Banach-Steinhaus theorem*

Dominique Unruh   José Manuel Rodríguez Caballero

October 13, 2025

Abstract

We formalize in Isabelle/HOL a result [2] due to S. Banach and H. Steinhaus [1] known as Banach-Steinhaus theorem or Uniform boundedness principle: a pointwise-bounded family of continuous linear operators from a Banach space to a normed space is uniformly bounded. Our approach is an adaptation to Isabelle/HOL of a proof due to A. Sokal [3].

Contents

1 Missing results for the proof of Banach-Steinhaus theorem

theory *Banach-Steinhaus-Missing*
imports
HOL−Analysis.Bounded-Linear-Function
HOL−Analysis.Line-Segment

begin

*Supported by the ERC consolidator grant CerQuS (819317), the PRG team grant Secure Quantum Technology (PRG946) from the Estonian Research Council, and the Estonian Centre of Exellence in IT (EXCITE) funded by ERDF.

1.1 Results missing for the proof of Banach-Steinhaus theorem

The results proved here are preliminaries for the proof of Banach-Steinhaus theorem using Sokal's approach, but they do not explicitly appear in Sokal's paper [3].

Notation for the norm

open-bundle *norm-syntax* **begin**
notation *norm* (‹‖-‖›)
end

Notation for apply bilinear function

open-bundle *blinfun-apply-syntax* **begin**
notation *blinfun-apply* (**infixr** ‹$*_v$› *70*)
end

lemma *bdd-above-plus*:
fixes *f*::‹$'a \Rightarrow$ *real*›
assumes ‹*bdd-above* (*f* ‘ *S*)› **and** ‹*bdd-above* (*g* ‘ *S*)›
shows ‹*bdd-above* ((λ *x*. *f x* + *g x*) ‘ *S*)›

Explanation: If the images of two real-valued functions *f*,*g* are bounded above on a set *S*, then the image of their sum is bounded on *S*.

proof−
obtain *M* **where** ‹$\bigwedge$ *x*. *x*∈*S* $\Longrightarrow$ *f x* ≤ *M*›
using ‹*bdd-above* (*f* ‘ *S*)› **unfolding** *bdd-above-def* **by** *blast*
obtain *N* **where** ‹$\bigwedge$ *x*. *x*∈*S* $\Longrightarrow$ *g x* ≤ *N*›
using ‹*bdd-above* (*g* ‘ *S*)› **unfolding** *bdd-above-def* **by** *blast*
have ‹$\bigwedge$ *x*. *x*∈*S* $\Longrightarrow$ *f x* + *g x* ≤ *M* + *N*›
using ‹$\bigwedge$*x*. *x* ∈ *S* $\Longrightarrow$ *f x* ≤ *M*› ‹$\bigwedge$*x*. *x* ∈ *S* $\Longrightarrow$ *g x* ≤ *N*› **by** *fastforce*
thus *?thesis* **unfolding** *bdd-above-def* **by** *blast*
qed

The maximum of two functions

definition *pointwise-max*:: ($'a \Rightarrow$ $'b$::*ord*) $\Rightarrow$ ($'a \Rightarrow$ $'b$) $\Rightarrow$ ($'a \Rightarrow$ $'b$) **where**
‹*pointwise-max f g* = (λ*x*. *max* (*f x*) (*g x*))›

lemma *max-Sup-absorb-left*:
fixes *f g*::‹$'a \Rightarrow$ *real*›
assumes ‹*X* ≠ {}› **and** ‹*bdd-above* (*f* ‘ *X*)› **and** ‹*bdd-above* (*g* ‘ *X*)› **and** ‹*Sup* (*f* ‘ *X*) ≥ *Sup* (*g* ‘ *X*)›
shows ‹*Sup* ((*pointwise-max f g*) ‘ *X*) = *Sup* (*f* ‘ *X*)›

Explanation: For real-valued functions *f* and *g*, if the supremum of *f* is greater-equal the supremum of *g*, then the supremum of *max f g* equals the supremum of *f*. (Under some technical conditions.)

proof−
have *y-Sup*: ‹*y* ∈ ((λ *x*. *max* (*f x*) (*g x*)) ‘ *X*) $\Longrightarrow$ *y* ≤ *Sup* (*f* ‘ *X*)› **for** *y*

proof−
  assume ‹$y \in ((\lambda\ x.\ max\ (f\ x)\ (g\ x))$ ‘ $X)$›
  then obtain x **where** ‹$y = max\ (f\ x)\ (g\ x)$› **and** ‹$x \in X$›
    by *blast*
  have ‹$f\ x \leq Sup\ (f$ ‘ $X)$›
    by (*simp add*: ‹$x \in X$› ‹*bdd-above* (f ‘ X)› *cSUP-upper*)
  moreover have ‹$g\ x \leq Sup\ (g$ ‘ $X)$›
    by (*simp add*: ‹$x \in X$› ‹*bdd-above* (g ‘ X)› *cSUP-upper*)
  ultimately have ‹$max\ (f\ x)\ (g\ x) \leq Sup\ (f$ ‘ $X)$›
    using ‹$Sup\ (f$ ‘ $X) \geq Sup\ (g$ ‘ $X)$› **by** *auto*
  thus *?thesis* **by** (*simp add*: ‹$y = max\ (f\ x)\ (g\ x)$›)
qed
have *y-f-X*: ‹$y \in f$ ‘ $X \Longrightarrow y \leq Sup\ ((\lambda\ x.\ max\ (f\ x)\ (g\ x))$ ‘ $X)$› **for** y
proof−
  assume ‹$y \in f$ ‘ X›
  then obtain x **where** ‹$x \in X$› **and** ‹$y = f\ x$›
    by *blast*
  have ‹*bdd-above* $((\lambda\ \xi.\ max\ (f\ \xi)\ (g\ \xi))$ ‘ $X)$›
  by (*metis* (*no-types*) ‹*bdd-above* (f ‘ X)› ‹*bdd-above* (g ‘ X)› *bdd-above-image-sup sup-max*)
  moreover have ‹$e > 0 \Longrightarrow \exists\ k \in (\lambda\ \xi.\ max\ (f\ \xi)\ (g\ \xi))$ ‘ $X.\ y \leq k + e$›
    for e::*real*
    using ‹$Sup\ (f$ ‘ $X) \geq Sup\ (g$ ‘ $X)$›
    by (*smt* (*verit*, *best*) ‹$x \in X$› ‹$y = f\ x$› *imageI*)
  ultimately show *?thesis*
    using ‹$x \in X$› ‹$y = f\ x$› *cSUP-upper* **by** *fastforce*
qed
have ‹$Sup\ ((\lambda\ x.\ max\ (f\ x)\ (g\ x))$ ‘ $X) \leq Sup\ (f$ ‘ $X)$›
  using *y-Sup* **by** (*simp add*: ‹$X \neq \{\}$› *cSup-least*)
moreover have ‹$Sup\ ((\lambda\ x.\ max\ (f\ x)\ (g\ x))$ ‘ $X) \geq Sup\ (f$ ‘ $X)$›
  using *y-f-X* **by** (*metis* (*mono-tags*) *cSup-least calculation empty-is-image*)
ultimately show *?thesis* **unfolding** *pointwise-max-def* **by** *simp*
qed

lemma *max-Sup-absorb-right*:
  fixes $f\ g$::‹$'a \Rightarrow real$›
  assumes ‹$X \neq \{\}$› **and** ‹*bdd-above* (f ‘ X)› **and** ‹*bdd-above* (g ‘ X)› **and** ‹$Sup\ (f$ ‘ $X) \leq Sup\ (g$ ‘ $X)$›
  shows ‹Sup ((*pointwise-max* $f\ g$) ‘ X) $= Sup\ (g$ ‘ $X)$›

Explanation: For real-valued functions f and g and a nonempty set X, such that the f and g are bounded above on X, if the supremum of f on X is lower-equal the supremum of g on X, then the supremum of *pointwise-max* f g on X equals the supremum of g. This is the right analog of *max-Sup-absorb-left*.

proof−
  have ‹Sup ((*pointwise-max* $g\ f$) ‘ X) $= Sup\ (g$ ‘ $X)$›
    using *assms* **by** (*simp add*: *max-Sup-absorb-left*)
  moreover have ‹*pointwise-max* $g\ f$ = *pointwise-max* $f\ g$›

unfolding *pointwise-max-def* **by** *auto*
ultimately show *?thesis* **by** *simp*
qed

lemma *max-Sup*:
fixes *f g*::‹*'a* ⇒ *real*›
assumes ‹$X \neq \{\}$› **and** ‹*bdd-above* (*f* ‘ *X*)› **and** ‹*bdd-above* (*g* ‘ *X*)›
shows ‹*Sup* ((*pointwise-max f g*) ‘ *X*) = *max* (*Sup* (*f* ‘ *X*)) (*Sup* (*g* ‘ *X*))›

Explanation: Let X be a nonempty set. Two supremum over X of the maximum of two real-value functions is equal to the maximum of their suprema over X, provided that the functions are bounded above on X.

proof(*cases* ‹*Sup* (*f* ‘ *X*) ≥ *Sup* (*g* ‘ *X*)›)
case *True* **thus** *?thesis* **by** (*simp add*: *assms*(*1*) *assms*(*2*) *assms*(*3*) *max-Sup-absorb-left*)
next
case *False*
have *f1*: ¬ *0* ≤ *Sup* (*f* ‘ *X*) + − *1* ∗ *Sup* (*g* ‘ *X*)
using *False* **by** *linarith*
hence *Sup* (*Banach-Steinhaus-Missing.pointwise-max f g* ‘ *X*) = *Sup* (*g* ‘ *X*)
by (*simp add*: *assms*(*1*) *assms*(*2*) *assms*(*3*) *max-Sup-absorb-right*)
thus *?thesis*
using *f1* **by** *linarith*
qed

lemma *identity-telescopic*:
fixes *x* :: ‹- ⇒ *'a*::*real-normed-vector*›
assumes ‹*x* ⟶ *l*›
shows ‹(λ *N*. *sum* (λ *k*. *x* (*Suc k*) − *x k*) {*n*..*N*}) ⟶ *l* − *x n*›

Expression of a limit as a telescopic series. Explanation: If x converges to l then the sum $\sum k = n..N.\ x\ (Suc\ k) - x\ k$ converges to $l - x\ n$ as N goes to infinity.

proof−
have ‹(λ *p*. *x* (*p* + *Suc n*)) ⟶ *l*›
using ‹*x* ⟶ *l*› **by** (*rule LIMSEQ-ignore-initial-segment*)
hence ‹(λ *p*. *x* (*Suc n* + *p*)) ⟶ *l*›
by (*simp add*: *add.commute*)
hence ‹(λ *p*. *x* (*Suc* (*n* + *p*))) ⟶ *l*›
by *simp*
hence ‹(λ *t*. (− (*x n*)) + (λ *p*. *x* (*Suc* (*n* + *p*))) *t*) ⟶ (− (*x n*)) + *l*›
using *tendsto-add-const-iff* **by** *metis*
hence *f1*: ‹(λ *p*. *x* (*Suc* (*n* + *p*)) − *x n*)⟶ *l* − *x n*›
by *simp*
have ‹*sum* (λ *k*. *x* (*Suc k*) − *x k*) {*n*..*n*+*p*} = *x* (*Suc* (*n*+*p*)) − *x n*› **for** *p*
by (*simp add*: *sum-Suc-diff*)
moreover have ‹(λ *N*. *sum* (λ *k*. *x* (*Suc k*) − *x k*) {*n*..*N*}) (*n* + *t*)
= (λ *p*. *sum* (λ *k*. *x* (*Suc k*) − *x k*) {*n*..*n*+*p*}) *t*› **for** *t*
by *blast*

ultimately have ‹$(\lambda\ p.\ (\lambda\ N.\ sum\ (\lambda\ k.\ x\ (Suc\ k) - x\ k)\ \{n..N\})\ (n + p)) \longrightarrow l - x\ n$›
using *f1* **by** *simp*
hence ‹$(\lambda\ p.\ (\lambda\ N.\ sum\ (\lambda\ k.\ x\ (Suc\ k) - x\ k)\ \{n..N\})\ (p + n)) \longrightarrow l - x\ n$›
by (*simp add*: *add.commute*)
hence ‹$(\lambda\ p.\ (\lambda\ N.\ sum\ (\lambda\ k.\ x\ (Suc\ k) - x\ k)\ \{n..N\})\ p) \longrightarrow l - x\ n$›
using *Topological-Spaces.LIMSEQ-offset*[**where** $f = (\lambda\ N.\ sum\ (\lambda\ k.\ x\ (Suc\ k) - x\ k)\ \{n..N\})$
and $a = l - x\ n$ **and** $k = n$] **by** *blast*
hence ‹$(\lambda\ M.\ (\lambda\ N.\ sum\ (\lambda\ k.\ x\ (Suc\ k) - x\ k)\ \{n..N\})\ M) \longrightarrow l - x\ n$›
by *simp*
thus *?thesis* **by** *blast*
qed

lemma *bound-Cauchy-to-lim*:
assumes ‹$y \longrightarrow x$› **and** ‹$\bigwedge n.\ \|y\ (Suc\ n) - y\ n\| \leq c\hat{\ }n$› **and** ‹$y\ 0 = 0$› **and** ‹$c < 1$›
shows ‹$\|x - y\ (Suc\ n)\| \leq (c\ /\ (1 - c)) * c\ \hat{\ }\ n$›

