^L$ (*lebesgue-on* {*a*..*b*}) (λ*x*. *f*(*x* + *c*)) = *0*
using *False not-integrable-integral-eq* **by** *blast*+
then show *?thesis*
by *simp*
qed

end

2 Lspace as it is in HOL Light

Mainly a repackaging of existing material from Lp

theory *Lspace*
imports *Lp.Lp*
begin

abbreviation *lspace* :: ′*a measure* ⇒ *ennreal* ⇒ (′*a* ⇒ *real*) *set*
where *lspace M p* ≡ $space_N$ (𝔏 *p M*)

lemma *lspace-1*:
assumes *S* ∈ *sets lebesgue*
shows *f* ∈ *lspace* (*lebesgue-on S*) *1* ⟷ *f absolutely-integrable-on S*
using *assms* **by** (*simp add*: *L1-space integrable-restrict-space set-integrable-def*)

lemma *lspace-ennreal-iff*:
assumes *p* > *0*
shows *lspace* (*lebesgue-on S*) (*ennreal p*) = {*f* ∈ *borel-measurable* (*lebesgue-on S*). *integrable* (*lebesgue-on S*) (λ*x*. (*norm*(*f x*) *powr p*))}
using *assms* **by** (*fastforce simp*: *Lp-measurable Lp-D intro*: *Lp-I*)

lemma *lspace-iff*:
assumes ∞ > *p p* > *0*
shows *lspace* (*lebesgue-on S*) *p* = {*f* ∈ *borel-measurable* (*lebesgue-on S*). *integrable* (*lebesgue-on S*) (λ*x*. (*norm*(*f x*) *powr* (*enn2real p*)))}
proof −
obtain *q*::*real* **where** *p* = *enn2real q*
using *Lp-cases assms* **by** *auto*
then show *?thesis*
using *assms lspace-ennreal-iff* **by** *auto*
qed

lemma *lspace-iff′*:
assumes *p*: ∞ > *p p* > *0* **and** *S*: *S* ∈ *sets lebesgue*

```
  shows lspace (lebesgue-on S) p = {f ∈ borel-measurable (lebesgue-on S). (λx.
(norm(f x) powr (enn2real p))) integrable-on S}
    (is ?lhs = ?rhs)
proof
  show ?lhs ⊆ ?rhs
    using assms integrable-on-lebesgue-on by (auto simp: lspace-iff)
next
  show ?rhs ⊆ ?lhs
  proof (clarsimp simp: lspace-iff [OF p])
    show integrable (lebesgue-on S) (λxa. |f xa| powr enn2real p)
        if f ∈ borel-measurable (lebesgue-on S) and (λx. |f x| powr enn2real p)
integrable-on S for f
    proof −
      have (λx. |f x| powr enn2real p) absolutely-integrable-on S
        by (simp add: absolutely-integrable-on-iff-nonneg that(2))
      then show ?thesis
        using L1-space S lspace-1 by blast
    qed
  qed
qed

lemma lspace-mono:
  assumes f ∈ lspace (lebesgue-on S) q and S: S ∈ lmeasurable and p > 0 p ≤ q
q < ∞
  shows f ∈ lspace (lebesgue-on S) p
proof −
  have p < ∞
    using assms by (simp add: top.not-eq-extremum)
  with assms show ?thesis
  proof (clarsimp simp add: lspace-iff')
    show (λx. |f x| powr enn2real p) integrable-on S
      if f ∈ borel-measurable (lebesgue-on S)
        and (λx. |f x| powr enn2real q) integrable-on S
    proof (rule measurable-bounded-by-integrable-imp-integrable)
      show (λx. |f x| powr enn2real p) ∈ borel-measurable (lebesgue-on S)
        using measurable-abs-powr that(1) by blast
      let ?g = λx. max 1 (norm(f x) powr enn2real q)
      have ?g absolutely-integrable-on S
      proof (rule absolutely-integrable-max-1)
        show (λx. norm (f x) powr enn2real q) absolutely-integrable-on S
          by (simp add: nonnegative-absolutely-integrable-1 that(2))
      qed (simp add: S)
      then show ?g integrable-on S
        using absolutely-integrable-abs-iff by blast
      show norm (|f x| powr enn2real p) ≤ ?g x if x ∈ S for x
      proof −
        have |f x| powr enn2real p ≤ |f x| powr enn2real q if 1 ≤ |f x|
          using assms enn2real-mono powr-mono that by auto
        then show ?thesis
```

```
        using powr-le1 by (fastforce simp add: le-max-iff-disj)
      qed
      show S ∈ sets lebesgue
        by (simp add: S fmeasurableD)
    qed
  qed
qed

lemma lspace-inclusion:
  assumes S ∈ lmeasurable and p > 0 p ≤ q q < ∞
  shows lspace (lebesgue-on S) q ⊆ lspace (lebesgue-on S) p
  using assms lspace-mono by auto

lemma lspace-const:
  fixes p::real
  assumes p > 0S ∈ lmeasurable
  shows (λx. c) ∈ lspace (lebesgue-on S) p
 by (simp add: Lp-space assms finite-measure.integrable-const finite-measure-lebesgue-on)

lemma lspace-max:
  fixes p::real
  assumes f ∈ lspace (lebesgue-on S) p g ∈ lspace (lebesgue-on S) p p > 0
  shows (λx. max (f x) (g x)) ∈ lspace (lebesgue-on S) p
proof −
  have integrable (lebesgue-on S) (λx. |max (f x) (g x)| powr p)
    if f: f ∈ borel-measurable (lebesgue-on S) integrable (lebesgue-on S) (λx. |f x|
powr p)
     and g: g ∈ borel-measurable (lebesgue-on S) integrable (lebesgue-on S) (λx. |g
x| powr p)
  proof −
    have integrable (lebesgue-on S) (λx. ||f x| powr p|) integrable (lebesgue-on S)
(λx. ||g x| powr p|)
      using integrable-abs that by blast+
    then have integrable (lebesgue-on S) (λx. max (||f x| powr p|) (||g x| powr p|))
      using integrable-max by blast
    then show ?thesis
    proof (rule Bochner-Integration.integrable-bound)
      show (λx. |max (f x) (g x)| powr p) ∈ borel-measurable (lebesgue-on S)
        using borel-measurable-max measurable-abs-powr that by blast
    qed auto
  qed
  then show ?thesis
    using assms by (auto simp: Lp-space borel-measurable-max)
qed

lemma lspace-min:
  fixes p::real
  assumes f ∈ lspace (lebesgue-on S) p g ∈ lspace (lebesgue-on S) p p > 0
  shows (λx. min (f x) (g x)) ∈ lspace (lebesgue-on S) p
```

proof −
have *integrable* (*lebesgue-on S*) (λ*x*. |*min* (*f x*) (*g x*)| *powr p*)
if *f*: *f* ∈ *borel-measurable* (*lebesgue-on S*) *integrable* (*lebesgue-on S*) (λ*x*. |*f x*| *powr p*)
and *g*: *g* ∈ *borel-measurable* (*lebesgue-on S*) *integrable* (*lebesgue-on S*) (λ*x*. |*g x*| *powr p*)
proof −
have *integrable* (*lebesgue-on S*) (λ*x*. ||*f x*| *powr p*|) *integrable* (*lebesgue-on S*) (λ*x*. ||*g x*| *powr p*|)
using *integrable-abs that* **by** *blast*+
then have *integrable* (*lebesgue-on S*) (λ*x*. *max* (||*f x*| *powr p*|) (||*g x*| *powr p*|))
using *integrable-max* **by** *blast*
then show *?thesis*
proof (*rule Bochner-Integration.integrable-bound*)
show (λ*x*. |*min* (*f x*) (*g x*)| *powr p*) ∈ *borel-measurable* (*lebesgue-on S*)
using *borel-measurable-min measurable-abs-powr that* **by** *blast*
qed *auto*
qed
then show *?thesis*
using *assms* **by** (*auto simp*: *Lp-space borel-measurable-min*)
qed

lemma *Lp-space-numeral*:
assumes *numeral n* > (*0*::*int*)
shows $space_N$ (𝔏 (*numeral n*) *M*) = {*f* ∈ *borel-measurable M*. *integrable M* (λ*x*. |*f x*| ^ *numeral n*)}
using *Lp-space* [*of numeral n M*] *assms* **by** *simp*

end

3 Square integrable functions over the reals

theory *Square-Integrable*
imports *Lspace*
begin

3.1 Basic definitions

definition *square-integrable*:: (*real* ⇒ *real*) ⇒ *real set* ⇒ *bool* (**infixr** ‹*square′-integrable*› *46*)
where *f square-integrable S* ≡ *S* ∈ *sets lebesgue* ∧ *f* ∈ *borel-measurable* (*lebesgue-on S*) ∧ *integrable* (*lebesgue-on S*) (λ*x*. *f x* ^ *2*)

lemma *square-integrable-imp-measurable*:
f square-integrable S ⟹ *f* ∈ *borel-measurable* (*lebesgue-on S*)
by (*simp add*: *square-integrable-def*)

lemma *square-integrable-imp-lebesgue*:
f square-integrable S ⟹ *S* ∈ *sets lebesgue*

by (*simp add*: *square-integrable-def*)

lemma *square-integrable-imp-lspace*:
assumes *f square-integrable S* **shows** *f* ∈ *lspace* (*lebesgue-on S*) *2*
proof −
have (λ*x*. (*f x*)^2) *absolutely-integrable-on S*
by (*metis assms integrable-on-lebesgue-on nonnegative-absolutely-integrable-1 square-integrable-def zero-le-power2*)
moreover have *S* ∈ *sets lebesgue*
using *assms square-integrable-def* **by** *blast*
ultimately show *?thesis*
by (*simp add*: *assms Lp-space-numeral integrable-restrict-space set-integrable-def square-integrable-imp-measurable*)
qed

lemma *square-integrable-iff-lspace*:
assumes *S* ∈ *sets lebesgue*
shows *f square-integrable S* ⟷ *f* ∈ *lspace* (*lebesgue-on S*) *2* (**is** *?lhs* = *?rhs*)
proof
assume *L*: *?lhs*
then show *?rhs*
using *square-integrable-imp-lspace* **by** *blast*
next
assume *?rhs* **then show** *?lhs*
using *assms* **by** (*auto simp*: *Lp-space-numeral square-integrable-def integrable-on-lebesgue-on*)
qed

lemma *square-integrable-0* [*simp*]:
S ∈ *sets lebesgue* ⟹ (λ*x*. *0*) *square-integrable S*
by (*simp add*: *square-integrable-def power2-eq-square integrable-0*)

lemma *square-integrable-neg-eq* [*simp*]:
(λ*x*. −(*f x*)) *square-integrable S* ⟷ *f square-integrable S*
by (*auto simp*: *square-integrable-def*)

lemma *square-integrable-lmult* [*simp*]:
assumes *f square-integrable S*
shows (λ*x*. *c* ∗ *f x*) *square-integrable S*
proof (*simp add*: *square-integrable-def*, *intro conjI*)
have *f*: *f* ∈ *borel-measurable* (*lebesgue-on S*) *integrable* (*lebesgue-on S*) (λ*x*. *f x* ^ *2*)
using *assms* **by** (*simp-all add*: *square-integrable-def*)
then show (λ*x*. *c* ∗ *f x*) ∈ *borel-measurable* (*lebesgue-on S*)
using *borel-measurable-scaleR* [*of* λ*x*. *c lebesgue-on S f*] **by** *simp*
have *integrable* (*lebesgue-on S*) (λ*x*. *c*^{2} ∗ (*f x*)^2)
by (*cases c*=*0*) (*auto simp*: *f integrable-0*)
then show *integrable* (*lebesgue-on S*) (λ*x*. (*c* ∗ *f x*)^2)
by (*simp add*: *power2-eq-square mult-ac*)
show *S* ∈ *sets lebesgue*

using *assms square-integrable-def* **by** *blast*
qed

lemma *square-integrable-rmult* [*simp*]:
f square-integrable S $\Longrightarrow$ (λx. *f x* $*$ *c*) *square-integrable S*
using *square-integrable-lmult* [*of f S c*] **by** (*simp add*: *mult.commute*)

lemma *square-integrable-imp-absolutely-integrable-product*:
assumes *f*: *f square-integrable S* **and** *g*: *g square-integrable S*
shows (λx. *f x* $*$ *g x*) *absolutely-integrable-on S*
proof −
have *fS*: *integrable* (*lebesgue-on S*) (λr. (*f r*)^2) *integrable* (*lebesgue-on S*) (λr. (*g r*)^2)
using *assms square-integrable-def* **by** *blast+*
have *integrable* (*lebesgue-on S*) (λx. |*f x* $*$ *g x*|)
proof (*intro integrable-abs Holder-inequality* [*of 2 2*])
show *f* $\in$ *borel-measurable* (*lebesgue-on S*) *g* $\in$ *borel-measurable* (*lebesgue-on S*)
using *f g square-integrable-def* **by** *blast+*
show *integrable* (*lebesgue-on S*) (λx. |*f x*| *powr 2*) *integrable* (*lebesgue-on S*) (λx. |*g x*| *powr 2*)
using *nonnegative-absolutely-integrable-1* [*of* (λx. (*f x*)^2)] *nonnegative-absolutely-integrable-1* [*of* (λx. (*g x*)^2)]
by (*simp-all add*: *fS integrable-restrict-space set-integrable-def*)
qed *auto*
then show *?thesis*
using *assms*
by (*simp add*: *absolutely-integrable-measurable-real borel-measurable-times square-integrable-def*)
qed

lemma *square-integrable-imp-integrable-product*:
assumes *f square-integrable S g square-integrable S*
shows *integrable* (*lebesgue-on S*) (λx. *f x* $*$ *g x*)
using *absolutely-integrable-measurable assms integrable-abs-iff*
by (*metis* (*full-types*) *absolutely-integrable-measurable-real square-integrable-def square-integrable-imp-absolutely-integrable-product*)

lemma *square-integrable-add* [*simp*]:
assumes *f*: *f square-integrable S* **and** *g*: *g square-integrable S*
shows (λx. *f x* + *g x*) *square-integrable S*
unfolding *square-integrable-def*
proof (*intro conjI*)
show *S* $\in$ *sets lebesgue*
using *assms square-integrable-def* **by** *blast*
show (λx. *f x* + *g x*) $\in$ *borel-measurable* (*lebesgue-on S*)
by (*simp add*: *f g borel-measurable-add square-integrable-imp-measurable*)
show *integrable* (*lebesgue-on S*) (λx. (*f x* + *g x*)^2)
unfolding *power2-eq-square distrib-right distrib-left*
proof (*intro Bochner-Integration.integrable-add*)

show *integrable* (*lebesgue-on* *S*) ($\lambda x.$ *f* *x* $*$ *f* *x*) *integrable* (*lebesgue-on* *S*) ($\lambda x.$ *g* *x* $*$ *g* *x*)
using *f* *g* *square-integrable-imp-integrable-product* **by** *blast*+
show *integrable* (*lebesgue-on* *S*) ($\lambda x.$ *f* *x* $*$ *g* *x*) *integrable* (*lebesgue-on* *S*) ($\lambda x.$ *g* *x* $*$ *f* *x*)
using *f* *g* *square-integrable-imp-integrable-product* **by** *blast*+
qed
qed

lemma *square-integrable-diff* [*simp*]:
⟦*f* *square-integrable* *S*; *g* *square-integrable* *S*⟧ $\Longrightarrow$ ($\lambda x.$ *f* *x* $-$ *g* *x*) *square-integrable* *S*
using *square-integrable-neg-eq* *square-integrable-add* [*of* *f* *S* $\lambda x.$ $-$ (*g* *x*)] **by** *auto*

lemma *square-integrable-abs* [*simp*]:
f *square-integrable* *S* $\Longrightarrow$ ($\lambda x.$ $|f\ x|$) *square-integrable* *S*
by (*simp* *add*: *square-integrable-def* *borel-measurable-abs*)

lemma *square-integrable-sum* [*simp*]:
assumes *I*: *finite* *I* $\bigwedge i.\ i \in I \Longrightarrow$ *f* *i* *square-integrable* *S* **and** *S*: *S* $\in$ *sets* *lebesgue*
shows ($\lambda x.$ $\sum i{\in}I.$ *f* *i* *x*) *square-integrable* *S*
using *I* **by** *induction* (*simp-all* *add*: *S*)

lemma *continuous-imp-square-integrable* [*simp*]:
continuous-on {*a*..*b*} *f* $\Longrightarrow$ *f* *square-integrable* {*a*..*b*}
using *continuous-imp-integrable* [*of* *a* *b* ($\lambda x.$ $(f\ x)^2$)]
by (*simp* *add*: *square-integrable-def* *continuous-on-power* *continuous-imp-measurable-on-sets-lebesgue*)

lemma *square-integrable-imp-absolutely-integrable*:
assumes *f*: *f* *square-integrable* *S* **and** *S*: *S* $\in$ *lmeasurable*
shows *f* *absolutely-integrable-on* *S*
proof $-$
have *f* $\in$ *lspace* (*lebesgue-on* *S*) *2*
using *f* *S* *square-integrable-iff-lspace* **by** *blast*
then have *f* $\in$ *lspace* (*lebesgue-on* *S*) *1*
by (*rule* *lspace-mono*) (*use* *S* **in** *auto*)
then show *?thesis*
using *S* **by** (*simp* *flip*: *lspace-1*)
qed

lemma *square-integrable-imp-integrable*:
assumes *f*: *f* *square-integrable* *S* **and** *S*: *S* $\in$ *lmeasurable*
shows *integrable* (*lebesgue-on* *S*) *f*
by (*meson* *S* *absolutely-integrable-measurable-real* *f* *fmeasurableD* *integrable-abs-iff* *square-integrable-imp-absolutely-integrable*)

3.2 The norm and inner product in L2

definition *l2product* :: *'a*::*euclidean-space* *set* $\Rightarrow$ (*'a* $\Rightarrow$ *real*) $\Rightarrow$ (*'a* $\Rightarrow$ *real*) $\Rightarrow$ *real*

```
  where l2product S f g ≡ (∫ x. f x ∗ g x ∂(lebesgue-on S))

definition l2norm :: ['a::euclidean-space set, 'a ⇒ real] ⇒ real
  where l2norm S f ≡ sqrt (l2product S f f)

definition lnorm :: ['a measure, real, 'a ⇒ real] ⇒ real
  where lnorm M p f ≡ (∫ x. |f x| powr p ∂M) powr (1/p)

corollary Holder-inequality-lnorm:
  assumes p > (0::real) q > 0 1/p+1/q = 1
      and f ∈ borel-measurable M g ∈ borel-measurable M
          integrable M (λx. |f x| powr p)
          integrable M (λx. |g x| powr q)
  shows (∫ x. |f x ∗ g x| ∂M) ≤ lnorm M p f ∗ lnorm M q g
        |∫ x. f x ∗ g x ∂M | ≤ lnorm M p f ∗ lnorm M q g
  by (simp-all add: Holder-inequality assms lnorm-def)

lemma l2norm-lnorm: l2norm S f = lnorm (lebesgue-on S) 2 f
proof −
  have (LINT x|lebesgue-on S. (f x)^2) ≥ 0
    by simp
  then show ?thesis
   by (auto simp: lnorm-def l2norm-def l2product-def power2-eq-square powr-half-sqrt)
qed

lemma lnorm-nonneg: lnorm M p f ≥ 0
  by (simp add: lnorm-def)

lemma lnorm-minus-commute: lnorm M p (g − f) = lnorm M p (f − g)
  by (simp add: lnorm-def abs-minus-commute)
```

Extending a continuous function in a periodic way

```
proposition continuous-on-compose-frac:
  fixes f:: real ⇒ real
  assumes contf: continuous-on {0..1} f and f10: f 1 = f 0
  shows continuous-on UNIV (f ∘ frac)
proof −
  have ∗: isCont (f ∘ frac) x
    if caf: ⋀x. ⟦0 ≤ x; x ≤ 1⟧ ⟹ continuous (at x within {0..1}) f for x
  proof (cases x ∈ ℤ)
    case True
    then have [simp]: frac x = 0
      by simp
    show ?thesis
    proof (clarsimp simp add: continuous-at-eps-delta dist-real-def)
      have f0: continuous (at 0 within {0..1}) f and f1: continuous (at 1 within
{0..1}) f
        by (auto intro: caf)
      show ∃ d>0. ∀ x'. |x'−x| < d ⟶ |f(frac x') − f 0| < e
```

```
      if 0 < e for e
    proof -
      obtain d0 where d0 > 0 and d0: ⋀x'. ⟦x'∈{0..1}; |x'| < d0⟧ ⟹ |f x' -
f 0| < e
        using ‹e > 0› caf [of 0] dist-not-less-zero
        by (auto simp: continuous-within-eps-delta dist-real-def)
      obtain d1 where d1 > 0 and d1: ⋀x'. ⟦x'∈{0..1}; |x' - 1| < d1⟧ ⟹ |f
x' - f 0| < e
        using ‹e > 0› caf [of 1] dist-not-less-zero f10
        by (auto simp: continuous-within-eps-delta dist-real-def)
      show ?thesis
      proof (intro exI conjI allI impI)
        show 0 < min 1 (min d0 d1)
          by (auto simp: ‹d0 > 0› ‹d1 > 0›)
        show |f(frac x') - f 0| < e
          if |x'-x| < min 1 (min d0 d1) for x'
        proof (cases x ≤ x')
          case True
          with ‹x ∈ ℤ› that have frac x' = x' - x
            by (simp add: frac-unique-iff)
          then show ?thesis
            using True d0 that by auto
        next
          case False
          then have [simp]: frac x' = 1 - (x - x')
            using that ‹x ∈ ℤ› by (simp add: not-le frac-unique-iff)
          show ?thesis
            using False d1 that by auto
        qed
      qed
    qed
  qed
next
  case False
  show ?thesis
  proof (rule continuous-at-compose)
    show isCont frac x
      by (simp add: False continuous-frac)
    have frac x ∈ {0<..<1}
      by (simp add: False frac-lt-1)
    then show isCont f(frac x)
      by (metis at-within-Icc-at greaterThanLessThan-iff le-cases not-le that)
  qed
qed
then show ?thesis
  using contf by (simp add: o-def continuous-on-eq-continuous-within)
qed
```

```
proposition Tietze-periodic-interval:
  fixes f:: real ⇒ real
  assumes contf: continuous-on {a..b} f and fab: f a = f b
  obtains g where continuous-on UNIV g ⋀x. x ∈ {a..b} ⟹ g x = f x
                  ⋀x. g(x + (b−a)) = g x
proof (cases a < b)
  case True
  let ?g = f ∘ (λy. a + (b−a) ∗ y) ∘ frac ∘
              (λx. (x − a) / (b−a))
  show ?thesis
  proof
    have a + (b − a) ∗ y ≤ b if a < b 0 ≤ y y ≤ 1 for y
      using that affine-ineq by (force simp: field-simps)
    then have ∗: continuous-on (range (λx. (x − a) / (b − a))) (f ∘ (λy. a + (b
− a) ∗ y) ∘ frac)
      apply (intro continuous-on-subset [OF continuous-on-compose-frac] continu-
ous-on-subset [OF contf]
          continuous-intros)
      using ‹a < b›
      by (auto simp: fab)
    show continuous-on UNIV ?g
      by (intro ∗ continuous-on-compose continuous-intros) (use True in auto)
    show ?g x = f x if x ∈ {a..b} for x :: real
    proof (cases x=b)
      case True
      then show ?thesis
        by (auto simp: frac-def intro: fab)
    next
      case False
      with ‹a < b› that have frac ((x − a) / (b − a)) = (x − a) / (b − a)
        by (subst frac-eq) (auto simp: divide-simps)
      with ‹a < b› show ?thesis
        by auto
    qed
    have a + (b−a) ∗ frac ((x + b − 2 ∗ a) / (b−a)) = a + (b−a) ∗ frac ((x −
a) / (b−a)) for x
      using True frac-1-eq [of (x − a) / (b−a)] by (auto simp: divide-simps)
    then show ?g (x + (b−a)) = (?g x::real) for x
      by force
  qed
next
  case False
  show ?thesis
  proof
    show f a = f x if x ∈ {a..b} for x
      using that False order-trans by fastforce
  qed auto
qed
```

3.3 Lspace stuff

lemma *eNorm-triangle-eps*:
  assumes *eNorm N* $(x' - x) < a$ *defect N* $= 1$
  obtains *e* **where** $e > 0$ $\bigwedge y.$ *eNorm N* $(y - x') < e \Longrightarrow$ *eNorm N* $(y - x) < a$
proof −
  let *?d* = *a* − *Norm N* $(x' - x)$
  have *nt*: *eNorm N* $(x' - x) < \top$
    using *assms top.not-eq-extremum* **by** *fastforce*
  with *assms* **have** *d*: *?d* > *0*
    by (*simp add*: *Norm-def diff-gr0-ennreal*)
  have [*simp*]: *ennreal* (*1* − *Norm N* $(x' - x)$) = *1* − *eNorm N* $(x' - x)$
    using *that nt* **unfolding** *Norm-def* **by** (*metis enn2real-nonneg ennreal-1 ennreal-enn2real ennreal-minus*)
  show *?thesis*
  proof
    show (*0*::*ennreal*) < *?d*
      using *d ennreal-less-zero-iff* **by** *blast*
    show *eNorm N* $(y - x) < a$
      if *eNorm N* $(y - x') <$ *?d* **for** *y*
      using *that assms eNorm-triangular-ineq* [*of N* $y - x'$ $x' - x$] *le-less-trans less-diff-eq-ennreal*
      by (*simp add*: *Norm-def nt*)
  qed
qed

lemma *topspace-topology*$_N$ [*simp*]:
  assumes *defect N* = *1* **shows** *topspace* (*topology*$_N$ *N*) = *UNIV*
proof −
  have $x \in$ *topspace* (*topology*$_N$ *N*) **for** *x*
  proof −
    have $\exists e>0. \forall y.$ *eNorm N* $(y - x') < e \longrightarrow$ *eNorm N* $(y - x) < 1$
      if *eNorm N* $(x' - x) < 1$ **for** x'
      using *eNorm-triangle-eps*
      by (*metis assms that*)
    then show *?thesis*
      unfolding *topspace-def*
    by (*rule-tac X*={*y. eNorm N* $(y - x) < 1$} **in** *UnionI*) (*auto intro*: *openin-topology*$_N$*-I*)
  qed
  then show *?thesis*
    by *auto*
qed

lemma *tendsto-ine*$_N$*-iff-limitin*:
  assumes *defect N* = *1*
  shows *tendsto-ine*$_N$ *N u x* = *limitin* (*topology*$_N$ *N*) *u x sequentially*
proof −
  have $\forall_F$ *x in sequentially. u x* $\in U$
    if *0*: ($\lambda n.$ *eNorm N* $(u\ n - x)$) $\longlonglongrightarrow$ *0* **and** *U*: *openin* (*topology*$_N$ *N*) *U x* $\in$ *U* **for** *U*

proof −
obtain *e* **where** $e > 0$ **and** *e*: $\bigwedge y.\ eNorm\ N\ (y{-}x) < e \Longrightarrow y \in U$
using *openin-topology*$_N$*-D U* **by** *metis*
then show *?thesis*
using *eventually-mono order-tendstoD*(*2*)[*OF 0*] **by** *force*
qed
moreover have $(\lambda n.\ eNorm\ N\ (u\ n - x)) \longrightarrow 0$
if *x*: $x \in topspace\ (topology_N\ N)$
and ∗: $\bigwedge U.\ [\![openin\ (topology_N\ N)\ U;\ x \in U]\!] \Longrightarrow (\forall_F\ x\ in\ sequentially.\ u\ x \in U)$
proof (*rule order-tendstoI*)
show $\forall_F\ n\ in\ sequentially.\ eNorm\ N\ (u\ n - x) < a$ **if** $a > 0$ **for** *a*
apply (*rule* ∗ [*OF openin-topology*$_N$*-I*, *of* $\{v.\ eNorm\ N\ (v - x) < a\}$, *simplified*])
using *assms eNorm-triangle-eps that* **apply** *blast*+
done
qed *simp*
ultimately show *?thesis*
by (*auto simp*: *tendsto-ine*$_N$*-def limitin-def assms*)
qed

corollary *tendsto-ine*$_N$*-iff-limitin-ge1*:
fixes *p* :: *ennreal*
assumes $p \geq 1$
shows $tendsto\text{-}ine_N\ (\mathfrak{L}\ p\ M)\ u\ x = limitin\ (topology_N\ (\mathfrak{L}\ p\ M))\ u\ x\ sequentially$
proof (*rule tendsto-ine*$_N$*-iff-limitin*)
show $defect\ (\mathfrak{L}\ p\ M) = 1$
by (*metis* (*full-types*) *L-infinity*(*2*) *L-zero*(*2*) *Lp*(*2*) *Lp-cases assms ennreal-ge-1*)
qed

corollary *tendsto-in*$_N$*-iff-limitin*:
assumes $defect\ N = 1\ x \in space_N\ N\ \bigwedge n.\ u\ n \in space_N\ N$
shows $tendsto\text{-}in_N\ N\ u\ x = limitin\ (topology_N\ N)\ u\ x\ sequentially$
using *assms tendsto-ine*$_N$*-iff-limitin tendsto-ine-in* **by** *blast*

corollary *tendsto-in*$_N$*-iff-limitin-ge1*:
fixes *p* :: *ennreal*
assumes $p \geq 1\ x \in lspace\ M\ p\ \bigwedge n.\ u\ n \in lspace\ M\ p$
shows $tendsto\text{-}in_N\ (\mathfrak{L}\ p\ M)\ u\ x = limitin\ (topology_N\ (\mathfrak{L}\ p\ M))\ u\ x\ sequentially$
proof (*rule tendsto-in*$_N$*-iff-limitin*)
show $defect\ (\mathfrak{L}\ p\ M) = 1$
by (*metis* (*full-types*) *L-infinity*(*2*) *L-zero*(*2*) *Lp*(*2*) *Lp-cases* ‹$p \geq 1$› *ennreal-ge-1*)
qed (*auto simp*: *assms*)

lemma *l2product-sym*: *l2product S f g* = *l2product S g f*
by (*simp add*: *l2product-def mult.commute*)

lemma *l2product-pos-le*:
  f square-integrable S $\Longrightarrow$ *0* $\leq$ *l2product S f f*
  by (*simp add*: *square-integrable-def l2product-def flip*: *power2-eq-square*)

lemma *l2norm-pow-2*:
  f square-integrable S $\Longrightarrow$ (*l2norm S f*) ^ *2* = *l2product S f f*
  by (*simp add*: *l2norm-def l2product-pos-le*)

lemma *l2norm-pos-le*:
  f square-integrable S $\Longrightarrow$ *0* $\leq$ *l2norm S f*
  by (*simp add*: *l2norm-def l2product-pos-le*)

lemma *l2norm-le*: (*l2norm S f* $\leq$ *l2norm S g* $\longleftrightarrow$ *l2product S f f* $\leq$ *l2product S g g*)
  by (*simp add*: *l2norm-def*)

lemma *l2norm-eq*:
  (*l2norm S f* = *l2norm S g* $\longleftrightarrow$ *l2product S f f* = *l2product S g g*)
  by (*simp add*: *l2norm-def*)

lemma *Schwartz-inequality-strong*:
  assumes *f square-integrable S g square-integrable S*
  shows *l2product S* (λx. |*f x*|) (λx. |*g x*|) $\leq$ *l2norm S f* $*$ *l2norm S g*
  using *Holder-inequality-lnorm* [*of 2 2 f lebesgue-on S g*] *assms*
  by (*simp add*: *square-integrable-def l2product-def abs-mult flip*: *l2norm-lnorm*)

lemma *Schwartz-inequality-abs*:
  assumes *f square-integrable S g square-integrable S*
  shows |*l2product S f g*| $\leq$ *l2norm S f* $*$ *l2norm S g*
proof −
  have |*l2product S f g*| $\leq$ *l2product S* (λx. |*f x*|) (λx. |*g x*|)
    unfolding *l2product-def*
  proof (*rule integral-abs-bound-integral*)
    show *integrable* (*lebesgue-on S*) (λx. *f x* $*$ *g x*) *integrable* (*lebesgue-on S*) (λx. |*f x*| $*$ |*g x*|)
      by (*simp-all add*: *assms square-integrable-imp-integrable-product*)
  qed (*simp add*: *abs-mult*)
  also have ... $\leq$ *l2norm S f* $*$ *l2norm S g*
    by (*simp add*: *Schwartz-inequality-strong assms*)
  finally show *?thesis* **.**
qed

lemma *Schwartz-inequality*:
  assumes *f square-integrable S g square-integrable S*
  shows *l2product S f g* $\leq$ *l2norm S f* $*$ *l2norm S g*
  using *Schwartz-inequality-abs assms* **by** *fastforce*

lemma *lnorm-triangle*:

assumes *f*: *f* ∈ *lspace M p* **and** *g*: *g* ∈ *lspace M p* **and** *p* ≥ *1*
shows *lnorm M p* (*λx*. *f x* + *g x*) ≤ *lnorm M p f* + *lnorm M p g*
proof −
have *p* > *0*
using *assms* **by** *linarith*
then have *integrable M* (*λx*. |*f x*| *powr p*) *integrable M* (*λx*. |*g x*| *powr p*)
by (*simp-all add*: *Lp-D*(*2*) *assms*)
moreover have *f* ∈ *borel-measurable M g* ∈ *borel-measurable M*
using *Lp-measurable f g* **by** *blast*+
ultimately show *?thesis*
unfolding *lnorm-def* **using** *Minkowski-inequality*(*2*) ‹*p* ≥ *1*› **by** *blast*
qed

lemma *lnorm-triangle-fun*:
assumes *f*: *f* ∈ *lspace M p* **and** *g*: *g* ∈ *lspace M p* **and** *p* ≥ *1*
shows *lnorm M p* (*f* + *g*) ≤ *lnorm M p f* + *lnorm M p g*
using *lnorm-triangle* [*OF assms*] **by** (*simp add*: *plus-fun-def*)

lemma *l2norm-triangle*:
assumes *f square-integrable S g square-integrable S*
shows *l2norm S* (*λx*. *f x* + *g x*) ≤ *l2norm S f* + *l2norm S g*
proof −
have *f* ∈ *lspace* (*lebesgue-on S*) *2 g* ∈ *lspace* (*lebesgue-on S*) *2*
using *assms* **by** (*simp-all add*: *square-integrable-imp-lspace*)
then show *?thesis*
using *lnorm-triangle* [*of f 2 lebesgue-on S*]
by (*simp add*: *l2norm-lnorm*)
qed

lemma *l2product-ladd*:
⟦*f square-integrable S*; *g square-integrable S*; *h square-integrable S*⟧
⟹ *l2product S* (*λx*. *f x* + *g x*) *h* = *l2product S f h* + *l2product S g h*
by (*simp add*: *l2product-def algebra-simps square-integrable-imp-integrable-product*)

lemma *l2product-radd*:
⟦*f square-integrable S*; *g square-integrable S*; *h square-integrable S*⟧
⟹ *l2product S f*(*λx*. *g x* + *h x*) = *l2product S f g* + *l2product S f h*
by (*simp add*: *l2product-def algebra-simps square-integrable-imp-integrable-product*)

lemma *l2product-ldiff*:
⟦*f square-integrable S*; *g square-integrable S*; *h square-integrable S*⟧
⟹ *l2product S* (*λx*. *f x* − *g x*) *h* = *l2product S f h* − *l2product S g h*
by (*simp add*: *l2product-def algebra-simps square-integrable-imp-integrable-product*)

lemma *l2product-rdiff*:
⟦*f square-integrable S*; *g square-integrable S*; *h square-integrable S*⟧
⟹ *l2product S f*(*λx*. *g x* − *h x*) = *l2product S f g* − *l2product S f h*
by (*simp add*: *l2product-def algebra-simps square-integrable-imp-integrable-product*)

lemma *l2product-lmult*:
  ⟦*f square-integrable S*; *g square-integrable S*⟧
  ⟹ *l2product S* (*λx*. *c* ∗ *f x*) *g* = *c* ∗ *l2product S f g*
  by (*simp add*: *l2product-def algebra-simps*)

lemma *l2product-rmult*:
  ⟦*f square-integrable S*; *g square-integrable S*⟧
  ⟹ *l2product S f*(*λx*. *c* ∗ *g x*) = *c* ∗ *l2product S f g*
  by (*simp add*: *l2product-def algebra-simps*)

lemma *l2product-lzero* [*simp*]: *l2product S* (*λx*. *0*) *f* = *0*
  by (*simp add*: *l2product-def*)

lemma *l2product-rzero* [*simp*]: *l2product S f*(*λx*. *0*) = *0*
  by (*simp add*: *l2product-def*)

lemma *l2product-lsum*:
 assumes *I*: *finite I* ⋀*i*. *i* ∈ *I* ⟹ (*f i*) *square-integrable S* **and** *S*: *g square-integrable S*
  shows *l2product S* (*λx*. ∑ *i*∈*I*. *f i x*) *g* = (∑ *i*∈*I*. *l2product S* (*f i*) *g*)
  using *I*
proof *induction*
  case (*insert i I*)
  with *S* **show** *?case*
    by (*simp add*: *l2product-ladd square-integrable-imp-lebesgue*)
qed *auto*

lemma *l2product-rsum*:
 assumes *I*: *finite I* ⋀*i*. *i* ∈ *I* ⟹ (*f i*) *square-integrable S* **and** *S*: *g square-integrable S*
  shows *l2product S g* (*λx*. ∑ *i*∈*I*. *f i x*) = (∑ *i*∈*I*. *l2product S g* (*f i*))
  using *l2product-lsum* [*OF assms*] **by** (*simp add*: *l2product-sym*)

lemma *l2norm-lmult*:
  f square-integrable S ⟹ *l2norm S* (*λx*. *c* ∗ *f x*) = |*c*| ∗ *l2norm S f*
  by (*simp add*: *l2norm-def l2product-rmult l2product-sym real-sqrt-mult*)

lemma *l2norm-rmult*:
  f square-integrable S ⟹ *l2norm S* (*λx*. *f x* ∗ *c*) = *l2norm S f* ∗ |*c*|
  using *l2norm-lmult* **by** (*simp add*: *mult.commute*)

lemma *l2norm-neg*:
  f square-integrable S ⟹ *l2norm S* (*λx*. − *f x*) = *l2norm S f*
  using *l2norm-lmult* [*of f S* −*1*] **by** *simp*

lemma *l2norm-diff*:
  assumes *f square-integrable S g square-integrable S*
  shows *l2norm S* (*λx*. *f x* − *g x*) = *l2norm S* (*λx*. *g x* − *f x*)

proof −
  have ($\lambda x.\ f\ x - g\ x$) *square-integrable S*
    using *assms square-integrable-diff* **by** *blast*
  then show *?thesis*
    using *l2norm-neg* [*of* $\lambda x.\ f\ x - g\ x\ S$] **by** (*simp add*: *algebra-simps*)
qed

