

The Euler–MacLaurin summation formula

Manuel Eberl

May 26, 2024

Abstract

The Euler–MacLaurin formula relates the value of a discrete sum $\sum_{i=a}^b f(i)$ to that of the integral $\int_a^b f(x) dx$ in terms of the derivatives of f at a and b and a remainder term. Since the remainder term is often very small as b grows, this can be used to compute asymptotic expansions for sums.

This entry contains a proof of this formula for functions from the reals to an arbitrary Banach space. Two variants of the formula are given: the standard textbook version and a variant outlined in *Concrete Mathematics* [3] that is more useful for deriving asymptotic estimates.

As example applications, we use that formula to derive the full asymptotic expansion of the harmonic numbers and the sum of inverse squares.

Contents

1	The Euler–MacLaurin summation formula	2
1.1	Auxiliary facts	2
1.2	The remainder terms	3
1.3	The conventional version of the Euler–MacLaurin formula . .	22
1.4	The “Concrete Mathematics” version of the Euler–MacLaurin formula	24
1.5	Bounds on the remainder term	30
1.6	Application to harmonic numbers	34
1.7	Application to sums of inverse squares	35
2	Connection of Euler–MacLaurin summation to Landau symbols	37
2.1	O -bound for the remainder term	37
2.2	Asymptotic expansion of the harmonic numbers	38
2.3	Asymptotic expansion of the sum of inverse squares	40

1 The Euler–MacLaurin summation formula

theory *Euler-MacLaurin*

imports

HOL-Complex-Analysis.Complex-Analysis

Bernoulli.Periodic-Bernpoly

Bernoulli.Bernoulli-FPS

begin

1.1 Auxiliary facts

lemma *pbernpoly-of-int [simp]*: $pbernpoly\ n\ (of-int\ a) = bernoulli\ n$
by (*simp add: pbernpoly-def*)

lemma *continuous-on-bernpoly'* [*continuous-intros*]:

assumes *continuous-on A f*

shows *continuous-on A* $(\lambda x. bernpoly\ n\ (f\ x)) :: 'a :: real-normed-algebra-1$

using *continuous-on-compose2* [*OF continuous-on-bernpoly assms, of UNIV n*] **by**
auto

lemma *sum-atLeastAtMost-int-last*:

assumes $a < (b :: int)$

shows $sum\ f\ \{a..b\} = sum\ f\ \{a..<b\} + f\ b$

proof –

from *assms* **have** $\{a..b\} = insert\ b\ \{a..<b\}$ **by** *auto*

also **have** $sum\ f\ \dots = sum\ f\ \{a..<b\} + f\ b$

by (*subst sum.insert*) (*auto simp: add-ac*)

finally **show** *?thesis* .

qed

lemma *sum-atLeastAtMost-int-head*:

assumes $a < (b :: int)$

shows $sum\ f\ \{a..b\} = f\ a + sum\ f\ \{a<..b\}$

proof –

from *assms* **have** $\{a..b\} = insert\ a\ \{a<..b\}$ **by** *auto*

also **have** $sum\ f\ \dots = f\ a + sum\ f\ \{a<..b\}$

by (*subst sum.insert*) *auto*

finally **show** *?thesis* .

qed

lemma *not-in-nonpos-Reals-imp-add-nonzero*:

assumes $z \notin \mathbb{R}_{\leq 0} \ x \geq 0$

shows $z + of-real\ x \neq 0$

using *assms* **by** (*auto simp: add-eq-0-iff2*)

lemma *negligible-atLeastAtMostI*: $b \leq a \implies negligible\ \{a..(b::real)\}$

by (*cases b < a*) *auto*

lemma *integrable-on-negligible*:

negligible $A \implies (f :: 'n :: \text{euclidean-space} \Rightarrow 'a :: \text{banach}) \text{ integrable-on } A$
by (*subst integrable-spike-set-eq*[*of - {}*]) (*simp-all add: integrable-on-empty*)

lemma *Union-atLeastAtMost-real-of-int*:

assumes $a < b$

shows $(\bigcup n \in \{a..<b\}. \{\text{real-of-int } n.. \text{real-of-int } (n + 1)\}) = \{\text{real-of-int } a.. \text{real-of-int } b\}$

proof (*intro equalityI subsetI*)

fix x **assume** $x \in \{\text{real-of-int } a.. \text{real-of-int } b\}$

thus $x \in (\bigcup n \in \{a..<b\}. \{\text{real-of-int } n.. \text{real-of-int } (n + 1)\})$

proof (*cases* $x = \text{real-of-int } b$)

case *True*

with *assms* **show** *?thesis* **by** (*auto intro!: bexI*[*of - b - 1*])

next

case *False*

with x **have** $x \geq \text{real-of-int } a$ $x < \text{real-of-int } b$ **by** *simp-all*

hence $x \geq \text{of-int } \lfloor x \rfloor$ $x \leq \text{of-int } \lfloor x \rfloor + 1$ **by** *linarith+*

moreover from x **have** $\lfloor x \rfloor \geq a$ $\lfloor x \rfloor < b$ **by** *linarith+*

ultimately have $\exists n \in \{a..<b\}. x \in \{\text{of-int } n.. \text{of-int } (n + 1)\}$

by (*intro bexI*[*of - \lfloor x \rfloor*]) *simp-all*

thus *?thesis* **by** *blast*

qed

qed *auto*

1.2 The remainder terms

The following describes the remainder term in the classical version of the Euler–MacLaurin formula.