Inequality about a sequence of approximations assuming that the sequence of differences is bounded by a geometric progression. Explanation: Let y be a sequence converging to x. If y satisfies the inequality $\|y\ (Suc\ n) - y\ n\| \leq c\ \hat{\ }\ n$ for some $c < 1$ and assuming $y\ 0 = 0$ then the inequality $\|x - y\ (Suc\ n)\| \leq (c\ /\ (1 - c)) * c\ \hat{\ }\ n$ holds.

proof−
have ‹$c \geq 0$›
using ‹$\bigwedge\ n.\ \|y\ (Suc\ n) - y\ n\| \leq c\hat{\ }n$›
by (*metis dual-order.trans norm-ge-zero power-one-right*)
have *norm-1*: ‹$norm\ (\sum k = Suc\ n..N.\ y\ (Suc\ k) - y\ k) \leq (c\ \hat{\ }\ Suc\ n)/(1 - c)$› **for** *N*
proof(*cases* ‹$N < Suc\ n$›)
case *True*
hence ‹$\|sum\ (\lambda k.\ y\ (Suc\ k) - y\ k)\ \{Suc\ n\ ..\ N\}\| = 0$›
by *auto*
thus *?thesis* **using** ‹$c \geq 0$› ‹$c < 1$› **by** *auto*
next
case *False*
hence ‹$N \geq Suc\ n$›
by *auto*
have ‹$c\hat{\ }(Suc\ N) \geq 0$›
using ‹$c \geq 0$› **by** *auto*
have ‹$1 - c > 0$›
by (*simp add*: ‹$c < 1$›)
hence ‹$(1 - c)/(1 - c) = 1$›
by *auto*
have ‹$\|sum\ (\lambda k.\ y\ (Suc\ k) - y\ k)\ \{Suc\ n\ ..\ N\}\| \leq (sum\ (\lambda k.\ \|y\ (Suc\ k) - y\ k\|)\ \{Suc\ n\ ..\ N\})$›
by (*simp add*: *sum-norm-le*)

hence ‹$\|$*sum* (λk. *y* (*Suc k*) − *y k*) {*Suc n* .. *N*}$\|$ ≤ (*sum* (*power c*) {*Suc n* .. *N*})›
by (*simp add*: *assms*(*2*) *sum-norm-le*)
hence ‹(*1* − *c*) ∗ $\|$*sum* (λk. *y* (*Suc k*) − *y k*) {*Suc n* .. *N*}$\|$
≤ (*1* − *c*) ∗ (*sum* (*power c*) {*Suc n* .. *N*})›
using ‹*0* < *1* − *c*› *mult-le-cancel-left-pos* **by** *blast*
also have ‹. . . = *c*^(*Suc n*) − *c*^(*Suc N*)›
using *Set-Interval.sum-gp-multiplied* ‹*Suc n* ≤ *N*› **by** *blast*
also have ‹. . . ≤ *c*^(*Suc n*)›
using ‹*c*^(*Suc N*) ≥ *0*› **by** *auto*
finally have ‹(*1* − *c*) ∗ $\|\sum$ *k* = *Suc n*..*N*. *y* (*Suc k*) − *y k*$\|$ ≤ *c* ^ *Suc n*›
by *blast*
hence ‹((*1* − *c*) ∗ $\|\sum$ *k* = *Suc n*..*N*. *y* (*Suc k*) − *y k*$\|$)/(*1* − *c*)
≤ (*c* ^ *Suc n*)/(*1* − *c*)›
using ‹*0* < *1* − *c*› *divide-le-cancel* **by** *fastforce*
thus ‹$\|\sum$ *k* = *Suc n*..*N*. *y* (*Suc k*) − *y k*$\|$ ≤ (*c* ^ *Suc n*)/(*1* − *c*)›
using ‹*0* < *1* − *c*› **by** *auto*
qed
have ‹(λ *N*. (*sum* (λk. *y* (*Suc k*) − *y k*) {*Suc n* .. *N*})) ⟶ *x* − *y* (*Suc n*)›
by (*metis* (*no-types*) ‹*y* ⟶ *x*› *identity-telescopic*)
hence ‹(λ *N*. $\|$*sum* (λk. *y* (*Suc k*) − *y k*) {*Suc n* .. *N*}$\|$) ⟶ $\|$*x* − *y* (*Suc n*)$\|$›
using *tendsto-norm* **by** *blast*
hence ‹$\|$*x* − *y* (*Suc n*)$\|$ ≤ (*c* ^ *Suc n*)/(*1* − *c*)›
using *norm-1 Lim-bounded* **by** *blast*
hence ‹$\|$*x* − *y* (*Suc n*)$\|$ ≤ (*c* ^ *Suc n*)/(*1* − *c*)›
by *auto*
moreover have ‹(*c* ^ *Suc n*)/(*1* − *c*) = (*c* / (*1* − *c*)) ∗ (*c* ^ *n*)›
by (*simp add*: *divide-inverse-commute*)
ultimately show ‹$\|$*x* − *y* (*Suc n*)$\|$ ≤ (*c* / (*1* − *c*)) ∗ (*c* ^ *n*)› **by** *linarith*
qed

lemma *onorm-open-ball*:
includes *norm-syntax*
shows ‹$\|f\|$ = *Sup* { $\|f *_v x\|$ | *x*. $\|x\|$ < *1* }›

Explanation: Let f be a bounded linear operator. The operator norm of f is the supremum of $\|f *_v x\|$ for x such that $\|x\| < 1$.

proof(*cases* ‹(*UNIV*::′*a set*) = *0*›)
case *True*
hence ‹*x* = *0*› **for** *x*::′*a*
by *auto*
hence ‹$f *_v x = 0$› **for** *x*
by (*metis* (*full-types*) *blinfun.zero-right*)
hence ‹$\|f\|$ = *0*›
by (*simp add*: *blinfun-eqI zero-blinfun.rep-eq*)
have ‹{ $\|f *_v x\|$ | *x*. $\|x\|$ < *1*} = {*0*}›
by (*smt* (*verit*, *ccfv-SIG*) *Collect-cong* ‹$\bigwedge x$. $f *_v x = 0$› *norm-zero singleton-conv*)

hence ‹$Sup\ \{\ \|f *_v x\|\ |\ x.\ \|x\| < 1\} = 0$›
  by *simp*
thus *?thesis* **using** ‹$\|f\| = 0$› **by** *auto*
next
case *False*
hence ‹$(UNIV::'a\ set) \neq 0$›
  by *simp*
have *nonnegative*: ‹$\|f *_v x\| \geq 0$› **for** x
  by *simp*
have ‹$\exists\ x::'a.\ x \neq 0$›
  using ‹$UNIV \neq 0$› **by** *auto*
then obtain $x::'a$ **where** ‹$x \neq 0$›
  by *blast*
hence ‹$\|x\| \neq 0$›
  by *auto*
define y **where** ‹$y = x\ /_R\ \|x\|$›
have ‹$norm\ y = \|\ x\ /_R\ \|x\|\ \|$›
  unfolding *y-def* **by** *auto*
also have ‹$\ldots = \|x\|\ /_R\ \|x\|$›
  by *auto*
also have ‹$\ldots = 1$›
  using ‹$\|x\| \neq 0$› **by** *auto*
finally have ‹$\|y\| = 1$›
  by *blast*
hence *norm-1-non-empty*: ‹$\{\ \|f *_v x\|\ |\ x.\ \|x\| = 1\} \neq \{\}$›
  by *blast*
have *norm-1-bounded*: ‹$bdd\text{-}above\ \{\ \|f *_v x\|\ |\ x.\ \|x\| = 1\}$›
  unfolding *bdd-above-def* **apply** *auto*
  by (*metis norm-blinfun*)
have *norm-less-1-non-empty*: ‹$\{\|f *_v x\|\ |\ x.\ \|x\| < 1\} \neq \{\}$›
 by (*metis* (*mono-tags*, *lifting*) *Collect-empty-eq-bot bot-empty-eq empty-iff norm-zero*

   zero-less-one)
have *norm-less-1-bounded*: ‹$bdd\text{-}above\ \{\|f *_v x\|\ |\ x.\ \|x\| < 1\}$›
proof−
  have ‹$\exists r.\ \|a\ r\| < 1 \longrightarrow \|f *_v (a\ r)\| \leq r$› **for** $a :: real \Rightarrow 'a$
  proof−
    obtain $r :: ('a \Rightarrow_L 'b) \Rightarrow real$ **where**
      $\bigwedge f\ x.\ 0 \leq r\ f \wedge (bounded\text{-}linear\ f \longrightarrow \|f *_v x\| \leq \|x\| * r\ f)$
      by (*metis mult.commute norm-blinfun norm-ge-zero*)
    have ‹$\neg\ \|f\| < 0$›
      by *simp*
    hence $(\exists r.\ \|f\| * \|a\ r\| \leq r) \vee (\exists r.\ \|a\ r\| < 1 \longrightarrow \|f *_v a\ r\| \leq r)$
      by (*meson less-eq-real-def mult-le-cancel-left2*)
    thus *?thesis* **using** *dual-order.trans norm-blinfun* **by** *blast*
  qed
  hence ‹$\exists\ M.\ \forall\ x.\ \|x\| < 1 \longrightarrow \|f *_v x\| \leq M$›
    by *metis*
  thus *?thesis* **by** *auto*

qed
have *Sup-non-neg*: ‹$Sup\ \{\|f *_v x\|\ |x.\ \|x\| = 1\} \geq 0$›
by (*metis* (*mono-tags*, *lifting*) ‹$\|y\| = 1$› *cSup-upper2 mem-Collect-eq norm-1-bounded norm-ge-zero*)
have ‹$\{0::real\} \neq \{\}$›
by *simp*
have ‹$bdd\text{-}above\ \{0::real\}$›
by *simp*
show ‹$\|f\| = Sup\ \{\|f *_v x\|\ |\ x.\ \|x\| < 1\}$›
proof(*cases* ‹$\forall x.\ f *_v x = 0$›)
case *True*
have ‹$\|f *_v x\| = 0$› **for** x
by (*simp add*: *True*)
hence ‹$\{\|f *_v x\|\ |\ x.\ \|x\| < 1\ \} \subseteq \{0\}$›
by *blast*
moreover have ‹$\{\|f *_v x\|\ |\ x.\ \|x\| < 1\ \} \supseteq \{0\}$›
using *calculation norm-less-1-non-empty* **by** *fastforce*
ultimately have ‹$\{\|f *_v x\|\ |\ x.\ \|x\| < 1\ \} = \{0\}$›
by *blast*
hence *Sup1*: ‹$Sup\ \{\|f *_v x\|\ |\ x.\ \|x\| < 1\ \} = 0$›
by *simp*
have ‹$\|f\| = 0$›
by (*simp add*: *True blinfun-eqI*)
moreover have ‹$Sup\ \{\|f *_v x\|\ |\ x.\ \|x\| < 1\} = 0$›
using *Sup1* **by** *blast*
ultimately show *?thesis* **by** *simp*
next
case *False*
have *norm-f-eq-leq*: ‹$y \in \{\|f *_v x\|\ |\ x.\ \|x\| = 1\} \Longrightarrow$
$y \leq Sup\ \{\|f *_v x\|\ |\ x.\ \|x\| < 1\}$› **for** y
proof−
assume ‹$y \in \{\|f *_v x\|\ |\ x.\ \|x\| = 1\}$›
hence ‹$\exists\ x.\ y = \|f *_v x\| \wedge \|x\| = 1$›
by *blast*
then obtain x **where** ‹$y = \|f *_v x\|$› **and** ‹$\|x\| = 1$›
by *auto*
define y' **where** ‹$y'\ n = (1 - (inverse\ (real\ (Suc\ n)))) *_R y$› **for** n
have ‹$y'\ n \in \{\|f *_v x\|\ |\ x.\ \|x\| < 1\}$› **for** n
proof−
have ‹$y'\ n = (1 - (inverse\ (real\ (Suc\ n)))) *_R \|f *_v x\|$›
using *y'-def* ‹$y = \|f *_v x\|$› **by** *blast*
also have ‹$... = |(1 - (inverse\ (real\ (Suc\ n))))| *_R \|f *_v x\|$›
by (*metis* (*mono-tags*, *opaque-lifting*) ‹$y = \|f *_v x\|$› *abs-1 abs-le-self-iff abs-of-nat*
abs-of-nonneg add-diff-cancel-left' add-eq-if cancel-comm-monoid-add-class.diff-cancel
diff-ge-0-iff-ge eq-iff-diff-eq-0 inverse-1 inverse-le-iff-le nat.distinct(1) *of-nat-0*
of-nat-Suc of-nat-le-0-iff zero-less-abs-iff zero-neq-one)
also have ‹$... = \|f *_v ((1 - (inverse\ (real\ (Suc\ n)))) *_R x)\|$›