3.4 Completeness (Riesz-Fischer)

lemma *eNorm-eq-lnorm*: $[\![f \in lspace\ M\ p;\ p > 0]\!] \Longrightarrow eNorm\ (\mathfrak{L}\ (ennreal\ p)\ M)\ f = ennreal\ (lnorm\ M\ p\ f)$
  by (*simp add*: *Lp-D*(*4*) *lnorm-def*)

lemma *Norm-eq-lnorm*: $[\![f \in lspace\ M\ p;\ p > 0]\!] \Longrightarrow Norm\ (\mathfrak{L}\ (ennreal\ p)\ M)\ f = lnorm\ M\ p\ f$
  by (*simp add*: *Lp-D*(*3*) *lnorm-def*)

lemma *eNorm-ge1-triangular-ineq*:
  assumes $p \geq (1::real)$
  shows $eNorm\ (\mathfrak{L}\ p\ M)\ (x + y) \leq eNorm\ (\mathfrak{L}\ p\ M)\ x + eNorm\ (\mathfrak{L}\ p\ M)\ y$
  using *eNorm-triangular-ineq* [*of* ($\mathfrak{L}$ *p M*)] *assms*
  by (*simp add*: *Lp*(*2*))

A mere repackaging of the theorem $complete_N$ ($\mathfrak{L}$ *?p ?M*), but nearly as much work again.

proposition *l2-complete*:
  assumes *f*: $\bigwedge i::nat.\ f\ i$ *square-integrable S*
    and *cauchy*: $\bigwedge e.\ 0 < e \Longrightarrow \exists N.\ \forall m \geq N.\ \forall n \geq N.\ l2norm\ S\ (\lambda x.\ f\ m\ x - f\ n\ x) < e$
  obtains *g* **where** *g square-integrable S* $((\lambda n.\ l2norm\ S\ (\lambda x.\ f\ n\ x - g\ x)) \longrightarrow 0)$
proof −
  have *finite*: $eNorm\ (\mathfrak{L}\ 2\ (lebesgue\text{-}on\ S))\ (f\ n - f\ m) < \top$ **for** *m n*
   by (*metis f infinity-ennreal-def spaceN-diff spaceN-iff square-integrable-imp-lspace*)
  have ∗: $cauchy\text{-}ine_N$ ($\mathfrak{L}$ *2* (*lebesgue-on S*)) *f*
  proof (*clarsimp simp*: $cauchy\text{-}ine_N\text{-}def$)
    show $\exists M.\ \forall n \geq M.\ \forall m \geq M.\ eNorm\ (\mathfrak{L}\ 2\ (lebesgue\text{-}on\ S))\ (f\ n - f\ m) < e$
      if $e > 0$ **for** *e*
    proof (*cases e*)
      case (*real r*)
      then have $r > 0$
        using *that* **by** *auto*
      with *cauchy* **obtain** *N*::*nat* **where** *N*: $\bigwedge m\ n.\ [\![m \geq N;\ n \geq N]\!] \Longrightarrow l2norm\ S\ (\lambda x.\ f\ n\ x - f\ m\ x) < r$
        by *blast*
      show *?thesis*
      proof (*intro exI allI impI*)
        show $eNorm\ (\mathfrak{L}\ 2\ (lebesgue\text{-}on\ S))\ (f\ n - f\ m) < e$

if $N \leq m$ $N \leq n$ **for** m n
proof −
have *fnm*: $(f\ n - f\ m) \in$ *borel-measurable* (*lebesgue-on* S)
using f **unfolding** *square-integrable-def* **by** (*blast intro*: *borel-measurable-diff'*)
have *l2norm* S ($\lambda x.\ f\ n\ x - f\ m\ x$) = *lnorm* (*lebesgue-on* S) *2* ($\lambda x.\ f\ n\ x - f\ m\ x$)
by (*metis l2norm-lnorm*)
also have ... = *Norm* ($\mathfrak{L}$ *2* (*lebesgue-on* S)) ($f\ n - f\ m$)
using *Lp-Norm* [*OF - fnm*, *of 2*] **by** (*simp add*: *lnorm-def*)
finally show *?thesis*
using N [*OF that*] *real finite*
by (*simp add*: *Norm-def*)
qed
qed
qed (*simp add*: *finite*)
qed
then obtain g **where** g: *tendsto-ine*$_N$ ($\mathfrak{L}$ *2* (*lebesgue-on* S)) f g
using *Lp-complete complete*$_N$*-def* **by** *blast*
show *?thesis*
proof
have *fng-to-0*: ($\lambda n.$ *eNorm* ($\mathfrak{L}$ *2* (*lebesgue-on* S)) ($\lambda x.\ f\ n\ x - g\ x$)) $\longrightarrow$ *0*
using g *Lp-D*(*4*) [*of 2 - lebesgue-on S*]
by (*simp add*: *tendsto-ine*$_N$*-def minus-fun-def*)
then obtain M **where** $\bigwedge n\ .\ n \geq M \Longrightarrow$ *eNorm* ($\mathfrak{L}$ *2* (*lebesgue-on* S)) ($\lambda x.\ f\ n\ x - g\ x$) $< \top$
apply (*simp add*: *lim-explicit*)
by (*metis* (*full-types*) *open-lessThan diff-self eNorm-zero lessThan-iff local.finite*)
then have *eNorm* ($\mathfrak{L}$ *2* (*lebesgue-on* S)) ($\lambda x.\ g\ x - f\ M\ x$) $< \top$
using *eNorm-uminus* [*of -* $\lambda x.\ g\ x - f$ *- x*] **by** (*simp add*: *uminus-fun-def*)
moreover have *eNorm* ($\mathfrak{L}$ *2* (*lebesgue-on* S)) ($\lambda x.\ f\ M\ x$) $< \top$
using f *square-integrable-imp-lspace* **by** (*simp add*: *spaceN-iff*)
ultimately have *eNorm* ($\mathfrak{L}$ *2* (*lebesgue-on* S)) $g < \top$
using *eNorm-ge1-triangular-ineq* [*of 2 lebesgue-on S* $g - f\ M\ f\ M$, *simplified*] *not-le top.not-eq-extremum*
by (*fastforce simp add*: *minus-fun-def*)
then have *g-space*: $g \in$ *space*$_N$ ($\mathfrak{L}$ *2* (*lebesgue-on* S))
by (*simp add*: *spaceN-iff*)
show g *square-integrable* S
unfolding *square-integrable-def*
proof (*intro conjI*)
show $g \in$ *borel-measurable* (*lebesgue-on* S)
using *Lp-measurable g-space* **by** *blast*
show $S \in$ *sets lebesgue*
using f *square-integrable-def* **by** *blast*
then show *integrable* (*lebesgue-on* S) ($\lambda x.\ (g\ x)^2$)
using *g-space square-integrable-def square-integrable-iff-lspace* **by** *blast*
qed
then have $f\ n - g \in$ *lspace* (*lebesgue-on* S) *2* **for** n

using *f spaceN-diff square-integrable-imp-lspace* **by** *blast*
with *fng-to-0* **have** (λ*n. ennreal* (*lnorm* (*lebesgue-on S*) *2* (λ*x. f n x* − *g x*))) $\longrightarrow$ *0*
by (*simp add*: *minus-fun-def flip*: *eNorm-eq-lnorm*)
then have (λ*n. lnorm* (*lebesgue-on S*) *2* (λ*x. f n x* − *g x*)) $\longrightarrow$ *0*
by (*simp add*: *ennreal-tendsto-0-iff lnorm-def*)
then show (λ*n. l2norm S* (λ*x. f n x* − *g x*)) $\longrightarrow$ *0*
using *g* **by** (*simp add*: *l2norm-lnorm lnorm-def*)
qed
qed

3.5 Approximation of functions in Lp by bounded and continuous ones

lemma *lspace-bounded-measurable*:
fixes *p*::*real*
assumes *f*: *f* ∈ *borel-measurable* (*lebesgue-on S*) **and** *g*: *g* ∈ *lspace* (*lebesgue-on S*) *p* **and** *p* > *0*
and *le*: *AE x in lebesgue-on S*. *norm* (|*f x*| *powr p*) ≤ *norm* (|*g x*| *powr p*)
shows *f* ∈ *lspace* (*lebesgue-on S*) *p*
using *assms* **by** (*auto simp*: *lspace-ennreal-iff intro*: *Bochner-Integration.integrable-bound*)

lemma *lspace-approximate-bounded*:
assumes *f*: *f* ∈ *lspace* (*lebesgue-on S*) *p* **and** *S*: *S* ∈ *lmeasurable* **and** *p* > *0 e* > *0*
obtains *g* **where** *g* ∈ *lspace* (*lebesgue-on S*) *p bounded* (*g* ‘ *S*)
lnorm (*lebesgue-on S*) *p* (*f* − *g*) < *e*
proof −
have *f-bm*: *f* ∈ *borel-measurable* (*lebesgue-on S*)
using *Lp-measurable f* **by** *blast*
let *?f* = λ*n*::*nat*. λ*x*. *max* (− *n*) (*min n* (*f x*))
have *tendsto-in*$_N$ ($\mathfrak{L}$ *p* (*lebesgue-on S*)) *?f f*
proof (*rule Lp-domination-limit*)
show $\bigwedge$*n*::*nat*. *?f n* ∈ *borel-measurable* (*lebesgue-on S*)
by (*intro f-bm borel-measurable-max borel-measurable-min borel-measurable-const*)
show *abs* ∘ *f* ∈ *lspace* (*lebesgue-on S*) *p*
using *Lp-Banach-lattice* [*OF f*] **by** (*simp add*: *o-def*)
have ∗: $\forall_F$ *n in sequentially*. *dist* (*?f n x*) (*f x*) < *e*
if *x*: *x* ∈ *space* (*lebesgue-on S*) **and** *e* > *0* **for** *x e*
proof
show *dist* (*?f n x*) (*f x*) < *e*
if *nat* ⌈|*f x*|⌉ ≤ *n* **for** *n* :: *nat*
using *that* ‹*0* < *e*› **by** (*simp add*: *dist-real-def max-def min-def abs-if split*: *if-split-asm*)
qed
then show *AE x in lebesgue-on S*. (λ*n*::*nat*. *max* (− *n*) (*min n* (*f x*))) $\longrightarrow$ *f x*
by (*blast intro*: *tendstoI*)
qed (*auto simp*: *f-bm*)

moreover
have *lspace*: *?f n* ∈ *lspace* (*lebesgue-on S*) *p* **for** *n*::*nat*
by (*intro f lspace-const lspace-min lspace-max* ‹*p* > *0*› *S*)
ultimately have (λ*n*. *lnorm* (*lebesgue-on S*) *p* (*?f n* − *f*)) ⟶ *0*
by (*simp add*: *tendsto-in*$_N$*-def Norm-eq-lnorm* ‹*p* > *0*› *f*)
with ‹*e* > *0*› **obtain** *N* **where** *N*: |*lnorm* (*lebesgue-on S*) *p* (*?f N* − *f*)| < *e*
by (*auto simp*: *LIMSEQ-iff*)
show *?thesis*
proof
have ∀ *x*∈*S*. |*max* (− *real N*) (*min* (*real N*) (*f x*))| ≤ *N*
by *auto*
then show *bounded* (*?f N* ' *S*::*real set*)
by (*force simp*: *bounded-iff*)
show *lnorm* (*lebesgue-on S*) *p* (*f* − *?f N*) < *e*
using *N* **by** (*simp add*: *lnorm-minus-commute*)
qed (*auto simp*: *lspace*)
qed

lemma *borel-measurable-imp-continuous-limit*:
fixes *h* :: *'a*::*euclidean-space* ⇒ *'b*::*euclidean-space*
assumes *h*: *h* ∈ *borel-measurable* (*lebesgue-on S*) **and** *S*: *S* ∈ *sets lebesgue*
obtains *g* **where** ⋀*n*. *continuous-on UNIV* (*g n*) *AE x in lebesgue-on S*. (λ*n*::*nat*. *g n x*) ⟶ *h x*
proof −
have *h measurable-on S*
using *S h measurable-on-iff-borel-measurable* **by** *blast*
then obtain *N g* **where** *N*: *N* ∈ *null-sets lebesgue* **and** *g*: ⋀*n*. *continuous-on UNIV* (*g n*)
and *tends*: ⋀*x*. *x* ∉ *N* ⟹ (λ*n*. *g n x*) ⟶ (*if x* ∈ *S then h x else 0*)
by (*auto simp*: *measurable-on-def negligible-iff-null-sets*)
moreover have *AE x in lebesgue-on S*. (λ*n*::*nat*. *g n x*) ⟶ *h x*
proof (*rule AE-I'*)
show *N* ∩ *S* ∈ *null-sets* (*lebesgue-on S*)
by (*simp add*: *S N null-set-Int2 null-sets-restrict-space*)
show {*x* ∈ *space* (*lebesgue-on S*). ¬ (λ*n*. *g n x*) ⟶ *h x*} ⊆ *N* ∩ *S*
using *tends* **by** *force*
qed
ultimately show *thesis*
using *that* **by** *blast*
qed

proposition *lspace-approximate-continuous*:
assumes *f*: *f* ∈ *lspace* (*lebesgue-on S*) *p* **and** *S*: *S* ∈ *lmeasurable* **and** *1* ≤ *p e* > *0*
obtains *g* **where** *continuous-on UNIV g g* ∈ *lspace* (*lebesgue-on S*) *p lnorm* (*lebesgue-on S*) *p* (*f* − *g*) < *e*
proof −
have *p* > *0*

```
    using assms by simp
  obtain h where h: h ∈ lspace (lebesgue-on S) p and bounded (h ‘ S)
    and lesse2: lnorm (lebesgue-on S) p (f − h) < e/2
    by (rule lspace-approximate-bounded [of f p S e/2]) (use assms in auto)
  then obtain B where B > 0 and B: ⋀x. x ∈ S ⟹ |h x| ≤ B
    by (auto simp: bounded-pos)
  have bmh: h ∈ borel-measurable (lebesgue-on S)
    using h lspace-ennreal-iff [of p] ‹p ≥ 1› by auto
  obtain g where contg: ⋀n. continuous-on UNIV (g n)
    and gle: ⋀n x. x ∈ S ⟹ |g n x| ≤ B
    and tends: AE x in lebesgue-on S. (λn::nat. g n x) ⟶ h x
  proof −
    obtain γ where cont: ⋀n. continuous-on UNIV (γ n)
      and tends: AE x in lebesgue-on S. (λn::nat. γ n x) ⟶ h x
      using borel-measurable-imp-continuous-limit S bmh by blast
    let ?g = λn::nat. λx. max (− B) (min B (γ n x))
    show thesis
    proof
      show continuous-on UNIV (?g n) for n
        by (intro continuous-intros cont)
      show |?g n x| ≤ B if x ∈ S  for n x
        using that ‹B > 0› by (auto simp: max-def min-def)
      have (λn. max (− B) (min B (γ n x))) ⟶ h x
        if (λn. γ n x) ⟶ h x x ∈ S for x
        using that ‹B > 0› B [OF ‹x ∈ S›]
        unfolding LIMSEQ-def by (fastforce simp: min-def max-def dist-real-def)
      then show AE x in lebesgue-on S. (λn. ?g n x) ⟶ h x
        using tends by auto
    qed
  qed
  have lspace-B: (λx. B) ∈ lspace (lebesgue-on S) p
    by (simp add: S ‹0 < p› lspace-const)
  have lspace-g: g n ∈ lspace (lebesgue-on S) p for n
  proof (rule lspace-bounded-measurable)
    show g n ∈ borel-measurable (lebesgue-on S)
      by (simp add: borel-measurable-continuous-onI contg measurable-completion
measurable-restrict-space1)
    show AE x in lebesgue-on S. norm (|g n x| powr p) ≤ norm (|B| powr p)
      using ‹B > 0› gle S ‹0 < p› powr-mono2 by auto
  qed (use ‹p > 0› lspace-B in auto)
  have tendsto-in_N (𝔏 p (lebesgue-on S)) g h
  proof (rule Lp-domination-limit [OF bmh - lspace-B tends])
    show ⋀n::nat. g n ∈ borel-measurable (lebesgue-on S)
      using Lp-measurable lspace-g by blast
    show ⋀n. AE x in lebesgue-on S. |g n x| ≤ B
      using S gle by auto
  qed
  then have 0: (λn. Norm (𝔏 p (lebesgue-on S)) (g n − h)) ⟶ 0
    by (simp add: tendsto-in_N-def)
```

have ⋀*e*. *0* < *e* ⟹ ∃ *N*. ∀ *n*≥*N*. *lnorm* (*lebesgue-on S*) *p* (*g n* − *h*) < *e*
  using *LIMSEQ-D* [*OF 0*] ‹*e* > *0*›
  by (*force simp*: *Norm-eq-lnorm* ‹*0* < *p*› *h lspace-g*)
then obtain *N* **where** *N*: *lnorm* (*lebesgue-on S*) *p* (*g N* − *h*) < *e*/*2*
  unfolding *minus-fun-def* **by** (*meson* ‹*e*>*0*› *half-gt-zero order-refl*)
show *?thesis*
proof
  show *continuous-on UNIV* (*g N*)
    by (*simp add*: *contg*)
  show *g N* ∈ *lspace* (*lebesgue-on S*) (*ennreal p*)
    by (*simp add*: *lspace-g*)
  have *lnorm* (*lebesgue-on S*) *p* (*f* − *h* + − (*g N* − *h*)) ≤ *lnorm* (*lebesgue-on S*) *p* (*f* − *h*) + *lnorm* (*lebesgue-on S*) *p* (− (*g N* − *h*))
    by (*rule lnorm-triangle-fun*) (*auto simp*: *lspace-g h assms*)
  also have ... < *e*/*2* + *e*/*2*
    using *lesse2 N* **by** (*simp add*: *lnorm-minus-commute*)
  finally show *lnorm* (*lebesgue-on S*) *p* (*f* − *g N*) < *e*
    by *simp*
qed
qed

proposition *square-integrable-approximate-continuous*:
  assumes *f*: *f square-integrable S* **and** *S*: *S* ∈ *lmeasurable* **and** *e* > *0*
  obtains *g* **where** *continuous-on UNIV g g square-integrable S l2norm S* (λ*x*. *f x* − *g x*) < *e*
proof −
  have *f2*: *f* ∈ *lspace* (*lebesgue-on S*) *2*
    by (*simp add*: *f square-integrable-imp-lspace*)
  then obtain *g* **where** *contg*: *continuous-on UNIV g*
        and *g2*: *g* ∈ *lspace* (*lebesgue-on S*) *2*
        and *less-e*: *lnorm* (*lebesgue-on S*) *2* (λ*x*. *f x* − *g x*) < *e*
    using *lspace-approximate-continuous* [*of f 2 S e*] *S* ‹*0* < *e*› **by** (*auto simp*: *minus-fun-def*)
  show *thesis*
  proof
    show *g square-integrable S*
      using *g2* **by** (*simp add*: *S fmeasurableD square-integrable-iff-lspace*)
    show *l2norm S* (λ*x*. *f x* − *g x*) < *e*
      using *less-e* **by** (*simp add*: *l2norm-lnorm*)
  qed (*simp add*: *contg*)
qed

lemma *absolutely-integrable-approximate-continuous*:
  fixes *f* :: *real* ⇒ *real*
  assumes *f*: *f absolutely-integrable-on S* **and** *S*: *S* ∈ *lmeasurable* **and** *0* < *e*
  obtains *g* **where** *continuous-on UNIV g g absolutely-integrable-on S* $integral^L$ (*lebesgue-on S*) (λ*x*. |*f x* − *g x*|) < *e*
proof −
  obtain *g* **where** *continuous-on UNIV g g* ∈ *lspace* (*lebesgue-on S*) *1*

and *lnorm*: *lnorm* (*lebesgue-on S*) *1* (*f* − *g*) < *e*
proof (*rule lspace-approximate-continuous*)
show *f* ∈ *lspace* (*lebesgue-on S*) (*ennreal 1*)
by (*simp add*: *S f fmeasurableD lspace-1*)
qed (*auto simp*: *assms*)
show *thesis*
proof
show *continuous-on UNIV g*
by *fact*
show *g absolutely-integrable-on S*
using *S* ‹*g* ∈ *lspace* (*lebesgue-on S*) *1*› *lspace-1* **by** *blast*
have ∗: (λ*x*. *f x* − *g x*) *absolutely-integrable-on S*
by (*simp add*: ‹*g absolutely-integrable-on S*› *f*)
moreover have *integrable* (*lebesgue-on S*) (λ*x*. |*f x* − *g x*|)
by (*simp add*: *L1-D*(*2*) *S* ∗ *fmeasurableD lspace-1*)
ultimately show $integral^L$ (*lebesgue-on S*) (λ*x*. |*f x* − *g x*|) < *e*
using *lnorm S* **unfolding** *lnorm-def absolutely-integrable-on-def*
by *simp*
qed
qed

end

4 Confining a series to a set

theory *Confine*
imports *Complex-Main*
begin

definition *confine* :: (′*a* ⇒ ′*b*::*zero*) ⇒ ′*a set* ⇒ ′*a* ⇒ ′*b*
where *confine f A* = (λ*x*. *if x* ∈ *A then f x else 0*)

lemma *confine-UNIV* [*simp*]: *confine f UNIV* = *f*
by (*simp add*: *confine-def*)

lemma *sum-confine-eq-Int*:
assumes *finite I*
shows *sum* (*confine f A*) *I* = (∑ *i* ∈ *I* ∩ *A*. *f i*)
proof −
have *sum f*(*I* ∩ *A*) = (∑ *a*∈*I*. *if a* ∈ *A then f a else 0*)
using *assms sum.inter-restrict* **by** *blast*
then show *?thesis*
by (*auto simp*: *confine-def*)
qed

lemma *sums-confine-add*:
fixes *f* :: *nat* ⇒ ′*a*::*real-normed-vector*
assumes *confine f N sums a confine g N sums b*
shows *confine* (λ*i*. *f i* + *g i*) *N sums* (*a*+*b*)

proof −
  have $\bigwedge n.$ (*if* $n \in N$ *then* $f\ n + g\ n$ *else* 0) = (*if* $n \in N$ *then* $f\ n$ *else* 0) + (*if* $n \in N$ *then* $g\ n$ *else* 0)
    by *simp*
  then show *?thesis*
    using *sums-add* [*of confine f N a confine g N b*] *assms*
    by (*simp add*: *confine-def*)
qed

lemma *sums-confine-minus*:
  fixes $f :: nat \Rightarrow 'a::real\text{-}normed\text{-}vector$
  shows *confine f N sums a* $\Longrightarrow$ *confine* (*uminus* $\circ$ *f*) *N sums* ($-a$)
    using *sums-minus* [*of confine f N a*]
    by (*simp add*: *confine-def if-distrib* [*of uminus*] *cong*: *if-cong*)

lemma *sums-confine-mult*:
  fixes $f :: nat \Rightarrow 'a::real\text{-}normed\text{-}algebra$
  shows *confine f N sums a* $\Longrightarrow$ *confine* ($\lambda n.\ c * f\ n$) *N sums* ($c * a$)
    using *sums-mult* [*of confine f N a c*]
    by (*simp add*: *confine-def if-distrib* [*of* (∗)*c*] *cong*: *if-cong*)

lemma *sums-confine-divide*:
  fixes $f :: nat \Rightarrow 'a::real\text{-}normed\text{-}field$
  shows *confine f N sums a* $\Longrightarrow$ *confine* ($\lambda n.\ f\ n\ /\ c$) *N sums* (a/c)
    using *sums-divide* [*of confine f N a c*]
    by (*simp add*: *confine-def if-distrib* [*of* $\lambda x.\ x/c$] *cong*: *if-cong*)

lemma *sums-confine-divide-iff*:
  fixes $f :: nat \Rightarrow 'a::real\text{-}normed\text{-}field$
  assumes $c \neq 0$
  shows *confine* ($\lambda n.\ f\ n\ /\ c$) *N sums* (a/c) $\longleftrightarrow$ *confine f N sums a*
proof
  show *confine f N sums a*
    if *confine* ($\lambda n.\ f\ n\ /\ c$) *N sums* ($a\ /\ c$)
    using *sums-confine-mult* [*OF that*, *of c*] *assms* **by** *simp*
qed (*auto simp*: *sums-confine-divide*)

lemma *sums-confine*:
  fixes $f :: nat \Rightarrow 'a::real\text{-}normed\text{-}vector$
  shows *confine f N sums l* $\longleftrightarrow$ (($\lambda n.\ \sum i \in \{..<n\} \cap N.\ f\ i$) $\longrightarrow$ l)
  by (*simp add*: *sums-def atLeast0LessThan confine-def sum.inter-restrict*)

lemma *sums-confine-le*:
  fixes $f :: nat \Rightarrow 'a::real\text{-}normed\text{-}vector$
  shows *confine f N sums l* $\longleftrightarrow$ (($\lambda n.\ \sum i \in \{..n\} \cap N.\ f\ i$) $\longrightarrow$ l)
  by (*simp add*: *sums-def-le atLeast0AtMost confine-def sum.inter-restrict*)

end

5 Lemmas possibly destined for future Isabelle releases

```
theory Fourier-Aux2
  imports HOL−Analysis.Analysis
begin

lemma integral-sin-Z [simp]:
  assumes n ∈ ℤ
  shows integral^L (lebesgue-on {−pi..pi}) (λx. sin(x ∗ n)) = 0
  proof (subst lebesgue-integral-eq-integral)
  show integrable (lebesgue-on {−pi..pi}) (λx. sin (x ∗ n))
    by (intro continuous-imp-integrable-real continuous-intros)
  show integral {−pi..pi} (λx. sin (x ∗ n)) = 0
    using assms Ints-cases integral-sin-nx by blast
qed auto

lemma integral-sin-Z' [simp]:
  assumes n ∈ ℤ
  shows integral^L (lebesgue-on {−pi..pi}) (λx. sin(n ∗ x)) = 0
  by (metis assms integral-sin-Z mult-commute-abs)

lemma integral-cos-Z [simp]:
  assumes n ∈ ℤ
  shows integral^L (lebesgue-on {−pi..pi}) (λx. cos(x ∗ n)) = (if n = 0 then 2 ∗
pi else 0)
  proof (subst lebesgue-integral-eq-integral)
  show integrable (lebesgue-on {−pi..pi}) (λx. cos (x ∗ n))
    by (intro continuous-imp-integrable-real continuous-intros)
  show integral {−pi..pi} (λx. cos (x ∗ n)) = (if n = 0 then 2 ∗ pi else 0)
    by (metis Ints-cases assms integral-cos-nx of-int-0-eq-iff)
qed auto

lemma integral-cos-Z' [simp]:
  assumes n ∈ ℤ
  shows integral^L (lebesgue-on {−pi..pi}) (λx. cos(n ∗ x)) = (if n = 0 then 2 ∗
pi else 0)
  by (metis assms integral-cos-Z mult-commute-abs)

lemma odd-even-cases [case-names 0 odd even]:
  assumes P 0 and odd: ⋀n. P(Suc (2 ∗ n)) and even: ⋀n. P(2 ∗ n + 2)
  shows P n
  by (metis nat-induct2 One-nat-def Suc-1 add-Suc-right assms(1) dvdE even odd
oddE)

end
```

6 The basics of Fourier series

Ported from HOL Light; thanks to Manuel Eberl for help with the real asymp proof methods

```
theory Fourier
  imports Periodic Square-Integrable HOL−Real-Asymp.Real-Asymp Confine Fourier-Aux2
begin
```

6.1 Orthonormal system of L2 functions and their Fourier coefficients

```
definition orthonormal-system :: 'a::euclidean-space set ⇒ ('b ⇒ 'a ⇒ real) ⇒
bool
  where orthonormal-system S w ≡ ∀ m n. l2product S (w m) (w n) = (if m = n
then 1 else 0)

definition orthonormal-coeff :: 'a::euclidean-space set ⇒ (nat ⇒ 'a ⇒ real) ⇒ ('a
⇒ real) ⇒ nat ⇒ real
  where orthonormal-coeff S w f n = l2product S (w n) f

lemma orthonormal-system-eq: orthonormal-system S w =⇒ l2product S (w m)
(w n) = (if m = n then 1 else 0)
  by (simp add: orthonormal-system-def)

lemma orthonormal-system-l2norm:
    orthonormal-system S w =⇒ l2norm S (w i) = 1
  by (simp add: l2norm-def orthonormal-system-def)

lemma orthonormal-partial-sum-diff:
  assumes os: orthonormal-system S w and w: ⋀i. (w i) square-integrable S
    and f: f square-integrable S and finite I
  shows (l2norm S (λx. f x − (∑ i∈I. a i ∗ w i x)))^2 =
        (l2norm S f)^2 + (∑ i∈I. (a i)^2) − 2 ∗ (∑ i∈I. a i ∗ orthonormal-coeff S
w f i)
proof −
  have S ∈ sets lebesgue
    using f square-integrable-def by blast
  then have (λx. ∑ i∈I. a i ∗ w i x) square-integrable S
    by (intro square-integrable-sum square-integrable-lmult w ‹finite I›)
  with assms show ?thesis
   apply (simp add: l2norm-pow-2 orthonormal-coeff-def orthonormal-system-def)
    apply (simp add: l2product-rdiff l2product-sym
            l2product-rsum l2product-rmult algebra-simps power2-eq-square if-distrib
[of λx. - ∗ x])
    done
qed

lemma orthonormal-optimal-partial-sum:
  assumes orthonormal-system S w ⋀i. (w i) square-integrable S
```

f square-integrable S finite I
shows *l2norm S* ($\lambda x.$ *f x* − ($\sum i \in I.$ *orthonormal-coeff S w f i* ∗ *w i x*))
≤ *l2norm S* ($\lambda x.$ *f x* − ($\sum i \in I.$ *a i* ∗ *w i x*))
proof −
have *2* ∗ (*a i* ∗ *l2product S f*(*w i*)) ≤ (*a i*)^2 + (*l2product S f*(*w i*))^2 **for** *i*
using *sum-squares-bound* [*of a i l2product S f*(*w i*)]
by (*force simp*: *algebra-simps*)
then have ∗: *2* ∗ ($\sum i \in I.$ *a i* ∗ *l2product S f*(*w i*)) ≤ ($\sum i \in I.$ (*a i*)^2 + (*l2product S f*(*w i*))^2)
by (*simp add*: *sum-distrib-left sum-mono*)
have *S*: *S* ∈ *sets lebesgue*
using *assms square-integrable-def* **by** *blast*
with *assms* ∗ **show** *?thesis*
apply (*simp add*: *l2norm-le square-integrable-sum square-integrable-diff square-integrable-lmult flip*: *l2norm-pow-2*)
apply (*simp add*: *orthonormal-coeff-def orthonormal-partial-sum-diff*)
apply (*simp add*: *l2product-sym power2-eq-square sum.distrib*)
done
qed

lemma *Bessel-inequality*:
assumes *orthonormal-system S w* $\bigwedge i.$ (*w i*) *square-integrable S*
f square-integrable S finite I
shows ($\sum i \in I.$ (*orthonormal-coeff S w f i*)^2) ≤ (*l2norm S f*)^2
using *orthonormal-partial-sum-diff* [*OF assms*, *of orthonormal-coeff S w f*]
apply (*simp add*: *algebra-simps flip*: *power2-eq-square*)
by (*metis* (*lifting*) *add-diff-cancel-left′ diff-ge-0-iff-ge zero-le-power2*)

lemma *Fourier-series-square-summable*:
assumes *os*: *orthonormal-system S w* **and** *w*: $\bigwedge i.$ (*w i*) *square-integrable S*
and *f*: *f square-integrable S*
shows *summable* (*confine* ($\lambda i.$ (*orthonormal-coeff S w f i*) ^ *2*) *I*)
proof −
have *incseq* ($\lambda n. \sum i \in I \cap \{..n\}.$ (*orthonormal-coeff S w f i*)^2)
by (*auto simp*: *incseq-def intro*: *sum-mono2*)
moreover have $\bigwedge i.$ ($\sum i \in I \cap \{..i\}.$ (*orthonormal-coeff S w f i*)^2) ≤ (*l2norm S f*)^2
by (*simp add*: *Bessel-inequality assms*)
ultimately obtain *L* **where** ($\lambda n. \sum i \in I \cap \{..n\}.$ (*orthonormal-coeff S w f i*)^2) $\longrightarrow$ *L*
using *incseq-convergent* **by** *blast*
then have $\forall r>0. \exists no. \forall n \geq no.$ |($\sum i \in \{..n\} \cap I.$ (*orthonormal-coeff S w f i*)^2) − *L*| < *r*
by (*simp add*: *LIMSEQ-iff Int-commute*)
then show *?thesis*
by (*auto simp*: *summable-def sums-confine-le LIMSEQ-iff*)
qed

lemma *orthonormal-Fourier-partial-sum-diff-squared*:

assumes *os*: *orthonormal-system S w* **and** *w*: $\bigwedge i.$ (*w i*) *square-integrable S*
and *f*: *f square-integrable S* **and** *finite I*
shows (*l2norm S* ($\lambda x.$ *f x* $-(\sum i{\in}I.$ *orthonormal-coeff S w f i* $*$ *w i x*)))^2 =
(*l2norm S f*)^2 $-$ ($\sum i{\in}I.$ (*orthonormal-coeff S w f i*)^2)
using *orthonormal-partial-sum-diff* [*OF assms*, **where** *a* = *orthonormal-coeff S w f*]
by (*simp add*: *power2-eq-square*)

lemma *Fourier-series-l2-summable*:
assumes *os*: *orthonormal-system S w* **and** *w*: $\bigwedge i.$ (*w i*) *square-integrable S*
and *f*: *f square-integrable S*
obtains *g* **where** *g square-integrable S*
($\lambda n.$ *l2norm S* ($\lambda x.$ ($\sum i{\in}I \cap \{..n\}.$ *orthonormal-coeff S w f i* $*$ *w i x*) $-$ *g x*))
$\longrightarrow$ *0*
proof $-$
have *S*: $S \in$ *sets lebesgue*
using *f square-integrable-def* **by** *blast*
let *?Θ* = $\lambda A\ x.$ ($\sum i{\in}I \cap A.$ *orthonormal-coeff S w f i* $*$ *w i x*)
let *?Ψ* = *confine* ($\lambda i.$ (*orthonormal-coeff S w f i*)^2) *I*
have *si*: *?Θ A square-integrable S* **if** *finite A* **for** *A*
by (*force intro*: *square-integrable-lmult w square-integrable-sum S that*)
have $\exists N.\ \forall m{\geq}N.\ \forall n{\geq}N.$ *l2norm S* ($\lambda x.$ *?Θ* $\{..m\}$ *x* $-$ *?Θ* $\{..n\}$ *x*) $<$ *e*
if $e > 0$ **for** *e*
proof $-$
have *summable ?Ψ*
by (*rule Fourier-series-square-summable* [*OF os w f*])
then have *Cauchy* ($\lambda n.$ *sum ?Ψ* $\{..n\}$)
by (*simp add*: *summable-def sums-def-le convergent-eq-Cauchy*)
then obtain *M* **where** *M*: $\bigwedge m\ n.$ ⟦$m{\geq}M$; $n{\geq}M$⟧ $\Longrightarrow$ *dist* (*sum ?Ψ* $\{..m\}$) (*sum ?Ψ* $\{..n\}$) $< e^2$
using ‹*e > 0*› **unfolding** *Cauchy-def* **by** (*drule-tac x=e^2* **in** *spec*) *auto*
have ⟦$m{\geq}M$; $n{\geq}M$⟧ $\Longrightarrow$ *l2norm S* ($\lambda x.$ *?Θ* $\{..m\}$ *x* $-$ *?Θ* $\{..n\}$ *x*) $<$ *e* **for** *m n*
proof (*induct m n rule*: *linorder-class.linorder-wlog*)
case (*le m n*)
have *sum-diff*: *sum f*($I \cap \{..n\}$) $-$ *sum f*($I \cap \{..m\}$) = *sum f*($I \cap \{Suc\ m..n\}$) **for** *f* :: *nat* $\Rightarrow$ *real*
proof $-$
have ($I \cap \{..n\}$) = ($I \cap \{..m\} \cup I \cap \{Suc\ m..n\}$) ($I \cap \{..m\}$) $\cap$ ($I \cap \{Suc\ m..n\}$) = {}
using *le* **by** *auto*
then show *?thesis*
by (*simp add*: *algebra-simps flip*: *sum.union-disjoint*)
qed
have (*l2norm S* ($\lambda x.$ *?Θ* $\{..n\}$ *x* $-$ *?Θ* $\{..m\}$ *x*))^2
$\leq$ $|$($\sum i{\in}I \cap \{..n\}.$ (*orthonormal-coeff S w f i*)^2) $-$ ($\sum i{\in}I \cap \{..m\}.$ (*orthonormal-coeff S w f i*)^2)$|$

proof (*simp add*: *sum-diff*)
have (*l2norm S* (*?Θ* {*Suc m..n*}))^2
= (∑ *i*∈*I* ∩ {*Suc m..n*}. *l2product S* (λ*x*. ∑ *j*∈*I* ∩ {*Suc m..n*}. *orthonormal-coeff S w f j* ∗ *w j x*)
(λ*x*. *orthonormal-coeff S w f i* ∗ *w i x*))
by (*simp add*: *l2norm-pow-2 l2product-rsum si w*)
also have … = (∑ *i*∈*I* ∩ {*Suc m..n*}. ∑ *j*∈*I* ∩ {*Suc m..n*}.
orthonormal-coeff S w f j ∗ *orthonormal-coeff S w f i* ∗ *l2product S* (*w j*) (*w i*))
by (*simp add*: *l2product-lsum l2product-lmult l2product-rmult si w flip*: *mult.assoc*)
also have … ≤ |∑ *i*∈*I* ∩ {*Suc m..n*}. (*orthonormal-coeff S w f i*)^2|
by (*simp add*: *orthonormal-system-eq* [*OF os*] *power2-eq-square if-distrib* [*of* (∗)-] *cong*: *if-cong*)
finally show (*l2norm S* (*?Θ* {*Suc m..n*}))^2 ≤ |∑ *i*∈*I* ∩ {*Suc m..n*}. (*orthonormal-coeff S w f i*)^2| **.**
qed
also have … < e^2
using *M* [*OF* ‹*n*≥*M*› ‹*m*≥*M*›]
by (*simp add*: *dist-real-def sum-confine-eq-Int Int-commute*)
finally have (*l2norm S* (λ*x*. *?Θ* {..*m*} *x* − *?Θ* {..*n*} *x*))^*2* < *e*^*2*
using *l2norm-diff* [*OF si si*] **by** *simp*
with ‹*e* > *0*› **show** *?case*
by (*simp add*: *power2-less-imp-less*)
next
case (*sym a b*)
then show *?case*
by (*subst l2norm-diff*) (*auto simp*: *si*)
qed
then show *?thesis*
by *blast*
qed
with *that* **show** *thesis*
by (*blast intro*: *si* [*of* {..-}] *l2-complete* [*of* λ*n*. *?Θ* {..*n*}])
qed

lemma *Fourier-series-l2-summable-strong*:
assumes *os*: *orthonormal-system S w* **and** *w*: ⋀*i*. (*w i*) *square-integrable S*
and *f*: *f square-integrable S*
obtains *g* **where** *g square-integrable S*
⋀*i*. *i* ∈ *I* ⟹ *orthonormal-coeff S w* (λ*x*. *f x* − *g x*) *i* = *0*
(λ*n*. *l2norm S* (λ*x*. (∑ *i*∈*I* ∩ {..*n*}. *orthonormal-coeff S w f i* ∗ *w i x*) − *g x*))
⟶ *0*
proof −
have *S*: *S* ∈ *sets lebesgue*
using *f square-integrable-def* **by** *blast*
obtain *g* **where** *g*: *g square-integrable S*
and *teng*: (λ*n*. *l2norm S* (λ*x*. (∑ *i*∈*I* ∩ {..*n*}. *orthonormal-coeff S w f i*

$*\ w\ i\ x)\ -\ g\ x))$
$\longrightarrow 0$
using *Fourier-series-l2-summable* [*OF assms*] **.**
show *thesis*
proof
show *orthonormal-coeff S w* ($\lambda x.\ f\ x - g\ x$) $i = 0$
if $i \in I$ **for** i
proof (*rule tendsto-unique*)
let $?\Theta = \lambda A\ x.\ (\sum i{\in}I \cap A.$ *orthonormal-coeff S w f i* $*$ *w i x*)
let $?f = \lambda n.$ *l2product S* (*w i*) ($\lambda x.$ ($f\ x - ?\Theta\ \{..n\}\ x$) + ($?\Theta\ \{..n\}\ x - g\ x$))
show $?f \longrightarrow$ *orthonormal-coeff S w* ($\lambda x.\ f\ x - g\ x$) i
by (*simp add*: *orthonormal-coeff-def*)
have *1*: ($\lambda n.$ *l2product S* (*w i*) ($\lambda x.\ f\ x - ?\Theta\ \{..n\}\ x$)) $\longrightarrow 0$
proof (*rule tendsto-eventually*)
have *l2product S* (*w i*) ($\lambda x.\ f\ x - ?\Theta\ \{..j\}\ x$) $= 0$
if $j \geq i$ **for** j
using *that* ‹$i \in I$›
apply (*simp add*: *l2product-rdiff l2product-rsum l2product-rmult orthonormal-coeff-def w f S*)
apply (*simp add*: *orthonormal-system-eq* [*OF os*] *if-distrib* [*of* ($*$)-] *cong*: *if-cong*)
done
then show $\forall_F$ *n in sequentially. l2product S* (*w i*) ($\lambda x.\ f\ x - ?\Theta\ \{..n\}\ x$) $= 0$
using *eventually-at-top-linorder* **by** *blast*
qed
have *2*: ($\lambda n.$ *l2product S* (*w i*) ($\lambda x.\ ?\Theta\ \{..n\}\ x - g\ x$)) $\longrightarrow 0$
proof (*intro Lim-null-comparison* [*OF - teng*] *eventuallyI*)
show *norm* (*l2product S* (*w i*) ($\lambda x.\ ?\Theta\ \{..n\}\ x - g\ x$)) $\leq$ *l2norm S* ($\lambda x.\ ?\Theta\ \{..n\}\ x - g\ x$) **for** n
using *Schwartz-inequality-abs* [*of w i S* ($\lambda x.\ ?\Theta\ \{..n\}\ x - g\ x$)]
by (*simp add*: *S g f w orthonormal-system-l2norm* [*OF os*])
qed
show $?f \longrightarrow 0$
using *that tendsto-add* [*OF 1 2*]
by (*subst l2product-radd*) (*simp-all add*: *l2product-radd w f g S*)
qed *auto*
qed (*auto simp*: *g teng*)
qed

6.2 Actual trigonometric orthogonality relations

lemma *integrable-sin-cx*:
integrable (*lebesgue-on* $\{-pi..pi\}$) ($\lambda x.\ sin(x * c)$)
by (*intro continuous-imp-integrable-real continuous-intros*)

lemma *integrable-cos-cx*:
integrable (*lebesgue-on* $\{-pi..pi\}$) ($\lambda x.\ cos(x * c)$)
by (*intro continuous-imp-integrable-real continuous-intros*)

lemma *integral-cos-Z'* [*simp*]:
assumes $n \in \mathbb{Z}$
shows $integral^L$ (*lebesgue-on* {−*pi*..*pi*}) (λ*x*. *cos*(*n* ∗ *x*)) = (*if n* = *0 then 2* ∗ *pi else 0*)
by (*metis assms integral-cos-Z mult-commute-abs*)

lemma *integral-sin-and-cos*:
fixes *m n*::*int*
shows
$integral^L$ (*lebesgue-on* {−*pi*..*pi*}) (λ*x*. *cos*(*m* ∗ *x*) ∗ *cos*(*n* ∗ *x*)) = (*if* |*m*| = |*n*| *then if n* = *0 then 2* ∗ *pi else pi else 0*)
$integral^L$ (*lebesgue-on* {−*pi*..*pi*}) (λ*x*. *cos*(*m* ∗ *x*) ∗ *sin*(*n* ∗ *x*)) = *0*
$integral^L$ (*lebesgue-on* {−*pi*..*pi*}) (λ*x*. *sin*(*m* ∗ *x*) ∗ *cos*(*n* ∗ *x*)) = *0*
⟦*m* ≥ *0*; *n* ≥ *0*⟧ ⟹ $integral^L$ (*lebesgue-on* {−*pi*..*pi*}) (λ*x*. *sin* (*m* ∗ *x*) ∗ *sin* (*n* ∗ *x*)) = (*if m* = *n* ∧ *n* ≠ *0 then pi else 0*)
|$integral^L$ (*lebesgue-on* {−*pi*..*pi*}) (λ*x*. *sin* (*m* ∗ *x*) ∗ *sin* (*n* ∗ *x*))| = (*if* |*m*| = |*n*| ∧ *n* ≠ *0 then pi else 0*)
by (*simp-all add*: *abs-if sin-times-sin cos-times-sin sin-times-cos cos-times-cos integrable-sin-cx integrable-cos-cx mult-ac*
flip: *distrib-left distrib-right left-diff-distrib right-diff-distrib*)

lemma *integral-sin-and-cos-Z* [*simp*]:
fixes *m n*::*real*
assumes $m \in \mathbb{Z}$ $n \in \mathbb{Z}$
shows
$integral^L$ (*lebesgue-on* {−*pi*..*pi*}) (λ*x*. *cos*(*m* ∗ *x*) ∗ *cos*(*n* ∗ *x*)) = (*if* |*m*| = |*n*| *then if n* = *0 then 2* ∗ *pi else pi else 0*)
$integral^L$ (*lebesgue-on* {−*pi*..*pi*}) (λ*x*. *cos*(*m* ∗ *x*) ∗ *sin*(*n* ∗ *x*)) = *0*
$integral^L$ (*lebesgue-on* {−*pi*..*pi*}) (λ*x*. *sin*(*m* ∗ *x*) ∗ *cos*(*n* ∗ *x*)) = *0*
|$integral^L$ (*lebesgue-on* {−*pi*..*pi*}) (λ*x*. *sin* (*m* ∗ *x*) ∗ *sin* (*n* ∗ *x*))| = (*if* |*m*| = |*n*| ∧ *n* ≠ *0 then pi else 0*)
using *assms* **unfolding** *Ints-def*
apply *safe*
unfolding *integral-sin-and-cos*
apply *auto*
done

lemma *integral-sin-and-cos-N* [*simp*]:
fixes *m n*::*real*
assumes $m \in \mathbb{N}$ $n \in \mathbb{N}$
shows $integral^L$ (*lebesgue-on* {−*pi*..*pi*}) (λ*x*. *sin* (*m* ∗ *x*) ∗ *sin* (*n* ∗ *x*)) = (*if m* = *n* ∧ *n* ≠ *0 then pi else 0*)
using *assms* **unfolding** *Nats-altdef1* **by** (*auto simp*: *integral-sin-and-cos*)

lemma *integrable-sin-and-cos*:
fixes *m n*::*int*
shows *integrable* (*lebesgue-on* {*a*..*b*}) (λ*x*. *cos*(*x* ∗ *m*) ∗ *cos*(*x* ∗ *n*))

integrable (*lebesgue-on* {*a..b*}) (λ*x*. *cos*(*x* ∗ *m*) ∗ *sin*(*x* ∗ *n*))
integrable (*lebesgue-on* {*a..b*}) (λ*x*. *sin*(*x* ∗ *m*) ∗ *cos*(*x* ∗ *n*))
integrable (*lebesgue-on* {*a..b*}) (λ*x*. *sin*(*x* ∗ *m*) ∗ *sin*(*x* ∗ *n*))
by (*intro continuous-imp-integrable-real continuous-intros*)+

lemma *sqrt-pi-ge1*: *sqrt pi* ≥ *1*
using *pi-gt3* **by** *auto*

definition *trigonometric-set* :: *nat* ⇒ *real* ⇒ *real*
where *trigonometric-set n* ≡
if n = *0 then* λ*x*. *1* / *sqrt*(*2* ∗ *pi*)
else if odd n then λ*x*. *sin*(*real*(*Suc* (*n div 2*)) ∗ *x*) / *sqrt*(*pi*)
else (λ*x*. *cos*((*n div 2*) ∗ *x*) / *sqrt pi*)

lemma *trigonometric-set*:
trigonometric-set 0 x = *1* / *sqrt*(*2* ∗ *pi*)
trigonometric-set (*Suc* (*2* ∗ *n*)) *x* = *sin*(*real*(*Suc n*) ∗ *x*) / *sqrt*(*pi*)
trigonometric-set (*2* ∗ *n* + *2*) *x* = *cos*(*real*(*Suc n*) ∗ *x*) / *sqrt*(*pi*)
trigonometric-set (*Suc* (*Suc* (*2* ∗ *n*))) *x* = *cos*(*real*(*Suc n*) ∗ *x*) / *sqrt*(*pi*)
by (*simp-all add*: *trigonometric-set-def algebra-simps add-divide-distrib*)

lemma *trigonometric-set-even*:
trigonometric-set(*2*∗*k*) = (*if k* = *0 then* (λ*x*. *1* / *sqrt*(*2* ∗ *pi*)) *else* (λ*x*. *cos*(*k* ∗ *x*) / *sqrt pi*))
by (*induction k*) (*auto simp*: *trigonometric-set-def add-divide-distrib split*: *if-split-asm*)

lemma *orthonormal-system-trigonometric-set*:
orthonormal-system {−*pi..pi*} *trigonometric-set*
proof −
have *l2product* {−*pi..pi*} (*trigonometric-set m*) (*trigonometric-set n*) = (*if m* = *n then 1 else 0*) **for** *m n*
proof (*induction m rule*: *odd-even-cases*)
case *0*
show *?case*
by (*induction n rule*: *odd-even-cases*) (*auto simp*: *trigonometric-set l2product-def measure-restrict-space*)
next
case (*odd m*)
show *?case*
by (*induction n rule*: *odd-even-cases*) (*auto simp*: *trigonometric-set l2product-def double-not-eq-Suc-double*)
next
case (*even m*)
show *?case*
by (*induction n rule*: *odd-even-cases*) (*auto simp*: *trigonometric-set l2product-def Suc-double-not-eq-double*)
qed
then show *?thesis*
unfolding *orthonormal-system-def* **by** *auto*

qed

lemma *square-integrable-trigonometric-set*:
  (*trigonometric-set i*) *square-integrable* {−*pi*..*pi*}
proof −
  have *continuous-on* {−*pi*..*pi*} (λ*x*. *sin* ((*1* + *real n*) ∗ *x*) / *sqrt pi*) **for** *n*
    by (*intro continuous-intros*) *auto*
  moreover
  have *continuous-on* {−*pi*..*pi*} (λ*x*. *cos* (*real i* ∗ *x* / *2*) / *sqrt pi*)
    by (*intro continuous-intros*) *auto*
  ultimately show *?thesis*
    by (*simp add*: *trigonometric-set-def*)
qed

6.3 Weierstrass for trigonometric polynomials

lemma *Weierstrass-trig-1*:
  fixes *g* :: *real* ⇒ *real*
  assumes *contf*: *continuous-on UNIV g* **and** *periodic*: ⋀*x*. *g*(*x* + *2* ∗ *pi*) = *g x*
and *1*: *norm z* = *1*
  shows *continuous* (*at z within* (*sphere 0 1*)) (*g* ∘ *Im* ∘ *Ln*)
proof (*cases Re z* < *0*)
  let *?f* = *complex-of-real* ∘ *g* ∘ (λ*x*. *x* + *pi*) ∘ *Im* ∘ *Ln* ∘ *uminus*
  case *True*
  have *continuous* (*at z within* (*sphere 0 1*)) (*complex-of-real* ∘ *g* ∘ *Im* ∘ *Ln*)
  proof (*rule continuous-transform-within*)
    have [*simp*]: $z \notin \mathbb{R}_{\geq 0}$
      using *True complex-nonneg-Reals-iff* **by** *auto*
    show *continuous* (*at z within* (*sphere 0 1*)) *?f*
    by (*intro continuous-within-Ln continuous-intros continuous-on-imp-continuous-within* [*OF contf*]) *auto*
    show *?f x′* = (*complex-of-real* ∘ *g* ∘ *Im* ∘ *Ln*) *x′*
      if *x′* ∈ (*sphere 0 1*) **and** *dist x′ z* < *1* **for** *x′*
    proof −
      have *x′* ≠ *0*
        using *that* **by** *auto*
      with *that* **show** *?thesis*
        by (*auto simp*: *Ln-minus add.commute periodic*)
    qed
  qed (*use 1* **in** *auto*)
  then have *continuous* (*at z within* (*sphere 0 1*)) (*Re* ∘ *complex-of-real* ∘ *g* ∘ *Im* ∘ *Ln*)
    unfolding *o-def* **by** (*rule continuous-Re*)
  then show *?thesis*
    by (*simp add*: *o-def*)
next
  case *False*
  let *?f* = *complex-of-real* ∘ *g* ∘ *Im* ∘ *Ln* ∘ *uminus*

have *z* ≠ *0*
  using *1* **by** *auto*
with *False* **have** $z \notin \mathbb{R}_{\leq 0}$
  by (*auto simp*: *not-less nonpos-Reals-def*)
then have *continuous* (*at z within* (*sphere 0 1*)) (*complex-of-real* ∘ *g* ∘ *Im* ∘ *Ln*)
  by (*intro continuous-within-Ln continuous-intros continuous-on-imp-continuous-within* [*OF contf*]) *auto*
then have *continuous* (*at z within* (*sphere 0 1*)) (*Re* ∘ *complex-of-real* ∘ *g* ∘ *Im* ∘ *Ln*)
  unfolding *o-def* **by** (*rule continuous-Re*)
then show *?thesis*
  by (*simp add*: *o-def*)
qed

inductive-set *cx-poly* :: (*complex* ⇒ *real*) *set* **where**
  Re: *Re* ∈ *cx-poly*
| *Im*: *Im* ∈ *cx-poly*
| *const*: (λ*x*. *c*) ∈ *cx-poly*
| *add*: ⟦*f* ∈ *cx-poly*; *g* ∈ *cx-poly*⟧ ⟹ (λ*x*. *f x* + *g x*) ∈ *cx-poly*
| *mult*: ⟦*f* ∈ *cx-poly*; *g* ∈ *cx-poly*⟧ ⟹ (λ*x*. *f x* ∗ *g x*) ∈ *cx-poly*

declare *cx-poly.intros* [*intro*]

lemma *Weierstrass-trig-polynomial*:
  assumes *contf*: *continuous-on* {−*pi*..*pi*} *f* **and** *fpi*: *f*(−*pi*) = *f pi* **and** *0* < *e*
  obtains *n*::*nat* **and** *a b* **where**
    ⋀*x*::*real*. *x* ∈ {−*pi*..*pi*} ⟹ |*f x* − (∑*k*≤*n*. *a k* ∗ *sin* (*k* ∗ *x*) + *b k* ∗ *cos* (*k* ∗ *x*))| < *e*
proof −
  interpret *CR*: *function-ring-on cx-poly sphere 0 1*
  proof
    show *continuous-on* (*sphere 0 1*) *f* **if** *f* ∈ *cx-poly* **for** *f*
      using *that* **by** *induction* (*assumption* | *intro continuous-intros*)+
    fix *x y*::*complex*
    assume *x* ∈ *sphere 0 1* **and** *y* ∈ *sphere 0 1* **and** *x* ≠ *y*
    then consider *Re x* ≠ *Re y* | *Im x* ≠ *Im y*
      using *complex-eqI* **by** *blast*
    then show ∃*f*∈*cx-poly*. *f x* ≠ *f y*
      by (*metis cx-poly.Im cx-poly.Re*)
  qed (*auto simp*: *cx-poly.intros*)
  have *continuous* (*at z within* (*sphere 0 1*)) (*f* ∘ *Im* ∘ *Ln*) **if** *norm z* = *1* **for** *z*
  proof −
    obtain *g* **where** *contg*: *continuous-on UNIV g* **and** *gf*: ⋀*x*. *x* ∈ {−*pi*..*pi*} ⟹ *g x* = *f x*
      and *periodic*: ⋀*x*. *g*(*x* + *2*∗*pi*) = *g x*
      using *Tietze-periodic-interval* [*OF contf fpi*] **by** *auto*
    let *?f* = (*g* ∘ *Im* ∘ *Ln*)
    show *?thesis*

proof (*rule continuous-transform-within*)
show *continuous* (*at z within* (*sphere 0 1*)) *?f*
using *Weierstrass-trig-1* [*OF contg periodic that*] **by** (*simp add*: *sphere-def*)
show *?f x′* = (*f* ∘ *Im* ∘ *Ln*) *x′*
if *x′* ∈ *sphere 0 1 dist x′ z* < *1* **for** *x′*
proof −
have *x′* ≠ *0*
using *that* **by** *auto*
then have *Im* (*Ln x′*) ∈ {−*pi*..*pi*}
using *Im-Ln-le-pi* [*of x′*] *mpi-less-Im-Ln* [*of x′*] **by** *simp*
then show *?thesis*
using *gf* **by** *simp*
qed
qed (*use that* **in** *auto*)
qed
then have *continuous-on* (*sphere 0 1*) (*f* ∘ *Im* ∘ *Ln*)
using *continuous-on-eq-continuous-within mem-sphere-0* **by** *blast*
then obtain *g* **where** *g* ∈ *cx-poly* **and** *g*: ⋀*x*. *x* ∈ *sphere 0 1* ⟹ |(*f* ∘ *Im* ∘ *Ln*) *x* − *g x*| < *e*
using *CR.Stone-Weierstrass-basic* [*of f* ∘ *Im* ∘ *Ln*] ‹*e* > *0*› **by** *meson*
define *trigpoly* **where**
trigpoly ≡ λ*f*. ∃ *n a b*. *f* = (λ*x*. (∑ *k*≤*n*. *a k* ∗ *sin*(*real k* ∗ *x*) + *b k* ∗ *cos*(*real k* ∗ *x*)))
have *tp-const*: *trigpoly*(λ*x*. *c*) **for** *c*
unfolding *trigpoly-def*
by (*rule-tac x=0* **in** *exI*) *auto*
have *tp-add*: *trigpoly*(λ*x*. *f x* + *g x*) **if** *trigpoly f trigpoly g* **for** *f g*
proof −
obtain *n1 a1 b1* **where** *feq*: *f* = (λ*x*. ∑ *k*≤*n1*. *a1 k* ∗ *sin* (*real k* ∗ *x*) + *b1 k* ∗ *cos* (*real k* ∗ *x*))
using ‹*trigpoly f*› *trigpoly-def* **by** *blast*
obtain *n2 a2 b2* **where** *geq*: *g* = (λ*x*. ∑ *k*≤*n2*. *a2 k* ∗ *sin* (*real k* ∗ *x*) + *b2 k* ∗ *cos* (*real k* ∗ *x*))
using ‹*trigpoly g*› *trigpoly-def* **by** *blast*
let *?a* = λ*n*. (*if n* ≤ *n1 then a1 n else 0*) + (*if n* ≤ *n2 then a2 n else 0*)
let *?b* = λ*n*. (*if n* ≤ *n1 then b1 n else 0*) + (*if n* ≤ *n2 then b2 n else 0*)
have *eq*: {*k*. *k* ≤ *max a b* ∧ *k* ≤ *a*} = {..*a*} {*k*. *k* ≤ *max a b* ∧ *k* ≤ *b*} = {..*b*} **for** *a b*::*nat*
by *auto*
have (λ*x*. *f x* + *g x*) = (λ*x*. ∑ *k* ≤ *max n1 n2*. *?a k* ∗ *sin* (*real k* ∗ *x*) + *?b k* ∗ *cos* (*real k* ∗ *x*))
by (*simp add*: *feq geq algebra-simps eq sum.distrib if-distrib* [*of* λ*u*. - ∗ *u*] *cong*: *if-cong flip*: *sum.inter-filter*)
then show *?thesis*
unfolding *trigpoly-def* **by** *meson*
qed
have *tp-sum*: *trigpoly*(λ*x*. ∑ *i*∈*S*. *f i x*) **if** *finite S* ⋀*i*. *i* ∈ *S* ⟹ *trigpoly*(*f i*) **for** *f* **and** *S* :: *nat set*
using *that*

by *induction* (*auto simp*: *tp-const tp-add*)
have *tp-cmul*: *trigpoly*($\lambda x.\ c * f\ x$) **if** *trigpoly f* **for** *f c*
proof −
obtain *n a b* **where** *feq*: $f = (\lambda x.\ \sum k{\le}n.\ a\ k * sin\ (real\ k * x) + b\ k * cos\ (real\ k * x))$
using ‹*trigpoly f*› *trigpoly-def* **by** *blast*
have $(\lambda x.\ c * f\ x) = (\lambda x.\ \sum k \le n.\ (c * a\ k) * sin\ (real\ k * x) + (c * b\ k) * cos\ (real\ k * x))$
by (*simp add*: *feq algebra-simps sum-distrib-left*)
then show *?thesis*
unfolding *trigpoly-def* **by** *meson*
qed
have *tp-cdiv*: *trigpoly*($\lambda x.\ f\ x\ /\ c$) **if** *trigpoly f* **for** *f c*
using *tp-cmul* [*OF* ‹*trigpoly f*›, *of inverse c*]
by (*simp add*: *divide-inverse-commute*)
have *tp-diff*: *trigpoly*($\lambda x.\ f\ x - g\ x$) **if** *trigpoly f trigpoly g* **for** *f g*
using *tp-add* [*OF* ‹*trigpoly f*› *tp-cmul* [*OF* ‹*trigpoly g*›, *of* −*1*]] **by** *auto*
have *tp-sin*: *trigpoly*($\lambda x.\ sin\ (n * x)$) **if** $n \in \mathbb{N}$ **for** *n*
unfolding *trigpoly-def*
proof
obtain *k* **where** *n* = *real k*
using *Nats-cases* ‹$n \in \mathbb{N}$› **by** *blast*
then show $\exists\ a\ b.\ (\lambda x.\ sin\ (n * x)) = (\lambda x.\ \sum i \le nat\lfloor n\rfloor.\ a\ i * sin\ (real\ i * x) + b\ i * cos\ (real\ i * x))$
apply (*rule-tac x*=λi. *if i* = *k then 1 else 0* **in** *exI*)
apply (*rule-tac x*=λi. *0* **in** *exI*)
apply (*simp add*: *if-distrib* [*of* $\lambda u.\ u * \text{-}$] *cong*: *if-cong*)
done
qed
have *tp-cos*: *trigpoly*($\lambda x.\ cos\ (n * x)$) **if** $n \in \mathbb{N}$ **for** *n*
unfolding *trigpoly-def*
proof
obtain *k* **where** *n* = *real k*
using *Nats-cases* ‹$n \in \mathbb{N}$› **by** *blast*
then show $\exists\ a\ b.\ (\lambda x.\ cos\ (n * x)) = (\lambda x.\ \sum i \le nat\lfloor n\rfloor.\ a\ i * sin\ (real\ i * x) + b\ i * cos\ (real\ i * x))$
apply (*rule-tac x*=λi. *0* **in** *exI*)
apply (*rule-tac x*=λi. *if i* = *k then 1 else 0* **in** *exI*)
apply (*simp add*: *if-distrib* [*of* $\lambda u.\ u * \text{-}$] *cong*: *if-cong*)
done
qed
have *tp-sincos*: $trigpoly(\lambda x.\ sin(i * x) * sin(j * x)) \wedge trigpoly(\lambda x.\ sin(i * x) * cos(j * x)) \wedge$
$trigpoly(\lambda x.\ cos(i * x) * sin(j * x)) \wedge trigpoly(\lambda x.\ cos(i * x) * cos(j * x))$ (**is** *?P i j*)
for *i j*::*nat*
proof (*rule linorder-wlog* [*of* - *j i*])
show *?P j i* **if** $i \le j$ **for** *j i*
using *that*

by (*simp add*: *sin-times-sin cos-times-cos sin-times-cos cos-times-sin diff-divide-distrib*