by (*simp add*: *blinfun.scaleR-right*)
finally have *y′-1*: ‹$y'\ n = \|f *_v\ (\ (1 - (inverse\ (real\ (Suc\ n)))) *_R\ x)\|$›
by *blast*
have ‹$\|(1 - (inverse\ (Suc\ n))) *_R\ x\| = (1 - (inverse\ (real\ (Suc\ n)))) * \|x\|$›
by (*simp add*: *linordered-field-class.inverse-le-1-iff*)
hence ‹$\|(1 - (inverse\ (Suc\ n))) *_R\ x\| < 1$›
by (*simp add*: ‹$\|x\| = 1$›)
thus *?thesis* **using** *y′-1* **by** *blast*
qed
have ‹$(\lambda n.\ (1 - (inverse\ (real\ (Suc\ n))))\ ) \longrightarrow 1$›
using *Limits.LIMSEQ-inverse-real-of-nat-add-minus* **by** *simp*
hence ‹$(\lambda n.\ (1 - (inverse\ (real\ (Suc\ n)))) *_R\ y) \longrightarrow 1 *_R\ y$›
using *Limits.tendsto-scaleR* **by** *blast*
hence ‹$(\lambda n.\ (1 - (inverse\ (real\ (Suc\ n)))) *_R\ y) \longrightarrow y$›
by *simp*
hence ‹$(\lambda n.\ y'\ n) \longrightarrow y$›
using *y′-def* **by** *simp*
hence ‹$y' \longrightarrow y$›
by *simp*
have ‹$y'\ n \le Sup\ \{\|f *_v\ x\|\ |\ x.\ \|x\| < 1\}$› **for** *n*
using *cSup-upper* ‹$\bigwedge n.\ y'\ n \in \{\|f *_v\ x\|\ |x.\ \|x\| < 1\}$› *norm-less-1-bounded* **by** *blast*
hence ‹$y \le Sup\ \{\|f *_v\ x\|\ |\ x.\ \|x\| < 1\}$›
using ‹$y' \longrightarrow y$› *Topological-Spaces.Sup-lim* **by** (*meson LIMSEQ-le-const2*)
thus *?thesis* **by** *blast*
qed
hence ‹$Sup\ \{\|f *_v\ x\|\ |\ x.\ \|x\| = 1\} \le Sup\ \{\|f *_v\ x\|\ |\ x.\ \|x\| < 1\}$›
by (*metis* (*lifting*) *cSup-least norm-1-non-empty*)
have ‹$y \in \{\|f *_v\ x\|\ |\ x.\ \|x\| < 1\} \Longrightarrow y \le Sup\ \{\|f *_v\ x\|\ |\ x.\ \|x\| = 1\}$› **for** *y*
proof(*cases* ‹$y = 0$›)
case *True* **thus** *?thesis* **by** (*simp add*: *Sup-non-neg*)
next
case *False*
hence ‹$y \neq 0$› **by** *blast*
assume ‹$y \in \{\|f *_v\ x\|\ |\ x.\ \|x\| < 1\}$›
hence ‹$\exists\ x.\ y = \|f *_v\ x\| \wedge \|x\| < 1$›
by *blast*
then obtain *x* **where** ‹$y = \|f *_v\ x\|$› **and** ‹$\|x\| < 1$›
by *blast*
have ‹$(1/\|x\|) * y = (1/\|x\|) * \|f\ x\|$›
by (*simp add*: ‹$y = \|f *_v\ x\|$›)
also have ‹$... = |1/\|x\|| * \|f *_v\ x\|$›
by *simp*
also have ‹$... = \|(1/\|x\|) *_R\ (f *_v\ x)\|$›
by *simp*
also have ‹$... = \|f *_v\ ((1/\|x\|) *_R\ x)\|$›
by (*simp add*: *blinfun.scaleR-right*)
finally have ‹$(1/\|x\|) * y\ = \|f *_v\ ((1/\|x\|) *_R\ x)\|$›

by *blast*
have ‹$x \neq 0$›
using ‹$y \neq 0$› ‹$y = \|f *_v x\|$› *blinfun.zero-right* **by** *auto*
have ‹$\| (1/\|x\|) *_R x \| = | (1/\|x\|) | * \|x\|$›
by *simp*
also have ‹$\ldots = (1/\|x\|) * \|x\|$›
by *simp*
finally have ‹$\|(1/\|x\|) *_R x\| = 1$›
using ‹$x \neq 0$› **by** *simp*
hence ‹$(1/\|x\|) * y \in \{ \|f *_v x\| \mid x. \|x\| = 1\}$›
using ‹$1 / \|x\| * y = \|f *_v (1 / \|x\|) *_R x\|$› **by** *blast*
hence ‹$(1/\|x\|) * y \leq Sup \{ \|f *_v x\| \mid x. \|x\| = 1\}$›
by (*simp add*: *cSup-upper norm-1-bounded*)
moreover have ‹$y \leq (1/\|x\|) * y$›
by (*metis* ‹$\|x\| < 1$› ‹$y = \|f *_v x\|$› *mult-le-cancel-right1 norm-not-less-zero*

order.strict-implies-order ‹$x \neq 0$› *less-divide-eq-1-pos zero-less-norm-iff*)
ultimately show *?thesis* **by** *linarith*
qed
hence ‹$Sup \{ \|f *_v x\| \mid x. \|x\| < 1\} \leq Sup \{ \|f *_v x\| \mid x. \|x\| = 1\}$›
by (*smt* (*verit*, *del-insts*) *less-cSupD norm-less-1-non-empty*)
hence ‹$Sup \{ \|f *_v x\| \mid x. \|x\| = 1\} = Sup \{ \|f *_v x\| \mid x. \|x\| < 1\}$›
using ‹$Sup \{\|f *_v x\| |x. norm\ x = 1\} \leq Sup \{ \|f *_v x\| |x. \|x\| < 1\}$› **by**
linarith
have *f1*: ‹$(SUP\ x. \|f *_v x\| / \|x\|) = Sup \{ \|f *_v x\| / \|x\| \mid x. True\}$›
by (*simp add*: *full-SetCompr-eq*)
have ‹$y \in \{ \|f *_v x\| / \|x\| |x. True\} \Longrightarrow y \in \{ \|f *_v x\| |x. \|x\| = 1\} \cup \{0\}$›
for *y*
proof−
assume ‹$y \in \{ \|f *_v x\| / \|x\| |x. True\}$› **show** *?thesis*
proof(*cases* ‹$y = 0$›)
case *True* **thus** *?thesis* **by** *simp*
next
case *False*
have ‹$\exists\ x. y = \|f *_v x\| / \|x\|$›
using ‹$y \in \{ \|f *_v x\| / \|x\| |x. True\}$› **by** *auto*
then obtain *x* **where** ‹$y = \|f *_v x\| / \|x\|$›
by *blast*
hence ‹$y = |(1/\|x\|)| * \| f *_v x \|$›
by *simp*
hence ‹$y = \|(1/\|x\|) *_R (f *_v x)\|$›
by *simp*
hence ‹$y = \|f ((1/\|x\|) *_R x)\|$›
by (*simp add*: *blinfun.scaleR-right*)
moreover have ‹$\| (1/\|x\|) *_R x \| = 1$›
using *False* ‹$y = \|f *_v x\| / \|x\|$› **by** *auto*
ultimately have ‹$y \in \{\|f *_v x\| |x. \|x\| = 1\}$›
by *blast*
thus *?thesis* **by** *blast*

qed
qed
moreover have ‹$y \in \{\|f\, x\| \; |x.\ \|x\| = 1\} \cup \{0\} \Longrightarrow y \in \{\|f *_v x\| \,/\, \|x\| \; |x.$ $True\}$›
for y
proof(*cases* ‹$y = 0$›)
case *True* **thus** *?thesis* **by** *auto*
next
case *False*
hence ‹$y \notin \{0\}$›
by *simp*
moreover assume ‹$y \in \{\|f *_v x\| \; |x.\ \|x\| = 1\} \cup \{0\}$›
ultimately have ‹$y \in \{\|f *_v x\| \; |x.\ \|x\| = 1\}$›
by *simp*
then obtain x **where** ‹$\|x\| = 1$› **and** ‹$y = \|f *_v x\|$›
by *auto*
have ‹$y = \|f *_v x\| \,/\, \|x\|$› **using** ‹$\|x\| = 1$› ‹$y = \|f *_v x\|$›
by *simp*
thus *?thesis* **by** *auto*
qed
ultimately have ‹$\{\|f *_v x\| \,/\, \|x\| \; |x.\ True\} = \{\|f *_v x\| \; |x.\ \|x\| = 1\} \cup \{0\}$›
by *blast*
hence ‹$Sup\ \{\|f *_v x\| \,/\, \|x\| \; |x.\ True\} = Sup\ (\{\|f *_v x\| \; |x.\ \|x\| = 1\} \cup \{0\})$›
by *simp*
have $\bigwedge r\ s.\ \neg\ (r{::}real) \le s \vee sup\ r\ s = s$
by (*metis* (*lifting*) *sup.absorb-iff1 sup-commute*)
hence ‹$Sup\ (\{\|f *_v x\| \; |x.\ \|x\| = 1\} \cup \{(0{::}real)\})$
$= max\ (Sup\ \{\|f *_v x\| \; |x.\ \|x\| = 1\})\ (Sup\ \{0{::}real\})$›
using ‹$0 \le Sup\ \{\|f *_v x\| \; |x.\ \|x\| = 1\}$› ‹*bdd-above* $\{0\}$› ‹$\{0\} \neq \{\}$› *cSup-singleton*
cSup-union-distrib max.absorb-iff1 sup-commute norm-1-bounded norm-1-non-empty
by (*metis* (*no-types*, *lifting*))
moreover have ‹$Sup\ \{(0{::}real)\} = (0{::}real)$›
by *simp*
ultimately have ‹$Sup\ (\{\|f *_v x\| \; |x.\ \|x\| = 1\} \cup \{0\}) = Sup\ \{\|f *_v x\| \; |x.$ $\|x\| = 1\}$›
using *Sup-non-neg* **by** *linarith*
moreover have ‹$Sup\ (\ \{\|f *_v x\| \; |x.\ \|x\| = 1\} \cup \{0\})$
$= max\ (Sup\ \{\|f *_v x\| \; |x.\ \|x\| = 1\})\ (Sup\ \{0\})$ ›
using *Sup-non-neg* ‹$Sup\ (\{\|f *_v x\| \; |x.\ \|x\| = 1\} \cup \{0\})$
$= max\ (Sup\ \{\|f *_v x\| \; |x.\ \|x\| = 1\})\ (Sup\ \{0\})$›
by *auto*
ultimately have $f2$: ‹$Sup\ \{\|f *_v x\| \,/\, \|x\| \mid x.\ True\} = Sup\ \{\|f *_v x\| \mid x.\ \|x\|$ $= 1\}$›
using ‹$Sup\ \{\|f *_v x\| \,/\, \|x\| \; |x.\ True\} = Sup\ (\{\|f *_v x\| \; |x.\ \|x\| = 1\} \cup \{0\})$› **by** *linarith*
have ‹$(SUP\ x.\ \|f *_v x\| \,/\, \|x\|) = Sup\ \{\|f *_v x\| \mid x.\ \|x\| = 1\}$›
using *f1 f2* **by** *linarith*
hence ‹$(SUP\ x.\ \|f *_v x\| \,/\, \|x\|) = Sup\ \{\|f *_v x\| \mid x.\ \|x\| < 1\ \}$›

by (*simp add*: ‹$Sup\ \{\|f *_v x\|\ |x.\ \|x\| = 1\} = Sup\ \{\|f *_v x\|\ |x.\ \|x\| < 1\}$›)

thus *?thesis* **apply** *transfer* **by** (*simp add*: *onorm-def*)
qed
qed

lemma *onorm-r*:
includes *norm-syntax*
assumes ‹$r > 0$›
shows ‹$\|f\| = Sup\ ((\lambda x.\ \|f *_v x\|)\ `\ (ball\ 0\ r))\ /\ r$›

Explanation: The norm of f is $1\ /\ r$ of the supremum of the norm of f $*_v$ x for x in the ball of radius r centered at the origin.

proof−
have ‹$\|f\| = Sup\ \{\|f *_v x\|\ |x.\ \|x\| < 1\}$›
using *onorm-open-ball* **by** *blast*
moreover have $*$: ‹$\{\|f *_v x\|\ |x.\ \|x\| < 1\} = (\lambda x.\ \|f *_v x\|)\ `\ (ball\ 0\ 1)$›
unfolding *ball-def* **by** *auto*
ultimately have *onorm-f*: ‹$\|f\| = Sup\ ((\lambda x.\ \|f *_v x\|)\ `\ (ball\ 0\ 1))$›
by *simp*
have *s2*: ‹$x \in (\lambda t.\ r *_R \|f *_v t\|)\ `\ ball\ 0\ 1 \Longrightarrow x \le r * Sup\ ((\lambda t.\ \|f *_v t\|)\ `\ ball\ 0\ 1)$› **for** x
proof−
assume ‹$x \in (\lambda t.\ r *_R \|f *_v t\|)\ `\ ball\ 0\ 1$›
hence ‹$\exists\ t.\ x = r *_R \|f *_v t\| \wedge \|t\| < 1$›
by *auto*
then obtain t **where** t: ‹$x = r *_R \|f *_v t\|$› ‹$\|t\| < 1$›
by *blast*
define y **where** ‹$y = x /_R r$›
have ‹$x = r * (inverse\ r * x)$›
using ‹$x = r *_R norm\ (f\ t)$› **by** *auto*
hence ‹$x - (r * (inverse\ r * x)) \le 0$›
by *linarith*
hence ‹$x \le r * (x /_R r)$›
by *auto*
have ‹$y \in (\lambda k.\ \|f *_v k\|)\ `\ ball\ 0\ 1$›
unfolding *y-def* **using** *assms t* $*$ **by** *fastforce*
moreover have ‹$x \le r * y$›
using ‹$x \le r * (x /_R r)$› *y-def* **by** *blast*
ultimately have *y-norm-f*: ‹$y \in (\lambda t.\ \|f *_v t\|)\ `\ ball\ 0\ 1 \wedge x \le r * y$›
by *blast*
have ‹$(\lambda t.\ \|f *_v t\|)\ `\ ball\ 0\ 1 \ne \{\}$›
by *simp*
moreover have ‹$bdd\text{-}above\ ((\lambda t.\ \|f *_v t\|)\ `\ ball\ 0\ 1)$›
by (*simp add*: *bounded-linear-image blinfun.bounded-linear-right bounded-imp-bdd-above*

bounded-norm-comp)
moreover have ‹$\exists\ y.\ y \in (\lambda t.\ \|f *_v t\|)\ `\ ball\ 0\ 1 \wedge x \le r * y$›
using *y-norm-f* **by** *blast*

ultimately show *?thesis*
by (*meson assms cSup-upper dual-order.trans mult-le-cancel-left-pos*)
qed
have *s3*: ‹($\bigwedge x.\ x \in (\lambda t.\ r * \|f *_v t\|)$ ‘ *ball 0 1* $\Longrightarrow x \le y) \Longrightarrow$
$r * Sup\ ((\lambda t.\ \|f *_v t\|)$ ‘ *ball 0 1*$) \le y$› **for** *y*
proof−
assume ‹$\bigwedge x.\ x \in (\lambda t.\ r * \|f *_v t\|)$ ‘ *ball 0 1* $\Longrightarrow x \le y$›
have *x-leq*: ‹$x \in (\lambda t.\ \|f *_v t\|)$ ‘ *ball 0 1* $\Longrightarrow x \le y\ /\ r$› **for** *x*
proof−
assume ‹$x \in (\lambda t.\ \|f *_v t\|)$ ‘ *ball 0 1*›
then obtain *t* **where** ‹$t \in$ *ball* $(0::'a)$ *1*› **and** ‹$x = \|f *_v t\|$›
by *auto*
define x' **where** ‹$x' = r *_R x$›
have ‹$x' = r * \|f *_v t\|$›
by (*simp add*: ‹$x = \|f *_v t\|$› x'*-def*)
hence ‹$x' \in (\lambda t.\ r * \|f *_v t\|)$ ‘ *ball 0 1*›
using ‹$t \in$ *ball* $(0::'a)$ *1*› **by** *auto*
hence ‹$x' \le y$›
using ‹$\bigwedge x.\ x \in (\lambda t.\ r * \|f *_v t\|)$ ‘ *ball 0 1* $\Longrightarrow x \le y$› **by** *blast*
thus ‹$x \le y\ /\ r$›
unfolding x'*-def* **using** ‹$r > 0$› **by** (*simp add*: *mult.commute pos-le-divide-eq*)

qed
have ‹$(\lambda t.\ \|f *_v t\|)$ ‘ *ball 0 1* $\neq \{\}$›
by *simp*
moreover have ‹*bdd-above* $((\lambda t.\ \|f *_v t\|)$ ‘ *ball 0 1*$)$›
by (*simp add*: *bounded-linear-image blinfun.bounded-linear-right bounded-imp-bdd-above*

bounded-norm-comp)
ultimately have ‹$Sup\ ((\lambda t.\ \|f *_v t\|)$ ‘ *ball 0 1*$) \le y/r$›
using *x-leq* **by** (*simp add*: ‹*bdd-above* $((\lambda t.\ \|f *_v t\|)$ ‘ *ball 0 1*$)$› *cSup-least*)
thus *?thesis* **using** ‹$r > 0$›
by (*simp add*: *mult.commute pos-le-divide-eq*)
qed
have *norm-scaleR*: ‹$norm \circ ((*_R)\ r) = ((*_R)\ |r|) \circ (norm::'a \Rightarrow real)$›
by *auto*
have *f-x1*: ‹$f\ (r *_R x) = r *_R f\ x$› **for** *x*
by (*simp add*: *blinfun.scaleR-right*)
have ‹*ball* $(0::'a)\ r = ((*_R)\ r)$ ‘ (*ball 0 1*)›
by (*smt* (*verit*) *assms ball-scale nonzero-mult-div-cancel-left right-inverse-eq scale-zero-right*)
hence ‹$Sup\ ((\lambda t.\ \|f *_v t\|)$ ‘ (*ball 0 r*)$) = Sup\ ((\lambda t.\ \|f *_v t\|)$ ‘ $(((*_R)\ r)$ ‘ (*ball 0 1*)))›
by *simp*
also have ‹$\ldots = Sup\ (((\lambda t.\ \|f *_v t\|) \circ ((*_R)\ r))$ ‘ (*ball 0 1*))›
using *Sup.SUP-image* **by** *auto*
also have ‹$\ldots = Sup\ ((\lambda t.\ \|f *_v (r *_R t)\|)$ ‘ (*ball 0 1*))›
using *f-x1* **by** (*simp add*: *comp-assoc*)
also have ‹$\ldots = Sup\ ((\lambda t.\ |r| *_R \|f *_v t\|)$ ‘ (*ball 0 1*))›

using *norm-scaleR f-x1* **by** *auto*
also have ‹… = *Sup* ((λ*t*. *r* $*_R$ ‖*f* $*_v$ *t*‖) ‘ (*ball 0 1*))›
using ‹*r* > *0*› **by** *auto*
also have ‹… = *r* ∗ *Sup* ((λ*t*. ‖*f* $*_v$ *t*‖) ‘ (*ball 0 1*))›
apply (*rule cSup-eq-non-empty*) **apply** *simp* **using** *s2* **apply** *auto* **using** *s3* **by** *auto*
also have ‹… = *r* ∗ ‖*f*‖›
using *onorm-f* **by** *auto*
finally have ‹*Sup* ((λ*t*. ‖*f* $*_v$ *t*‖) ‘ *ball 0 r*) = *r* ∗ ‖*f*‖›
by *blast*
thus ‹‖*f*‖ = *Sup* ((λ*x*. ‖*f* $*_v$ *x*‖) ‘ (*ball 0 r*)) / *r*› **using** ‹*r* > *0*› **by** *simp*
qed

Pointwise convergence

definition *pointwise-convergent-to* ::
‹(*nat* ⇒ (′*a* ⇒ ′*b*::*topological-space*)) ⇒ (′*a* ⇒ ′*b*) ⇒ *bool*›
(‹((-)/ −*pointwise*→ (-))› [*60*, *60*] *60*) **where**
‹*pointwise-convergent-to* *x* *l* = (∀ *t*::′*a*. (λ *n*. (*x n*) *t*) ⟶ *l t*)›

lemma *linear-limit-linear*:
fixes *f* :: ‹- ⇒ (′*a*::*real-vector* ⇒ ′*b*::*real-normed-vector*)›
assumes ‹⋀*n*. *linear* (*f n*)› **and** ‹*f* −*pointwise*→ *F*›
shows ‹*linear F*›

Explanation: If a family of linear operators converges pointwise, then the limit is also a linear operator.

proof
show *F* (*x* + *y*) = *F x* + *F y* **for** *x y*
proof−
have ∀ *a*. *F a* = *lim* (λ*n*. *f n a*)
using ‹*f* −*pointwise*→ *F*› **unfolding** *pointwise-convergent-to-def* **by** (*metis* (*full-types*) *limI*)
moreover have ∀ *f b c g*. (*lim* (λ*n*. *g n* + *f n*) = (*b*::′*b*) + *c* ∨ ¬ *f* ⟶ *c*) ∨ ¬ *g* ⟶ *b*
by (*metis* (*no-types*) *limI tendsto-add*)
moreover have ⋀*a*. (λ*n*. *f n a*) ⟶ *F a*
using *assms*(*2*) *pointwise-convergent-to-def* **by** *force*
ultimately have
lim-sum: ‹*lim* (λ *n*. (*f n*) *x* + (*f n*) *y*) = *lim* (λ *n*. (*f n*) *x*) + *lim* (λ *n*. (*f n*) *y*)›
by *metis*
have ‹(*f n*) (*x* + *y*) = (*f n*) *x* + (*f n*) *y*› **for** *n*
using ‹⋀ *n*. *linear* (*f n*)› **unfolding** *linear-def* **using** *Real-Vector-Spaces.linear-iff* *assms*(*1*)
by *auto*
hence ‹*lim* (λ *n*. (*f n*) (*x* + *y*)) = *lim* (λ *n*. (*f n*) *x* + (*f n*) *y*)›
by *simp*
hence ‹*lim* (λ *n*. (*f n*) (*x* + *y*)) = *lim* (λ *n*. (*f n*) *x*) + *lim* (λ *n*. (*f n*) *y*)›
using *lim-sum* **by** *simp*

moreover have ‹$(\lambda\ n.\ (f\ n)\ (x + y)) \longrightarrow F\ (x + y)$›
using ‹f −*pointwise*→ F› **unfolding** *pointwise-convergent-to-def* **by** *blast*
moreover have ‹$(\lambda\ n.\ (f\ n)\ x) \longrightarrow F\ x$›
using ‹f −*pointwise*→ F› **unfolding** *pointwise-convergent-to-def* **by** *blast*
moreover have ‹$(\lambda\ n.\ (f\ n)\ y) \longrightarrow F\ y$›
using ‹f −*pointwise*→ F› **unfolding** *pointwise-convergent-to-def* **by** *blast*
ultimately show *?thesis*
by (*metis limI*)
qed
show $F\ (r *_R x) = r *_R F\ x$ **for** r **and** x
proof−
have ‹$(f\ n)\ (r *_R x) = r *_R (f\ n)\ x$› **for** n
using ‹$\bigwedge\ n.$ *linear* $(f\ n)$›
by (*simp add*: *Real-Vector-Spaces.linear-def real-vector.linear-scale*)
hence ‹*lim* $(\lambda\ n.\ (f\ n)\ (r *_R x)) =$ *lim* $(\lambda\ n.\ r *_R (f\ n)\ x)$›
by *simp*
have ‹*convergent* $(\lambda\ n.\ (f\ n)\ x)$›
by (*metis assms(2) convergentI pointwise-convergent-to-def*)
moreover have ‹*isCont* $(\lambda\ t{::}'b.\ r *_R t)$ *tt*› **for** *tt*
by (*simp add*: *bounded-linear-scaleR-right*)
ultimately have ‹*lim* $(\lambda\ n.\ r *_R ((f\ n)\ x)) = r *_R$ *lim* $(\lambda\ n.\ (f\ n)\ x)$›
using ‹f −*pointwise*→ F› **unfolding** *pointwise-convergent-to-def*
by (*metis* (*mono-tags*) *isCont-tendsto-compose limI*)
hence ‹*lim* $(\lambda\ n.\ (f\ n)\ (r *_R x)) = r *_R$ *lim* $(\lambda\ n.\ (f\ n)\ x)$›
using ‹*lim* $(\lambda\ n.\ (f\ n)\ (r *_R x)) =$ *lim* $(\lambda\ n.\ r *_R (f\ n)\ x)$› **by** *simp*
moreover have ‹$(\lambda\ n.\ (f\ n)\ x) \longrightarrow F\ x$›
using ‹f −*pointwise*→ F› **unfolding** *pointwise-convergent-to-def* **by** *blast*
moreover have ‹$(\lambda\ n.\ (f\ n)\ (r *_R x)) \longrightarrow F\ (r *_R x)$›
using ‹f −*pointwise*→ F› **unfolding** *pointwise-convergent-to-def* **by** *blast*
ultimately show *?thesis*
by (*metis limI*)
qed
qed

lemma *non-Cauchy-unbounded*:
fixes a ::‹- ⇒ *real*›
assumes ‹$\bigwedge n.\ a\ n \geq 0$› **and** ‹$e > 0$›
and ‹$\forall M.\ \exists m.\ \exists n.\ m \geq M \wedge n \geq M \wedge m > n \wedge sum\ a\ \{Suc\ n..m\} \geq e$›
shows ‹$(\lambda n.\ (sum\ a\ \{0..n\})) \longrightarrow \infty$›

Explanation: If the sequence of partial sums of nonnegative terms is not Cauchy, then it converges to infinite.