tp-add tp-diff tp-cdiv tp-cos tp-sin flip: *left-diff-distrib distrib-right*)
qed (*simp add*: *mult-ac*)
have *tp-mult*: *trigpoly*(λ*x*. *f x* ∗ *g x*) **if** *trigpoly f trigpoly g* **for** *f g*
proof −
obtain *n1 a1 b1* **where** *feq*: *f* = (λ*x*. ∑ *k*≤*n1*. *a1 k* ∗ *sin* (*real k* ∗ *x*) + *b1 k* ∗ *cos* (*real k* ∗ *x*))
using ‹*trigpoly f*› *trigpoly-def* **by** *blast*
obtain *n2 a2 b2* **where** *geq*: *g* = (λ*x*. ∑ *k*≤*n2*. *a2 k* ∗ *sin* (*real k* ∗ *x*) + *b2 k* ∗ *cos* (*real k* ∗ *x*))
using ‹*trigpoly g*› *trigpoly-def* **by** *blast*
then show *?thesis*
unfolding *feq geq*
by (*simp add*: *algebra-simps sum-product tp-sum tp-add tp-cmul tp-sincos del*: *mult.commute*)
qed
have ∗: *trigpoly*(λ*x*. *f*(*exp*(i ∗ *of-real x*))) **if** *f* ∈ *cx-poly* **for** *f*
using *that*
proof *induction*
case *Re*
then show *?case*
using *tp-cos* [*of 1*] **by** (*auto simp*: *Re-exp*)
next
case *Im*
then show *?case*
using *tp-sin* [*of 1*] **by** (*auto simp*: *Im-exp*)
qed (*auto simp*: *tp-const tp-add tp-mult*)
obtain *n a b* **where** *eq*: (*g* (*iexp x*)) = (∑ *k*≤*n*. *a k* ∗ *sin* (*real k* ∗ *x*) + *b k* ∗ *cos* (*real k* ∗ *x*)) **for** *x*
using ∗ [*OF* ‹*g* ∈ *cx-poly*›] *trigpoly-def* **by** *meson*
show *thesis*
proof
show |*f* ϑ − (∑ *k*≤*n*. *a k* ∗ *sin* (*real k* ∗ ϑ) + *b k* ∗ *cos* (*real k* ∗ ϑ))| < *e*
if ϑ ∈ {−*pi*..*pi*} **for** ϑ
proof −
have *f* ϑ − *g* (*iexp* ϑ) = (*f* ∘ *Im* ∘ *Ln*) (*iexp* ϑ) − *g* (*iexp* ϑ)
proof (*cases* ϑ = −*pi*)
case *True*
then show *?thesis*
by (*simp add*: *exp-minus fpi*)
next
case *False*
then show *?thesis*
using *that* **by** *auto*
qed
then show *?thesis*
using *g* [*of exp*(i ∗ *of-real* ϑ)] **by** (*simp flip*: *eq*)
qed

qed
qed

6.4 A bit of extra hacking round so that the ends of a function are OK

lemma *integral-tweak-ends*:
fixes *a b* :: *real*
assumes $a < b$ $e > 0$
obtains *f* **where** *continuous-on* $\{a..b\}$ *f f a* = *d f b* = *0 l2norm* $\{a..b\}$ *f* < *e*
proof −
have *cont*: *continuous-on* $\{a..b\}$
(λ*x*. *if* $x \le a+1$ / *real*(*Suc n*)
then ((*Suc n*) ∗ *d*) ∗ ((*a*+*1* / (*Suc n*)) − *x*) *else 0*) **for** *n*
proof (*cases a*+*1*/(*Suc n*) ≤ *b*)
case *True*
have ∗: *1* / (*1* + *real n*) > *0*
by *auto*
have *abeq*: $\{a..b\} = \{a..a+1/(Suc\ n)\} \cup \{a+1/(Suc\ n)..b\}$
using ‹*a* < *b*› *True*
apply *auto*
using ∗ **by** *linarith*
show *?thesis*
unfolding *abeq*
proof (*rule continuous-on-cases*)
show *continuous-on* {*a*..*a*+*1* / *real* (*Suc n*)} (λ*x*. *real* (*Suc n*) ∗ *d* ∗ (*a*+*1* / *real* (*Suc n*) − *x*))
by (*intro continuous-intros*)
qed *auto*
next
case *False*
show *?thesis*
proof (*rule continuous-on-eq* [**where** *f* = λ*x*. ((*Suc n*) ∗ *d*) ∗ ((*a*+*1*/(*Suc n*)) − *x*)])
show *continuous-on* {*a*..*b*} (λ*x*. (*Suc n*) ∗ *d* ∗ (*a*+*1* / *real* (*Suc n*) − *x*))
by (*intro continuous-intros*)
qed (*use False* **in** *auto*)
qed
let *?f* = λ*k x*. (*if* $x \le a+1$ / (*Suc k*) *then* (*Suc k*) ∗ *d* ∗ (*a*+*1* / (*Suc k*) − *x*) *else 0*)^2
let *?g* = λ*x*. *if x* = *a then* d^2 *else 0*
have *bmg*: *?g* ∈ *borel-measurable* (*lebesgue-on* {*a*..*b*})
apply (*rule measurable-restrict-space1*)
using *borel-measurable-if-I* [*of* - {*a*}, *OF borel-measurable-const*] **by** *auto*
have *bmf*: *?f k* ∈ *borel-measurable* (*lebesgue-on* {*a*..*b*}) **for** *k*
proof −
have *bm*: (λ*x*. (*Suc k*) ∗ *d* ∗ (*a*+*1* / *real* (*Suc k*) − *x*))
∈ *borel-measurable* (*lebesgue-on* {..*a*+*1* / (*Suc k*)})
by (*intro measurable*) (*auto simp*: *measurable-completion measurable-restrict-space1*)

```
    show ?thesis
      apply (intro borel-measurable-power measurable-restrict-space1)
      using borel-measurable-if-I [of - {.. a+1 / (Suc k)}, OF bm] apply auto
      done
  qed
  have int-d2: integrable (lebesgue-on {a..b}) (λx. d²)
    by (intro continuous-imp-integrable-real continuous-intros)
  have (λk. ?f k x) ⟶ ?g x
    if x: x ∈ {a..b} for x
  proof (cases x = a)
    case False
    then have x > a
      using x by auto
    with x obtain N where N > 0 and N: 1 / real N < x−a
      using real-arch-invD [of x−a]
      by (force simp: inverse-eq-divide)
    then have x > a+1 / (1 + real k)
      if k ≥ N for k
    proof −
      have a+1 / (1 + real k) < a+1 / real N
        using that ‹0 < N› by (simp add: field-simps)
      also have ... < x
        using N by linarith
      finally show ?thesis .
    qed
    then show ?thesis
      apply (intro tendsto-eventually eventually-sequentiallyI [where c=N])
      by (fastforce simp: False)
  qed auto
  then have tends: AE x in (lebesgue-on {a..b}). (λk. ?f k x) ⟶ ?g x
    by force
  have le-d2: ⋀k. AE x in (lebesgue-on {a..b}). norm (?f k x) ≤ d²
  proof
    show norm ((if x ≤ a+1 / real (Suc k) then real (Suc k) ∗ d ∗ (a+1 / real
(Suc k) − x) else 0)²) ≤ d²
      if x ∈ space (lebesgue-on {a..b}) for k x
      using that
      apply (simp add: abs-mult divide-simps flip: abs-le-square-iff)
      apply (auto simp: abs-if zero-less-mult-iff mult-left-le)
      done
  qed
  have integ: integrable (lebesgue-on {a..b}) ?g
    using integrable-dominated-convergence [OF bmg bmf int-d2 tends le-d2] by
auto
  have int: (λk. integral^L (lebesgue-on {a..b}) (?f k)) ⟶ integral^L (lebesgue-on
{a..b}) ?g
    using integral-dominated-convergence [OF bmg bmf int-d2 tends le-d2] by auto
  then obtain N where N: ⋀k. k ≥ N ⟹ |integral^L (lebesgue-on {a..b}) (?f k)
− integral^L (lebesgue-on {a..b}) ?g| < e²
```

apply (*simp add*: *lim-sequentially dist-real-def*)
apply (*drule-tac x*=*e*^*2* **in** *spec*)
using ‹*e* > *0*›
by *auto*
obtain *M* **where** *M* > *0* **and** *M*: *1* / *real M* < *b*−*a*
using *real-arch-invD* [*of b*−*a*]
by (*metis* ‹*a* < *b*› *diff-gt-0-iff-gt inverse-eq-divide neq0-conv*)
have ∗: |*integral*^{L} (*lebesgue-on* {*a*..*b*}) (*?f* (*max M N*)) − *integral*^{L} (*lebesgue-on* {*a*..*b*}) *?g*| < e^2
using *N* **by** *force*
let *?φ* = λ*x*. *if x* ≤ *a*+*1*/(*Suc* (*max M N*)) *then* ((*Suc* (*max M N*)) ∗ *d*) ∗ ((*a*+*1*/(*Suc* (*max M N*))) − *x*) *else 0*
show *?thesis*
proof
show *continuous-on* {*a*..*b*} *?φ*
by (*rule cont*)
have *1* / (*1* + *real* (*max M N*)) ≤ *1* / (*real M*)
by (*simp add*: ‹*0* < *M*› *frac-le*)
then have ¬ (*b* ≤ *a*+*1* / (*1* + *real* (*max M N*)))
using *M* ‹*a* < *b*› ‹*M* > *0*› *max.cobounded1* [*of M N*]
by *linarith*
then show *?φ b* = *0*
by *simp*
have *null-a*: {*a*} ∈ *null-sets* (*lebesgue-on* {*a*..*b*})
using ‹*a* < *b*› **by** (*simp add*: *null-sets-restrict-space*)
have *LINT x*|*lebesgue-on* {*a*..*b*}. *?g x* = *0*
by (*intro integral-eq-zero-AE AE-I'* [*OF null-a*]) *auto*
then have *l2norm* {*a*..*b*} *?φ* < *sqrt* (*e*^*2*)
unfolding *l2norm-def l2product-def power2-eq-square* [*symmetric*]
apply (*intro real-sqrt-less-mono*)
using ∗ **by** *linarith*
then show *l2norm* {*a*..*b*} *?φ* < *e*
using ‹*e* > *0*› **by** *auto*
qed *auto*
qed

lemma *square-integrable-approximate-continuous-ends*:
assumes *f*: *f square-integrable* {*a*..*b*} **and** *a* < *b 0* < *e*
obtains *g* **where** *continuous-on* {*a*..*b*} *g g b* = *g a g square-integrable* {*a*..*b*} *l2norm* {*a*..*b*} (λ*x*. *f x* − *g x*) < *e*
proof −
obtain *g* **where** *contg*: *continuous-on UNIV g* **and** *g square-integrable* {*a*..*b*}
and *lg*: *l2norm* {*a*..*b*} (λ*x*. *f x* − *g x*) < *e*/*2*
using *f* ‹*e* > *0*› *square-integrable-approximate-continuous*
by (*metis* (*full-types*) *box-real*(*2*) *half-gt-zero-iff lmeasurable-cbox*)
obtain *h* **where** *conth*: *continuous-on* {*a*..*b*} *h* **and** *h a* = *g b* − *g a h b* = *0*
and *lh*: *l2norm* {*a*..*b*} *h* < *e*/*2*
using *integral-tweak-ends* ‹*e* > *0*›

by (*metis* ‹*a* < *b*› *zero-less-divide-iff zero-less-numeral*)
have *h square-integrable* {*a*..*b*}
using ‹*continuous-on* {*a*..*b*} *h*› *continuous-imp-square-integrable* **by** *blast*
show *thesis*
proof
show *continuous-on* {*a*..*b*} (λ*x*. *g x* + *h x*)
by (*blast intro*: *continuous-on-subset* [*OF contg*] *conth continuous-intros*)
then show (λ*x*. *g x* + *h x*) *square-integrable* {*a*..*b*}
using *continuous-imp-square-integrable* **by** *blast*
show *g b* + *h b* = *g a* + *h a*
by (*simp add*: ‹*h a* = *g b* − *g a*› ‹*h b* = *0*›)
have *l2norm* {*a*..*b*} (λ*x*. (*f x* − *g x*) + − *h x*) < *e*
proof (*rule le-less-trans* [*OF l2norm-triangle* [*of* λ*x*. *f x* − *g x* {*a*..*b*} λ*x*. − (*h x*)]]])
show (λ*x*. *f x* − *g x*) *square-integrable* {*a*..*b*}
using ‹*g square-integrable* {*a*..*b*}› *f square-integrable-diff* **by** *blast*
show (λ*x*. − *h x*) *square-integrable* {*a*..*b*}
by (*simp add*: ‹*h square-integrable* {*a*..*b*}›)
show *l2norm* {*a*..*b*} (λ*x*. *f x* − *g x*) + *l2norm* {*a*..*b*} (λ*x*. − *h x*) < *e*
using ‹*h square-integrable* {*a*..*b*}› *l2norm-neg lg lh* **by** *auto*
qed
then show *l2norm* {*a*..*b*} (λ*x*. *f x* − (*g x* + *h x*)) < *e*
by (*simp add*: *field-simps*)
qed
qed

6.5 Hence the main approximation result

lemma *Weierstrass-l2-trig-polynomial*:
assumes *f*: *f square-integrable* {−*pi*..*pi*} **and** *0* < *e*
obtains *n a b* **where**
l2norm {−*pi*..*pi*} (λ*x*. *f x* − (∑*k*≤*n*. *a k* ∗ *sin*(*real k* ∗ *x*) + *b k* ∗ *cos*(*real k* ∗ *x*))) < *e*
proof −
let *?φ* = λ*n a b x*. ∑*k*≤*n*. *a k* ∗ *sin*(*real k* ∗ *x*) + *b k* ∗ *cos*(*real k* ∗ *x*)
obtain *g* **where** *contg*: *continuous-on* {−*pi*..*pi*} *g* **and** *geq*: *g*(−*pi*) = *g pi*
and *g*: *g square-integrable* {−*pi*..*pi*} **and** *norm-fg*: *l2norm* {−*pi*..*pi*} (λ*x*. *f x* − *g x*) < *e*/*2*
using ‹*e* > *0*› **by** (*auto intro*: *square-integrable-approximate-continuous-ends* [*OF f*, *of e*/*2*])
then obtain *n a b* **where** *g-phi-less*: ⋀*x*. *x* ∈ {−*pi*..*pi*} ⟹ |*g x* − (*?φ n a b x*)| < *e*/*6*
using ‹*e* > *0*› *Weierstrass-trig-polynomial* [*OF contg geq*, *of e*/*6*]
by (*meson zero-less-divide-iff zero-less-numeral*)
show *thesis*
proof
have *si*: (*?φ n u v*) *square-integrable* {−*pi*..*pi*} **for** *n*::*nat* **and** *u v*
proof (*intro square-integrable-sum continuous-imp-square-integrable*)
show *continuous-on* {−*pi*..*pi*} (λ*x*. *u k* ∗ *sin* (*real k* ∗ *x*) + *v k* ∗ *cos* (*real k*

∗ x))
        if *k* ∈ {..*n*} **for** *k*
        using *that* **by** (*intro continuous-intros*)
    qed *auto*
    have *l2norm* {−*pi*..*pi*} (λ*x*. *f x* − (*?φ n a b x*)) = *l2norm* {−*pi*..*pi*} (λ*x*. (*f x* − *g x*) + (*g x* − (*?φ n a b x*)))
      by *simp*
    also have ... ≤ *l2norm* {−*pi*..*pi*} (λ*x*. *f x* − *g x*) + *l2norm* {−*pi*..*pi*} (λ*x*. *g x* − (*?φ n a b x*))
    proof (*rule l2norm-triangle*)
      show (λ*x*. *f x* − *g x*) *square-integrable* {−*pi*..*pi*}
        using *f g square-integrable-diff* **by** *blast*
      show (λ*x*. *g x* − (*?φ n a b x*)) *square-integrable* {−*pi*..*pi*}
        using *g si square-integrable-diff* **by** *blast*
    qed
    also have ... < *e*
    proof −
      have *g-phi*: (λ*x*. *g x* − (*?φ n a b x*)) *square-integrable* {−*pi*..*pi*}
        using *g si square-integrable-diff* **by** *blast*
      have *l2norm* {−*pi*..*pi*} (λ*x*. *g x* − (*?φ n a b x*)) ≤ *e*/*2*
        unfolding *l2norm-def l2product-def power2-eq-square* [*symmetric*]
      proof (*rule real-le-lsqrt*)
        have *LINT x*|*lebesgue-on* {−*pi*..*pi*}. (*g x* − (*?φ n a b x*))^2
            ≤ *LINT x*|*lebesgue-on* {−*pi*..*pi*}. (*e* / *6*) ^ *2*
        proof (*rule integral-mono*)
          show *integrable* (*lebesgue-on* {−*pi*..*pi*}) (λ*x*. (*g x* − (*?φ n a b x*))^2)
            using *g-phi square-integrable-def* **by** *auto*
          show *integrable* (*lebesgue-on* {−*pi*..*pi*}) (λ*x*. (*e* / *6*)^2)
            by (*intro continuous-intros continuous-imp-integrable-real*)
          show (*g x* − (*?φ n a b x*))^2 ≤ (*e* / *6*)^2 **if** *x* ∈ *space* (*lebesgue-on* {−*pi*..*pi*}) **for** *x*
            using ‹*e* > *0*› *g-phi-less that*
            apply (*simp add*: *abs-le-square-iff* [*symmetric*])
            using *less-eq-real-def* **by** *blast*
        qed
        also have ... ≤ (*e* / *2*)^2
          using ‹*e* > *0*› *pi-less-4* **by** (*auto simp*: *power2-eq-square measure-restrict-space*)
        finally show *LINT x*|*lebesgue-on* {−*pi*..*pi*}. (*g x* − (*?φ n a b x*))^2 ≤ (*e* / *2*)^2 **.**
      qed (*use* ‹*e* > *0*› **in** *auto*)
      with *norm-fg* **show** *?thesis*
        by *auto*
    qed
    finally show *l2norm* {−*pi*..*pi*} (λ*x*. *f x* − (*?φ n a b x*)) < *e* **.**
  qed
qed

proposition *Weierstrass-l2-trigonometric-set*:

assumes *f*: *f square-integrable* {−*pi*..*pi*} **and** *0* < *e*
obtains *n a* **where** *l2norm* {−*pi*..*pi*} (λ*x*. *f x* − (∑*k*≤*n*. *a k* ∗ *trigonometric-set k x*)) < *e*
proof −
obtain *n a b* **where** *lee*:
l2norm {−*pi*..*pi*} (λ*x*. *f x* − (∑*k*≤*n*. *a k* ∗ *sin*(*real k* ∗ *x*) + *b k* ∗ *cos*(*real k* ∗ *x*))) < *e*
using *Weierstrass-l2-trig-polynomial* [*OF assms*] **.**
let *?a* = λ*k*. *if k* = *0 then sqrt*(*2* ∗ *pi*) ∗ *b 0*
else if even k then sqrt pi ∗ *b*(*k div 2*)
else if k ≤ *2* ∗ *n then sqrt pi* ∗ *a*((*Suc k*) *div 2*)
else 0
show *thesis*
proof
have [*simp*]: *Suc* (*i* ∗ *2*) ≤ *n* ∗ *2* ⟷ *i*<*n* {..*n*} ∩ {..<*n*} = {..<*n*} **for** *i n*
by *auto*
have (∑*k*≤*n*. *b k* ∗ *cos* (*real k* ∗ *x*)) = (∑*i*≤*n*. *if i* = *0 then b 0 else b i* ∗ *cos* (*real i* ∗ *x*)) **for** *x*
by (*rule sum.cong*) *auto*
moreover have (∑*k*≤*n*. *a k* ∗ *sin* (*real k* ∗ *x*)) = (∑*i*≤*n*. (*if Suc* (*2* ∗ *i*) ≤ *2* ∗ *n then sqrt pi* ∗ *a* (*Suc i*) ∗ *sin* ((*1* + *real i*) ∗ *x*) *else 0*) / *sqrt pi*)
(**is** *?lhs* = *?rhs*) **for** *x*
proof (*cases n*=*0*)
case *False*
then obtain *n′* **where** *n′*: *n* = *Suc n′*
using *not0-implies-Suc* **by** *blast*
have *?lhs* = (∑*k* = *Suc 0*..*n*. *a k* ∗ *sin* (*real k* ∗ *x*))
by (*simp add*: *atMost-atLeast0 sum-shift-lb-Suc0-0*)
also have ... = (∑*i*<*n*. *a* (*Suc i*) ∗ *sin* (*x* + *x* ∗ *real i*))
proof (*subst sum.reindex-bij-betw* [*symmetric*])
show *bij-betw Suc* {..*n′*} {*Suc 0*..*n*}
by (*simp add*: *atMost-atLeast0 n′*)
show (∑*j*≤*n′*. *a* (*Suc j*) ∗ *sin* (*real* (*Suc j*) ∗ *x*)) = (∑*i*<*n*. *a* (*Suc i*) ∗ *sin* (*x* + *x* ∗ *real i*))
unfolding *n′ lessThan-Suc-atMost* **by** (*simp add*: *algebra-simps*)
qed
also have ... = *?rhs*
by (*simp add*: *field-simps if-distrib* [*of* λ*x*. *x*/-] *sum.inter-restrict* [**where** *B* = {..<*n*}, *simplified*, *symmetric*] *cong*: *if-cong*)
finally
show *?thesis* **.**
qed *auto*
ultimately
have (∑*k*≤*n*. *a k* ∗ *sin*(*real k* ∗ *x*) + *b k* ∗ *cos*(*real k* ∗ *x*)) = (∑*k* ≤ *Suc*(*2*∗*n*). *?a k* ∗ *trigonometric-set k x*) **for** *x*
unfolding *sum.in-pairs-0 trigonometric-set-def*
by (*simp add*: *sum.distrib if-distrib* [*of* λ*x*. *x*∗-] *cong*: *if-cong*)
with *lee* **show** *l2norm* {−*pi*..*pi*} (λ*x*. *f x* − (∑*k* ≤ *Suc*(*2*∗*n*). *?a k* ∗ *trigonometric-set k x*)) < *e*

by *auto*
qed
qed

6.6 Convergence wrt the L2 norm of trigonometric Fourier series

definition *Fourier-coefficient*
where *Fourier-coefficient* ≡ *orthonormal-coeff* {−*pi*..*pi*} *trigonometric-set*

lemma *Fourier-series-l2*:
assumes *f square-integrable* {−*pi*..*pi*}
shows (λ*n*. *l2norm* {−*pi*..*pi*} (λ*x*. *f x* − (∑ *i*≤*n*. *Fourier-coefficient f i* ∗ *trigonometric-set i x*)))
⟶ *0*
proof (*clarsimp simp add*: *lim-sequentially dist-real-def Fourier-coefficient-def*)
let *?h* = λ*n x*. (∑ *i*≤*n*. *orthonormal-coeff* {−*pi*..*pi*} *trigonometric-set f i* ∗ *trigonometric-set i x*)
show ∃ *N*. ∀ *n*≥*N*. |*l2norm* {−*pi*..*pi*} (λ*x*. *f x* − *?h n x*)| < *e*
if *0* < *e* **for** *e*
proof −
obtain *N a* **where** *lte*: *l2norm* {−*pi*..*pi*} (λ*x*. *f x* − (∑ *k*≤*N*. *a k* ∗ *trigonometric-set k x*)) < *e*
using *Weierstrass-l2-trigonometric-set* **by** (*meson* ‹*0* < *e*› *assms*)
show *?thesis*
proof (*intro exI allI impI*)
show |*l2norm* {−*pi*..*pi*} (λ*x*. *f x* − *?h m x*)| < *e*
if *N* ≤ *m* **for** *m*
proof −
have |*l2norm* {−*pi*..*pi*} (λ*x*. *f x* − *?h m x*)| = *l2norm* {−*pi*..*pi*} (λ*x*. *f x* − *?h m x*)
proof (*rule abs-of-nonneg*)
show *0* ≤ *l2norm* {−*pi*..*pi*} (λ*x*. *f x* − *?h m x*)
apply (*intro l2norm-pos-le square-integrable-diff square-integrable-sum square-integrable-lmult*
square-integrable-trigonometric-set assms, *auto*)
done
qed
also have ... ≤ *l2norm* {−*pi*..*pi*} (λ*x*. *f x* − (∑ *k*≤*N*. *a k* ∗ *trigonometric-set k x*))
proof −
have (∑ *i*≤*m*. (*if i* ≤ *N then a i else 0*) ∗ *trigonometric-set i x*) = (∑ *i*≤*N*. *a i* ∗ *trigonometric-set i x*) **for** *x*
using *sum.inter-restrict* [**where** *A* = {..*m*} **and** *B* = {..*N*}, *symmetric*] *that*
by (*force simp*: *if-distrib* [*of* λ*x*. *x* ∗ -] *min-absorb2 cong*: *if-cong*)
moreover
have *l2norm* {−*pi*..*pi*} (λ*x*. *f x* − *?h m x*)
≤ *l2norm* {−*pi*..*pi*} (λ*x*. *f x* − (∑ *i*≤*m*. (*if i* ≤ *N then a i else 0*) ∗

trigonometric-set i x))
using *orthonormal-optimal-partial-sum*
[*OF orthonormal-system-trigonometric-set square-integrable-trigonometric-set assms*]
by *simp*
ultimately show *?thesis*
by *simp*
qed
also have ... < *e*
by (*rule lte*)
finally show *?thesis* **.**
qed
qed
qed
qed

6.7 Fourier coefficients go to 0 (weak form of Riemann-Lebesgue)

lemma *trigonometric-set-mul-absolutely-integrable*:
assumes *f absolutely-integrable-on* {−*pi..pi*}
shows (λ*x. trigonometric-set n x* ∗ *f x*) *absolutely-integrable-on* {−*pi..pi*}
proof (*rule absolutely-integrable-bounded-measurable-product-real*)
show *trigonometric-set n* ∈ *borel-measurable* (*lebesgue-on* {−*pi..pi*})
using *square-integrable-def square-integrable-trigonometric-set* **by** *blast*
show *bounded* (*trigonometric-set n* ‘ {−*pi..pi*})
unfolding *bounded-iff* **using** *pi-gt3 sqrt-pi-ge1*
by (*rule-tac x=1* **in** *exI*)
(*auto simp*: *trigonometric-set-def dist-real-def*
intro: *order-trans* [*OF abs-sin-le-one*] *order-trans* [*OF abs-cos-le-one*])
qed (*auto simp*: *assms*)

lemma *trigonometric-set-mul-integrable*:
f absolutely-integrable-on {−*pi..pi*} ⟹ *integrable* (*lebesgue-on* {−*pi..pi*}) (λ*x. trigonometric-set n x* ∗ *f x*)
using *trigonometric-set-mul-absolutely-integrable*
by (*simp add*: *integrable-restrict-space set-integrable-def*)

lemma *trigonometric-set-integrable* [*simp*]: *integrable* (*lebesgue-on* {−*pi..pi*}) (*trigonometric-set n*)
using *trigonometric-set-mul-integrable* [**where** *f* = *id*]
by *simp* (*metis absolutely-integrable-imp-integrable fmeasurableD interval-cbox lmeasurable-cbox square-integrable-imp-absolutely-integrable square-integrable-trigonometric-set*)

lemma *absolutely-integrable-sin-product*:
assumes *f absolutely-integrable-on* {−*pi..pi*}
shows (λ*x. sin*(*k* ∗ *x*) ∗ *f x*) *absolutely-integrable-on* {−*pi..pi*}
proof (*rule absolutely-integrable-bounded-measurable-product-real*)
show (λ*x. sin* (*k* ∗ *x*)) ∈ *borel-measurable* (*lebesgue-on* {−*pi..pi*})

by (*metis borel-measurable-integrable integrable-sin-cx mult-commute-abs*)
show *bounded* (($\lambda x.$ *sin* ($k * x$)) ' {$-pi..pi$})
by (*metis* (*mono-tags*, *lifting*) *abs-sin-le-one bounded-iff imageE real-norm-def*)
qed (*auto simp*: *assms*)

lemma *absolutely-integrable-cos-product*:
assumes *f absolutely-integrable-on* {$-pi..pi$}
shows ($\lambda x.$ *cos*($k * x$) $*$ *f x*) *absolutely-integrable-on* {$-pi..pi$}
proof (*rule absolutely-integrable-bounded-measurable-product-real*)
show ($\lambda x.$ *cos* ($k * x$)) $\in$ *borel-measurable* (*lebesgue-on* {$-pi..pi$})
by (*metis borel-measurable-integrable integrable-cos-cx mult-commute-abs*)
show *bounded* (($\lambda x.$ *cos* ($k * x$)) ' {$-pi..pi$})
by (*metis* (*mono-tags*, *lifting*) *abs-cos-le-one bounded-iff imageE real-norm-def*)
qed (*auto simp*: *assms*)

lemma
assumes *f absolutely-integrable-on* {$-pi..pi$}
shows *Fourier-products-integrable-cos*: *integrable* (*lebesgue-on* {$-pi..pi$}) ($\lambda x.$ *cos*($k * x$) $*$ *f x*)
and *Fourier-products-integrable-sin*: *integrable* (*lebesgue-on* {$-pi..pi$}) ($\lambda x.$ *sin*($k * x$) $*$ *f x*)
using *absolutely-integrable-cos-product absolutely-integrable-sin-product assms*
by (*auto simp*: *integrable-restrict-space set-integrable-def*)

lemma *Riemann-lebesgue-square-integrable*:
assumes *orthonormal-system S w* $\bigwedge i.$ *w i square-integrable S f square-integrable S*
shows *orthonormal-coeff S w f* $\longrightarrow$ *0*
using *Fourier-series-square-summable* [*OF assms*, *of UNIV*] *summable-LIMSEQ-zero*
by *force*

proposition *Riemann-lebesgue*:
assumes *f absolutely-integrable-on* {$-pi..pi$}
shows *Fourier-coefficient f* $\longrightarrow$ *0*
unfolding *lim-sequentially*
proof (*intro allI impI*)
fix *e*::*real*
assume $e > 0$
then obtain *g* **where** *continuous-on UNIV g* **and** *gabs*: *g absolutely-integrable-on* {$-pi..pi$}
and *fg-e2*: $integral^L$ (*lebesgue-on* {$-pi..pi$}) ($\lambda x.$ $|f\ x - g\ x|$) $< e/2$
using *absolutely-integrable-approximate-continuous* [*OF assms*, *of e/2*]
by (*metis* (*full-types*) *box-real*(*2*) *half-gt-zero-iff lmeasurable-cbox*)
have *g square-integrable* {$-pi..pi$}
using ‹*continuous-on UNIV g*› *continuous-imp-square-integrable continuous-on-subset* **by** *blast*
then have *orthonormal-coeff* {$-pi..pi$} *trigonometric-set g* $\longrightarrow$ *0*
using *Riemann-lebesgue-square-integrable orthonormal-system-trigonometric-set*

square-integrable-trigonometric-set **by** *blast*
with ‹$e > 0$› **obtain** N **where** N: $\bigwedge n.\ n \geq N \Longrightarrow |$*orthonormal-coeff* $\{-pi..pi\}$ *trigonometric-set g n*$| < e/2$
unfolding *lim-sequentially* **by** (*metis half-gt-zero-iff norm-conv-dist real-norm-def*)
have $|$*Fourier-coefficient f n*$| < e$
if $n \geq N$ **for** n
proof −
have $|$*LINT x*$|$*lebesgue-on* $\{-pi..pi\}$. *trigonometric-set n x* $*$ *g x*$| < e/2$
using N [*OF* ‹$n \geq N$›] **by** (*auto simp*: *orthonormal-coeff-def l2product-def*)

have *integrable* (*lebesgue-on* $\{-pi..pi\}$) (λx. *trigonometric-set n x* $*$ *g x*)
using *gabs trigonometric-set-mul-integrable* **by** *blast*
moreover have *integrable* (*lebesgue-on* $\{-pi..pi\}$) (λx. *trigonometric-set n x* $*$ *f x*)
using *assms trigonometric-set-mul-integrable* **by** *blast*
ultimately have $|$(*LINT x*$|$*lebesgue-on* $\{-pi..pi\}$. *trigonometric-set n x* $*$ *g x*) −
(*LINT x*$|$*lebesgue-on* $\{-pi..pi\}$. *trigonometric-set n x* $*$ *f x*)$|$
$= |$(*LINT x*$|$*lebesgue-on* $\{-pi..pi\}$. *trigonometric-set n x* $*$ (*g x* − *f x*))$|$
by (*simp add*: *algebra-simps flip*: *Bochner-Integration.integral-diff*)
also have ... $\leq$ *LINT x*$|$*lebesgue-on* $\{-pi..pi\}$. $|$*f x* − *g x*$|$
proof (*rule integral-abs-bound-integral*)
show *integrable* (*lebesgue-on* $\{-pi..pi\}$) (λx. *trigonometric-set n x* $*$ (*g x* − *f x*))
by (*simp add*: *gabs assms trigonometric-set-mul-integrable*)
have (λx. *f x* − *g x*) *absolutely-integrable-on* $\{-pi..pi\}$
using *gabs assms* **by** *blast*
then show *integrable* (*lebesgue-on* $\{-pi..pi\}$) (λx. $|$*f x* − *g x*$|$)
by (*simp add*: *absolutely-integrable-imp-integrable*)
fix x
assume $x \in$ *space* (*lebesgue-on* $\{-pi..pi\}$)
then have $-pi \leq x$ $x \leq pi$
by *auto*
have $|$*trigonometric-set n x*$| \leq 1$
using *pi-ge-two*
apply (*simp add*: *trigonometric-set-def*)
using *sqrt-pi-ge1 abs-sin-le-one order-trans abs-cos-le-one* **by** *metis*
then show $|$*trigonometric-set n x* $*$ (*g x* − *f x*)$| \leq |$*f x* − *g x*$|$
using *abs-ge-zero mult-right-mono* **by** (*fastforce simp add*: *abs-mult abs-minus-commute*)
qed
finally have $|$(*LINT x*$|$*lebesgue-on* $\{-pi..pi\}$. *trigonometric-set n x* $*$ *g x*) −
(*LINT x*$|$*lebesgue-on* $\{-pi..pi\}$. *trigonometric-set n x* $*$ *f x*)$|$
$\leq$ *LINT x*$|$*lebesgue-on* $\{-pi..pi\}$. $|$*f x* − *g x*$|$ **.**
then show *?thesis*
using N [*OF* ‹$n \geq N$›] *fg-e2*
unfolding *Fourier-coefficient-def orthonormal-coeff-def l2product-def*
by *linarith*
qed

then show ∃ *no*. ∀ *n*≥*no*. *dist* (*Fourier-coefficient f n*) *0* < *e*
by *auto*
qed

lemma *Riemann-lebesgue-sin*:
assumes *f absolutely-integrable-on* {−*pi*..*pi*}
shows (λ*n*. *integral*^{L} (*lebesgue-on* {−*pi*..*pi*}) (λ*x*. *sin*(*real n* ∗ *x*) ∗ *f x*)) ⟶ *0*
unfolding *lim-sequentially*
proof (*intro allI impI*)
fix *e*::*real*
assume *e* > *0*
then obtain *N* **where** *N*: ⋀*n*. *n* ≥ *N* ⟹ |*Fourier-coefficient f n*| < *e*/*4*
using *Riemann-lebesgue* [*OF assms*]
unfolding *lim-sequentially*
by (*metis norm-conv-dist real-norm-def zero-less-divide-iff zero-less-numeral*)
have |*LINT x*|*lebesgue-on* {−*pi*..*pi*}. *sin* (*real n* ∗ *x*) ∗ *f x*| < *e* **if** *n* > *N* **for** *n*
using *that*
proof (*induction n*)
case (*Suc n*)
have |*Fourier-coefficient f*(*Suc* (*2* ∗ *n*))| < *e*/*4*
using *N Suc.prems* **by** *auto*
then have |*LINT x*|*lebesgue-on* {−*pi*..*pi*}. *sin* ((*1* + *real n*) ∗ *x*) ∗ *f x*| < *sqrt pi* ∗ *e* / *4*
by (*simp add*: *Fourier-coefficient-def orthonormal-coeff-def trigonometric-set-def l2product-def field-simps*)
also have . . . ≤ *e*
using ‹*0* < *e*› *pi-less-4 real-sqrt-less-mono* **by** (*fastforce simp add*: *field-simps*)
finally show *?case*
by *simp*
qed *auto*
then show ∃ *no*. ∀ *n*≥*no*. *dist* (*LINT x*|*lebesgue-on* {−*pi*..*pi*}. *sin* (*real n* ∗ *x*) ∗ *f x*) *0* < *e*
by (*metis* (*full-types*) *le-neq-implies-less less-add-same-cancel2 less-trans norm-conv-dist real-norm-def zero-less-one*)
qed

lemma *Riemann-lebesgue-cos*:
assumes *f absolutely-integrable-on* {−*pi*..*pi*}
shows (λ*n*. *integral*^{L} (*lebesgue-on* {−*pi*..*pi*}) (λ*x*. *cos*(*real n* ∗ *x*) ∗ *f x*)) ⟶ *0*
unfolding *lim-sequentially*
proof (*intro allI impI*)
fix *e*::*real*
assume *e* > *0*
then obtain *N* **where** *N*: ⋀*n*. *n* ≥ *N* ⟹ |*Fourier-coefficient f n*| < *e*/*4*
using *Riemann-lebesgue* [*OF assms*]
unfolding *lim-sequentially*

by (*metis norm-conv-dist real-norm-def zero-less-divide-iff zero-less-numeral*)
have |*LINT x*|*lebesgue-on* {−*pi..pi*}. *cos* (*real n* ∗ *x*) ∗ *f x*| < *e* **if** *n* > *N* **for** *n*
using *that*
proof (*induction n*)
case (*Suc n*)
have *eq*: (*x* ∗ *2* + *x* ∗ (*real n* ∗ *2*)) / *2* = *x* + *x* ∗ (*real n*) **for** *x*
by *simp*
have |*Fourier-coefficient f* (*2*∗*n* + *2*)| < *e*/*4*
using *N Suc.prems* **by** *auto*
then have |*LINT x*|*lebesgue-on* {−*pi..pi*}. *f x* ∗ *cos* (*x* + *x* ∗ (*real n*))| < *sqrt pi* ∗ *e* / *4*
by (*simp add*: *Fourier-coefficient-def orthonormal-coeff-def trigonometric-set-def l2product-def field-simps eq*)
also have . . . ≤ *e*
using ‹*0* < *e*› *pi-less-4 real-sqrt-less-mono* **by** (*fastforce simp add*: *field-simps*)
finally show *?case*
by (*simp add*: *field-simps*)
qed *auto*
then show ∃ *no*. ∀ *n*≥*no*. *dist* (*LINT x*|*lebesgue-on* {−*pi..pi*}. *cos* (*real n* ∗ *x*) ∗ *f x*) *0* < *e*
by (*metis* (*full-types*) *le-neq-implies-less less-add-same-cancel2 less-trans norm-conv-dist real-norm-def zero-less-one*)
qed

lemma *Riemann-lebesgue-sin-half*:
assumes *f absolutely-integrable-on* {−*pi..pi*}
shows (λ*n*. *LINT x*|*lebesgue-on* {−*pi..pi*}. *sin* ((*real n* + *1*/*2*) ∗ *x*) ∗ *f x*) ⟶ *0*
proof (*simp add*: *algebra-simps sin-add*)
let *?INT* = *integral*^{L} (*lebesgue-on* {−*pi..pi*})
let *?f* = (λ*n*. *?INT* (λ*x*. *sin*(*n* ∗ *x*) ∗ *cos*(*1*/*2* ∗ *x*) ∗ *f x*) +
?INT (λ*x*. *cos*(*n* ∗ *x*) ∗ *sin*(*1*/*2* ∗ *x*) ∗ *f x*))
show (λ*n*. *?INT* (λ*x*. *f x* ∗ (*cos* (*x* ∗ *real n*) ∗ *sin* (*x*/*2*)) + *f x* ∗ (*sin* (*x* ∗ *real n*) ∗ *cos* (*x*/*2*)))) ⟶ *0*
proof (*rule Lim-transform-eventually*)
have *sin*: (λ*x*. *sin* (*1*/*2* ∗ *x*) ∗ *f x*) *absolutely-integrable-on* {−*pi..pi*}
by (*intro absolutely-integrable-sin-product assms*)
have *cos*: (λ*x*. *cos* (*1*/*2* ∗ *x*) ∗ *f x*) *absolutely-integrable-on* {−*pi..pi*}
by (*intro absolutely-integrable-cos-product assms*)
show ∀$_F$ *n in sequentially*. *?f n* = *?INT* (λ*x*. *f x* ∗ (*cos* (*x* ∗ *real n*) ∗ *sin* (*x*/*2*)) + *f x* ∗ (*sin* (*x* ∗ *real n*) ∗ *cos* (*x*/*2*)))
unfolding *mult.assoc*
apply (*rule eventuallyI*)
apply (*subst Bochner-Integration.integral-add* [*symmetric*])
apply (*safe intro*!: *cos absolutely-integrable-sin-product sin absolutely-integrable-cos-product absolutely-integrable-imp-integrable*)
apply (*auto simp*: *algebra-simps*)
done

have *?f* ⟶ *0* + *0*
unfolding *mult.assoc*
by (*intro tendsto-add Riemann-lebesgue-sin Riemann-lebesgue-cos sin cos*)
then show *?f* ⟶ (*0*::*real*) **by** *simp*
qed
qed

lemma *Fourier-sum-limit-pair*:
assumes *f absolutely-integrable-on* {−*pi*..*pi*}
shows (λ*n*. ∑ *k*≤*2* ∗ *n*. *Fourier-coefficient f k* ∗ *trigonometric-set k t*) ⟶ *l*
⟷ (λ*n*. ∑ *k*≤*n*. *Fourier-coefficient f k* ∗ *trigonometric-set k t*) ⟶ *l*
(**is** *?lhs* = *?rhs*)
proof
assume *L*: *?lhs*
show *?rhs*
unfolding *lim-sequentially dist-real-def*
proof (*intro allI impI*)
fix *e*::*real*
assume *e* > *0*
then obtain *N1* **where** *N1*: ⋀*n*. *n* ≥ *N1* ⟹ |*Fourier-coefficient f n*| < *e*/*2*
using *Riemann-lebesgue* [*OF assms*] **unfolding** *lim-sequentially*
by (*metis norm-conv-dist real-norm-def zero-less-divide-iff zero-less-numeral*)
obtain *N2* **where** *N2*: ⋀*n*. *n* ≥ *N2* ⟹ |(∑ *k*≤*2* ∗ *n*. *Fourier-coefficient f k* ∗ *trigonometric-set k t*) − *l*| < *e*/*2*
using *L* **unfolding** *lim-sequentially dist-real-def*
by (*meson* ‹*0* < *e*› *half-gt-zero-iff*)
show ∃ *no*. ∀ *n*≥*no*. |(∑ *k*≤*n*. *Fourier-coefficient f k* ∗ *trigonometric-set k t*) − *l*| < *e*
proof (*intro exI allI impI*)
fix *n*
assume *n*: *N1* + *2*∗*N2* + *1* ≤ *n*
then have *n* ≥ *N1 n* ≥ *N2 n div 2* ≥ *N2*
by *linarith*+
consider *n* = *2* ∗ (*n div 2*) | *n* = *Suc*(*2* ∗ (*n div 2*))
by *linarith*
then show |(∑ *k*≤*n*. *Fourier-coefficient f k* ∗ *trigonometric-set k t*) − *l*| < *e*
proof *cases*
case *1*
show *?thesis*
apply (*subst 1*)
using *N2* [*OF* ‹*n div 2* ≥ *N2*›] **by** *linarith*
next
case *2*
have |(∑ *k*≤*2* ∗ (*n div 2*). *Fourier-coefficient f k* ∗ *trigonometric-set k t*) − *l*| < *e*/*2*
using *N2* [*OF* ‹*n div 2* ≥ *N2*›] **by** *linarith*
moreover have |*Fourier-coefficient f* (*Suc* (*2* ∗ (*n div 2*))) ∗ *trigonometric-set* (*Suc* (*2* ∗ (*n div 2*))) *t*| < (*e*/*2*) ∗ *1*

```
        proof −
          have |trigonometric-set (Suc (2 ∗ (n div 2))) t| ≤ 1
            apply (simp add: trigonometric-set)
            using abs-sin-le-one sqrt-pi-ge1 by (blast intro: order-trans)
          moreover have |Fourier-coefficient f (Suc (2 ∗ (n div 2)))| < e/2
            using N1 ‹N1 ≤ n› by auto
          ultimately show ?thesis
            unfolding abs-mult
        by (meson abs-ge-zero le-less-trans mult-left-mono mult-less-cancel-right-pos
zero-less-one)
        qed
        ultimately show ?thesis
          apply (subst 2)
          unfolding sum.atMost-Suc by linarith
      qed
    qed
  qed
next
  assume ?rhs then show ?lhs
    unfolding lim-sequentially
    by (auto simp: elim!: imp-forward ex-forward)
qed
```

6.8 Express Fourier sum in terms of the special expansion at the origin

```
lemma Fourier-sum-0:
  (∑ k ≤ n. Fourier-coefficient f k ∗ trigonometric-set k 0) =
     (∑ k ≤ n div 2. Fourier-coefficient f (2∗k) ∗ trigonometric-set (2∗k) 0)
  (is ?lhs = ?rhs)
proof −
  have ?lhs = (∑ k = 2 ∗ 0.. Suc (2 ∗ (n div 2)). Fourier-coefficient f k ∗
trigonometric-set k 0)
  proof (rule sum.mono-neutral-left)
    show ∀ i∈{2 ∗ 0..Suc (2 ∗ (n div 2))} − {..n}. Fourier-coefficient f i ∗ trigono-
metric-set i 0 = 0
    proof clarsimp
      fix i
      assume i ≤ Suc (2 ∗ (n div 2)) ¬ i ≤ n
   then have i = Suc n even n
     by presburger+
   moreover
   assume trigonometric-set i 0 ≠ 0
   ultimately
   show Fourier-coefficient f i = 0
     by (simp add: trigonometric-set-def)
  qed
 qed auto
  also have ... = ?rhs
```

unfolding *sum.in-pairs* **by** (*simp add*: *trigonometric-set atMost-atLeast0*)
finally show *?thesis* **.**
qed

lemma *Fourier-sum-0-explicit*:
($\sum$ *k*≤*n*. *Fourier-coefficient f k* ∗ *trigonometric-set k 0*)
= (*Fourier-coefficient f 0* / *sqrt 2* + ($\sum$ *k* = *1*..*n div 2*. *Fourier-coefficient f*(*2*∗*k*))) / *sqrt pi*
(**is** *?lhs* = *?rhs*)
proof −
have ($\sum$ *k*=*0*..*n*. *Fourier-coefficient f k* ∗ *trigonometric-set k 0*)
= *Fourier-coefficient f 0* ∗ *trigonometric-set 0 0* + ($\sum$ *k* = *1*..*n*. *Fourier-coefficient f k* ∗ *trigonometric-set k 0*)
by (*simp add*: *Fourier-sum-0 sum.atLeast-Suc-atMost*)
also have ... = *?rhs*
proof −
have *Fourier-coefficient f 0* ∗ *trigonometric-set 0 0* = *Fourier-coefficient f 0* / (*sqrt 2* ∗ *sqrt pi*)
by (*simp add*: *real-sqrt-mult trigonometric-set*(*1*))
moreover have ($\sum$ *k* = *Suc 0*..*n*. *Fourier-coefficient f k* ∗ *trigonometric-set k 0*) = ($\sum$ *k* = *Suc 0*..*n div 2*. *Fourier-coefficient f*(*2*∗*k*)) / *sqrt pi*
proof (*induction n*)
case (*Suc n*)
show *?case*
by (*simp add*: *Suc*) (*auto simp*: *trigonometric-set-def field-simps*)
qed *auto*
ultimately show *?thesis*
by (*simp add*: *add-divide-distrib*)
qed
finally show *?thesis*
by (*simp add*: *atMost-atLeast0*)
qed

lemma *Fourier-sum-0-integrals*:
assumes *f absolutely-integrable-on* {−*pi*..*pi*}
shows ($\sum$ *k*≤*n*. *Fourier-coefficient f k* ∗ *trigonometric-set k 0*) =
(*integral*^{L} (*lebesgue-on* {−*pi*..*pi*}) *f* / *2* +
($\sum$ *k* = *Suc 0*..*n div 2*. *integral*^{L} (*lebesgue-on* {−*pi*..*pi*}) (λ*x*. *cos*(*k* ∗ *x*) ∗ *f x*))) / *pi*
proof −
have *integral*^{L} (*lebesgue-on* {−*pi*..*pi*}) *f* / (*sqrt* (*2* ∗ *pi*) ∗ *sqrt 2* ∗ *sqrt pi*) = *integral*^{L} (*lebesgue-on* {−*pi*..*pi*}) *f* / (*2* ∗ *pi*)
by (*simp add*: *algebra-simps real-sqrt-mult*)
moreover have ($\sum$ *k* = *Suc 0*..*n div 2*. *LINT x*|*lebesgue-on* {−*pi*..*pi*}. *trigonometric-set* (*2*∗*k*) *x* ∗ *f x*) / *sqrt pi*
= ($\sum$ *k* = *Suc 0*..*n div 2*. *LINT x*|*lebesgue-on* {−*pi*..*pi*}. *cos* (*k* ∗ *x*) ∗ *f x*) / *pi*
by (*simp add*: *trigonometric-set-def sum-divide-distrib*)

ultimately show *?thesis*
unfolding *Fourier-sum-0-explicit*
by (*simp add*: *Fourier-coefficient-def orthonormal-coeff-def l2product-def trigonometric-set add-divide-distrib*)
qed

lemma *Fourier-sum-0-integral*:
assumes *f absolutely-integrable-on* $\{-pi..pi\}$
shows ($\sum k \le n$. *Fourier-coefficient f k* $*$ *trigonometric-set k 0*) =
integral^L (*lebesgue-on* $\{-pi..pi\}$) (λx. (*1/2* + ($\sum k$ = *Suc 0..n div 2. cos*(*k* $*$ *x*))) $*$ *f x*) / *pi*
proof −
note $*$ = *Fourier-products-integrable-cos* [*OF assms*]
have *integrable* (*lebesgue-on* $\{-pi..pi\}$) (λx. $\sum n$ = *Suc 0..n div 2. f x* $*$ *cos* (*x* $*$ *real n*))
using $*$ **by** (*force simp*: *mult-ac*)
moreover have *integral*^{L} (*lebesgue-on* $\{-pi..pi\}$) *f* / *2* + ($\sum k$ = *Suc 0..n div 2. LINT x*|*lebesgue-on* $\{-pi..pi\}$. *f x* $*$ *cos* (*x* $*$ *real k*))
= (*LINT x*|*lebesgue-on* $\{-pi..pi\}$. *f x* / *2*) + (*LINT x*|*lebesgue-on* $\{-pi..pi\}$. ($\sum n$ = *Suc 0..n div 2. f x* $*$ *cos* (*x* $*$ *real n*)))
proof (*subst Bochner-Integration.integral-sum*)
show *integrable* (*lebesgue-on* $\{-pi..pi\}$) (λx. *f x* $*$ *cos* (*x* $*$ *real i*))
if *i* $\in$ {*Suc 0..n div 2*} **for** *i*
using *that* $*$ [*of i*] **by** (*auto simp*: *Bochner-Integration.integral-sum mult-ac*)
qed *auto*
ultimately show *?thesis*
using *Fourier-products-integrable-cos* [*OF assms*] *absolutely-integrable-imp-integrable* [*OF assms*]
by (*simp add*: *Fourier-sum-0-integrals sum-distrib-left assms algebra-simps*)
qed

6.9 How Fourier coefficients behave under addition etc

lemma *Fourier-coefficient-add*:
assumes *f absolutely-integrable-on* $\{-pi..pi\}$ *g absolutely-integrable-on* $\{-pi..pi\}$
shows *Fourier-coefficient* (λx. *f x* + *g x*) *i* =
Fourier-coefficient f i + *Fourier-coefficient g i*
using *assms trigonometric-set-mul-integrable*
by (*simp add*: *Fourier-coefficient-def orthonormal-coeff-def l2product-def distrib-left*)

lemma *Fourier-coefficient-minus*:
assumes *f absolutely-integrable-on* $\{-pi..pi\}$
shows *Fourier-coefficient* (λx. − *f x*) *i* = − *Fourier-coefficient f i*
using *assms trigonometric-set-mul-integrable*
by (*simp add*: *Fourier-coefficient-def orthonormal-coeff-def l2product-def*)

lemma *Fourier-coefficient-diff*:
assumes *f*: *f absolutely-integrable-on* $\{-pi..pi\}$ **and** *g*: *g absolutely-integrable-on*

{−*pi*..*pi*}
shows *Fourier-coefficient* (λ*x*. *f x* − *g x*) *i* = *Fourier-coefficient f i* − *Fourier-coefficient g i*
proof −
have *mg*: (λ*x*. − *g x*) *absolutely-integrable-on* {−*pi*..*pi*}
using *g* **by** (*simp add*: *absolutely-integrable-measurable-real*)
show *?thesis*
using *Fourier-coefficient-add* [*OF f mg*] *Fourier-coefficient-minus* [*OF g*] **by** *simp*
qed

lemma *Fourier-coefficient-const*:
Fourier-coefficient (λ*x*. *c*) *i* = (*if i* = *0 then c* ∗ *sqrt*(*2* ∗ *pi*) *else 0*)
by (*auto simp*: *Fourier-coefficient-def orthonormal-coeff-def l2product-def trigonometric-set-def divide-simps measure-restrict-space*)

lemma *Fourier-offset-term*:
fixes *f* :: *real* ⇒ *real*
assumes *f*: *f absolutely-integrable-on* {−*pi*..*pi*} **and** *periodic*: ⋀*x*. *f*(*x* + *2*∗*pi*) = *f x*
shows *Fourier-coefficient* (λ*x*. *f*(*x*+*t*)) (*2* ∗ *n* + *2*) ∗ *trigonometric-set* (*2* ∗ *n* + *2*) *0*
= *Fourier-coefficient f*(*2* ∗ *n*+*1*) ∗ *trigonometric-set* (*2* ∗ *n*+*1*) *t*
+ *Fourier-coefficient f*(*2* ∗ *n* + *2*) ∗ *trigonometric-set* (*2* ∗ *n* + *2*) *t*
proof −
have *eq*: (*2* + *2* ∗ *real n*) ∗ *x* / *2* = (*1* + *real n*) ∗ *x* **for** *x*
by (*simp add*: *divide-simps*)
have *1*: *integrable* (*lebesgue-on* {−*pi*..*pi*}) (λ*x*. *f x* ∗ (*cos* (*x* + *x* ∗ *n*) ∗ *cos* (*t* + *t* ∗ *n*)))
using *Fourier-products-integrable-cos* [*of f Suc n*] *f*
by (*simp add*: *algebra-simps*) (*simp add*: *mult.assoc* [*symmetric*])
have *2*: *integrable* (*lebesgue-on* {−*pi*..*pi*}) (λ*x*. *f x* ∗ (*sin* (*x* + *x* ∗ *n*) ∗ *sin* (*t* + *t* ∗ *n*)))
using *Fourier-products-integrable-sin* [*of f Suc n*] *f*
by (*simp add*: *algebra-simps*) (*simp add*: *mult.assoc* [*symmetric*])
have *has-bochner-integral* (*lebesgue-on* {−*pi*..*pi*}) (λ*x*. *cos* (*real* (*Suc n*) ∗ (*x* + *t* − *t*)) ∗ *f*(*x* + *t*))
(*LINT x*|*lebesgue-on* {−*pi*..*pi*}. *cos* (*real* (*Suc n*) ∗ (*x* − *t*)) ∗ *f x*)
proof (*rule has-integral-periodic-offset*)
have (λ*x*. *cos* (*real* (*Suc n*) ∗ (*x* − *t*)) ∗ *f x*) *absolutely-integrable-on* {−*pi*..*pi*}
proof (*rule absolutely-integrable-bounded-measurable-product-real*)
show (λ*x*. *cos* (*real* (*Suc n*) ∗ (*x* − *t*))) ∈ *borel-measurable* (*lebesgue-on* {−*pi*..*pi*})
by (*intro continuous-imp-measurable-on-sets-lebesgue continuous-intros*) *auto*
show *bounded* ((λ*x*. *cos* (*real* (*Suc n*) ∗ (*x* − *t*))) ‘ {−*pi*..*pi*})
by (*rule boundedI* [**where** *B*=*1*]) *auto*
qed (*use f* **in** *auto*)
then show *has-bochner-integral* (*lebesgue-on* {−*pi*..*pi*}) (λ*x*. *cos* ((*Suc n*) ∗ (*x* − *t*)) ∗ *f x*) (*LINT x*|*lebesgue-on* {−*pi*..*pi*}. *cos* (*real* (*Suc n*) ∗ (*x* − *t*)) ∗ *f x*)

by (*simp add*: *has-bochner-integral-integrable integrable-restrict-space set-integrable-def*)
next
fix *x*
show *cos* (*real* (*Suc n*) ∗ (*x* + (*pi* − − *pi*) − *t*)) ∗ *f*(*x* + (*pi* − − *pi*)) = *cos* (*real* (*Suc n*) ∗ (*x* − *t*)) ∗ *f x*
using *periodic cos.plus-of-nat* [*of* (*Suc n*) ∗ (*x* − *t*) *Suc n*] **by** (*simp add*: *algebra-simps*)
qed
then have *has-bochner-integral* (*lebesgue-on* {−*pi*..*pi*}) (λ*x*. *cos* (*real* (*Suc n*) ∗ *x*) ∗ *f*(*x* + *t*))
(*LINT x*|*lebesgue-on* {−*pi*..*pi*}. *cos* (*real* (*Suc n*) ∗ (*x* − *t*)) ∗ *f x*)
by *simp*
then have *LINT x*|*lebesgue-on* {−*pi*..*pi*}. *cos* ((*Suc n*) ∗ *x*) ∗ *f*(*x*+*t*)
= *LINT x*|*lebesgue-on* {−*pi*..*pi*}. *cos* ((*Suc n*) ∗ (*x* − *t*)) ∗ *f x*
using *has-bochner-integral-integral-eq* **by** *blast*
also have ... = *LINT x*|*lebesgue-on* {−*pi*..*pi*}. *cos* ((*Suc n*) ∗ *x* − ((*Suc n*) ∗ *t*)) ∗ *f x*
by (*simp add*: *algebra-simps*)
also have ... = *cos* ((*Suc n*) ∗ *t*) ∗ (*LINT x*|*lebesgue-on* {−*pi*..*pi*}. *cos* ((*Suc n*) ∗ *x*) ∗ *f x*)
+ *sin* ((*Suc n*) ∗ *t*) ∗ (*LINT x*|*lebesgue-on* {−*pi*..*pi*}. *sin* ((*Suc n*) ∗ *x*) ∗ *f x*)
by (*simp add*: *cos-diff algebra-simps 1 2 flip*: *integral-mult-left-zero integral-mult-right-zero*)
finally have *LINT x*|*lebesgue-on* {−*pi*..*pi*}. *cos* ((*Suc n*) ∗ *x*) ∗ *f*(*x*+*t*)
= *cos* ((*Suc n*) ∗ *t*) ∗ (*LINT x*|*lebesgue-on* {−*pi*..*pi*}. *cos* ((*Suc n*) ∗ *x*) ∗ *f x*)
+ *sin* ((*Suc n*) ∗ *t*) ∗ (*LINT x*|*lebesgue-on* {−*pi*..*pi*}. *sin* ((*Suc n*) ∗ *x*) ∗ *f x*) **.**
then show *?thesis*
unfolding *Fourier-coefficient-def orthonormal-coeff-def trigonometric-set-def*
by (*simp add*: *eq*) (*simp add*: *divide-simps algebra-simps l2product-def*)
qed

lemma *Fourier-sum-offset*:
fixes *f* :: *real* ⇒ *real*
assumes *f*: *f absolutely-integrable-on* {−*pi*..*pi*} **and** *periodic*: ⋀*x*. *f*(*x* + *2*∗*pi*) = *f x*
shows (∑ *k*≤*2*∗*n*. *Fourier-coefficient f k* ∗ *trigonometric-set k t*) =
(∑ *k*≤*2*∗*n*. *Fourier-coefficient* (λ*x*. *f*(*x*+*t*)) *k* ∗ *trigonometric-set k 0*) (**is** *?lhs* = *?rhs*)
proof −
have *?lhs* = *Fourier-coefficient f 0* ∗ *trigonometric-set 0 t* + (∑ *k* = *Suc 0*..*2*∗*n*. *Fourier-coefficient f k* ∗ *trigonometric-set k t*)
by (*simp add*: *atMost-atLeast0 sum.atLeast-Suc-atMost*)
moreover have (∑ *k* = *Suc 0*..*2*∗*n*. *Fourier-coefficient f k* ∗ *trigonometric-set k t*) =
(∑ *k* = *Suc 0*..*2*∗*n*. *Fourier-coefficient* (λ*x*. *f*(*x*+*t*)) *k* ∗ *trigonometric-set k 0*)

proof (*cases n*)
case (*Suc n′*)
have ($\sum$ *k = Suc 0..2 ∗ n. Fourier-coefficient f k ∗ trigonometric-set k t*)
= ($\sum$ *k = Suc 0.. Suc* (*Suc* (*2 ∗ n′*)). *Fourier-coefficient f k ∗ trigonometric-set k t*)
by (*simp add*: *Suc*)
also have ... = ($\sum$ *k ≤ Suc* (*2 ∗ n′*). *Fourier-coefficient f*(*Suc k*) ∗ *trigonometric-set* (*Suc k*) *t*)
unfolding *atMost-atLeast0* **using** *sum.shift-bounds-cl-Suc-ivl* **by** *blast*
also have ... = ($\sum$ *k ≤ Suc* (*2 ∗ n′*). *Fourier-coefficient* (*λx. f*(*x+t*)) (*Suc k*) ∗ *trigonometric-set* (*Suc k*) *0*)
unfolding *sum.in-pairs-0*
proof (*rule sum.cong* [*OF refl*])
show *Fourier-coefficient f*(*Suc* (*2 ∗ k*)) ∗ *trigonometric-set* (*Suc* (*2 ∗ k*)) *t* + *Fourier-coefficient f*(*Suc* (*Suc* (*2 ∗ k*))) ∗ *trigonometric-set* (*Suc* (*Suc* (*2 ∗ k*))) *t* = *Fourier-coefficient* (*λx. f*(*x + t*)) (*Suc* (*2 ∗ k*)) ∗ *trigonometric-set* (*Suc* (*2 ∗ k*)) *0* + *Fourier-coefficient* (*λx. f*(*x + t*)) (*Suc* (*Suc* (*2 ∗ k*))) ∗ *trigonometric-set* (*Suc* (*Suc* (*2 ∗ k*))) *0*
if *k* ∈ {..*n′*} **for** *k*
using *that Fourier-offset-term* [*OF assms*, *of t*] **by** (*auto simp*: *trigonometric-set-def*)
qed
also have ... = ($\sum$ *k = Suc 0..Suc* (*Suc* (*2 ∗ n′*)). *Fourier-coefficient* (*λx. f*(*x+t*)) *k* ∗ *trigonometric-set k 0*)
by (*simp only*: *atMost-atLeast0 sum.shift-bounds-cl-Suc-ivl*)
also have ... = ($\sum$ *k = Suc 0..2∗n. Fourier-coefficient* (*λx. f*(*x+t*)) *k* ∗ *trigonometric-set k 0*)
by (*simp add*: *Suc*)
finally show *?thesis* **.**
qed *auto*
moreover have *?rhs*
= *Fourier-coefficient* (*λx. f*(*x+t*)) *0* ∗ *trigonometric-set 0 0* + ($\sum$ *k = Suc 0..2∗n. Fourier-coefficient* (*λx. f*(*x+t*)) *k* ∗ *trigonometric-set k 0*)
by (*simp add*: *atMost-atLeast0 sum.atLeast-Suc-atMost*)
moreover have *Fourier-coefficient f 0* ∗ *trigonometric-set 0 t*
= *Fourier-coefficient* (*λx. f*(*x+t*)) *0* ∗ *trigonometric-set 0 0*
by (*simp add*: *Fourier-coefficient-def orthonormal-coeff-def trigonometric-set-def l2product-def integral-periodic-offset periodic*)
ultimately show *?thesis*
by *simp*
qed

lemma *Fourier-sum-offset-unpaired*:
fixes *f* :: *real* ⇒ *real*
assumes *f*: *f absolutely-integrable-on* {−*pi*..*pi*} **and** *periodic*: ⋀*x. f*(*x* + *2∗pi*) = *f x*
shows ($\sum$ *k≤2∗n. Fourier-coefficient f k* ∗ *trigonometric-set k t*) =
($\sum$ *k≤n. Fourier-coefficient* (*λx. f*(*x+t*)) (*2∗k*) ∗ *trigonometric-set* (*2∗k*)

0)
  (**is** *?lhs* = *?rhs*)
proof −
  have *?lhs* = (∑ *k*≤*n*. *Fourier-coefficient* (λ*x*. *f*(*x*+*t*)) (*2*∗*k*) ∗ *trigonometric-set* (*2*∗*k*) *0* +
        Fourier-coefficient (λ*x*. *f*(*x*+*t*)) (*Suc* (*2*∗*k*)) ∗ *trigonometric-set* (*Suc* (*2*∗*k*)) *0*)
    apply (*simp add*: *assms Fourier-sum-offset*)
    apply (*subst sum.in-pairs-0* [*symmetric*])
    by (*simp add*: *trigonometric-set*)
  also have ... = *?rhs*
    by (*auto simp*: *trigonometric-set*)
  finally show *?thesis* **.**
qed