proof−
define S::*ereal set* **where** ‹$S = range\ (\lambda n.\ sum\ a\ \{0..n\})$›
have ‹$\exists\ s{\in}S.\ k{*}e \leq s$› **for** k::*nat*
proof(*induction k*)
case *0*
from ‹$\forall M.\ \exists m.\ \exists n.\ m \geq M \wedge n \geq M \wedge m > n \wedge sum\ a\ \{Suc\ n..m\} \geq e$›

obtain *m n* **where** ‹$m \geq 0$› **and** ‹$n \geq 0$› **and** ‹$m > n$› **and** ‹*sum a* $\{Suc\ n..m\} \geq e$› **by** *blast*
have ‹$n < Suc\ n$›
by *simp*
hence ‹$\{0..n\} \cup \{Suc\ n..m\} = \{0..m\}$›
using *Set-Interval.ivl-disj-un*(7) ‹$n < m$› **by** *auto*
moreover have ‹*finite* $\{0..n\}$›
by *simp*
moreover have ‹*finite* $\{Suc\ n..m\}$›
by *simp*
moreover have ‹$\{0..n\} \cap \{Suc\ n..m\} = \{\}$›
by *simp*
ultimately have ‹*sum a* $\{0..n\}$ + *sum a* $\{Suc\ n..m\}$ = *sum a* $\{0..m\}$›
by (*metis sum.union-disjoint*)
moreover have ‹*sum a* $\{Suc\ n..m\} > 0$›
using ‹$e > 0$› ‹*sum a* $\{Suc\ n..m\} \geq e$› **by** *linarith*
moreover have ‹*sum a* $\{0..n\} \geq 0$›
by (*simp add*: *assms*(1) *sum-nonneg*)
ultimately have ‹*sum a* $\{0..m\} > 0$›
by *linarith*
moreover have ‹*sum a* $\{0..m\} \in S$›
unfolding *S-def* **by** *blast*
ultimately have ‹$\exists\, s \in S.\ 0 \leq s$›
using *ereal-less-eq*(5) **by** *fastforce*
thus *?case*
by (*simp add*: *zero-ereal-def*)
next
case (*Suc k*)
assume ‹$\exists\, s \in S.\ k*e \leq s$›
then obtain *s* **where** ‹$s \in S$› **and** ‹*ereal* $(k * e) \leq s$›
by *blast*
have ‹$\exists\, N.\ s =$ *sum a* $\{0..N\}$›
using ‹$s \in S$› **unfolding** *S-def* **by** *blast*
then obtain *N* **where** ‹$s =$ *sum a* $\{0..N\}$›
by *blast*
from ‹$\forall\, M.\ \exists\, m.\ \exists\, n.\ m \geq M \wedge n \geq M \wedge m > n \wedge$ *sum a* $\{Suc\ n..m\} \geq e$›
obtain *m n* **where** ‹$m \geq Suc\ N$› **and** ‹$n \geq Suc\ N$› **and** ‹$m > n$› **and** ‹*sum a* $\{Suc\ n..m\} \geq e$›
by *blast*
have ‹*finite* $\{Suc\ N..n\}$›
by *simp*
moreover have ‹*finite* $\{Suc\ n..m\}$›
by *simp*
moreover have ‹$\{Suc\ N..n\} \cup \{Suc\ n..m\} = \{Suc\ N..m\}$›
using *Set-Interval.ivl-disj-un*
by (*metis* ‹$Suc\ N \leq n$› ‹$n < m$› *atLeastSucAtMost-greaterThanAtMost order-less-imp-le*)
moreover have ‹$\{\} = \{Suc\ N..n\} \cap \{Suc\ n..m\}$›
by *simp*

ultimately have ‹*sum a* {*Suc N..m*} = *sum a* {*Suc N..n*} + *sum a* {*Suc n..m*}›
by (*metis sum.union-disjoint*)
moreover have ‹*sum a* {*Suc N..n*} ≥ *0*›
using ‹⋀*n. a n* ≥ *0*› **by** (*simp add*: *sum-nonneg*)
ultimately have ‹*sum a* {*Suc N..m*} ≥ *e*›
using ‹*e* ≤ *sum a* {*Suc n..m*}› **by** *linarith*
have ‹*finite* {*0..N*}›
by *simp*
have ‹*finite* {*Suc N..m*}›
by *simp*
moreover have ‹{*0..N*} ∪ {*Suc N..m*} = {*0..m*}›
using *Set-Interval.ivl-disj-un*(*7*) ‹*Suc N* ≤ *m*› **by** *auto*
moreover have ‹{*0..N*} ∩ {*Suc N..m*} = {}›
by *simp*
ultimately have ‹*sum a* {*0..N*} + *sum a* {*Suc N..m*} = *sum a* {*0..m*}›
by (*metis* ‹*finite* {*0..N*}› *sum.union-disjoint*)
hence ‹*e* + *k* ∗ *e* ≤ *sum a* {*0..m*}›
using ‹*ereal* (*real k* ∗ *e*) ≤ *s*› ‹*s* = *ereal* (*sum a* {*0..N*})› ‹*e* ≤ *sum a* {*Suc N..m*}› **by** *auto*
moreover have ‹*e* + *k* ∗ *e* = (*Suc k*) ∗ *e*›
by (*simp add*: *semiring-normalization-rules*(*3*))
ultimately have ‹(*Suc k*) ∗ *e* ≤ *sum a* {*0..m*}›
by *linarith*
hence ‹*ereal* ((*Suc k*) ∗ *e*) ≤ *sum a* {*0..m*}›
by *auto*
moreover have ‹*sum a* {*0..m*}∈*S*›
unfolding *S-def* **by** *blast*
ultimately show *?case* **by** *blast*
qed
hence ‹∃ *s*∈*S*. (*real n*) ≤ *s*› **for** *n*
by (*meson assms*(*2*) *ereal-le-le ex-less-of-nat-mult less-le-not-le*)
hence ‹*Sup S* = ∞›
using *Sup-le-iff Sup-subset-mono dual-order.strict-trans1 leD less-PInf-Ex-of-nat subsetI*
by *metis*
hence *Sup*: ‹*Sup* ((*range* (λ *n*. (*sum a* {*0..n*})))::*ereal set*) = ∞› **using** *S-def*
by *blast*
have ‹*incseq* (λ*n*. (*sum a* {..<*n*}))›
using ‹⋀*n. a n* ≥ *0*› **using** *Extended-Real.incseq-sumI* **by** *auto*
hence ‹*incseq* (λ*n*. (*sum a* {..< *Suc n*}))›
by (*meson incseq-Suc-iff*)
hence ‹*incseq* (λ*n*. (*sum a* {*0..n*})::*ereal*)›
using *incseq-ereal* **by** (*simp add*: *atLeast0AtMost lessThan-Suc-atMost*)
hence ‹(λ*n*. *sum a* {*0..n*}) ⟶ *Sup* (*range* (λ*n*. (*sum a* {*0..n*})::*ereal*))›
using *LIMSEQ-SUP* **by** *auto*
thus *?thesis* **using** *Sup PInfty-neq-ereal* **by** *auto*
qed

```
lemma sum-Cauchy-positive:
  fixes a ::‹- ⇒ real›
  assumes ‹⋀n. a n ≥ 0› and ‹∃K. ∀n. (sum a  {0..n}) ≤ K›
  shows ‹Cauchy (λn. sum a {0..n})›
```

Explanation: If a series of nonnegative reals is bounded, then the series is Cauchy.

```
proof (unfold Cauchy-altdef2, rule, rule)
  fix e::real
  assume ‹e>0›
  have ‹∃M. ∀m≥M. ∀n≥M. m > n ⟶ sum a {Suc n..m} < e›
  proof(rule classical)
    assume ‹¬(∃M. ∀m≥M. ∀n≥M. m > n ⟶ sum a {Suc n..m} < e)›
    hence ‹∀M. ∃m. ∃n. m ≥ M ∧ n ≥ M ∧ m > n ∧ ¬(sum a {Suc n..m} <
e)›
      by blast
    hence ‹∀M. ∃m. ∃n. m ≥ M ∧ n ≥ M ∧ m > n ∧ sum a {Suc n..m} ≥ e›
      by fastforce
    hence ‹(λn. (sum a  {0..n}) ) ⟶ ∞›
      using non-Cauchy-unbounded ‹0 < e› assms(1) by blast
    from ‹∃K. ∀n. sum a  {0..n} ≤ K›
    obtain K where  ‹∀n. sum a {0..n} ≤ K›
      by blast
    from  ‹(λn. sum a {0..n})  ⟶ ∞›
    have ‹∀B. ∃N. ∀n≥N. (λ n. (sum a  {0..n}) ) n ≥ B›
      using Lim-PInfty by simp
    hence  ‹∃n. (sum a {0..n}) ≥ K+1›
      using ereal-less-eq(3) by blast
    thus ?thesis using  ‹∀n. (sum a  {0..n}) ≤ K› by (smt (verit, best))
  qed
  have ‹sum a {Suc n..m} = sum a {0..m} − sum a {0..n}›
    if m > n for m n
   by (metis add-diff-cancel-left′ atLeast0AtMost less-imp-add-positive sum-up-index-split
that)
  hence ‹∃M. ∀m≥M. ∀n≥M. m > n ⟶ sum a {0..m} − sum a {0..n} < e›
   using ‹∃M. ∀m≥M. ∀n≥M. m > n ⟶ sum a {Suc n..m} < e› by presburger
  then obtain M where ‹∀m≥M. ∀n≥M. m > n ⟶ sum a {0..m} − sum a
{0..n} < e›
    by blast
  moreover have ‹m > n ⟹ sum a {0..m} ≥ sum a {0..n}› for m n
    using ‹⋀ n. a n ≥ 0› by (simp add: sum-mono2)
  ultimately have ‹∃M. ∀m≥M. ∀n≥M. m > n ⟶ |sum a {0..m} − sum a
{0..n}| < e›
    by auto
  hence ‹∃M. ∀m≥M. ∀n≥M. m ≥ n ⟶ |sum a {0..m} − sum a {0..n}| < e›
   by (metis ‹0 < e› abs-zero cancel-comm-monoid-add-class.diff-cancel diff-is-0-eq′

        less-irrefl-nat linorder-neqE-nat zero-less-diff)
  hence ‹∃M. ∀m≥M. ∀n≥M. |sum a {0..m} − sum a {0..n}| < e›
```

```
    by (metis abs-minus-commute nat-le-linear)
  hence ‹∃M. ∀m≥M. ∀n≥M. dist (sum a {0..m}) (sum a {0..n}) < e›
    by (simp add: dist-real-def)
  hence ‹∃M. ∀m≥M. ∀n≥M. dist (sum a {0..m}) (sum a {0..n}) < e› by blast
  thus ‹∃N. ∀n≥N. dist (sum a {0..n}) (sum a {0..N}) < e› by auto
qed

lemma convergent-series-Cauchy:
  fixes a::‹nat ⇒ real› and φ::‹nat ⇒ 'a::metric-space›
  assumes ‹∃M. ∀n. sum a {0..n} ≤ M› and ‹⋀n. dist (φ (Suc n)) (φ n) ≤ a
n›
  shows ‹Cauchy φ›
```

Explanation: Let a be a real-valued sequence and let φ be sequence in a metric space. If the partial sums of a are uniformly bounded and the distance between consecutive terms of φ are bounded by the sequence a, then φ is Cauchy.

```
proof (unfold Cauchy-altdef2, rule, rule)
  fix e::real
  assume ‹e > 0›
  have ‹⋀k. a k ≥ 0›
    using ‹⋀n. dist (φ (Suc n)) (φ n) ≤ a n› dual-order.trans zero-le-dist by blast
  hence ‹Cauchy (λk. sum a {0..k})›
    using ‹∃M. ∀n. sum a {0..n} ≤ M› sum-Cauchy-positive by blast
  hence ‹∃M. ∀m≥M. ∀n≥M. dist (sum a {0..m}) (sum a {0..n}) < e›
    unfolding Cauchy-def using ‹e > 0› by blast
  hence ‹∃M. ∀m≥M. ∀n≥M. m > n ⟶ dist (sum a {0..m}) (sum a {0..n})
< e›
    by blast
  have ‹dist (sum a {0..m}) (sum a {0..n}) = sum a {Suc n..m}› if ‹n<m› for
m n
  proof −
    have ‹n < Suc n›
      by simp
    have ‹finite {0..n}›
      by simp
    moreover have ‹finite {Suc n..m}›
      by simp
    moreover have ‹{0..n} ∪ {Suc n..m} = {0..m}›
      using ‹n < Suc n› ‹n < m› by auto
    moreover have ‹{0..n} ∩ {Suc n..m} = {}›
      by simp
    ultimately have sum-plus: ‹(sum a {0..n}) + sum a {Suc n..m} = (sum a
{0..m})›
      by (metis sum.union-disjoint)
    have ‹dist (sum a {0..m}) (sum a {0..n}) = |(sum a {0..m}) − (sum a {0..n})|›
      using dist-real-def by blast
    moreover have ‹(sum a {0..m}) − (sum a {0..n}) = sum a {Suc n..m}›
      using sum-plus by linarith
```

```
    ultimately show ?thesis
      by (simp add: ‹⋀k. 0 ≤ a k› sum-nonneg)
  qed
  hence sum-a: ‹∃M. ∀m≥M. ∀n≥M. m > n ⟶ sum a {Suc n..m} < e›
    by (metis ‹∃M. ∀m≥M. ∀n≥M. dist (sum a {0..m}) (sum a {0..n}) < e›)
  obtain M where ‹∀m≥M. ∀n≥M. m > n ⟶ sum a {Suc n..m} < e›
    using sum-a ‹e > 0› by blast
  hence ‹∀m. ∀n. Suc m ≥ Suc M ∧ Suc n ≥ Suc M ∧ Suc m > Suc n ⟶ sum
a {Suc n..Suc m − 1} < e›
    by simp
  hence ‹∀m≥1. ∀n≥1. m ≥ Suc M ∧ n ≥ Suc M ∧ m > n ⟶ sum a {n..m
− 1} < e›
    by (metis Suc-le-D)
  hence sum-a2: ‹∃M. ∀m≥M. ∀n≥M. m > n ⟶ sum a {n..m−1} < e›
    by (meson add-leE)
  have ‹dist (φ (n+p+1)) (φ n) ≤ sum a {n..n+p}› for p n :: nat
  proof(induction p)
    case 0 thus ?case  by (simp add: assms(2))
  next
    case (Suc p) thus ?case
        by (smt(verit, ccfv-SIG) Suc-eq-plus1 add-Suc-right add-less-same-cancel1
assms(2) dist-self dist-triangle2
          gr-implies-not0 sum.cl-ivl-Suc)
  qed
  hence ‹m > n ⟹ dist (φ m) (φ n) ≤ sum a {n..m−1}› for m n :: nat
   by (metis Suc-eq-plus1 Suc-le-D diff-Suc-1  gr0-implies-Suc less-eq-Suc-le less-imp-Suc-add

        zero-less-Suc)
  hence ‹∃M. ∀m≥M. ∀n≥M. m > n ⟶ dist (φ m) (φ n) < e›
    using sum-a2 ‹e > 0› by (smt (verit))
  thus ∃N. ∀n≥N. dist (φ n) (φ N) < e
    using ‹0 < e› by fastforce
qed

unbundle blinfun-apply-syntax

unbundle no norm-syntax

end
```

2 Banach-Steinhaus theorem

```
theory Banach-Steinhaus
  imports Banach-Steinhaus-Missing
begin
```

We formalize Banach-Steinhaus theorem as theorem *banach-steinhaus*. This theorem was originally proved in Banach-Steinhaus's paper [1]. For the proof, we follow Sokal's approach [3]. Furthermore, we prove as a corollary a result about pointwise convergent sequences of bounded operators whose domain is a Banach space.