6.10 Express partial sums using Dirichlet kernel

definition *Dirichlet-kernel*
  where *Dirichlet-kernel* ≡
      λ*n x*. *if x* = *0 then real n* + *1/2*
          else sin((*real n* + *1/2*) ∗ *x*) / (*2* ∗ *sin*(*x*/*2*))

lemma *Dirichlet-kernel-0* [*simp*]:
  |*x*| < *2* ∗ *pi* ⟹ *Dirichlet-kernel 0 x* = *1/2*
  by (*auto simp*: *Dirichlet-kernel-def sin-zero-iff*)

lemma *Dirichlet-kernel-minus* [*simp*]: *Dirichlet-kernel n* (−*x*) = *Dirichlet-kernel n x*
  by (*auto simp*: *Dirichlet-kernel-def*)

lemma *Dirichlet-kernel-continuous-strong*:
  continuous-on {−(*2* ∗ *pi*)<..<*2* ∗ *pi*} (*Dirichlet-kernel n*)
proof −
  have *isCont* (*Dirichlet-kernel n*) *z* **if** −(*2* ∗ *pi*) < *z z* < *2* ∗ *pi* **for** *z*
  proof (*cases z*=*0*)
    case *True*
    have ∗: (λ*x*. *sin* ((*n* + *1/2*) ∗ *x*) / (*2* ∗ *sin* (*x*/*2*))) −*0*→ *real n* + *1/2*
      by *real-asymp*
    show *?thesis*
      unfolding *Dirichlet-kernel-def continuous-at True*
      apply (*rule Lim-transform-eventually* [**where** *f* = λ*x*. *sin*((*n* + *1/2*) ∗ *x*) / (*2* ∗ *sin*(*x*/*2*))])
       apply (*auto simp*: ∗ *eventually-at-filter*)
      done
  next
    case *False*
    have *continuous* (*at z*) (λ*x*. *sin*((*real n* + *1/2*) ∗ *x*) / (*2* ∗ *sin*(*x*/*2*)))
      using *that False* **by** (*intro continuous-intros*) (*auto simp*: *sin-zero-iff*)

moreover have $\forall_F$ *x in nhds z. x* $\neq$ *0*
using *False t1-space-nhds* **by** *blast*
ultimately show *?thesis*
using *False*
by (*force simp*: *Dirichlet-kernel-def continuous-at eventually-at-filter elim*: *Lim-transform-eventually*)
qed
then show *?thesis*
by (*simp add*: *continuous-on-eq-continuous-at*)
qed

lemma *Dirichlet-kernel-continuous*: *continuous-on* {−*pi*..*pi*} (*Dirichlet-kernel n*)
apply (*rule continuous-on-subset* [*OF Dirichlet-kernel-continuous-strong*], *clarsimp*)
using *pi-gt-zero* **by** *linarith*

lemma *absolutely-integrable-mult-Dirichlet-kernel*:
assumes *f absolutely-integrable-on* {−*pi*..*pi*}
shows (λ*x*. *Dirichlet-kernel n x* ∗ *f x*) *absolutely-integrable-on* {−*pi*..*pi*}
proof (*rule absolutely-integrable-bounded-measurable-product-real*)
show *Dirichlet-kernel n* ∈ *borel-measurable* (*lebesgue-on* {−*pi*..*pi*})
by (*simp add*: *Dirichlet-kernel-continuous continuous-imp-measurable-on-sets-lebesgue*)
have *compact* (*Dirichlet-kernel n* ' {−*pi*..*pi*})
by (*auto simp*: *compact-continuous-image* [*OF Dirichlet-kernel-continuous*])
then show *bounded* (*Dirichlet-kernel n* ' {−*pi*..*pi*})
using *compact-imp-bounded* **by** *blast*
qed (*use assms* **in** *auto*)

lemma *cosine-sum-lemma*:
(*1/2* + (∑ *k* = *Suc 0*..*n*. *cos*(*real k* ∗ *x*))) ∗ *sin*(*x/2*) = *sin*((*real n* + *1/2*) ∗ *x*) / *2*
proof −
consider *n=0* | *n* ≥ *1* **by** *linarith*
then show *?thesis*
proof *cases*
case *2*
then have (∑ *k* = *Suc 0*..*n*. (*sin* (*real k* ∗ *x* + *x/2*) ∗ *2* − *sin* (*real k* ∗ *x* − *x/2*) ∗ *2*) / *2*)
= *sin* (*real n* ∗ *x* + *x/2*) − *sin* (*x/2*)
proof (*induction n*)
case (*Suc n*)
show *?case*
proof (*cases n=0*)
case *False*
with *Suc* **show** *?thesis*
by (*simp add*: *divide-simps algebra-simps*)
qed *auto*

qed *auto*
then show *?thesis*
by (*simp add*: *distrib-right sum-distrib-right cos-times-sin*)
qed *auto*
qed

lemma *Dirichlet-kernel-cosine-sum*:
assumes $|x| < 2 * pi$
shows *Dirichlet-kernel n x* = $1/2 + (\sum k = Suc\ 0..n.\ cos(real\ k * x))$
proof −
have $sin\ ((real\ n + 1/2) * x) / (2 * sin\ (x/2)) = 1/2 + (\sum k = Suc\ 0..n.\ cos\ (real\ k * x))$
if $x \neq 0$
proof −
have $sin(x/2) \neq 0$
using *assms that* **by** (*auto simp*: *sin-zero-iff*)
then show *?thesis*
using *cosine-sum-lemma* [*of x n*] **by** (*simp add*: *divide-simps*)
qed
then show *?thesis*
by (*auto simp*: *Dirichlet-kernel-def*)
qed

lemma *integrable-Dirichlet-kernel*: *integrable* (*lebesgue-on* $\{-pi..pi\}$) (*Dirichlet-kernel n*)
using *Dirichlet-kernel-continuous continuous-imp-integrable-real* **by** *blast*

lemma *integral-Dirichlet-kernel* [*simp*]:
$integral^L$ (*lebesgue-on* $\{-pi..pi\}$) (*Dirichlet-kernel n*) = *pi*
proof −
have $integral^L$ (*lebesgue-on* $\{-pi..pi\}$) (*Dirichlet-kernel n*) = *LINT x|lebesgue-on* $\{-pi..pi\}.\ 1/2 + (\sum k = Suc\ 0..n.\ cos\ (k * x))$
using *pi-ge-two*
by (*force intro*: *Bochner-Integration.integral-cong* [*OF refl Dirichlet-kernel-cosine-sum*])
also have ... = (*LINT x|lebesgue-on* $\{-pi..pi\}.\ 1/2$) + (*LINT x|lebesgue-on* $\{-pi..pi\}.\ (\sum k = Suc\ 0..n.\ cos\ (k * x))$)
proof (*rule Bochner-Integration.integral-add*)
show *integrable* (*lebesgue-on* $\{-pi..pi\}$) ($\lambda x.\ \sum k = Suc\ 0..n.\ cos\ (real\ k * x)$)
by (*rule Bochner-Integration.integrable-sum*) (*metis integrable-cos-cx mult-commute-abs*)
qed *auto*
also have ... = *pi*
proof −
have $\bigwedge i.\ [\![Suc\ 0 \leq i;\ i \leq n]\!]$
$\Longrightarrow$ *integrable* (*lebesgue-on* $\{-pi..pi\}$) ($\lambda x.\ cos\ (real\ i * x)$)
by (*metis integrable-cos-cx mult-commute-abs*)
then have *LINT x|lebesgue-on* $\{-pi..pi\}.\ (\sum k = Suc\ 0..n.\ cos\ (real\ k * x)) = 0$
by (*simp add*: *Bochner-Integration.integral-sum*)

then show *?thesis*
by (*auto simp*: *measure-restrict-space*)
qed
finally show *?thesis* **.**
qed

lemma *integral-Dirichlet-kernel-half* [*simp*]:
integral^L (*lebesgue-on* {*0..pi*}) (*Dirichlet-kernel n*) = *pi/2*
proof −
have *integral*^{L} (*lebesgue-on* {− *pi..0*}) (*Dirichlet-kernel n*) +
integral^L (*lebesgue-on* {*0..pi*}) (*Dirichlet-kernel n*) = *pi*
using *integral-combine* [*OF integrable-Dirichlet-kernel*] *integral-Dirichlet-kernel* **by** *simp*
moreover have *integral*^{L} (*lebesgue-on* {− *pi..0*}) (*Dirichlet-kernel n*) = *integral*^{L} (*lebesgue-on* {*0..pi*}) (*Dirichlet-kernel n*)
using *integral-reflect-real* [*of pi 0 Dirichlet-kernel n*] **by** *simp*
ultimately show *?thesis*
by *simp*
qed

lemma *Fourier-sum-offset-Dirichlet-kernel*:
assumes *f*: *f absolutely-integrable-on* {−*pi..pi*} **and** *periodic*: ⋀*x. f*(*x* + *2*∗*pi*) = *f x*
shows
(∑ *k*≤*2*∗*n. Fourier-coefficient f k* ∗ *trigonometric-set k t*) =
integral^L (*lebesgue-on* {−*pi..pi*}) (λ*x. Dirichlet-kernel n x* ∗ *f*(*x*+*t*)) / *pi*
(**is** *?lhs* = *?rhs*)
proof −
have *ft*: (λ*x. f*(*x*+*t*)) *absolutely-integrable-on* {−*pi..pi*}
using *absolutely-integrable-periodic-offset* [*OF f*, *of t*] *periodic* **by** *simp*
have *?lhs* = (∑ *k*=*0..n. Fourier-coefficient* (λ*x. f*(*x*+*t*)) (*2*∗*k*) ∗ *trigonometric-set* (*2*∗*k*) *0*)
using *Fourier-sum-offset-unpaired assms atMost-atLeast0* **by** *auto*
also have . . . = *Fourier-coefficient* (λ*x. f*(*x*+*t*)) *0* / *sqrt* (*2* ∗ *pi*)
+ (∑ *k* = *Suc 0..n. Fourier-coefficient* (λ*x. f*(*x*+*t*)) (*2*∗*k*) / *sqrt pi*)
by (*simp add*: *sum.atLeast-Suc-atMost trigonometric-set-def*)
also have . . . = (*LINT x*|*lebesgue-on* {−*pi..pi*}. *f*(*x*+*t*)) / (*2* ∗ *pi*) +
(∑ *k* = *Suc 0..n.* (*LINT x*|*lebesgue-on* {−*pi..pi*}. *cos* (*real k* ∗ *x*) ∗ *f*(*x*+*t*)) / *pi*)
by (*simp add*: *Fourier-coefficient-def orthonormal-coeff-def trigonometric-set-def l2product-def*)
also have . . . = *LINT x*|*lebesgue-on* {−*pi..pi*}.
f(*x*+*t*) / (*2* ∗ *pi*) + (∑ *k* = *Suc 0..n.* (*cos* (*real k* ∗ *x*) ∗ *f*(*x*+*t*)) / *pi*)
using *Fourier-products-integrable-cos* [*OF ft*] *absolutely-integrable-imp-integrable* [*OF ft*] **by** *simp*
also have . . . = (*LINT x*|*lebesgue-on* {−*pi..pi*}.
f(*x*+*t*) / *2* + (∑ *k* = *Suc 0..n. cos* (*real k* ∗ *x*) ∗ *f*(*x*+*t*))) / *pi*

by (*simp add*: *divide-simps sum-distrib-right mult.assoc*)
also have ... = *?rhs*
proof −
have *LINT x|lebesgue-on* {−*pi..pi*}. *f*(*x* + *t*) / *2* + (∑ *k* = *Suc 0..n*. *cos* (*real k* ∗ *x*) ∗ *f*(*x* + *t*))
= *LINT x|lebesgue-on* {−*pi..pi*}. *Dirichlet-kernel n x* ∗ *f*(*x* + *t*)
proof −
have *eq*: *f*(*x*+*t*) / *2* + (∑ *k* = *Suc 0..n*. *cos* (*real k* ∗ *x*) ∗ *f*(*x* + *t*))
= *Dirichlet-kernel n x* ∗ *f*(*x* + *t*) **if** − *pi* ≤ *x x* ≤ *pi* **for** *x*
proof (*cases x* = *0*)
case *False*
then have *sin* (*x*/*2*) ≠ *0*
using *that* **by** (*auto simp*: *sin-zero-iff*)
then have *f*(*x* + *t*) ∗ (*1*/*2* + (∑ *k* = *Suc 0..n*. *cos*(*real k* ∗ *x*))) = *f*(*x* + *t*) ∗ *sin*((*real n* + *1*/*2*) ∗ *x*) / *2* / *sin*(*x*/*2*)
using *cosine-sum-lemma* [*of x n*] **by** (*simp add*: *divide-simps*)
then show *?thesis*
by (*simp add*: *Dirichlet-kernel-def False field-simps sum-distrib-left*)
qed (*simp add*: *Dirichlet-kernel-def algebra-simps*)
show *?thesis*
by (*rule Bochner-Integration.integral-cong* [*OF refl*]) (*simp add*: *eq*)
qed
then show *?thesis* **by** *simp*
qed
finally show *?thesis* **.**
qed

lemma *Fourier-sum-limit-Dirichlet-kernel*:
assumes *f*: *f absolutely-integrable-on* {−*pi..pi*} **and** *periodic*: ⋀*x*. *f*(*x* + *2*∗*pi*) = *f x*
shows ((λ*n*. (∑ *k*≤*n*. *Fourier-coefficient f k* ∗ *trigonometric-set k t*)) ⟶ *l*)
⟷ (λ*n*. *LINT x|lebesgue-on* {−*pi..pi*}. *Dirichlet-kernel n x* ∗ *f*(*x* + *t*)) ⟶ *pi* ∗ *l*
(**is** *?lhs* = *?rhs*)
proof −
have *?lhs* ⟷ (λ*n*. (*LINT x|lebesgue-on* {−*pi..pi*}. *Dirichlet-kernel n x* ∗ *f*(*x* + *t*)) / *pi*) ⟶ *l*
by (*simp add*: *Fourier-sum-limit-pair* [*OF f*, *symmetric*] *Fourier-sum-offset-Dirichlet-kernel assms*)
also have ... ⟷ (λ*k*. (*LINT x|lebesgue-on* {−*pi..pi*}. *Dirichlet-kernel k x* ∗ *f*(*x* + *t*)) ∗ *inverse pi*)
⟶ *pi* ∗ *l* ∗ *inverse pi*
by (*auto simp*: *divide-simps*)
also have ... ⟷ *?rhs*
using *tendsto-mult-right-iff* [*of inverse pi*, *symmetric*] **by** *auto*
finally show *?thesis* **.**
qed

6.11 A directly deduced sufficient condition for convergence at a point

lemma *simple-Fourier-convergence-periodic*:
assumes *f*: *f absolutely-integrable-on* $\{-pi..pi\}$
and *ft*: $(\lambda x.\ (f(x+t) - f\ t)\ /\ sin(x/2))$ *absolutely-integrable-on* $\{-pi..pi\}$
and *periodic*: $\bigwedge x.\ f(x + 2*pi) = f\ x$
shows $(\lambda n.\ (\sum k\leq n.\ Fourier\text{-}coefficient\ f\ k * trigonometric\text{-}set\ k\ t)) \longrightarrow f\ t$
proof −
let $?\Phi = \lambda n.\ \sum k\leq n.\ Fourier\text{-}coefficient\ (\lambda x.\ f\ x - f\ t)\ k * trigonometric\text{-}set\ k\ t$
let $?\Psi = \lambda n.\ LINT\ x|lebesgue\text{-}on\ \{-pi..pi\}.\ Dirichlet\text{-}kernel\ n\ x * (f(x + t) - f\ t)$
have *fft*: $(\lambda x.\ f\ x - f\ t)$ *absolutely-integrable-on* $\{-pi..pi\}$
by (*simp add*: *f absolutely-integrable-measurable-real*)
have *fxt*: $(\lambda x.\ f(x + t))$ *absolutely-integrable-on* $\{-pi..pi\}$
using *absolutely-integrable-periodic-offset assms* **by** *auto*
have ∗: $?\Phi \longrightarrow 0 \longleftrightarrow ?\Psi \longrightarrow 0$
by (*simp add*: *Fourier-sum-limit-Dirichlet-kernel fft periodic*)
moreover have $(\lambda n.\ (\sum k\leq n.\ Fourier\text{-}coefficient\ f\ k * trigonometric\text{-}set\ k\ t) - f\ t) \longrightarrow 0$
if $?\Phi \longrightarrow 0$
proof (*rule Lim-transform-eventually* [*OF that eventually-sequentiallyI*])
show $(\sum k\leq n.\ Fourier\text{-}coefficient\ (\lambda x.\ f\ x - f\ t)\ k * trigonometric\text{-}set\ k\ t)$
$= (\sum k\leq n.\ Fourier\text{-}coefficient\ f\ k * trigonometric\text{-}set\ k\ t) - f\ t$
if $Suc\ 0 \leq n$ **for** n
proof −
have $(\sum k = Suc\ 0..n.\ Fourier\text{-}coefficient\ (\lambda x.\ f\ x - f\ t)\ k * trigonometric\text{-}set\ k\ t)$
$= (\sum k = Suc\ 0..n.\ Fourier\text{-}coefficient\ f\ k * trigonometric\text{-}set\ k\ t)$
proof (*rule sum.cong* [*OF refl*])
fix *k*
assume *k*: $k \in \{Suc\ 0..n\}$
then have [*simp*]: $integral^L\ (lebesgue\text{-}on\ \{-pi..pi\})\ (trigonometric\text{-}set\ k) = 0$
by (*auto simp*: *trigonometric-set-def*)
show $Fourier\text{-}coefficient\ (\lambda x.\ f\ x - f\ t)\ k * trigonometric\text{-}set\ k\ t = Fourier\text{-}coefficient\ f\ k * trigonometric\text{-}set\ k\ t$
using *k* **unfolding** *Fourier-coefficient-def orthonormal-coeff-def l2product-def*
by (*simp add*: *right-diff-distrib f absolutely-integrable-measurable-real trigonometric-set-mul-integrable*)
qed
moreover have $Fourier\text{-}coefficient\ (\lambda x.\ f\ x - f\ t)\ 0 * trigonometric\text{-}set\ 0\ t =$
$Fourier\text{-}coefficient\ f\ 0 * trigonometric\text{-}set\ 0\ t - f\ t$
using *f* **unfolding** *Fourier-coefficient-def orthonormal-coeff-def l2product-def*
by (*auto simp*: *divide-simps trigonometric-set absolutely-integrable-imp-integrable measure-restrict-space*)
ultimately show *?thesis*
by (*simp add*: *sum.atLeast-Suc-atMost atMost-atLeast0*)

```
    qed
  qed
  moreover
  have ?Ψ ⟶ 0
  proof -
    have eq: ⋀n. ?Ψ n = integral^L (lebesgue-on {-pi..pi}) (λx. sin((n + 1/2) * x) * ((f(x+t) - f t) / sin(x/2) / 2))
      unfolding Dirichlet-kernel-def
      by (rule Bochner-Integration.integral-cong [OF refl]) (auto simp: divide-simps sin-zero-iff)
    show ?thesis
      unfolding eq
      by (intro ft set-integrable-divide Riemann-lebesgue-sin-half)
  qed
  ultimately show ?thesis
    by (simp add: LIM-zero-cancel)
qed
```

6.12 A more natural sufficient Hoelder condition at a point

```
lemma bounded-inverse-sin-half:
  assumes d > 0
  obtains B where B>0 ⋀x. x ∈ ({-pi..pi} - {-d<..<d}) ⟹ |inverse (sin (x/2))| ≤ B
proof -
  have bounded ((λx. inverse (sin (x/2))) ` ({-pi..pi} - {- d<..<d}))
  proof (intro compact-imp-bounded compact-continuous-image)
    have ⟦x ∈ {-pi..pi}; x ≠ 0⟧ ⟹ sin (x/2) ≠ 0 for x
      using ‹0 < d› by (auto simp: sin-zero-iff)
    then show continuous-on ({-pi..pi} - {-d<..<d}) (λx. inverse (sin (x/2)))
      using ‹0 < d› by (intro continuous-intros) auto
    show compact ({-pi..pi} - {-d<..<d})
      by (simp add: compact-diff)
  qed
  then show thesis
    using that by (auto simp: bounded-pos)
qed

proposition Hoelder-Fourier-convergence-periodic:
  assumes f: f absolutely-integrable-on {-pi..pi} and d > 0 a > 0
    and ft: ⋀x. |x-t| < d ⟹ |f x - f t| ≤ M * |x-t| powr a
    and periodic: ⋀x. f(x + 2*pi) = f x
  shows (λn. (∑k≤n. Fourier-coefficient f k * trigonometric-set k t)) ⟶ f t
proof (rule simple-Fourier-convergence-periodic [OF f])
  have half: ((λx. sin(x/2)) has-real-derivative 1/2) (at 0)
    by (rule derivative-eq-intros refl | force)+
  then obtain e0::real where e0 > 0 and e0: ⋀x. ⟦x ≠ 0; |x| < e0⟧ ⟹ |sin (x/2) / x - 1/2| < 1/4
    apply (simp add: DERIV-def Lim-at dist-real-def)
```

```
    apply (drule-tac x=1/4 in spec, auto)
    done
  obtain e where e > 0 and e: ⋀x. |x| < e ⟹ |(f(x+t) − f t) / sin (x/2)| ≤
4 ∗ (|M| ∗ |x| powr (a−1))
  proof
    show min d e0 > 0
      using ‹d > 0› ‹e0 > 0› by auto
  next
    fix x
    assume x: |x| < min d e0
    show |(f(x + t) − f t) / sin (x/2)| ≤ 4 ∗ (|M| ∗ |x| powr (a − 1))
    proof (cases sin(x/2) = 0 ∨ x=0)
      case False
      have eq: |(f(x + t) − f t) / sin (x/2)| = |inverse (sin (x/2) / x)| ∗ (|f(x +
t) − f t| / |x|)
        by simp
      show ?thesis
        unfolding eq
      proof (rule mult-mono)
        have |sin (x/2) / x − 1/2| < 1/4
          using e0 [of x] x False by force
        then have 1/4 ≤ |sin (x/2) / x|
          by (simp add: abs-if field-simps split: if-split-asm)
        then show |inverse (sin (x/2) / x)| ≤ 4
          using False by (simp add: field-simps)
        have |f(x + t) − f t| / |x| ≤ M ∗ |x + t − t| powr (a − 1)
          using ft [of x+t] x by (auto simp: divide-simps algebra-simps Transcen-
dental.powr-mult-base)
        also have . . . ≤ |M| ∗ |x| powr (a − 1)
          by (simp add: mult-mono)
        finally show |f(x + t) − f t| / |x| ≤ |M| ∗ |x| powr (a − 1) .
      qed auto
    qed auto
  qed
  obtain B where B>0 and B: ⋀x. x ∈ ({−pi..pi} − {− e<..<e}) ⟹ |inverse
(sin (x/2))| ≤ B
    using bounded-inverse-sin-half [OF ‹e > 0›] by metis
  let ?g = λx. max (B ∗ |f(x + t) − f t|) ((4 ∗ |M|) ∗ |x| powr (a − 1))
  show (λx. (f(x + t) − f t) / sin (x/2)) absolutely-integrable-on {−pi..pi}
  proof (rule measurable-bounded-by-integrable-imp-absolutely-integrable)
    have fxt: (λx. f(x + t)) absolutely-integrable-on {−pi..pi}
      using absolutely-integrable-periodic-offset assms by auto
    show (λx. (f(x + t) − f t) / sin (x/2)) ∈ borel-measurable (lebesgue-on
{−pi..pi})
    proof (intro measurable)
      show (λx. f(x + t)) ∈ borel-measurable (lebesgue-on {−pi..pi})
        using absolutely-integrable-on-def fxt integrable-imp-measurable by blast
      show (λx. sin (x/2)) ∈ borel-measurable (lebesgue-on {−pi..pi})
        by (intro continuous-imp-measurable-on-sets-lebesgue continuous-intros) auto
```

```
    qed auto
    have (λx. f(x + t) − f t) absolutely-integrable-on {−pi..pi}
      by (intro set-integral-diff fxt) (simp add: set-integrable-def)
    moreover
    have (λx. |x| powr (a − 1)) absolutely-integrable-on {−pi..pi}
    proof −
      have ((λx. x powr (a − 1)) has-integral
       inverse a ∗ pi powr a − inverse a ∗ 0 powr a)
       {0..pi}
      proof (rule fundamental-theorem-of-calculus-interior)
        show continuous-on {0..pi} (λx. inverse a ∗ x powr a)
          using ‹a > 0›
          by (intro continuous-on-powr′ continuous-intros) auto
        show ((λb. inverse a ∗ b powr a) has-vector-derivative x powr (a − 1)) (at
x)
          if x ∈ {0<..<pi} for x
          using that ‹a > 0›
          apply (simp flip: has-real-derivative-iff-has-vector-derivative)
          apply (rule derivative-eq-intros | simp)+
          done
      qed auto
      then have int: (λx. x powr (a − 1)) integrable-on {0..pi}
        by blast
      show ?thesis
      proof (rule nonnegative-absolutely-integrable-1)
        show (λx. |x| powr (a − 1)) integrable-on {−pi..pi}
        proof (rule Henstock-Kurzweil-Integration.integrable-combine)
          show (λx. |x| powr (a − 1)) integrable-on {0..pi}
            using int integrable-eq by fastforce
          then show (λx. |x| powr (a − 1)) integrable-on {− pi..0}
            using Henstock-Kurzweil-Integration.integrable-reflect-real by fastforce
        qed auto
      qed auto
    qed
    ultimately show ?g integrable-on {−pi..pi}
      apply (intro set-lebesgue-integral-eq-integral absolutely-integrable-max-1 abso-
lutely-integrable-bounded-measurable-product-real set-integrable-abs measurable)
            apply (auto simp: image-constant-conv)
      done
    show norm ((f(x + t) − f t) / sin (x/2)) ≤ ?g x if x ∈ {−pi..pi} for x
    proof (cases |x| < e)
      case True
      then show ?thesis
        using e [OF True] by (simp add: max-def)
    next
      case False
      then have |inverse (sin (x/2))| ≤ B
        using B that by force
      then have |inverse (sin (x/2))| ∗ |f(x + t) − f t| ≤ B ∗ |f(x + t) − f t|
```

by (*simp add*: *mult-right-mono*)
then have $|f(x + t) - f\ t| / |sin\ (x/2)| \le B * |f(x + t) - f\ t|$
by (*simp add*: *divide-simps split*: *if-split-asm*)
then show *?thesis*
by *auto*
qed
qed *auto*
qed (*auto simp*: *periodic*)

In particular, a Lipschitz condition at the point

corollary *Lipschitz-Fourier-convergence-periodic*:
assumes *f*: *f absolutely-integrable-on* $\{-pi..pi\}$ **and** $d > 0$
and *ft*: $\bigwedge x.\ |x-t| < d \Longrightarrow |f\ x - f\ t| \le M * |x-t|$
and *periodic*: $\bigwedge x.\ f(x + 2*pi) = f\ x$
shows ($\lambda n.$ ($\sum k \le n.$ *Fourier-coefficient f k* $*$ *trigonometric-set k t*)) $\longlongrightarrow f\ t$
using *Hoelder-Fourier-convergence-periodic* [*OF f* ‹$d > 0$›, *of 1*] *assms*
by (*fastforce simp add*:)

In particular, if left and right derivatives both exist

proposition *bi-differentiable-Fourier-convergence-periodic*:
assumes *f*: *f absolutely-integrable-on* $\{-pi..pi\}$
and *f-lt*: *f differentiable at-left t*
and *f-gt*: *f differentiable at-right t*
and *periodic*: $\bigwedge x.\ f(x + 2*pi) = f\ x$
shows ($\lambda n.$ ($\sum k \le n.$ *Fourier-coefficient f k* $*$ *trigonometric-set k t*)) $\longlongrightarrow f\ t$
proof −
obtain *D-lt* **where** *D-lt*: $\bigwedge e.\ e > 0 \Longrightarrow \exists\, d>0.\ \forall\, x<t.\ 0 < dist\ x\ t \wedge dist\ x\ t < d \longrightarrow dist\ ((f\ x - f\ t) / (x - t))\ D\text{-}lt < e$
using *f-lt* **unfolding** *has-field-derivative-iff real-differentiable-def Lim-within*
by (*meson lessThan-iff*)
then obtain *d-lt* **where** $d\text{-}lt > 0$
and *d-lt*: $\bigwedge x.\ [\![x < t;\ 0 < |x - t|;\ |x - t| < d\text{-}lt]\!] \Longrightarrow |(f\ x - f\ t) / (x - t) - D\text{-}lt| < 1$
unfolding *dist-real-def* **by** (*metis dist-commute dist-real-def zero-less-one*)
obtain *D-gt* **where** *D-gt*: $\bigwedge e.\ e > 0 \Longrightarrow \exists\, d>0.\ \forall\, x>t.\ 0 < dist\ x\ t \wedge dist\ x\ t < d \longrightarrow dist\ ((f\ x - f\ t) / (x - t))\ D\text{-}gt < e$
using *f-gt* **unfolding** *has-field-derivative-iff real-differentiable-def Lim-within*
by (*meson greaterThan-iff*)
then obtain *d-gt* **where** $d\text{-}gt > 0$
and *d-gt*: $\bigwedge x.\ [\![x > t;\ 0 < |x - t|;\ |x - t| < d\text{-}gt]\!] \Longrightarrow |(f\ x - f\ t) / (x - t) - D\text{-}gt| < 1$
unfolding *dist-real-def* **by** (*metis dist-commute dist-real-def zero-less-one*)
show *?thesis*
proof (*rule Lipschitz-Fourier-convergence-periodic* [*OF f*])
fix *x*
assume $|x - t| < min\ d\text{-}lt\ d\text{-}gt$
then have $*$: $|x - t| < d\text{-}lt$ $|x - t| < d\text{-}gt$
by *auto*
consider $x<t$ | $x=t$ | $x>t$

```
      by linarith
    then show |f x − f t| ≤ (|D-lt| + |D-gt| + 1)  ∗ |x − t|
    proof cases
      case 1
      then have |(f x − f t) / (x − t) − D-lt| < 1
        using d-lt [OF 1] ∗ by auto
      then have |(f x − f t) / (x − t)| − |D-lt| < 1
        by linarith
      then have |f x − f t| ≤ (|D-lt| + 1) ∗ |x − t|
        by (simp add: comm-semiring-class.distrib divide-simps split: if-split-asm)
      also have ... ≤ (|D-lt| + |D-gt| + 1) ∗ |x − t|
        by (simp add: mult-right-mono)
      finally show ?thesis .
    next
      case 3
      then have |(f x − f t) / (x − t) − D-gt| < 1
        using d-gt [OF 3] ∗ by auto
      then have |(f x − f t) / (x − t)| − |D-gt| < 1
        by linarith
      then have |f x − f t| ≤ (|D-gt| + 1) ∗ |x − t|
        by (simp add: comm-semiring-class.distrib divide-simps split: if-split-asm)
      also have ... ≤ (|D-lt| + |D-gt| + 1) ∗ |x − t|
        by (simp add: mult-right-mono)
      finally show ?thesis .
    qed auto
  qed (auto simp: ‹0 < d-gt› ‹0 < d-lt› periodic)
qed
```

And in particular at points where the function is differentiable

```
lemma differentiable-Fourier-convergence-periodic:
  assumes f: f absolutely-integrable-on {−pi..pi}
    and fdif: f differentiable (at t)
    and periodic: ⋀x. f(x + 2∗pi) = f x
  shows (λn. (∑ k≤n. Fourier-coefficient f k ∗ trigonometric-set k t)) ⟶ f t
  by (rule bi-differentiable-Fourier-convergence-periodic) (auto simp: differentiable-at-withinI
assms)
```

Use reflection to halve the region of integration

```
lemma absolutely-integrable-mult-Dirichlet-kernel-reflected:
  assumes f: f absolutely-integrable-on {−pi..pi}
    and periodic: ⋀x. f(x + 2∗pi) = f x
  shows (λx. Dirichlet-kernel n x ∗ f(t+x)) absolutely-integrable-on {−pi..pi}
        (λx. Dirichlet-kernel n x ∗ f(t−x)) absolutely-integrable-on {−pi..pi}
        (λx. Dirichlet-kernel n x ∗ c) absolutely-integrable-on {−pi..pi}
proof −
  show (λx. Dirichlet-kernel n x ∗ f(t+x)) absolutely-integrable-on {−pi..pi}
    apply (rule absolutely-integrable-mult-Dirichlet-kernel)
    using absolutely-integrable-periodic-offset [OF f, of t] periodic
    apply simp
```

done
then show (*λx. Dirichlet-kernel n x* ∗ *f*(*t*−*x*)) *absolutely-integrable-on* {−*pi*..*pi*}
by (*subst absolutely-integrable-reflect-real* [*symmetric*]) *simp*
show (*λx. Dirichlet-kernel n x* ∗ *c*) *absolutely-integrable-on* {−*pi*..*pi*}
by (*simp add*: *absolutely-integrable-measurable-real borel-measurable-integrable integrable-Dirichlet-kernel*)
qed

lemma *absolutely-integrable-mult-Dirichlet-kernel-reflected-part*:
assumes *f*: *f absolutely-integrable-on* {−*pi*..*pi*}
and *periodic*: ⋀*x*. *f*(*x* + *2*∗*pi*) = *f x* **and** *d* ≤ *pi*
shows (*λx. Dirichlet-kernel n x* ∗ *f*(*t*+*x*)) *absolutely-integrable-on* {*0*..*d*}
(*λx. Dirichlet-kernel n x* ∗ *f*(*t*−*x*)) *absolutely-integrable-on* {*0*..*d*}
(*λx. Dirichlet-kernel n x* ∗ *c*) *absolutely-integrable-on* {*0*..*d*}
using *absolutely-integrable-mult-Dirichlet-kernel-reflected* [*OF f periodic*, *of n*] ‹*d* ≤ *pi*›
by (*force intro*: *absolutely-integrable-on-subinterval*)+

lemma *absolutely-integrable-mult-Dirichlet-kernel-reflected-part2*:
assumes *f*: *f absolutely-integrable-on* {−*pi*..*pi*}
and *periodic*: ⋀*x*. *f*(*x* + *2*∗*pi*) = *f x* **and** *d* ≤ *pi*
shows (*λx. Dirichlet-kernel n x* ∗ (*f*(*t*+*x*) + *f*(*t*−*x*))) *absolutely-integrable-on* {*0*..*d*}
(*λx. Dirichlet-kernel n x* ∗ ((*f*(*t*+*x*) + *f*(*t*−*x*)) − *c*)) *absolutely-integrable-on* {*0*..*d*}
using *absolutely-integrable-mult-Dirichlet-kernel-reflected-part assms*
by (*simp-all add*: *distrib-left right-diff-distrib*)

lemma *integral-reflect-and-add*:
fixes *f* :: *real* ⇒ *'b*::*euclidean-space*
assumes *integrable* (*lebesgue-on* {−*a*..*a*}) *f*
shows $integral^L$ (*lebesgue-on* {−*a*..*a*}) *f* = $integral^L$ (*lebesgue-on* {*0*..*a*}) (*λx. f x* + *f*(−*x*))
proof (*cases a* ≥ *0*)
case *True*
have *f*: *integrable* (*lebesgue-on* {*0*..*a*}) *f integrable* (*lebesgue-on* {−*a*..*0*}) *f*
using *integrable-subinterval assms* **by** *fastforce*+
then have *fm*: *integrable* (*lebesgue-on* {*0*..*a*}) (*λx. f*(−*x*))
using *integrable-reflect-real* **by** *fastforce*
have $integral^L$ (*lebesgue-on* {−*a*..*a*}) *f* = $integral^L$ (*lebesgue-on* {−*a*..*0*}) *f* + $integral^L$ (*lebesgue-on* {*0*..*a*}) *f*
by (*simp add*: *True assms integral-combine*)
also have ... = $integral^L$ (*lebesgue-on* {*0*..*a*}) (*λx. f*(−*x*)) + $integral^L$ (*lebesgue-on* {*0*..*a*}) *f*
by (*metis* (*no-types*) *add.inverse-inverse add.inverse-neutral integral-reflect-real*)
also have ... = $integral^L$ (*lebesgue-on* {*0*..*a*}) (*λx. f x* + *f*(−*x*))
by (*simp add*: *fm f*)
finally show *?thesis* **.**

qed (*use assms* **in** *auto*)

lemma *Fourier-sum-offset-Dirichlet-kernel-half*:
assumes *f*: *f absolutely-integrable-on* {−*pi*..*pi*}
and *periodic*: ⋀*x*. *f*(*x* + *2*∗*pi*) = *f x*
shows (∑ *k*≤*2*∗*n*. *Fourier-coefficient f k* ∗ *trigonometric-set k t*) − *l*
= (*LINT x*|*lebesgue-on* {*0*..*pi*}. *Dirichlet-kernel n x* ∗ (*f*(*t*+*x*) + *f*(*t*−*x*) − *2*∗*l*)) / *pi*
proof −
have *fxt*: (λ*x*. *f*(*x* + *t*)) *absolutely-integrable-on* {−*pi*..*pi*}
using *absolutely-integrable-periodic-offset assms* **by** *auto*
have *int*: *integrable* (*lebesgue-on* {*0*..*pi*}) (*Dirichlet-kernel n*)
using *not-integrable-integral-eq* **by** *fastforce*
have *LINT x*|*lebesgue-on* {−*pi*..*pi*}. *Dirichlet-kernel n x* ∗ *f*(*x* + *t*)
= *LINT x*|*lebesgue-on* {*0*..*pi*}. *Dirichlet-kernel n x* ∗ *f*(*x* + *t*) + *Dirichlet-kernel n* (− *x*) ∗ *f*(− *x* + *t*)
by (*simp add*: *integral-reflect-and-add absolutely-integrable-imp-integrable absolutely-integrable-mult-Dirichlet-kernel fxt*)
also have ... = (*LINT x*|*lebesgue-on* {*0*..*pi*}. *Dirichlet-kernel n x* ∗ (*f*(*t*+*x*) + *f*(*t*−*x*) − *2*∗*l*)) + *pi* ∗ *l*
using *absolutely-integrable-mult-Dirichlet-kernel-reflected-part* [*OF f periodic order-refl*, *of n t*]
apply (*simp add*: *algebra-simps absolutely-integrable-imp-integrable int*)
done
finally show *?thesis*
by (*simp add*: *Fourier-sum-offset-Dirichlet-kernel assms divide-simps*)
qed

lemma *Fourier-sum-limit-Dirichlet-kernel-half*:
assumes *f*: *f absolutely-integrable-on* {−*pi*..*pi*}
and *periodic*: ⋀*x*. *f*(*x* + *2*∗*pi*) = *f x*
shows (λ*n*. (∑ *k*≤*n*. *Fourier-coefficient f k* ∗ *trigonometric-set k t*)) ⟶ *l*
⟷ (λ*n*. (*LINT x*|*lebesgue-on* {*0*..*pi*}. *Dirichlet-kernel n x* ∗ (*f*(*t*+*x*) + *f*(*t*−*x*) − *2*∗*l*))) ⟶ *0*
apply (*simp flip*: *Fourier-sum-limit-pair* [*OF f*])
apply (*simp add*: *Lim-null* [**where** *l*=*l*] *Fourier-sum-offset-Dirichlet-kernel-half assms*)
done

6.13 Localization principle: convergence only depends on values "nearby"

proposition *Riemann-localization-integral*:
assumes *f*: *f absolutely-integrable-on* {−*pi*..*pi*} **and** *g*: *g absolutely-integrable-on* {−*pi*..*pi*}
and *d* > *0* **and** *d*: ⋀*x*. |*x*| < *d* ⟹ *f x* = *g x*
shows (λ*n*. $integral^L$ (*lebesgue-on* {−*pi*..*pi*}) (λ*x*. *Dirichlet-kernel n x* ∗ *f x*)
− $integral^L$ (*lebesgue-on* {−*pi*..*pi*}) (λ*x*. *Dirichlet-kernel n x* ∗ *g x*))
⟶ *0* (**is** *?a* ⟶ *0*)

```
proof –
  let ?f = λx. (if |x| < d then 0 else f x − g x) / sin(x/2) / 2
  have eq: ?a n = integral^L (lebesgue-on {−pi..pi}) (λx. sin((n+1/2) ∗ x) ∗ ?f x)
for n
  proof –
    have ?a n = integral^L (lebesgue-on {−pi..pi}) (λx. Dirichlet-kernel n x ∗ (if
|x| < d then 0 else f x − g x))
    apply (simp add: absolutely-integrable-imp-integrable absolutely-integrable-mult-Dirichlet-kernel
f g flip: Bochner-Integration.integral-diff)
      apply (auto simp: d algebra-simps intro: Bochner-Integration.integral-cong)
      done
    also have . . . = integral^L (lebesgue-on {−pi..pi}) (λx. sin((n+1/2) ∗ x) ∗ ?f
x)
    using ‹d > 0› by (auto simp: Dirichlet-kernel-def intro: Bochner-Integration.integral-cong)
    finally show ?thesis .
  qed
  show ?thesis
    unfolding eq
  proof (rule Riemann-lebesgue-sin-half)
    obtain B where B>0 and B: ⋀x. x ∈ ({−pi..pi} − {−d<..<d}) ⟹ |inverse
(sin (x/2))| ≤ B
      using bounded-inverse-sin-half [OF ‹d > 0›] by metis
    have (λx. (if |x| < d then 0 else f x − g x) / sin (x/2)) absolutely-integrable-on
{−pi..pi}
    proof (rule measurable-bounded-by-integrable-imp-absolutely-integrable)
      show (λx. (if |x| < d then 0 else f x − g x) / sin (x/2)) ∈ borel-measurable
(lebesgue-on {−pi..pi})
      proof (intro measurable)
          show f ∈ borel-measurable (lebesgue-on {−pi..pi}) g ∈ borel-measurable
(lebesgue-on {−pi..pi})
          using absolutely-integrable-on-def f g integrable-imp-measurable by blast+
        show (λx. x) ∈ borel-measurable (lebesgue-on {−pi..pi})
             (λx. sin (x/2)) ∈ borel-measurable (lebesgue-on {−pi..pi})
          by (intro continuous-intros continuous-imp-measurable-on-sets-lebesgue |
force)+
      qed auto
      have (λx. B ∗ |f x − g x|) absolutely-integrable-on {−pi..pi}
        using ‹B > 0› by (simp add: f g set-integrable-abs)
      then show (λx. B ∗ |f x − g x|) integrable-on {−pi..pi}
        using absolutely-integrable-on-def by blast
    next
      fix x
      assume x: x ∈ {−pi..pi}
      then have [simp]: sin (x/2) = 0 ⟷ x=0
        using ‹0 < d› by (force simp: sin-zero-iff)
      show norm ((if |x| < d then 0 else f x − g x) / sin (x/2)) ≤ B ∗ |f x − g x|
      proof (cases |x| < d)
        case False
        then have 1 ≤ B ∗ |sin (x/2)|
```

using ‹B > 0› B [of x] x ‹d > 0›
by (auto simp: abs-less-iff divide-simps split: if-split-asm)
then have 1 ∗ |f x − g x| ≤ B ∗ |sin (x/2)| ∗ |f x − g x|
by (metis (full-types) abs-ge-zero mult.commute mult-left-mono)
then show ?thesis
using False by (auto simp: divide-simps algebra-simps)
qed (simp add: d)
qed auto
then show (λx. (if |x| < d then 0 else f x − g x) / sin (x/2) / 2) absolutely-integrable-on {−pi..pi}
using set-integrable-divide by blast
qed
qed

lemma *Riemann-localization-integral-range*:
assumes *f*: *f absolutely-integrable-on* {−*pi*..*pi*}
and *0* < *d d* ≤ *pi*
shows (λ*n*. *integral*^{L} (*lebesgue-on* {−*pi*..*pi*}) (λ*x*. *Dirichlet-kernel n x* ∗ *f x*)
− *integral*^{L} (*lebesgue-on* {−*d*..*d*}) (λ*x*. *Dirichlet-kernel n x* ∗ *f x*))
⟶ *0*
proof −
have ∗: (λ*n*. (*LINT x*|*lebesgue-on* {−*pi*..*pi*}. *Dirichlet-kernel n x* ∗ *f x*)
− (*LINT x*|*lebesgue-on* {−*pi*..*pi*}. *Dirichlet-kernel n x* ∗ (*if x* ∈ {−*d*..*d*} *then f x else 0*)))
⟶ *0*
proof (*rule Riemann-localization-integral* [*OF f* - ‹*0* < *d*›])
have *f absolutely-integrable-on* {−*d*..*d*}
using *f absolutely-integrable-on-subinterval* ‹*d* ≤ *pi*› **by** *fastforce*
moreover have (λ*x*. *if* − *pi* ≤ *x* ∧ *x* ≤ *pi then if x* ∈ {−*d*..*d*} *then f x else 0 else 0*)
= (λ*x*. *if x* ∈ {−*d*..*d*} *then f x else 0*)
using ‹*d* ≤ *pi*› **by** *force*
ultimately
show (λ*x*. *if x* ∈ {−*d*..*d*} *then f x else 0*) *absolutely-integrable-on* {−*pi*..*pi*}
apply (*subst absolutely-integrable-restrict-UNIV* [*symmetric*])
apply (*simp flip*: *absolutely-integrable-restrict-UNIV* [*of* {−*d*..*d*} *f*])
done
qed *auto*
then show *?thesis*
using *integral-restrict* [*of* {−*d*..*d*} {−*pi*..*pi*} λ*x*. *Dirichlet-kernel* - *x* ∗ *f x*] *assms*
by (*simp add*: *if-distrib cong*: *if-cong*)
qed

lemma *Riemann-localization*:
assumes *f*: *f absolutely-integrable-on* {−*pi*..*pi*} **and** *g*: *g absolutely-integrable-on* {−*pi*..*pi*}
and *perf*: ⋀*x*. *f*(*x* + *2*∗*pi*) = *f x*
and *perg*: ⋀*x*. *g*(*x* + *2*∗*pi*) = *g x*

and *d* > *0* **and** *d*: ⋀*x*. |*x*−*t*| < *d* ⟹ *f x* = *g x*
shows (λ*n*. ∑ *k*≤*n*. *Fourier-coefficient f k* ∗ *trigonometric-set k t*) ⟶ *c*
⟷ (λ*n*. ∑ *k*≤*n*. *Fourier-coefficient g k* ∗ *trigonometric-set k t*) ⟶ *c*
proof −
have (λ*n*. *LINT x*|*lebesgue-on* {−*pi*..*pi*}. *Dirichlet-kernel n x* ∗ *f*(*x* + *t*)) ⟶ *pi* ∗ *c*
⟷ (λ*n*. *LINT x*|*lebesgue-on* {−*pi*..*pi*}. *Dirichlet-kernel n x* ∗ *g* (*x* + *t*)) ⟶ *pi* ∗ *c*
proof (*intro Lim-transform-eq Riemann-localization-integral*)
show (λ*x*. *f*(*x* + *t*)) *absolutely-integrable-on* {−*pi*..*pi*} (λ*x*. *g* (*x* + *t*)) *absolutely-integrable-on* {−*pi*..*pi*}
using *assms* **by** (*auto intro*: *absolutely-integrable-periodic-offset*)
qed (*use assms* **in** *auto*)
then show *?thesis*
by (*simp add*: *Fourier-sum-limit-Dirichlet-kernel assms*)
qed

6.14 Localize the earlier integral

lemma *Riemann-localization-integral-range-half*:
assumes *f*: *f absolutely-integrable-on* {−*pi*..*pi*}
and *0* < *d d* ≤ *pi*
shows (λ*n*. (*LINT x*|*lebesgue-on* {*0*..*pi*}. *Dirichlet-kernel n x* ∗ (*f x* + *f*(−*x*)))
− (*LINT x*|*lebesgue-on* {*0*..*d*}. *Dirichlet-kernel n x* ∗ (*f x* + *f*(−*x*))))
⟶ *0*
proof −
have ∗: (λ*x*. *Dirichlet-kernel n x* ∗ *f x*) *absolutely-integrable-on* {−*d*..*d*} **for** *n*
by (*metis* (*full-types*) *absolutely-integrable-mult-Dirichlet-kernel absolutely-integrable-on-subinterval*
‹*d* ≤ *pi*› *atLeastatMost-subset-iff f neg-le-iff-le*)
show *?thesis*
using *absolutely-integrable-mult-Dirichlet-kernel* [*OF f*]
using *Riemann-localization-integral-range* [*OF assms*]
apply (*simp add*: ∗ *absolutely-integrable-imp-integrable integral-reflect-and-add*)
apply (*simp add*: *algebra-simps*)
done
qed

lemma *Fourier-sum-limit-Dirichlet-kernel-part*:
assumes *f*: *f absolutely-integrable-on* {−*pi*..*pi*}
and *periodic*: ⋀*x*. *f*(*x* + *2*∗*pi*) = *f x*
and *d*: *0* < *d d* ≤ *pi*
shows (λ*n*. ∑ *k*≤*n*. *Fourier-coefficient f k* ∗ *trigonometric-set k t*) ⟶ *l*
⟷ (λ*n*. (*LINT x*|*lebesgue-on* {*0*..*d*}. *Dirichlet-kernel n x* ∗ ((*f*(*t*+*x*) + *f*(*t*−*x*)) − *2*∗*l*))) ⟶ *0*
proof −
have (λ*x*. *f*(*t*+*x*)) *absolutely-integrable-on* {−*pi*..*pi*}
using *absolutely-integrable-periodic-offset* [*OF f*, *of t*] *periodic* **by** *simp*
then have *int*: (λ*x*. *f*(*t*+*x*) − *l*) *absolutely-integrable-on* {−*pi*..*pi*}

by *auto*
have ($\lambda n.$ *LINT x|lebesgue-on* {*0..pi*}. *Dirichlet-kernel n x* ∗ (*f*(*t*+*x*) + *f*(*t*−*x*) − *2*∗*l*)) ⟶ *0*
⟷ ($\lambda n.$ *LINT x|lebesgue-on* {*0..d*}. *Dirichlet-kernel n x* ∗ (*f*(*t*+*x*) + *f*(*t*−*x*) − *2*∗*l*)) ⟶ *0*
by (*rule Lim-transform-eq*) (*use Riemann-localization-integral-range-half* [*OF int d*] **in** *auto*)
then show *?thesis*
by (*simp add*: *Fourier-sum-limit-Dirichlet-kernel-half assms*)
qed

6.15 Make a harmless simplifying tweak to the Dirichlet kernel

lemma *inte-Dirichlet-kernel-mul-expand*:
assumes *f*: *f* ∈ *borel-measurable* (*lebesgue-on S*) **and** *S*: *S* ∈ *sets lebesgue*
shows (*LINT x|lebesgue-on S*. *Dirichlet-kernel n x* ∗ *f x*
= *LINT x|lebesgue-on S*. *sin*((*n*+*1/2*) ∗ *x*) ∗ *f x* / (*2* ∗ *sin*(*x/2*)))
∧ (*integrable* (*lebesgue-on S*) ($\lambda x.$ *Dirichlet-kernel n x* ∗ *f x*)
⟷ *integrable* (*lebesgue-on S*) ($\lambda x.$ *sin*((*n*+*1/2*) ∗ *x*) ∗ *f x* / (*2* ∗ *sin*(*x/2*))))
proof (*cases 0* ∈ *S*)
case *True*
have ∗: {*x*. *x* = *0* ∧ *x* ∈ *S*} ∈ *sets* (*lebesgue-on S*)
using *True* **by** (*simp add*: *S sets-restrict-space-iff cong*: *conj-cong*)
have *bm1*: ($\lambda x.$ *Dirichlet-kernel n x* ∗ *f x*) ∈ *borel-measurable* (*lebesgue-on S*)
unfolding *Dirichlet-kernel-def*
by (*force intro*: ∗ *assms measurable continuous-imp-measurable-on-sets-lebesgue continuous-intros*)
have *bm2*: ($\lambda x.$ *sin* ((*n* + *1/2*) ∗ *x*) ∗ *f x* / (*2* ∗ *sin* (*x/2*))) ∈ *borel-measurable* (*lebesgue-on S*)
by (*intro assms measurable continuous-imp-measurable-on-sets-lebesgue continuous-intros*) *auto*
have *0*: {*0*} ∈ *null-sets* (*lebesgue-on S*)
using *True* **by** (*simp add*: *S null-sets-restrict-space*)
show *?thesis*
apply (*intro conjI integral-cong-AE integrable-cong-AE bm1 bm2 AE-I'* [*OF 0*])
unfolding *Dirichlet-kernel-def* **by** *auto*
next
case *False*
then show *?thesis*
unfolding *Dirichlet-kernel-def* **by** (*auto intro*!: *Bochner-Integration.integral-cong Bochner-Integration.integrable-cong*)
qed

lemma
assumes *f*: *f* ∈ *borel-measurable* (*lebesgue-on S*) **and** *S*: *S* ∈ *sets lebesgue*
shows *integral-Dirichlet-kernel-mul-expand*:
(*LINT x|lebesgue-on S*. *Dirichlet-kernel n x* ∗ *f x*)

= (*LINT x|lebesgue-on S. sin((n+1/2) ∗ x) ∗ f x / (2 ∗ sin(x/2))*) (**is** *?th1*)
and *integrable-Dirichlet-kernel-mul-expand*:
integrable (lebesgue-on S) (λx. Dirichlet-kernel n x ∗ f x)
⟷ *integrable (lebesgue-on S) (λx. sin((n+1/2) ∗ x) ∗ f x / (2 ∗ sin(x/2)))*
(**is** *?th2*)
using *inte-Dirichlet-kernel-mul-expand* [*OF assms*] **by** *auto*

proposition *Fourier-sum-limit-sine-part*:
assumes *f*: *f absolutely-integrable-on {−pi..pi}*
and *periodic*: *⋀x. f(x + 2∗pi) = f x*
and *d*: *0 < d d ≤ pi*
shows (*λn. (∑ k≤n. Fourier-coefficient f k ∗ trigonometric-set k t)) ⟶ l*
⟷ (*λn. LINT x|lebesgue-on {0..d}. sin((n + 1/2) ∗ x) ∗ ((f(t+x) + f(t−x) − 2∗l) / x)) ⟶ 0*
(**is** *?lhs ⟷ ?Ψ ⟶ 0*)
proof −
have *ftx*: (*λx. f(t+x)) absolutely-integrable-on {−pi..pi}*
using *absolutely-integrable-periodic-offset assms* **by** *auto*
then have *ftmx*: (*λx. f(t−x)) absolutely-integrable-on {−pi..pi}*
by (*simp flip*: *absolutely-integrable-reflect-real* [**where** *f= (λx. f(t+x))*])
have *fbm*: (*λx. f(t+x) + f(t−x) − 2∗l) absolutely-integrable-on {−pi..pi}*
by (*force intro*: *ftmx ftx*)
let *?Φ = λn. LINT x|lebesgue-on {0..d}. Dirichlet-kernel n x ∗ ((f(t+x) + f(t−x)) − 2∗l)*
have *?lhs ⟷ ?Φ ⟶ 0*
by (*intro Fourier-sum-limit-Dirichlet-kernel-part assms*)
moreover have *?Φ ⟶ 0 ⟷ ?Ψ ⟶ 0*
proof (*rule Lim-transform-eq* [*OF Lim-transform-eventually*])
let *?f = λn. LINT x|lebesgue-on {0..d}. sin((real n + 1/2) ∗ x) ∗ (1 / (2 ∗ sin(x/2)) − 1/x) ∗ (f(t+x) + f(t−x) − 2∗l)*
have *?f n = (LINT x|lebesgue-on {−pi..pi}.*
sin ((n + 1/2) ∗ x) ∗ ((if x ∈ {0..d} then 1 / (2 ∗ sin (x/2)) − 1/x else 0) ∗ (f(t+x) + f(t−x) − 2∗l))) **for** *n*
using *d* **by** (*simp add*: *integral-restrict if-distrib* [*of λu. - ∗ (u ∗ -)*] *mult.assoc flip*: *atLeastAtMost-iff cong*: *if-cong*)
moreover have ... ⟶ *0*
proof (*intro Riemann-lebesgue-sin-half absolutely-integrable-bounded-measurable-product-real*)
have (*λx. 1 / (2 ∗ sin(x/2)) − 1/x) ∈ borel-measurable (lebesgue-on {0..d})*
by (*intro measurable continuous-imp-measurable-on-sets-lebesgue continuous-intros*) *auto*
then show (*λx. if x ∈ {0..d} then 1 / (2 ∗ sin(x/2)) − 1/x else 0) ∈ borel-measurable (lebesgue-on {−pi..pi})*
using *d* **by** (*simp add*: *borel-measurable-if-lebesgue-on flip*: *atLeastAtMost-iff*)

let *?g = λx::real. 1 / (2 ∗ sin(x/2)) − 1/x*
have *limg*: (*?g ⟶ ?g a) (at a within {0..d})* — thanks to Manuel Eberl
if *a*: *0 ≤ a a ≤ d* **for** *a*
proof −

have (*?g* ⟶ *0*) (*at-right 0*) **and** (*?g* ⟶ *?g d*) (*at-left d*)
using *d sin-gt-zero*[*of d/2*] **by** (*real-asymp simp*: *field-simps*)+
moreover have (*?g* ⟶ *?g a*) (*at a*) **if** *a* > *0*
using *a that d sin-gt-zero*[*of a/2*] *that* **by** (*real-asymp simp*: *field-simps*)
ultimately show *?thesis* **using** *that d*
by (*cases a* = *0* ∨ *a* = *d*) (*auto simp*: *at-within-Icc-at at-within-Icc-at-right at-within-Icc-at-left*)
qed
have ((λ*x*. *if x* ∈ {*0*..*d*} *then 1* / (*2* ∗ *sin*(*x*/*2*)) − *1*/*x else 0*) ' {−*pi*..*pi*}) = ((λ*x*. *1* / (*2* ∗ *sin*(*x*/*2*)) − *1*/*x*) ' {*0*..*d*})
using *d* **by** (*auto intro*: *image-eqI* [**where** *x*=*0*])
moreover have *bounded* . . .
by (*intro compact-imp-bounded compact-continuous-image*) (*auto simp*: *continuous-on limg*)
ultimately show *bounded* ((λ*x*. *if x* ∈ {*0*..*d*} *then 1* / (*2* ∗ *sin*(*x*/*2*)) − *1*/*x else 0*) ' {−*pi*..*pi*})
by *simp*
qed (*auto simp*: *fbm*)
ultimately show *?f* ⟶ (*0*::*real*)
by *simp*
show ∀$_F$ *n in sequentially*. *?f n* = *?*Φ *n* − *?*Ψ *n*
proof (*rule eventually-sequentiallyI* [**where** *c* = *Suc 0*])
fix *n*
assume *n* ≥ *Suc 0*
have *integrable* (*lebesgue-on* {*0*..*d*}) (λ*x*. *sin* ((*real n* + *1*/*2*) ∗ *x*) ∗ (*f*(*t*+*x*) + *f*(*t*−*x*) − *2*∗*l*) / (*2* ∗ *sin*(*x*/*2*)))
using *d*
apply (*subst integrable-Dirichlet-kernel-mul-expand* [*symmetric*])
apply (*intro assms absolutely-integrable-imp-borel-measurable absolutely-integrable-on-subinterval* [*OF fbm*]
absolutely-integrable-imp-integrable absolutely-integrable-mult-Dirichlet-kernel-reflected-part2 | *force*)+
done
moreover have *integrable* (*lebesgue-on* {*0*..*d*}) (λ*x*. *sin* ((*real n* + *1*/*2*) ∗ *x*) ∗ (*f*(*t*+*x*) + *f*(*t*−*x*) − *2*∗*l*) / *x*)
proof (*rule integrable-cong-AE-imp*)
let *?g* = λ*x*. *Dirichlet-kernel n x* ∗ (*2* ∗ *sin*(*x*/*2*) / *x* ∗ (*f*(*t*+*x*) + *f*(*t*−*x*) − *2*∗*l*))
have ∗: |*2* ∗ *sin* (*x*/*2*) / *x*| ≤ *1* **for** *x*::*real*
using *abs-sin-x-le-abs-x* [*of x*/*2*]
by (*simp add*: *divide-simps mult.commute abs-mult*)
have *bounded* ((λ*x*. *1* + (*x*/*2*)^2) ' {−*pi*..*pi*})
by (*intro compact-imp-bounded compact-continuous-image continuous-intros*) *auto*
then have *bo*: *bounded* ((λ*x*. *2* ∗ *sin* (*x*/*2*) / *x*) ' {−*pi*..*pi*})
using ∗ **unfolding** *bounded-real* **by** *blast*
show *integrable* (*lebesgue-on* {*0*..*d*}) *?g*
using *d*
apply (*intro absolutely-integrable-imp-integrable*

absolutely-integrable-on-subinterval [*OF absolutely-integrable-mult-Dirichlet-kernel*]
absolutely-integrable-bounded-measurable-product-real bo fbm
measurable continuous-imp-measurable-on-sets-lebesgue continuous-intros, *auto*)
done
show (*λx. sin* ((*n* + *1/2*) ∗ *x*) ∗ (*f*(*t*+*x*) + *f*(*t*−*x*) − *2*∗*l*) / *x*) ∈ *borel-measurable* (*lebesgue-on* {*0*..*d*})
using *d*
apply (*intro measurable absolutely-integrable-imp-borel-measurable*
absolutely-integrable-on-subinterval [*OF ftx*] *absolutely-integrable-on-subinterval* [*OF ftmx*]
absolutely-integrable-continuous-real continuous-intros, *auto*)
done
have *Dirichlet-kernel n x* ∗ (*2* ∗ *sin*(*x*/*2*)) / *x* = *sin* ((*real n* + *1/2*) ∗ *x*) / *x*
if *0* < *x x* ≤ *d* **for** *x*
using *that d* **by** (*simp add*: *Dirichlet-kernel-def divide-simps sin-zero-pi-iff*)
then show *AE x in lebesgue-on* {*0*..*d*}. *?g x* = *sin* ((*real n* + *1/2*) ∗ *x*) ∗ (*f*(*t*+*x*) + *f*(*t*−*x*) − *2*∗*l*) / *x*
using *d* **by** (*force intro*: *AE-I′* [**where** *N*={*0*}])
qed
ultimately have *?f n* = (*LINT x*|*lebesgue-on* {*0*..*d*}. *sin* ((*n* + *1/2*) ∗ *x*) ∗ (*f*(*t*+*x*) + *f*(*t*−*x*) − *2*∗*l*) / (*2* ∗ *sin*(*x*/*2*)))
− (*LINT x*|*lebesgue-on* {*0*..*d*}. *sin* ((*n* + *1/2*) ∗ *x*) ∗ (*f*(*t*+*x*) + *f*(*t*−*x*) − *2*∗*l*) / *x*)
by (*simp add*: *right-diff-distrib* [*of - - 1/-*] *left-diff-distrib*)
also have . . . = *?Φ n* − *?Ψ n*
using *d*
by (*simp add*: *measurable-restrict-mono* [*OF absolutely-integrable-imp-borel-measurable* [*OF fbm*]]
integral-Dirichlet-kernel-mul-expand)
finally show *?f n* = *?Φ n* − *?Ψ n* **.**
qed
qed
ultimately show *?thesis*
by *simp*
qed

6.16 Dini's test for the convergence of a Fourier series

proposition *Fourier-Dini-test*:
assumes *f*: *f absolutely-integrable-on* {−*pi*..*pi*}
and *periodic*: ⋀*x*. *f*(*x* + *2*∗*pi*) = *f x*
and *int0d*: *integrable* (*lebesgue-on* {*0*..*d*}) (*λx*. |*f*(*t*+*x*) + *f*(*t*−*x*) − *2*∗*l*| / *x*)
and *0* < *d*
shows (*λn*. (∑ *k*≤*n*. *Fourier-coefficient f k* ∗ *trigonometric-set k t*)) ⟶ *l*
proof −
define *φ* **where** *φ* ≡ *λn x*. *sin*((*n* + *1/2*) ∗ *x*) ∗ ((*f*(*t*+*x*) + *f*(*t*−*x*) − *2*∗*l*) / *x*)
have (*λn*. *LINT x*|*lebesgue-on* {*0*..*pi*}. *φ n x*) ⟶ *0*

unfolding *lim-sequentially dist-real-def*
proof (*intro allI impI*)
fix *e* :: *real*
assume *e* > *0*
define ϑ **where** ϑ ≡ λ*x*. *LINT x*|*lebesgue-on* {*0*..*x*}. |*f*(*t*+*x*) + *f*(*t*−*x*) − *2*∗*l*|
/ *x*
have [*simp*]: ϑ *0* = *0*
by (*simp add*: ϑ*-def integral-eq-zero-null-sets*)
have *cont*: *continuous-on* {*0*..*d*} ϑ
unfolding ϑ*-def* **using** *indefinite-integral-continuous-real int0d* **by** *blast*
with ‹*d* > *0*›
have ∀ *e*>*0*. ∃ *da*>*0*. ∀ *x*′∈{*0*..*d*}. |*x*′ − *0*| < *da* ⟶ |ϑ *x*′ − ϑ *0*| < *e*
by (*force simp*: *continuous-on-real-range dest*: *bspec* [**where** *x*=*0*])
with ‹*e* > *0*›
obtain *k* **where** *k* > *0* **and** *k*: ⋀*x*′. ⟦*0* ≤ *x*′; *x*′ < *k*; *x*′ ≤ *d*⟧ ⟹ |ϑ *x*′| < *e*/*2*
by (*drule-tac x*=*e*/*2* **in** *spec*) *auto*
define δ **where** δ ≡ *min d* (*min* (*k*/*2*) *pi*)
have *e2*: |ϑ δ| < *e*/*2* **and** δ > *0* δ ≤ *pi*
unfolding δ*-def* **using** *k* ‹*k* > *0*› ‹*d* > *0*› **by** *auto*
have *ftx*: (λ*x*. *f*(*t*+*x*)) *absolutely-integrable-on* {−*pi*..*pi*}
using *absolutely-integrable-periodic-offset assms* **by** *auto*
then have *ftmx*: (λ*x*. *f*(*t*−*x*)) *absolutely-integrable-on* {−*pi*..*pi*}
by (*simp flip*: *absolutely-integrable-reflect-real* [**where** *f*= (λ*x*. *f*(*t*+*x*))])
have *1*: φ *n absolutely-integrable-on* {*0*..δ} **for** *n*::*nat*
unfolding φ*-def*
proof (*rule absolutely-integrable-bounded-measurable-product-real*)
show (λ*x*. *sin* ((*real n* + *1*/*2*) ∗ *x*)) ∈ *borel-measurable* (*lebesgue-on* {*0*..δ})
by (*intro continuous-imp-measurable-on-sets-lebesgue continuous-intros*) *auto*
show *bounded* ((λ*x*. *sin* ((*real n* + *1*/*2*) ∗ *x*)) ‘ {*0*..δ})
by (*simp add*: *bounded-iff*) (*use abs-sin-le-one* **in** *blast*)
have (λ*x*. (*f*(*t*+*x*) + *f*(*t*−*x*) − *2*∗*l*) / *x*) ∈ *borel-measurable* (*lebesgue-on*
{*0*..δ})
using ‹*d* > *0*› **unfolding** δ*-def*
by (*intro measurable absolutely-integrable-imp-borel-measurable*
absolutely-integrable-on-subinterval [*OF ftx*] *absolutely-integrable-on-subinterval*
[*OF ftmx*]
absolutely-integrable-continuous-real continuous-intros, *auto*)
moreover have *integrable* (*lebesgue-on* {*0*..δ}) (*norm* ∘ (λ*x*. (*f*(*t*+*x*) + *f*(*t*−*x*)
− *2*∗*l*) / *x*))
proof (*rule integrable-subinterval* [*of 0 d*])
show *integrable* (*lebesgue-on* {*0*..*d*}) (*norm* ∘ (λ*x*. (*f*(*t*+*x*) + *f*(*t*−*x*) − *2*∗*l*)
/ *x*))
using *int0d* **by** (*subst Bochner-Integration.integrable-cong*) (*auto simp*:
o-def)
show {*0*..δ} ⊆ {*0*..*d*}
using ‹*d* > *0*› **by** (*auto simp*: δ*-def*)
qed
ultimately show (λ*x*. (*f*(*t*+*x*) + *f*(*t*−*x*) − *2*∗*l*) / *x*) *absolutely-integrable-on*
{*0*..δ}

by (*auto simp*: *absolutely-integrable-measurable*)
qed *auto*
have *2*: φ *n absolutely-integrable-on* $\{\delta..pi\}$ **for** *n*::*nat*
unfolding φ*-def*
proof (*rule absolutely-integrable-bounded-measurable-product-real*)
show ($\lambda x.$ *sin* ((*n* + *1/2*) $*$ *x*)) $\in$ *borel-measurable* (*lebesgue-on* $\{\delta..pi\}$)
by (*intro continuous-imp-measurable-on-sets-lebesgue continuous-intros*) *auto*
show *bounded* (($\lambda x.$ *sin* ((*n* + *1/2*) $*$ *x*)) ‘ $\{\delta..pi\}$)
by (*simp add*: *bounded-iff*) (*use abs-sin-le-one* **in** *blast*)
have ($\lambda x.$ *inverse x* $*$ (*f*(*t*+*x*) + *f*(*t*−*x*) − *2*$*$*l*)) *absolutely-integrable-on* $\{\delta..pi\}$
proof (*rule absolutely-integrable-bounded-measurable-product-real*)
have ($\lambda x.$ *1/x*) $\in$ *borel-measurable* (*lebesgue-on* $\{\delta..pi\}$)
by (*auto simp*: *measurable-completion measurable-restrict-space1*)
then show *inverse* $\in$ *borel-measurable* (*lebesgue-on* $\{\delta..pi\}$)
by (*metis* (*no-types*, *lifting*) *inverse-eq-divide measurable-lebesgue-cong*)
have $\forall x\in\{\delta..pi\}$. *inverse* $|x| \le$ *inverse* δ
using ‹*0* < δ› **by** *auto*
then show *bounded* (*inverse* ‘ $\{\delta..pi\}$)
by (*auto simp*: *bounded-iff*)
show ($\lambda x.$ *f*(*t*+*x*) + *f*(*t*−*x*) − *2*$*$*l*) *absolutely-integrable-on* $\{\delta..pi\}$
proof (*rule absolutely-integrable-on-subinterval*)
show ($\lambda x.$ (*f*(*t*+*x*) + *f*(*t*−*x*) − *2*$*$*l*)) *absolutely-integrable-on* $\{-pi..pi\}$
by (*fast intro*: *ftx ftmx absolutely-integrable-on-const*)
show $\{\delta..pi\} \subseteq \{-pi..pi\}$
using ‹*0* < δ› **by** *auto*
qed
qed *auto*
then show ($\lambda x.$ (*f*(*t*+*x*) + *f*(*t*−*x*) − *2*$*$*l*) / *x*) *absolutely-integrable-on* $\{\delta..pi\}$
by (*metis* (*no-types*, *lifting*) *divide-inverse mult.commute set-integrable-cong*)
qed *auto*
have ($\lambda x.$ *f*(*t*+*x*) − *l*) *absolutely-integrable-on* $\{-pi..pi\}$
using *ftx* **by** *auto*
moreover have *bounded* (($\lambda x.$ *0*) ‘ $\{x.\ |x| < \delta\}$)
using *bounded-def* **by** *blast*
moreover have *bounded* (*inverse* ‘ $\{x.\ \neg\ |x| < \delta\}$)
using ‹δ > *0*› **by** (*auto simp*: *divide-simps intro*: *boundedI* [**where** *B* = *1*/δ])
ultimately have ($\lambda x.$ (*if* $|x| < \delta$ *then 0 else inverse x*) $*$ (*if x* $\in \{-pi..pi\}$ *then* *f*(*t*+*x*) − *l else 0*)) *absolutely-integrable-on UNIV*
apply (*intro absolutely-integrable-bounded-measurable-product-real measurable set-integral-diff*)
apply (*auto simp*: *lebesgue-on-UNIV-eq measurable-completion simp flip*: *absolutely-integrable-restrict-UNIV* [**where** *S* = $\{-pi..pi\}$])
done
moreover have (*if x* $\in \{-pi..pi\}$ *then if* $|x| < \delta$ *then 0 else* (*f*(*t*+*x*) − *l*) / *x else 0*)
= (*if* $|x| < \delta$ *then 0 else inverse x*) $*$ (*if x* $\in \{-pi..pi\}$ *then f*(*t*+*x*) − *l else 0*) **for** *x*
by (*auto simp*: *divide-simps*)
ultimately have $*$: ($\lambda x.$ *if* $|x| < \delta$ *then 0 else* (*f*(*t*+*x*) − *l*) / *x*) *absolutely-integrable-on*