2.1 Preliminaries for Sokal's proof of Banach-Steinhaus theorem

lemma *linear-plus-norm*:
  includes *norm-syntax*
  assumes ‹*linear f*›
  shows ‹$\|f\ \xi\| \leq max\ \|f\ (x + \xi)\|\ \|f\ (x - \xi)\|$›

Explanation: For arbitrary x and a linear operator f, $\|f\ \xi\|$ is upper bounded by the maximum of the norms of the shifts of f (i.e., $f\ (x + \xi)$ and $f\ (x - \xi)$).

proof−
  have ‹$norm\ (f\ \xi) = norm\ (\ (inverse\ (of\text{-}nat\ 2))\ *_R\ (f\ (x + \xi) - f\ (x - \xi))\ )$›
   by (*metis* (*no-types*, *opaque-lifting*) *add.commute assms diff-diff-eq2 group-cancel.sub1 linear-cmul linear-diff of-nat-numeral real-vector-affinity-eq scaleR-2 scaleR-right-diff-distrib zero-neq-numeral*)
  also have ‹$\ldots = inverse\ (of\text{-}nat\ 2) * norm\ (f\ (x + \xi) - f\ (x - \xi))$›
    using *Real-Vector-Spaces.real-normed-vector-class.norm-scaleR* **by** *simp*
  also have ‹$\ldots \leq inverse\ (of\text{-}nat\ 2) * (norm\ (f\ (x + \xi)) + norm\ (f\ (x - \xi)))$›
    by (*simp add*: *norm-triangle-ineq4*)
  also have ‹$\ldots \leq\ max\ (norm\ (f\ (x + \xi)))\ (norm\ (f\ (x - \xi)))$›
    by *auto*
  finally show *?thesis* **by** *blast*
qed

lemma *onorm-Sup-on-ball*:
  includes *norm-syntax*
  assumes ‹$r > 0$›
  shows $\|f\| \leq Sup\ (\ (\lambda x.\ \|f *_v x\|)\ `\ (ball\ x\ r)\ )\ /\ r$

Explanation: Let f be a bounded operator and let x be a point. For any $0 < r$, the operator norm of f is bounded above by the supremum of f applied to the open ball of radius r around x, divided by r.

proof−
  have *bdd-above-3*: ‹$bdd\text{-}above\ ((\lambda x.\ \|f *_v x\|)\ `\ (ball\ 0\ r))$›
  proof −
    obtain M **where** ‹$\bigwedge \xi.\ \ \|f *_v \xi\| \leq M * norm\ \xi$› **and** ‹$M \geq 0$›
      using *norm-blinfun norm-ge-zero* **by** *blast*
    hence ‹$\bigwedge \xi.\ \xi \in ball\ 0\ r \Longrightarrow \|f *_v \xi\| \leq M * r$›
      using ‹$r > 0$› **by** (*smt* (*verit*) *mem-ball-0 mult-left-mono*)
    thus *?thesis* **by** (*meson bdd-aboveI2*)
  qed
  have *bdd-above-2*: ‹$bdd\text{-}above\ ((\lambda\ \xi.\ \|f *_v (x + \xi)\|)\ `\ (ball\ 0\ r))$›
  proof−
    have ‹$bdd\text{-}above\ ((\lambda\ \xi.\ \|f *_v x\|)\ `\ (ball\ 0\ r))$›

by *auto*
moreover have ‹*bdd-above* $((\lambda\ \xi.\ \|f *_v \xi\|)$ ‘ (*ball 0 r*))›
using *bdd-above-3* **by** *blast*
ultimately have ‹*bdd-above* $((\lambda\ \xi.\ \|f *_v x\| + \|f *_v \xi\|)$ ‘ (*ball 0 r*))›
by (*rule bdd-above-plus*)
then obtain M **where** ‹$\bigwedge \xi.\ \xi \in$ *ball 0 r* $\Longrightarrow \|f *_v x\| + \|f *_v \xi\| \le M$›
unfolding *bdd-above-def* **by** (*meson image-eqI*)
moreover have ‹$\|f *_v (x + \xi)\| \le \|f *_v x\| + \|f *_v \xi\|$› **for** ξ
by (*simp add*: *blinfun.add-right norm-triangle-ineq*)
ultimately have ‹$\bigwedge \xi.\ \xi \in$ *ball 0 r* $\Longrightarrow \|f *_v (x + \xi)\| \le M$›
by (*simp add*: *blinfun.add-right norm-triangle-le*)
thus *?thesis* **by** (*meson bdd-aboveI2*)
qed
have *bdd-above-4*: ‹*bdd-above* $((\lambda\ \xi.\ \|f *_v (x - \xi)\|)$ ‘ (*ball 0 r*))›
proof−
obtain K **where** *K-def*: ‹$\bigwedge \xi.\ \xi \in$ *ball 0 r* $\Longrightarrow \|f *_v (x + \xi)\| \le K$›
using ‹*bdd-above* ((λ ξ. *norm* (*f* (*x* + ξ))) ‘ (*ball 0 r*))› **unfolding** *bdd-above-def*
by (*meson image-eqI*)
have ‹$\xi \in$ *ball* (*0*::'*a*) *r* $\Longrightarrow -\xi \in$ *ball 0 r*› **for** ξ
by *auto*
thus *?thesis* **by** (*metis K-def ab-group-add-class.ab-diff-conv-add-uminus bdd-aboveI2*)
qed
have *bdd-above-1*: ‹*bdd-above* $((\lambda\ \xi.\ max\ \|f *_v (x + \xi)\|\ \ \|f *_v (x - \xi)\|)$ ‘ (*ball 0 r*))›
proof−
have ‹*bdd-above* $((\lambda\ \xi.\ \|f *_v (x + \xi)\|)$ ‘ (*ball 0 r*))›
using *bdd-above-2* **by** *blast*
moreover have ‹*bdd-above* $((\lambda\ \xi.\ \|f *_v (x - \xi)\|)$ ‘ (*ball 0 r*))›
using *bdd-above-4* **by** *blast*
ultimately show *?thesis*
unfolding *max-def* **apply** *auto* **apply** (*meson bdd-above-Int1 bdd-above-mono image-Int-subset*)
by (*meson bdd-above-Int1 bdd-above-mono image-Int-subset*)
qed
have *bdd-above-6*: ‹*bdd-above* $((\lambda t.\ \|f *_v t\|)$ ‘ *ball x r*)›
proof−
have ‹*bounded* (*ball x r*)›
by *simp*
hence ‹*bounded* $((\lambda t.\ \|f *_v t\|)$ ‘ *ball x r*)›
by (*metis* (*no-types*) *add.left-neutral bdd-above-2 bdd-above-norm bounded-norm-comp*

image-add-ball image-image)
thus *?thesis*
by (*simp add*: *bounded-imp-bdd-above*)
qed
have *norm-1*: ‹$(\lambda\xi.\ \|f *_v (x + \xi)\|)$ ‘ *ball 0 r* = $(\lambda t.\ \|f *_v t\|)$ ‘ *ball x r*›
by (*metis add.right-neutral ball-translation image-image*)
have *bdd-above-5*: ‹*bdd-above* $((\lambda\xi.\ norm\ (f\ (x + \xi)))$ ‘ *ball 0 r*)›

by (*simp add*: *bdd-above-2*)
have *norm-2*: ‹$\|\xi\| < r \Longrightarrow \|f *_v (x - \xi)\| \in (\lambda\xi. \|f *_v (x + \xi)\|)$ ‘ *ball 0 r*› **for** ξ
proof−
assume ‹$\|\xi\| < r$›
hence ‹$\xi \in$ *ball* (*0*::′*a*) *r*›
by *auto*
hence ‹$-\xi \in$ *ball* (*0*::′*a*) *r*›
by *auto*
thus *?thesis*
by (*metis* (*no-types*, *lifting*) *ab-group-add-class.ab-diff-conv-add-uminus image-iff*)
qed
have *norm-2′*: ‹$\|\xi\| < r \Longrightarrow \|f *_v (x + \xi)\| \in (\lambda\xi. \|f *_v (x - \xi)\|)$ ‘ *ball 0 r*› **for** ξ
proof−
assume ‹*norm* $\xi < r$›
hence ‹$\xi \in$ *ball* (*0*::′*a*) *r*›
by *auto*
hence ‹$-\xi \in$ *ball* (*0*::′*a*) *r*›
by *auto*
thus *?thesis*
by (*metis* (*no-types*, *lifting*) *diff-minus-eq-add image-iff*)
qed
have *bdd-above-6*: ‹*bdd-above* $((\lambda\xi. \|f *_v (x - \xi)\|)$ ‘ *ball 0 r*)›
by (*simp add*: *bdd-above-4*)
have *Sup-2*: ‹(*SUP* $\xi\in$*ball 0 r*. *max* $\|f *_v (x + \xi)\|$ $\|f *_v (x - \xi)\|$) =
max (*SUP* $\xi\in$*ball 0 r*. $\|f *_v (x + \xi)\|$) (*SUP* $\xi\in$*ball 0 r*. $\|f *_v (x - \xi)\|$)›
proof−
have ‹*ball* (*0*::′*a*) *r* $\neq \{\}$›
using ‹$r > 0$› **by** *auto*
moreover have ‹*bdd-above* $((\lambda\xi. \|f *_v (x + \xi)\|)$ ‘ *ball 0 r*)›
using *bdd-above-5* **by** *blast*
moreover have ‹*bdd-above* $((\lambda\xi. \|f *_v (x - \xi)\|)$ ‘ *ball 0 r*)›
using *bdd-above-6* **by** *blast*
ultimately show *?thesis*
using *max-Sup*
by (*metis* (*mono-tags*, *lifting*) *Banach-Steinhaus-Missing.pointwise-max-def image-cong*)
qed
have *Sup-3′*: ‹$\|\xi\| < r \Longrightarrow \|f *_v (x + \xi)\| \in (\lambda\xi. \|f *_v (x - \xi)\|)$ ‘ *ball 0 r*› **for** ξ::′*a*
by (*simp add*: *norm-2′*)
have *Sup-3′′*: ‹$\|\xi\| < r \Longrightarrow \|f *_v (x - \xi)\| \in (\lambda\xi. \|f *_v (x + \xi)\|)$ ‘ *ball 0 r*› **for** ξ::′*a*
by (*simp add*: *norm-2*)
have *Sup-3*: ‹*max* (*SUP* $\xi\in$*ball 0 r*. $\|f *_v (x + \xi)\|$) (*SUP* $\xi\in$*ball 0 r*. $\|f *_v (x - \xi)\|$) =

$(SUP\ \xi{\in}ball\ 0\ r.\ \|f *_v (x + \xi)\|)$›
proof−
have ‹$(\lambda\xi.\ \|f *_v (x + \xi)\|)$ ‘ $(ball\ 0\ r) = (\lambda\xi.\ \|f *_v (x - \xi)\|)$ ‘ $(ball\ 0\ r)$›
apply *auto* **using** *Sup-3′* **apply** *auto* **using** *Sup-3′′* **by** *blast*
hence ‹$Sup\ ((\lambda\xi.\ \|f *_v (x + \xi)\|)$ ‘ $(ball\ 0\ r))=Sup\ ((\lambda\xi.\ \|f *_v (x - \xi)\|)$ ‘ $(ball\ 0\ r))$›
by *simp*
thus *?thesis* **by** *simp*
qed
have *Sup-1*: ‹$Sup\ ((\lambda t.\ \|f *_v t\|)$ ‘ $(ball\ 0\ r)) \le Sup\ (\ (\lambda\xi.\ \|f *_v \xi\|)$ ‘ $(ball\ x\ r)\ )$›
proof−
have ‹$(\lambda t.\ \|f *_v t\|)\ \xi \le max\ \|f *_v (x + \xi)\|\ \|f *_v (x - \xi)\|$› **for** ξ
apply(*rule linear-plus-norm*) **apply** (*rule bounded-linear.linear*)
by (*simp add*: *blinfun.bounded-linear-right*)
moreover have ‹$bdd\text{-}above\ ((\lambda\ \xi.\ max\ \|f *_v (x + \xi)\|\ \ \|f *_v (x - \xi)\|)$ ‘ $(ball\ 0\ r))$›
using *bdd-above-1* **by** *blast*
moreover have ‹$ball\ (0{::}'a)\ r \neq \{\}$›
using ‹$r > 0$› **by** *auto*
ultimately have ‹$Sup\ ((\lambda t.\ \|f *_v t\|)$ ‘ $(ball\ 0\ r)) \le$
$Sup\ ((\lambda\xi.\ max\ \|f *_v (x + \xi)\|\ \|f *_v (x - \xi)\|)$ ‘ $(ball\ 0\ r))$›
using *cSUP-mono* **by** (*smt* (*verit*))
also have ‹$\ldots = max\ (Sup\ ((\lambda\xi.\ \ \|f *_v (x + \xi)\|)$ ‘ $(ball\ 0\ r)))$
$(Sup\ ((\lambda\xi.\ \|f *_v (x - \xi)\|)$ ‘ $(ball\ 0\ r)))$›
using *Sup-2* **by** *blast*
also have ‹$\ldots = Sup\ ((\lambda\xi.\ \|f *_v (x + \xi)\|)$ ‘ $(ball\ 0\ r))$›
using *Sup-3* **by** *blast*
also have ‹$\ldots = Sup\ ((\lambda\xi.\ \|f *_v \xi\|)$ ‘ $(ball\ x\ r))$›
by (*metis add.right-neutral ball-translation image-image*)
finally show *?thesis* **by** *blast*
qed
have ‹$\|f\| = (SUP\ x{\in}ball\ 0\ r.\ \|f *_v x\|) / r$›
using ‹$0 < r$› *onorm-r* **by** *blast*
moreover have ‹$Sup\ ((\lambda t.\ \|f *_v t\|)$ ‘ $(ball\ 0\ r)) / r \le Sup\ ((\lambda\xi.\ \|f *_v \xi\|)$ ‘ $(ball\ x\ r)) / r$›
using *Sup-1* ‹$0 < r$› *divide-right-mono* **by** *fastforce*
ultimately have ‹$\|f\| \le Sup\ ((\lambda t.\ \|f *_v t\|)$ ‘ $ball\ x\ r) / r$›
by *simp*
thus *?thesis* **by** *simp*
qed

lemma *onorm-Sup-on-ball′*:
includes *norm-syntax*
assumes ‹$r > 0$› **and** ‹$\tau < 1$›
shows ‹$\exists\,\xi{\in}ball\ x\ r.\ \ \tau * r * \|f\| \le \|f *_v \xi\|$›

In the proof of Banach-Steinhaus theorem, we will use this variation of the lemma *onorm-Sup-on-ball*.

Explanation: Let f be a bounded operator, let x be a point and let r be a positive real number. For any real number $\tau < 1$, there is a point ξ in the open ball of radius r around x such that $\tau * r * \|f\| \leq \|f *_v \xi\|$.

proof(*cases* ‹$f = 0$›)
  case *True*
   thus *?thesis* **by** (*metis assms*(*1*) *centre-in-ball mult-zero-right norm-zero order-refl*
        zero-blinfun.rep-eq)
next
  case *False*
  have *bdd-above-1*: ‹*bdd-above* ((λt. $\|(*_v)\ f\ t\|$) ‘ *ball x r*)› **for** f::‹$'a \Rightarrow_L 'b$›
    using *assms*(*1*) *bounded-linear-image* **by** (*simp add*: *bounded-linear-image*
        blinfun.bounded-linear-right bounded-imp-bdd-above bounded-norm-comp)
  have ‹*norm f* > 0›
    using ‹$f \neq 0$› **by** *auto*
  have ‹*norm f* $\leq$ *Sup* (($\lambda\xi$. $\|(*_v)\ f\ \xi\|$) ‘ (*ball x r*)) / r›
    using ‹$r > 0$› **by** (*simp add*: *onorm-Sup-on-ball*)
  hence ‹r ∗ *norm f* $\leq$ *Sup* (($\lambda\xi$. $\|(*_v)\ f\ \xi\|$) ‘ (*ball x r*))›
   using ‹$0 < r$› **by** (*smt* (*verit*) *divide-strict-right-mono nonzero-mult-div-cancel-left*)

  moreover have ‹$\tau * r *$ *norm f* $< r *$ *norm f*›
    using ‹$\tau < 1$› **using** ‹$0 <$ *norm f*› ‹$0 < r$› **by** *auto*
  ultimately have ‹$\tau * r *$ *norm f* $<$ *Sup* ((*norm* ∘ ($(*_v)$ f)) ‘ (*ball x r*))›
    by *simp*
  moreover have ‹(*norm* ∘ ($(*_v)$ f)) ‘ (*ball x r*) $\neq \{\}$›
    using ‹$0 < r$› **by** *auto*
  moreover have ‹*bdd-above* ((*norm* ∘ ($(*_v)$ f)) ‘ (*ball x r*))›
    using *bdd-above-1* **apply** *transfer* **by** *simp*
  ultimately have ‹$\exists\, t \in$ (*norm* ∘ ($(*_v)$ f)) ‘ (*ball x r*). $\tau * r *$ *norm f* $< t$›
    by (*simp add*: *less-cSup-iff*)
  thus *?thesis* **by** (*smt* (*verit*) *comp-def image-iff*)
qed

2.2 Banach-Steinhaus theorem

theorem *banach-steinhaus*:
  fixes f::‹$'c \Rightarrow ('a$::*banach* $\Rightarrow_L 'b$::*real-normed-vector*)›
  assumes ‹$\bigwedge x$. *bounded* (*range* (λn. (f n) $*_v$ x))›
  shows ‹*bounded* (*range f*)›

This is Banach-Steinhaus Theorem.

Explanation: If a family of bounded operators on a Banach space is pointwise bounded, then it is uniformly bounded.

proof(*rule classical*)
  assume ‹¬(*bounded* (*range f*))›
  have *sum-1*: ‹$\exists K. \forall n.$ *sum* (λk. *inverse* (*real-of-nat* $3\hat{}k$)) $\{0..n\} \leq K$›
  proof−
    have ‹*summable* (λn. *inverse* ((*3*::*real*) ^ n))›
      by (*simp flip*: *power-inverse*)
    hence ‹*bounded* (*range* (λn. *sum* (λ k. *inverse* (*real 3* ^ k)) $\{0..<n\}$))›

```
    using summable-imp-sums-bounded[where f = (λn. inverse (real-of-nat 3^n))]
      lessThan-atLeast0 by auto
    hence ‹∃ M. ∀ h∈(range (λn. sum (λ k. inverse (real 3 ^ k)) {0..<n})). norm h ≤ M›
      using bounded-iff by blast
    then obtain M where ‹h∈range (λn. sum (λ k. inverse (real 3 ^ k)) {0..<n}) ⟹ norm h ≤ M›
      for h
      by blast
    have sum-2: ‹sum (λk. inverse (real-of-nat 3^k)) {0..n} ≤ M› for n
    proof−
      have ‹norm (sum (λ k. inverse (real 3 ^ k)) {0..< Suc n}) ≤ M›
        using ‹⋀h. h∈(range (λn. sum (λ k. inverse (real 3 ^ k)) {0..<n})) ⟹ norm h ≤ M›
        by blast
      hence ‹norm (sum (λ k. inverse (real 3 ^ k)) {0..n}) ≤ M›
        by (simp add: atLeastLessThanSuc-atLeastAtMost)
      hence ‹(sum (λ k. inverse (real 3 ^ k)) {0..n}) ≤ M›
        by auto
      thus ?thesis by blast
    qed
    have ‹sum (λk. inverse (real-of-nat 3^k)) {0..n} ≤ M› for n
      using sum-2 by blast
    thus ?thesis by blast
  qed
  have ‹of-rat 2/3 < (1::real)›
    by auto
  hence ‹∀ g::'a ⇒L 'b. ∀ x. ∀ r. ∃ξ. g ≠ 0 ∧ r > 0
             ⟶ (ξ∈ball x r ∧ (of-rat 2/3) ∗ r ∗ norm g ≤ norm ((∗v) g ξ))›
    using onorm-Sup-on-ball' by blast
  hence ‹∃ ξ. ∀ g::'a ⇒L 'b. ∀ x. ∀ r. g ≠ 0 ∧ r > 0
           ⟶ ((ξ g x r)∈ball x r ∧ (of-rat 2/3) ∗ r ∗ norm g ≤ norm ((∗v) g (ξ g x r)))›
    by metis
  then obtain ξ where f1: ‹⟦g ≠ 0; r > 0⟧ ⟹
        ξ g x r ∈ ball x r ∧ (of-rat 2/3) ∗ r ∗ norm g ≤ norm ((∗v) g (ξ g x r))›
    for g::‹'a ⇒L 'b› and x and r
    by blast
  have ‹∀ n. ∃ k. norm (f k) ≥ 4^n›
    using ‹¬(bounded (range f))› by (metis (mono-tags, opaque-lifting) boundedI image-iff linear)
  hence ‹∃ k. ∀ n. norm (f (k n)) ≥ 4^n›
    by metis
  hence ‹∃ k. ∀ n. norm ((f ∘ k) n) ≥ 4^n›
    by simp
  then obtain k where ‹norm ((f ∘ k) n) ≥ 4^n› for n
    by blast
  define T where ‹T = f ∘ k›
  have ‹T n ∈ range f› for n
```

unfolding *T-def* **by** *simp*
have ‹$norm\ (T\ n) \geq of\text{-}nat\ (4\hat{\ }n)$› **for** n
unfolding *T-def* **using** ‹$\bigwedge n.\ norm\ ((f \circ k)\ n) \geq 4\hat{\ }n$› **by** *auto*
hence ‹$T\ n \neq 0$› **for** n
by (*smt* (*verit*) *T-def* ‹$\bigwedge n.\ 4\ \hat{\ }\ n \leq norm\ ((f \circ k)\ n)$› *norm-zero power-not-zero zero-le-power*)
have ‹$inverse\ (of\text{-}nat\ 3\hat{\ }n) > (0::real)$› **for** n
by *auto*
define y::‹$nat \Rightarrow 'a$› **where** ‹$y = rec\text{-}nat\ 0\ (\lambda n\ x.\ \xi\ (T\ n)\ x\ (inverse\ (of\text{-}nat\ 3\hat{\ }n)))$›
have ‹$y\ (Suc\ n) \in ball\ (y\ n)\ (inverse\ (of\text{-}nat\ 3\hat{\ }n))$› **for** n
using *f1* ‹$\bigwedge n.\ T\ n \neq 0$› ‹$\bigwedge n.\ inverse\ (of\text{-}nat\ 3\hat{\ }n) > 0$› **unfolding** *y-def* **by** *auto*
hence ‹$norm\ (y\ (Suc\ n) - y\ n) \leq inverse\ (of\text{-}nat\ 3\hat{\ }n)$› **for** n
unfolding *ball-def* **apply** *auto* **using** *dist-norm* **by** (*smt* (*verit*) *norm-minus-commute*)

moreover have ‹$\exists K.\ \forall n.\ sum\ (\lambda k.\ inverse\ (real\text{-}of\text{-}nat\ 3\hat{\ }k))\ \{0..n\} \leq K$›
using *sum-1* **by** *blast*
moreover have ‹*Cauchy y*›
using *convergent-series-Cauchy*[**where** $a = \lambda n.\ inverse\ (of\text{-}nat\ 3\hat{\ }n)$ **and** $\varphi = y$] *dist-norm*
by (*metis calculation*(*1*) *calculation*(*2*))
hence ‹$\exists\ x.\ y \longrightarrow x$›
by (*simp add*: *convergent-eq-Cauchy*)
then obtain x **where** ‹$y \longrightarrow x$›
by *blast*
have *norm-2*: ‹$norm\ (x - y\ (Suc\ n)) \leq (inverse\ (of\text{-}nat\ 2)) * (inverse\ (of\text{-}nat\ 3\hat{\ }n))$› **for** n
proof−
have ‹$inverse\ (real\text{-}of\text{-}nat\ 3) < 1$›
by *simp*
moreover have ‹$y\ 0 = 0$›
using *y-def* **by** *auto*
ultimately have ‹$norm\ (x - y\ (Suc\ n))$
$\leq (inverse\ (of\text{-}nat\ 3)) * inverse\ (1 - (inverse\ (of\text{-}nat\ 3))) * ((inverse\ (of\text{-}nat\ 3))\ \hat{\ }\ n)$›
using *bound-Cauchy-to-lim*[**where** $c = inverse\ (of\text{-}nat\ 3)$ **and** $y = y$ **and** $x = x$]
power-inverse semiring-norm(*77*) ‹$y \longrightarrow x$›
‹$\bigwedge n.\ norm\ (y\ (Suc\ n) - y\ n) \leq inverse\ (of\text{-}nat\ 3\hat{\ }n)$› **by** (*metis divide-inverse*)
moreover have ‹$inverse\ (real\text{-}of\text{-}nat\ 3) * inverse\ (1 - (inverse\ (of\text{-}nat\ 3)))$
$= inverse\ (of\text{-}nat\ 2)$›
by *auto*
ultimately show *?thesis*
by (*metis power-inverse*)
qed
have ‹$norm\ (x - y\ (Suc\ n)) \leq (inverse\ (of\text{-}nat\ 2)) * (inverse\ (of\text{-}nat\ 3\hat{\ }n))$› **for** n