$\{-pi..pi\}$
by (*simp add*: *flip*: *absolutely-integrable-restrict-UNIV* [**where** $S = \{-pi..pi\}$])
have (λn. *LINT x|lebesgue-on* $\{0..pi\}$. *sin* $((n + 1/2) * x) *$ (*if* $|x| < \delta$ *then 0 else* $(f(t+x) - l) / x)$
$+$ *sin* $((n + 1/2) * -x) *$ (*if* $|x| < \delta$ *then 0 else* $(f(t-x) - l) / -x))$
$\longrightarrow 0$
using *Riemann-lebesgue-sin-half* [*OF* $*$] $*$
by (*simp add*: *integral-reflect-and-add absolutely-integrable-imp-integrable absolutely-integrable-sin-product cong*: *if-cong*)
moreover have *sin* $((n + 1/2) * x) *$ (*if* $|x| < \delta$ *then 0 else* $(f(t+x) - l) / x)$
$+$ *sin* $((n + 1/2) * -x) *$ (*if* $|x| < \delta$ *then 0 else* $(f(t-x) - l) / -x)$
$=$ (*if* $\neg |x| < \delta$ *then* φ *n x else 0*) **for** *x n*
by *simp* (*auto simp*: *divide-simps algebra-simps* φ*-def*)
ultimately have (λn. *LINT x|lebesgue-on* $\{0..pi\}$. (*if* $\neg |x| < \delta$ *then* φ *n x else 0*)) $\longrightarrow 0$
by *simp*
moreover have (*if* $0 \le x \wedge x \le pi$ *then if* $\neg |x| < \delta$ *then* φ *n x else 0 else 0*)
$=$ (*if* $\delta \le x \wedge x \le pi$ *then* φ *n x else 0*) **for** *x n*
using ‹$\delta > 0$› **by** *auto*
ultimately have †: (λn. *LINT x|lebesgue-on* $\{\delta..pi\}$. φ *n x*) $\longrightarrow 0$
by (*simp flip*: *Lebesgue-Measure.integral-restrict-UNIV*)
then obtain *N::nat* **where** *N*: $\bigwedge n.\ n \ge N \Longrightarrow$ |*LINT x|lebesgue-on* $\{\delta..pi\}$. φ *n x*| $< e/2$
unfolding *lim-sequentially dist-real-def*
by (*metis* (*full-types*) ‹$0 < e$› *diff-zero half-gt-zero-iff*)
have $|integral^L$ (*lebesgue-on* $\{0..pi\}$) (φ *n*)| $< e$ **if** $n \ge N$ **for** *n::nat*
proof −
have *int*: *integrable* (*lebesgue-on* $\{0..pi\}$) (φ (*real n*))
by (*intro integrable-combine* [*of concl*: *0 pi*] *absolutely-integrable-imp-integrable*) (*use* ‹$\delta > 0$› ‹$\delta \le pi$› *1 2* **in** *auto*)
then have $integral^L$ (*lebesgue-on* $\{0..pi\}$) (φ *n*) $= integral^L$ (*lebesgue-on* $\{0..\delta\}$) (φ *n*) $+ integral^L$ (*lebesgue-on* $\{\delta..pi\}$) (φ *n*)
by (*rule integral-combine*) (*use* ‹$0 < \delta$› ‹$\delta \le pi$› **in** *auto*)
moreover have $|integral^L$ (*lebesgue-on* $\{0..\delta\}$) (φ *n*)| $\le$ *LINT x|lebesgue-on* $\{0..\delta\}$. $|f(t + x) + f(t - x) - 2 * l| / x$
proof (*rule integral-abs-bound-integral*)
show *integrable* (*lebesgue-on* $\{0..\delta\}$) (φ (*real n*))
by (*meson integrable-subinterval* ‹$\delta \le pi$› *int atLeastatMost-subset-iff order-refl*)
have $\{0..\delta\} \subseteq \{0..d\}$
by (*auto simp*: δ*-def*)
then show *integrable* (*lebesgue-on* $\{0..\delta\}$) (λx. $|f(t + x) + f(t - x) - 2 * l| / x$)
by (*rule integrable-subinterval* [*OF int0d*])
show $|\varphi$ (*real n*) $x| \le |f(t + x) + f(t - x) - 2 * l| / x$
if $x \in$ *space* (*lebesgue-on* $\{0..\delta\}$) **for** *x*
using *that*
apply (*auto simp*: φ*-def divide-simps abs-mult*)

```
      by (simp add: mult.commute mult-left-le)
    qed
    ultimately have |integral^L (lebesgue-on {0..pi}) (φ n)| < e/2 + e/2
      using N [OF that] e2 unfolding ϑ-def by linarith
    then show ?thesis
      by simp
  qed
  then show ∃N. ∀n≥N. |integral^L (lebesgue-on {0..pi}) (φ (real n)) − 0| < e
    by force
qed
  then show ?thesis
    unfolding φ-def using Fourier-sum-limit-sine-part assms pi-gt-zero by blast
qed
```

6.17 Cesaro summability of Fourier series using Fejér kernel

```
definition Fejer-kernel :: nat ⇒ real ⇒ real
  where
  Fejer-kernel ≡ λn x. if n = 0 then 0 else (∑r<n. Dirichlet-kernel r x) / n

lemma Fejer-kernel:
     Fejer-kernel n x =
        (if n = 0 then 0
         else if x = 0 then n/2
         else sin(n / 2 ∗ x) ^ 2 / (2 ∗ n ∗ sin(x/2) ^ 2))
proof (cases x=0 ∨ sin(x/2) = 0)
  case True
  have (∑r<n. (1 + real r ∗ 2)) = real n ∗ real n
    by (induction n) (auto simp: algebra-simps)
  with True show ?thesis
   by (auto simp: Fejer-kernel-def Dirichlet-kernel-def field-simps simp flip: sum-divide-distrib)
next
  case False
  have sin (x/2) ∗ (∑r<n. sin ((real r ∗ 2 + 1) ∗ x / 2)) =
        sin (real n ∗ x / 2) ∗ sin (real n ∗ x / 2)
  proof (induction n)
  next
    case (Suc n)
    then show ?case
      apply (simp add: field-simps sin-times-sin)
      apply (simp add: add-divide-distrib)
      done
  qed auto
  then show ?thesis
    using False
    unfolding Fejer-kernel-def Dirichlet-kernel-def
   by (simp add: divide-simps power2-eq-square mult.commute flip: sum-divide-distrib)
qed
```

lemma *Fejer-kernel-0* [*simp*]: *Fejer-kernel 0 x* = *0* *Fejer-kernel n 0* = *n*/*2*
by (*auto simp*: *Fejer-kernel*)

lemma *Fejer-kernel-continuous-strong*:
continuous-on {−(*2* ∗ *pi*)<..<*2* ∗ *pi*} (*Fejer-kernel n*)
proof (*cases n*=*0*)
case *False*
then show *?thesis*
by (*simp add*: *Fejer-kernel-def continuous-intros Dirichlet-kernel-continuous-strong*)
qed (*simp add*: *Fejer-kernel-def*)

lemma *Fejer-kernel-continuous*:
continuous-on {−*pi*..*pi*} (*Fejer-kernel n*)
apply (*rule continuous-on-subset* [*OF Fejer-kernel-continuous-strong*])
apply (*simp add*: *subset-iff*)
using *pi-gt-zero* **apply** *linarith*
done

lemma *absolutely-integrable-mult-Fejer-kernel*:
assumes *f absolutely-integrable-on* {−*pi*..*pi*}
shows (λ*x*. *Fejer-kernel n x* ∗ *f x*) *absolutely-integrable-on* {−*pi*..*pi*}
proof (*rule absolutely-integrable-bounded-measurable-product-real*)
show *Fejer-kernel n* ∈ *borel-measurable* (*lebesgue-on* {−*pi*..*pi*})
by (*simp add*: *Fejer-kernel-continuous continuous-imp-measurable-on-sets-lebesgue*)
show *bounded* (*Fejer-kernel n* ‘ {−*pi*..*pi*})
using *Fejer-kernel-continuous compact-continuous-image compact-imp-bounded*
by *blast*
qed (*use assms* **in** *auto*)

lemma *absolutely-integrable-mult-Fejer-kernel-reflected1*:
assumes *f*: *f absolutely-integrable-on* {−*pi*..*pi*}
and *periodic*: ⋀*x*. *f*(*x* + *2*∗*pi*) = *f x*
shows (λ*x*. *Fejer-kernel n x* ∗ *f*(*t* + *x*)) *absolutely-integrable-on* {−*pi*..*pi*}
using *assms*
by (*force intro*: *absolutely-integrable-mult-Fejer-kernel absolutely-integrable-periodic-offset*)

lemma *absolutely-integrable-mult-Fejer-kernel-reflected2*:
assumes *f*: *f absolutely-integrable-on* {−*pi*..*pi*}
and *periodic*: ⋀*x*. *f*(*x* + *2*∗*pi*) = *f x*
shows (λ*x*. *Fejer-kernel n x* ∗ *f*(*t* − *x*)) *absolutely-integrable-on* {−*pi*..*pi*}
proof −
have (λ*x*. *f*(*t* − *x*)) *absolutely-integrable-on* {−*pi*..*pi*}
using *assms*
apply (*subst absolutely-integrable-reflect-real* [*symmetric*])
apply (*simp add*: *absolutely-integrable-periodic-offset*)
done
then show *?thesis*

by (*rule absolutely-integrable-mult-Fejer-kernel*)
qed

lemma *absolutely-integrable-mult-Fejer-kernel-reflected3*:
shows (λx. *Fejer-kernel n x* $*$ *c*) *absolutely-integrable-on* $\{-pi..pi\}$
using *absolutely-integrable-on-const absolutely-integrable-mult-Fejer-kernel* **by** *blast*

lemma *absolutely-integrable-mult-Fejer-kernel-reflected-part1*:
assumes *f*: *f absolutely-integrable-on* $\{-pi..pi\}$
and *periodic*: $\bigwedge x.\ f(x + 2{*}pi) = f\ x$ **and** $d \le pi$
shows (λx. *Fejer-kernel n x* $*$ $f(t + x)$) *absolutely-integrable-on* $\{0..d\}$
by (*rule absolutely-integrable-on-subinterval* [*OF absolutely-integrable-mult-Fejer-kernel-reflected1*])
(*auto simp*: *assms*)

lemma *absolutely-integrable-mult-Fejer-kernel-reflected-part2*:
assumes *f*: *f absolutely-integrable-on* $\{-pi..pi\}$
and *periodic*: $\bigwedge x.\ f(x + 2{*}pi) = f\ x$ **and** $d \le pi$
shows (λx. *Fejer-kernel n x* $*$ $f(t - x)$) *absolutely-integrable-on* $\{0..d\}$
by (*rule absolutely-integrable-on-subinterval* [*OF absolutely-integrable-mult-Fejer-kernel-reflected2*])
(*auto simp*: *assms*)

lemma *absolutely-integrable-mult-Fejer-kernel-reflected-part3*:
assumes $d \le pi$
shows (λx. *Fejer-kernel n x* $*$ *c*) *absolutely-integrable-on* $\{0..d\}$
by (*rule absolutely-integrable-on-subinterval* [*OF absolutely-integrable-mult-Fejer-kernel-reflected2*])
(*auto simp*: *assms*)

lemma *absolutely-integrable-mult-Fejer-kernel-reflected-part4*:
assumes *f*: *f absolutely-integrable-on* $\{-pi..pi\}$
and *periodic*: $\bigwedge x.\ f(x + 2{*}pi) = f\ x$ **and** $d \le pi$
shows (λx. *Fejer-kernel n x* $*$ $(f(t + x) + f(t - x))$) *absolutely-integrable-on*
$\{0..d\}$
unfolding *distrib-left*
by (*intro set-integral-add absolutely-integrable-mult-Fejer-kernel-reflected-part1*
absolutely-integrable-mult-Fejer-kernel-reflected-part2 assms)

lemma *absolutely-integrable-mult-Fejer-kernel-reflected-part5*:
assumes *f*: *f absolutely-integrable-on* $\{-pi..pi\}$
and *periodic*: $\bigwedge x.\ f(x + 2{*}pi) = f\ x$ **and** $d \le pi$
shows (λx. *Fejer-kernel n x* $*$ $((f(t + x) + f(t - x)) - c)$) *absolutely-integrable-on*
$\{0..d\}$
unfolding *distrib-left right-diff-distrib*
by (*intro set-integral-add set-integral-diff absolutely-integrable-on-const*
absolutely-integrable-mult-Fejer-kernel-reflected-part1 absolutely-integrable-mult-Fejer-kernel-reflected-part2
assms, *auto*)

lemma *Fourier-sum-offset-Fejer-kernel-half*:

```
  fixes n::nat
  assumes f: f absolutely-integrable-on {−pi..pi}
    and periodic: ⋀x. f(x + 2∗pi) = f x and n > 0
  shows (∑r<n. ∑k≤2∗r. Fourier-coefficient f k ∗ trigonometric-set k t) / n −
l
        = (LINT x|lebesgue-on {0..pi}. Fejer-kernel n x ∗ (f(t + x) + f(t − x) − 2
∗ l)) / pi
proof −
  have (∑r<n. ∑k≤2 ∗ r. Fourier-coefficient f k ∗ trigonometric-set k t) − real
n ∗ l
        = (∑r<n. (∑k≤2 ∗ r. Fourier-coefficient f k ∗ trigonometric-set k t) − l)
    by (simp add: sum-subtractf)
  also have ... = (∑r<n.
                          (LINT x|lebesgue-on {0..pi}. Dirichlet-kernel r x ∗ (f(t + x) +
f(t − x) − 2 ∗ l)) / pi)
    by (simp add: Fourier-sum-offset-Dirichlet-kernel-half assms)
  also have ... = real n ∗ ((LINT x|lebesgue-on {0..pi}. Fejer-kernel n x ∗ (f(t
+ x) + f(t − x) − 2 ∗ l)) / pi)
  proof −
    have integrable (lebesgue-on {0..pi}) (λx. Dirichlet-kernel i x ∗ (f(t + x) + f(t
− x) − 2 ∗ l)) for i
      using absolutely-integrable-mult-Dirichlet-kernel-reflected-part2(2) f periodic
      by (force simp: intro!: absolutely-integrable-imp-integrable)
    then show ?thesis
      using ‹n > 0›
      apply (simp add: Fejer-kernel-def flip: sum-divide-distrib)
        apply (simp add: sum-distrib-right flip: Bochner-Integration.integral-sum
[symmetric])
      done
  qed
  finally have (∑r<n. ∑k≤2 ∗ r. Fourier-coefficient f k ∗ trigonometric-set k
t) − real n ∗ l = real n ∗ ((LINT x|lebesgue-on {0..pi}. Fejer-kernel n x ∗ (f(t +
x) + f(t − x) − 2 ∗ l)) / pi) .
  with ‹n > 0› show ?thesis
    by (auto simp: mult.commute divide-simps)
qed


lemma Fourier-sum-limit-Fejer-kernel-half:
  fixes n::nat
  assumes f: f absolutely-integrable-on {−pi..pi}
    and periodic: ⋀x. f(x + 2∗pi) = f x
  shows (λn. ((∑r<n. ∑k≤2∗r. Fourier-coefficient f k ∗ trigonometric-set k t))
/ n) ⟶ l
          ⟷
          ((λn. integral^L (lebesgue-on {0..pi}) (λx. Fejer-kernel n x ∗ ((f(t + x) +
f(t − x)) − 2∗l))) ⟶ 0)
        (is ?lhs = ?rhs)
proof −
```

have *?lhs* ⟷
($\lambda n.$ (($\sum r<n.$ $\sum k\leq 2*r.$ *Fourier-coefficient f k* $*$ *trigonometric-set k t*)) / *n* $-$ *l*) ⟶ *0*
by (*simp add*: *LIM-zero-iff*)
also have ... ⟷
($\lambda n.$ (((($\sum r<n.$ $\sum k\leq 2*r.$ *Fourier-coefficient f k* $*$ *trigonometric-set k t*)) / *n*) $-$ *l*) $*$ *pi*) ⟶ *0*
using *tendsto-mult-right-iff* [*OF pi-neq-zero*] **by** *simp*
also have ... ⟷ *?rhs*
apply (*intro Lim-transform-eq* [*OF Lim-transform-eventually* [*of* $\lambda n.$ *0*]] *eventually-sequentiallyI* [*of 1*])
apply (*simp-all add*: *Fourier-sum-offset-Fejer-kernel-half assms*)
done
finally show *?thesis* **.**
qed

lemma *has-integral-Fejer-kernel*:
has-bochner-integral (*lebesgue-on* {$-$*pi*..*pi*}) (*Fejer-kernel n*) (*if n* = *0 then 0 else pi*)
proof $-$
have *has-bochner-integral* (*lebesgue-on* {$-$*pi*..*pi*}) ($\lambda x.$ ($\sum r<n.$ *Dirichlet-kernel r x*) / *real n*) (($\sum r<n.$ *pi*) / *n*)
by (*simp add*: *has-bochner-integral-iff integrable-Dirichlet-kernel has-bochner-integral-divide has-bochner-integral-sum*)
then show *?thesis*
by (*auto simp*: *Fejer-kernel-def*)
qed

lemma *has-integral-Fejer-kernel-half*:
has-bochner-integral (*lebesgue-on* {*0*..*pi*}) (*Fejer-kernel n*) (*if n* = *0 then 0 else pi/2*)
proof $-$
have *has-bochner-integral* (*lebesgue-on* {*0*..*pi*}) ($\lambda x.$ ($\sum r<n.$ *Dirichlet-kernel r x*) / *real n*) (($\sum r<n.$ *pi/2*) / *n*)
apply (*intro has-bochner-integral-sum has-bochner-integral-divide*)
using *not-integrable-integral-eq* **by** (*force simp*: *has-bochner-integral-iff*)
then show *?thesis*
by (*auto simp*: *Fejer-kernel-def*)
qed

lemma *Fejer-kernel-pos-le* [*simp*]: *Fejer-kernel n x* $\geq$ *0*
by (*simp add*: *Fejer-kernel*)

theorem *Fourier-Fejer-Cesaro-summable*:
assumes *f*: *f absolutely-integrable-on* {$-$*pi*..*pi*}
and *periodic*: $\bigwedge x.$ *f*(*x* + *2***pi*) = *f x*
and *fl*: (*f* ⟶ *l*) (*at t within atMost t*)

and *fr*: (*f* ⟶ *r*) (*at t within atLeast t*)
shows (*λn*. (∑ *m*<*n*. ∑ *k*≤*2*∗*m*. *Fourier-coefficient f k* ∗ *trigonometric-set k t*) / *n*) ⟶ (*l*+*r*) / *2*
proof −
define *h* **where** *h* ≡ *λu*. (*f*(*t*+*u*) + *f*(*t*−*u*)) − (*l*+*r*)
have (*λn*. *LINT u*|*lebesgue-on* {*0*..*pi*}. *Fejer-kernel n u* ∗ *h u*) ⟶ *0*
proof −
have *h0*: (*h* ⟶ *0*) (*at 0 within atLeast 0*)
proof −
have *l0*: ((*λu*. *f*(*t*−*u*) − *l*) ⟶ *0*) (*at 0 within* {*0*..})
using *fl*
unfolding *Lim-within*
apply (*elim all-forward imp-forward ex-forward conj-forward asm-rl*, *clarify*)
apply (*drule-tac x*=*t*−*x* **in** *bspec*)
apply (*auto simp*: *dist-norm*)
done
have *r0*: ((*λu*. *f*(*t* + *u*) − *r*) ⟶ *0*) (*at 0 within* {*0*..})
using *fr*
unfolding *Lim-within*
apply (*elim all-forward imp-forward ex-forward conj-forward asm-rl*, *clarify*)
apply (*drule-tac x*=*t*+*x* **in** *bspec*)
apply (*auto simp*: *dist-norm*)
done
show *?thesis*
using *tendsto-add* [*OF l0 r0*] **by** (*simp add*: *h-def algebra-simps*)
qed
show *?thesis*
unfolding *lim-sequentially dist-real-def diff-0-right*
proof *clarify*
fix *e*::*real*
assume *e* > *0*
then obtain *x′* **where** *0* < *x′* ⋀*x*. ⟦*0* < *x*; *x* < *x′*⟧ ⟹ |*h x*| < *e* / (*2* ∗ *pi*)
using *h0* **unfolding** *Lim-within dist-real-def*
by (*auto simp*: *dest*: *spec* [**where** *x*=*e*/*2*/*pi*])
then obtain *ξ* **where** *0* < *ξ ξ* < *pi* **and** *ξ*: ⋀*x*. *0* < *x* ∧ *x* ≤ *ξ* ⟹ |*h x*| < *e*/*2*/*pi*
apply (*intro that* [**where** *ξ*=*min x′ pi*/*2*], *auto*)
using *m2pi-less-pi* **by** *linarith*
have *ftx*: (*λx*. *f*(*t*+*x*)) *absolutely-integrable-on* {−*pi*..*pi*}
using *absolutely-integrable-periodic-offset assms* **by** *auto*
then have *ftmx*: (*λx*. *f*(*t*−*x*)) *absolutely-integrable-on* {−*pi*..*pi*}
by (*simp flip*: *absolutely-integrable-reflect-real* [**where** *f*= (*λx*. *f*(*t*+*x*))])
have *h-aint*: *h absolutely-integrable-on* {−*pi*..*pi*}
unfolding *h-def*
by (*intro absolutely-integrable-on-const set-integral-diff set-integral-add*, *auto simp*: *ftx ftmx*)
have (*λn*. *LINT x*|*lebesgue-on* {*ξ*..*pi*}. *Fejer-kernel n x* ∗ *h x*) ⟶ *0*
proof (*rule Lim-null-comparison*)
define *φ* **where** *φ* ≡ *λn*. (*LINT x*|*lebesgue-on* {*ξ*..*pi*}. |*h x*| / (*2* ∗ *sin*(*x*/*2*)

```
^ 2)) / n
      show ∀F n in sequentially. norm (LINT x|lebesgue-on {ξ..pi}. Fejer-kernel
n x * h x) ≤ φ n
      proof (rule eventually-sequentiallyI)
        fix n::nat assume n ≥ 1
        have hint: (λx. h x / (2 * sin(x/2) ^ 2)) absolutely-integrable-on {ξ..pi}
          unfolding divide-inverse mult.commute [of h -]
        proof (rule absolutely-integrable-bounded-measurable-product-real)
            have cont: continuous-on {ξ..pi} (λx. inverse (2 * (sin (x * inverse
2))²))
            using ‹0 < ξ› by (intro continuous-intros) (auto simp: sin-zero-pi-iff)
             show (λx. inverse (2 * (sin (x * inverse 2))²)) ∈ borel-measurable
(lebesgue-on {ξ..pi})
            using ‹0 < ξ›
            by (intro cont continuous-imp-measurable-on-sets-lebesgue) auto
          show bounded ((λx. inverse (2 * (sin (x * inverse 2))²)) ‘ {ξ..pi})
        using cont by (simp add: compact-continuous-image compact-imp-bounded)
          show h absolutely-integrable-on {ξ..pi}
         using ‹0 < ξ› ‹ξ < pi› by (auto intro: absolutely-integrable-on-subinterval
[OF h-aint])
        qed auto
      then have *: integrable (lebesgue-on {ξ..pi}) (λx. |h x| / (2 * (sin (x/2))²))
          by (simp add: absolutely-integrable-measurable o-def)
        define ψ where
            ψ ≡ λx. (if n = 0 then 0 else if x = 0 then n/2
                    else (sin (real n / 2 * x))² / (2 * real n * (sin (x/2))²)) * h x
        have |LINT x|lebesgue-on {ξ..pi}. ψ x|
            ≤ (LINT x|lebesgue-on {ξ..pi}. |h x| / (2 * (sin (x/2))²) / n)
        proof (rule integral-abs-bound-integral)
          show **: integrable (lebesgue-on {ξ..pi}) (λx. |h x| / (2 * (sin (x/2))²)
/ n)
            using Bochner-Integration.integrable-mult-left [OF *, of 1/n]
            by (simp add: field-simps)
          show †: |ψ x| ≤ |h x| / (2 * (sin (x/2))²) / real n
            if x ∈ space (lebesgue-on {ξ..pi}) for x
            using that ‹0 < ξ›
            apply (simp add: ψ-def divide-simps mult-less-0-iff abs-mult)
            apply (auto simp: square-le-1 mult-left-le-one-le)
            done
          show integrable (lebesgue-on {ξ..pi}) ψ
         proof (rule measurable-bounded-by-integrable-imp-lebesgue-integrable [OF
- **])
            let ?g = λx. |h x| / (2 * sin(x/2) ^ 2) / n
              have ***: integrable (lebesgue-on {ξ..pi}) (λx. (sin (n/2 * x))² *
(inverse (2 * (sin (x/2))²) * h x))
        proof (rule absolutely-integrable-imp-integrable [OF absolutely-integrable-bounded-measurable-product-
              show (λx. (sin (real n / 2 * x))²) ∈ borel-measurable (lebesgue-on
{ξ..pi})
           by (intro continuous-imp-measurable-on-sets-lebesgue continuous-intros)
```

auto
show *bounded* $((\lambda x.\ (sin\ (real\ n\ /\ 2 * x))^2)$ ‘ $\{\xi..pi\})$
by (*force simp*: *square-le-1 intro*: *boundedI* [**where** *B=1*])
show $(\lambda x.\ inverse\ (2 * (sin\ (x/2))^2) * h\ x)$ *absolutely-integrable-on* $\{\xi..pi\}$
using *hint* **by** (*simp add*: *divide-simps*)
qed *auto*
show $\psi \in$ *borel-measurable* (*lebesgue-on* $\{\xi..pi\}$)
apply (*rule borel-measurable-integrable*)
apply (*rule Bochner-Integration.integrable-cong* [**where** $f = \lambda x.\ sin(n\ /\ 2 * x)$ ^ $2\ /\ (2 * n * sin(x/2)$ ^ $2) * h\ x$, *OF refl*, *THEN iffD1*])
using ‹$0 < \xi$› $**$
apply (*force simp*: ψ*-def divide-simps algebra-simps mult-less-0-iff abs-mult*)
using *Bochner-Integration.integrable-mult-left* [*OF* $***$, *of* $1/n$]
by (*simp add*: *field-simps*)
show *norm* $(\psi\ x) \le\ ?g\ x$ **if** $x \in \{\xi..pi\}$ **for** x
using *that* † **by** (*simp add*: ψ*-def*)
qed *auto*
qed
then show *norm* (*LINT x|lebesgue-on* $\{\xi..pi\}$. *Fejer-kernel n x* $*$ *h x*) $\le \varphi\ n$
by (*simp add*: *Fejer-kernel* φ*-def* ψ*-def flip*: *Bochner-Integration.integral-divide* [*OF* $*$])
qed
show $\varphi \longrightarrow 0$
unfolding φ*-def divide-inverse*
by (*simp add*: *tendsto-mult-right-zero lim-inverse-n*)
qed
then obtain N **where** N: $\bigwedge n.\ n \ge N \Longrightarrow$ |*LINT x|lebesgue-on* $\{\xi..pi\}$. *Fejer-kernel n x* $*$ *h x*| $< e/2$
unfolding *lim-sequentially* **by** (*metis half-gt-zero-iff norm-conv-dist real-norm-def* ‹$e > 0$›)
show $\exists N.\ \forall n \ge N.$ |(*LINT u|lebesgue-on* $\{0..pi\}$. *Fejer-kernel n u* $*$ *h u*)| $< e$
proof (*intro exI allI impI*)
fix *n* :: *nat*
assume *n*: $n \ge max\ 1\ N$
with *N* **have** *1*: |*LINT x|lebesgue-on* $\{\xi..pi\}$. *Fejer-kernel n x* $*$ *h x*| $< e/2$
by *simp*
have $integral^L$ (*lebesgue-on* $\{0..\xi\}$) (*Fejer-kernel n*) $\le integral^L$ (*lebesgue-on* $\{0..pi\}$) (*Fejer-kernel n*)
using ‹$\xi < pi$› *has-bochner-integral-iff has-integral-Fejer-kernel-half*
by (*force intro*!: *integral-mono-lebesgue-on-AE*)
also have ... $\le pi$
using *has-integral-Fejer-kernel-half* **by** (*simp add*: *has-bochner-integral-iff*)
finally have *int-le-pi*: $integral^L$ (*lebesgue-on* $\{0..\xi\}$) (*Fejer-kernel n*) $\le pi$ **.**
have *2*: |*LINT x|lebesgue-on* $\{0..\xi\}$. *Fejer-kernel n x* $*$ *h x*| $\le$ (*LINT x|lebesgue-on* $\{0..\xi\}$. *Fejer-kernel n x* $*$ $e/2/pi$)
proof −

have *eq*: *integral*^{L} (*lebesgue-on* {*0*..*ξ*}) (*λx*. *Fejer-kernel n x* ∗ *h x*)
= *integral*^{L} (*lebesgue-on* {*0*..*ξ*}) (*λx*. *Fejer-kernel n x* ∗ (*if x* = *0 then 0 else h x*))
proof (*rule integral-cong-AE*)
have †: {*u*. *u* ≠ *0* ⟶ *P u*} ∩ {*0*..*ξ*} = {*0*} ∪ *Collect P* ∩ {*0*..*ξ*}
{*u*. *u* ≠ *0* ∧ *P u*} ∩ {*0*..*ξ*} = (*Collect P* ∩ {*0*..*ξ*}) − {*0*} **for** *P*
using ‹*0* < *ξ*› **by** *auto*
have ∗: − {*0*} ∩ *A* ∩ {*0*..*ξ*} = *A* ∩ {*0*..*ξ*} − {*0*} **for** *A*
by *auto*
show (*λx*. *Fejer-kernel n x* ∗ *h x*) ∈ *borel-measurable* (*lebesgue-on* {*0*..*ξ*})
using ‹*ξ* < *pi*›
by (*intro absolutely-integrable-imp-borel-measurable h-aint absolutely-integrable-on-subinterval* [*OF absolutely-integrable-mult-Fejer-kernel*], *auto*)
then show (*λx*. *Fejer-kernel n x* ∗ (*if x* = *0 then 0 else h x*)) ∈ *borel-measurable* (*lebesgue-on* {*0*..*ξ*})
apply (*simp add*: *in-borel-measurable Ball-def vimage-def Collect-conj-eq Collect-imp-eq* ∗ *flip*: *Collect-neg-eq*)
apply (*elim all-forward imp-forward asm-rl*)
using ‹*0* < *ξ*›
apply (*auto simp*: † *sets.insert-in-sets sets-restrict-space-iff cong*: *conj-cong*)
done
have *0*: {*0*} ∈ *null-sets* (*lebesgue-on* {*0*..*ξ*})
using ‹*0* < *ξ*› **by** (*simp add*: *null-sets-restrict-space*)
then show *AE x in lebesgue-on* {*0*..*ξ*}. *Fejer-kernel n x* ∗ *h x* = *Fejer-kernel n x* ∗ (*if x* = *0 then 0 else h x*)
by (*intro AE-I′* [*OF 0*]) *auto*
qed
show *?thesis*
unfolding *eq*
proof (*rule integral-abs-bound-integral*)
have (*λx*. *if x* = *0 then 0 else h x*) *absolutely-integrable-on* {− *pi*..*pi*}
proof (*rule absolutely-integrable-spike* [*OF h-aint*])
show *negligible* {*0*}
by *auto*
qed *auto*
with ‹*0* < *ξ*› ‹*ξ* < *pi*› **show** *integrable* (*lebesgue-on* {*0*..*ξ*}) (*λx*. *Fejer-kernel n x* ∗ (*if x* = *0 then 0 else h x*))
by (*intro absolutely-integrable-imp-integrable h-aint absolutely-integrable-on-subinterval* [*OF absolutely-integrable-mult-Fejer-kernel*]) *auto*
show *integrable* (*lebesgue-on* {*0*..*ξ*}) (*λx*. *Fejer-kernel n x* ∗ *e* / *2* / *pi*)
by (*simp add*: *absolutely-integrable-imp-integrable* ‹*ξ* < *pi*› *absolutely-integrable-mult-Fejer-kernel-reflected-part3 less-eq-real-def*)
show |*Fejer-kernel n x* ∗ (*if x* = *0 then 0 else h x*)| ≤ *Fejer-kernel n x* ∗ *e* / *2* / *pi*
if *x* ∈ *space* (*lebesgue-on* {*0*..*ξ*}) **for** *x*
using *that ξ* [*of x*] ‹*e* > *0*›
by (*auto simp*: *abs-mult eq simp flip*: *times-divide-eq-right intro*:

mult-left-mono)
          qed
        qed
        have *3*: ... ≤ *e/2*
          using *int-le-pi* ‹*0* < *e*›
          by (*simp add*: *divide-simps mult.commute* [*of e*])

        have *integrable* (*lebesgue-on* {*0..pi*}) (λ*x. Fejer-kernel n x* ∗ *h x*)
          unfolding *h-def*
        by (*simp add*: *absolutely-integrable-imp-integrable absolutely-integrable-mult-Fejer-kernel-reflected-part5 assms*)
        then have *LINT x*|*lebesgue-on* {*0..pi*}. *Fejer-kernel n x* ∗ *h x*
                = (*LINT x*|*lebesgue-on* {*0..ξ*}. *Fejer-kernel n x* ∗ *h x*) + (*LINT x*|*lebesgue-on* {*ξ..pi*}. *Fejer-kernel n x* ∗ *h x*)
          by (*rule integral-combine*) (*use* ‹*0* < *ξ*› ‹*ξ* < *pi*› **in** *auto*)
        then show |*LINT u*|*lebesgue-on* {*0..pi*}. *Fejer-kernel n u* ∗ *h u*| < *e*
          using *1 2 3* **by** *linarith*
      qed
    qed
  qed
  then show *?thesis*
   unfolding *h-def* **by** (*simp add*: *Fourier-sum-limit-Fejer-kernel-half assms add-divide-distrib*)
qed

corollary *Fourier-Fejer-Cesaro-summable-simple*:
  assumes *f*: *continuous-on UNIV f*
    and *periodic*: ⋀*x. f*(*x* + *2*∗*pi*) = *f x*
  shows (λ*n.* (∑ *m*<*n.* ∑ *k*≤*2*∗*m. Fourier-coefficient f k* ∗ *trigonometric-set k x*) / *n*) ⟶ *f x*
proof −
  have (λ*n.* (∑ *m*<*n.* ∑ *k*≤*2*∗*m. Fourier-coefficient f k* ∗ *trigonometric-set k x*) / *n*) ⟶ (*f x* + *f x*) / *2*
  proof (*rule Fourier-Fejer-Cesaro-summable*)
    show *f absolutely-integrable-on* {− *pi..pi*}
      using *absolutely-integrable-continuous-real continuous-on-subset f* **by** *blast*
    show (*f* ⟶ *f x*) (*at x within* {..*x*}) (*f* ⟶ *f x*) (*at x within* {*x*..})
      using *Lim-at-imp-Lim-at-within continuous-on-def f* **by** *blast+*
  qed (*auto simp*: *periodic Lim-at-imp-Lim-at-within continuous-on-def f*)
  then show *?thesis*
    by *simp*
qed

end

7 Acknowledgements

The author was supported by the ERC Advanced Grant ALEXANDRIA (Project 742178) funded by the European Research Council.