using *norm-2* **by** *blast*
have ‹∃ *M*. ∀ *n*. *norm* (($*_v$) (*T n*) *x*) ≤ *M*›
unfolding *T-def* **apply** *auto*
by (*metis* ‹⋀*x*. *bounded* (*range* (λ*n*. ($*_v$) (*f n*) *x*))› *bounded-iff rangeI*)
then obtain *M* **where** ‹*norm* (($*_v$) (*T n*) *x*) ≤ *M*› **for** *n*
by *blast*
have *norm-1*: ‹*norm* (*T n*) ∗ *norm* (*y* (*Suc n*) − *x*) + *norm* (($*_v$) (*T n*) *x*)
≤ *inverse* (*real 2*) ∗ *inverse* (*real 3* ^ *n*) ∗ *norm* (*T n*) + *norm* (($*_v$) (*T n*) *x*)› **for** *n*
proof−
have ‹*norm* (*y* (*Suc n*) − *x*) ≤ (*inverse* (*of-nat 2*))∗(*inverse* (*of-nat 3*^*n*))›
using ‹*norm* (*x* − *y* (*Suc n*)) ≤ (*inverse* (*of-nat 2*))∗(*inverse* (*of-nat 3*^*n*))›
by (*simp add*: *norm-minus-commute*)
moreover have ‹*norm* (*T n*) ≥ *0*›
by *auto*
ultimately have ‹*norm* (*T n*) ∗ *norm* (*y* (*Suc n*) − *x*)
≤ (*inverse* (*of-nat 2*))∗(*inverse* (*of-nat 3*^*n*))∗*norm* (*T n*)›
by (*simp add*: ‹⋀*n*. *T n* ≠ *0*›)
thus *?thesis* **by** *simp*
qed
have *inverse-2*: ‹(*inverse* (*of-nat 6*)) ∗ *inverse* (*real 3* ^ *n*) ∗ *norm* (*T n*)
≤ *norm* (($*_v$) (*T n*) *x*)› **for** *n*
proof−
have ‹(*of-rat 2/3*)∗(*inverse* (*of-nat 3*^*n*))∗*norm* (*T n*) ≤ *norm* (($*_v$) (*T n*) (*y* (*Suc n*)))›
using *f1* ‹⋀ *n*. *T n* ≠ *0*› ‹⋀ *n*. *inverse* (*of-nat 3*^*n*) > *0*› **unfolding** *y-def* **by** *auto*
also have ‹... = *norm* (($*_v$) (*T n*) ((*y* (*Suc n*) − *x*) + *x*))›
by *auto*
also have ‹... = *norm* (($*_v$) (*T n*) (*y* (*Suc n*) − *x*) + ($*_v$) (*T n*) *x*)›
apply *transfer* **apply** *auto* **by** (*metis diff-add-cancel linear-simps*(*1*))
also have ‹... ≤ *norm* (($*_v$) (*T n*) (*y* (*Suc n*) − *x*)) + *norm* (($*_v$) (*T n*) *x*)›
by (*simp add*: *norm-triangle-ineq*)
also have ‹... ≤ *norm* (*T n*) ∗ *norm* (*y* (*Suc n*) − *x*) + *norm* (($*_v$) (*T n*) *x*)›
apply *transfer* **apply** *auto* **using** *onorm* **by** *auto*
also have ‹... ≤ (*inverse* (*of-nat 2*))∗(*inverse* (*of-nat 3*^*n*))∗*norm* (*T n*)
+ *norm* (($*_v$) (*T n*) *x*)›
using *norm-1* **by** *blast*
finally have ‹(*of-rat 2/3*) ∗ *inverse* (*real 3* ^ *n*) ∗ *norm* (*T n*)
≤ *inverse* (*real 2*) ∗ *inverse* (*real 3* ^ *n*) ∗ *norm* (*T n*)
+ *norm* (($*_v$) (*T n*) *x*)›
by *blast*
hence ‹(*of-rat 2/3*) ∗ *inverse* (*real 3* ^ *n*) ∗ *norm* (*T n*)
− *inverse* (*real 2*) ∗ *inverse* (*real 3* ^ *n*) ∗ *norm* (*T n*) ≤ *norm* (($*_v$) (*T n*) *x*)›
by *linarith*
moreover have ‹(*of-rat 2/3*) ∗ *inverse* (*real 3* ^ *n*) ∗ *norm* (*T n*)
− *inverse* (*real 2*) ∗ *inverse* (*real 3* ^ *n*) ∗ *norm* (*T n*)
= (*inverse* (*of-nat 6*)) ∗ *inverse* (*real 3* ^ *n*) ∗ *norm* (*T n*)›

by *fastforce*
ultimately show ‹(*inverse* (*of-nat 6*)) ∗ *inverse* (*real 3* ^ *n*) ∗ *norm* (*T n*) ≤ *norm* ((∗$_v$) (*T n*) *x*)›
by *linarith*
qed
have *inverse-3*: ‹(*inverse* (*of-nat 6*)) ∗ (*of-rat* (*4/3*)^*n*)
≤ (*inverse* (*of-nat 6*)) ∗ *inverse* (*real 3* ^ *n*) ∗ *norm* (*T n*)› **for** *n*
proof−
have ‹*of-rat* (*4/3*)^*n* = *inverse* (*real 3* ^ *n*) ∗ (*of-nat 4*^*n*)›
apply *auto* **by** (*metis divide-inverse-commute of-rat-divide power-divide of-rat-numeral-eq*)
also have ‹. . . ≤ *inverse* (*real 3* ^ *n*) ∗ *norm* (*T n*)›
using ‹⋀*n*. *norm* (*T n*) ≥ *of-nat* (*4*^*n*)› **by** *simp*
finally have ‹*of-rat* (*4/3*)^*n* ≤ *inverse* (*real 3* ^ *n*) ∗ *norm* (*T n*)›
by *blast*
moreover have ‹*inverse* (*of-nat 6*) > (*0*::*real*)›
by *auto*
ultimately show *?thesis* **by** *auto*
qed
have *inverse-1*: ‹(*inverse* (*of-nat 6*)) ∗ (*of-rat* (*4/3*)^*n*) ≤ *M*› **for** *n*
proof−
have ‹(*inverse* (*of-nat 6*)) ∗ (*of-rat* (*4/3*)^*n*)
≤ (*inverse* (*of-nat 6*)) ∗ *inverse* (*real 3* ^ *n*) ∗ *norm* (*T n*)›
using *inverse-3* **by** *blast*
also have ‹. . . ≤ *norm* ((∗$_v$) (*T n*) *x*)›
using *inverse-2* **by** *blast*
finally have ‹(*inverse* (*of-nat 6*)) ∗ (*of-rat* (*4/3*)^*n*) ≤ *norm* ((∗$_v$) (*T n*) *x*)›
by *auto*
thus *?thesis* **using** ‹⋀ *n*. *norm* ((∗$_v$) (*T n*) *x*) ≤ *M*› **by** (*smt* (*verit*))
qed
have ‹∃ *n*. *M* < (*inverse* (*of-nat 6*)) ∗ (*of-rat* (*4/3*)^*n*)›
using *Real.real-arch-pow* **by** *auto*
moreover have ‹(*inverse* (*of-nat 6*)) ∗ (*of-rat* (*4/3*)^*n*) ≤ *M*› **for** *n*
using *inverse-1* **by** *blast*
ultimately show *?thesis* **by** (*smt* (*verit*))
qed

2.3 A consequence of Banach-Steinhaus theorem

corollary *bounded-linear-limit-bounded-linear*:
fixes *f*::‹*nat* ⇒ (′*a*::*banach* ⇒$_L$ ′*b*::*real-normed-vector*)›
assumes ‹⋀*x*. *convergent* (λ*n*. (*f n*) ∗$_v$ *x*)›
shows ‹∃ *g*. (λ*n*. (∗$_v$) (*f n*)) −*pointwise*→ (∗$_v$) *g*›

Explanation: If a sequence of bounded operators on a Banach space converges pointwise, then the limit is also a bounded operator.

proof−
have ‹∃ *l*. (λ*n*. (∗$_v$) (*f n*) *x*) ⟶ *l*› **for** *x*
by (*simp add*: ‹⋀*x*. *convergent* (λ*n*. (∗$_v$) (*f n*) *x*)› *convergentD*)

hence ‹∃ *F*. (λ*n*. $(*_v)$ (*f n*)) −*pointwise*→ *F*›
unfolding *pointwise-convergent-to-def* **by** *metis*
obtain *F* **where** ‹(λ*n*. $(*_v)$ (*f n*)) −*pointwise*→ *F*›
using ‹∃ *F*. (λ*n*. $(*_v)$ (*f n*)) −*pointwise*→ *F*› **by** *auto*
have ‹⋀*x*. (λ *n*. $(*_v)$ (*f n*) *x*) ⟶ *F x*›
using ‹(λ*n*. $(*_v)$ (*f n*)) −*pointwise*→ *F*› **apply** *transfer*
by (*simp add*: *pointwise-convergent-to-def*)
have ‹*bounded* (*range f*)›
using ‹⋀*x*. *convergent* (λ*n*. $(*_v)$ (*f n*) *x*)› *banach-steinhaus*
‹⋀*x*. ∃ *l*. (λ*n*. $(*_v)$ (*f n*) *x*) ⟶ *l*› *convergent-imp-bounded* **by** *blast*
have *norm-f-n*: ‹∃ *M*. ∀ *n*. *norm* (*f n*) ≤ *M*›
unfolding *bounded-def*
by (*meson UNIV-I* ‹*bounded* (*range f*)› *bounded-iff image-eqI*)
have ‹*isCont* (λ *t*::′*b*. *norm t*) *y*› **for** *y*::′*b*
using *Limits.isCont-norm* **by** *simp*
hence ‹(λ *n*. *norm* ($(*_v)$ (*f n*) *x*)) ⟶ (*norm* (*F x*))› **for** *x*
using ‹⋀ *x*::′*a*. (λ *n*. $(*_v)$ (*f n*) *x*) ⟶ *F x*› **by** (*simp add*: *tendsto-norm*)
hence *norm-f-n-x*: ‹∃ *M*. ∀ *n*. *norm* ($(*_v)$ (*f n*) *x*) ≤ *M*› **for** *x*
using *Elementary-Metric-Spaces.convergent-imp-bounded*
by (*metis UNIV-I* ‹⋀ *x*::′*a*. (λ *n*. $(*_v)$ (*f n*) *x*) ⟶ *F x*› *bounded-iff image-eqI*)
have *norm-f*: ‹∃ *K*. ∀ *n*. ∀ *x*. *norm* ($(*_v)$ (*f n*) *x*) ≤ *norm x* ∗ *K*›
proof−
have ‹∃ *M*. ∀ *n*. *norm* ($(*_v)$ (*f n*) *x*) ≤ *M*› **for** *x*
using *norm-f-n-x* ‹⋀*x*. (λ*n*. $(*_v)$ (*f n*) *x*) ⟶ *F x*› **by** *blast*
hence ‹∃ *M*. ∀ *n*. *norm* (*f n*) ≤ *M*›
using *norm-f-n* **by** *simp*
then obtain *M*::*real* **where** ‹∃ *M*. ∀ *n*. *norm* (*f n*) ≤ *M*›
by *blast*
have ‹∀ *n*. ∀ *x*. *norm* ($(*_v)$ (*f n*) *x*) ≤ *norm x* ∗ *norm* (*f n*)›
apply *transfer* **apply** *auto* **by** (*metis mult.commute onorm*)
thus *?thesis* **using** ‹∃ *M*. ∀ *n*. *norm* (*f n*) ≤ *M*›
by (*metis* (*no-types*, *opaque-lifting*) *dual-order.trans norm-eq-zero order-refl mult-le-cancel-left-pos vector-space-over-itself.scale-zero-left zero-less-norm-iff*)
qed
have *norm-F-x*: ‹∃ *K*. ∀ *x*. *norm* (*F x*) ≤ *norm x* ∗ *K*›
proof−
have ∃ *K*. ∀ *n*. ∀ *x*. *norm* ($(*_v)$ (*f n*) *x*) ≤ *norm x* ∗ *K*
using *norm-f* ‹⋀*x*. (λ*n*. $(*_v)$ (*f n*) *x*) ⟶ *F x*› **by** *auto*
thus *?thesis*
using ‹⋀ *x*::′*a*. (λ *n*. $(*_v)$ (*f n*) *x*) ⟶ *F x*› **apply** *transfer*
by (*metis Lim-bounded tendsto-norm*)
qed
have ‹*linear F*›
proof(*rule linear-limit-linear*)
show ‹*linear* ($(*_v)$ (*f n*))› **for** *n*
apply *transfer* **apply** *auto* **by** (*simp add*: *bounded-linear.linear*)
show ‹*f* −*pointwise*→ *F*›
using ‹(λ*n*. $(*_v)$ (*f n*)) −*pointwise*→ *F*› **by** *auto*

```
  qed
  moreover have ‹bounded-linear-axioms F›
   using norm-F-x by (simp add: ‹⋀x. (λn. (*_v) (f n) x) ⟶ F x› bounded-linear-axioms-def)

  ultimately have ‹bounded-linear F›
    unfolding bounded-linear-def by blast
  hence ‹∃ g. (*_v) g = F›
    using bounded-linear-Blinfun-apply by auto
  thus ?thesis using ‹(λn. (*_v) (f n)) −pointwise→ F› apply transfer by auto
qed

end
```

References

[1] S. Banach and H. Steinhaus. Sur le principe de la condensation de singularités. *Fundamenta Mathematicae*, 1(9):50–61, 1927.

[2] M. S. Moslehian and E. W. Weisstein. Uniform boundedness principle. *From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/UniformBoundednessPrinciple.html.*

[3] A. D. Sokal. A really simple elementary proof of the uniform boundedness theorem. *The American Mathematical Monthly*, 118(5):450–452, 2011.