definition *EM-remainder'* $:: \text{nat} \Rightarrow (\text{real} \Rightarrow 'a :: \text{banach}) \Rightarrow \text{real} \Rightarrow \text{real} \Rightarrow 'a$
where

*EM-remainder' n f a b = ((-1) ^ Suc n / fact n) *_R integral {a..b} (λt . *pbernpoly n t* *_R *f t*)*

Next, we define the remainder term that occurs when one lets the right bound of summation in the Euler–MacLaurin formula tend to infinity.

definition *EM-remainder-converges* $:: \text{nat} \Rightarrow (\text{real} \Rightarrow 'a :: \text{banach}) \Rightarrow \text{int} \Rightarrow \text{bool}$
where

EM-remainder-converges n f a $\longleftrightarrow (\exists L. ((\lambda x. \text{EM-remainder}' n f a (\text{of-int } x)) \longrightarrow L) \text{ at-top})$

definition *EM-remainder* $:: \text{nat} \Rightarrow (\text{real} \Rightarrow 'a :: \text{banach}) \Rightarrow \text{int} \Rightarrow 'a$ **where**

EM-remainder n f a =

(if EM-remainder-converges n f a then

Lim at-top ($\lambda x. \text{EM-remainder}' n f a (\text{of-int } x)$) else 0)

The following lemmas are fairly obvious – but tedious to prove – properties of the remainder terms.

lemma *EM-remainder-eqI*:

fixes L
assumes $((\lambda x. EM\text{-remainder}' n f b (of\text{-int } x)) \longrightarrow L)$ *at-top*
shows $EM\text{-remainder } n f b = L$
using *assms by (auto simp: EM-remainder-def EM-remainder-converges-def intro!: tendsto-Lim)*

lemma *integrable-EM-remainder'-int:*

fixes $a b :: int$ **and** $f :: real \Rightarrow 'a :: banach$
assumes *continuous-on {of-int a..of-int b} f*
shows $(\lambda t. pbernpoly n t *_R f t)$ *integrable-on {a..b}*
proof –
have [*continuous-intros*]: *continuous-on A f if $A \subseteq \{of\text{-int } a..of\text{-int } b\}$ for A*
using *continuous-on-subset[OF assms that]* .
consider $a > b \mid a = b \mid a < b \ n = 1 \mid a < b \ n \neq 1$
by (*cases a b rule: linorder-cases*) *auto*
thus *?thesis*
proof *cases*
assume $a < b$ **and** $n \neq 1$
thus *?thesis* **by** (*intro integrable-continuous-real continuous-intros*) *auto*
next
assume $ab: a < b$ **and** [*simp*]: $n = 1$
let $?A = (\lambda n. \{real\text{-of-int } n..real\text{-of-int } (n+1)\})$ ‘ $\{a..<b\}$
show *?thesis*
proof (*rule integrable-combine-division; (intro ballI)?*)
show $?A$ *division-of {of-int a..of-int b}*
using *Union-atLeastAtMost-real-of-int[OF ab]* **by** (*simp add: division-of-def*)
next
fix I **assume** $I \in ?A$
then obtain i **where** $i \in \{a..<b\}$ $I = \{of\text{-int } i..of\text{-int } (i + 1)\}$ **by** *auto*
show $(\lambda t. pbernpoly n t *_R f t)$ *integrable-on I*
proof (*rule integrable-spike*)
show $(\lambda t. (t - of\text{-int } i - 1/2) *_R f t)$ *integrable-on I*
using i **by** (*auto intro!: integrable-continuous-real continuous-intros*)
next
fix x **assume** $x \in I - \{of\text{-int } (i + 1)\}$
with i **have** $of\text{-int } i \leq x < of\text{-int } i + 1$ **by** *simp-all*
hence $\text{floor } x = i$ **by** *linarith*
thus $pbernpoly n x *_R f x = (x - of\text{-int } i - 1 / 2) *_R f x$
by (*simp add: pbernpoly-def bernpoly-def frac-def*)
qed *simp-all*
qed
qed (*simp-all add: integrable-on-negligible*)
qed

lemma *integrable-EM-remainder':*

fixes $a b :: real$ **and** $f :: real \Rightarrow 'a :: banach$
assumes *continuous-on {a..b} f*
shows $(\lambda t. pbernpoly n t *_R f t)$ *integrable-on {a..b}*
proof (*cases $\lceil a \rceil \leq \lfloor b \rfloor$*)

```

case True
define a' b' where a' = ⌈a⌉ and b' = ⌊b⌋
from True have *: a' ≤ b' a' ≥ a b' ≤ b by (auto simp: a'-def b'-def)
from * have A: (λt. pbernpoly n t *R f t) integrable-on ({a'..b'})
  by (intro integrable-EM-remainder'-int continuous-on-subset[OF assms]) auto
have B: (λt. pbernpoly n t *R f t) integrable-on ({a..a'})
proof (rule integrable-spike)
  show pbernpoly n x *R f x = bernpoly n (x - of-int (floor a)) *R f x
    if x ∈ {a..real-of-int a'} - {real-of-int a'} for x
  proof -
    have x ≥ a x < real-of-int a' using that by auto
    with True have floor x = floor a unfolding a'-def
      using ceiling-diff-floor-le-1 [of a] by (intro floor-unique; linarith)
    thus ?thesis by (simp add: pbernpoly-def frac-def)
  qed
qed (insert *, auto intro!: continuous-intros integrable-continuous-real
  continuous-on-subset[OF assms])
have C: (λt. pbernpoly n t *R f t) integrable-on ({b'..b})
proof (rule integrable-spike)
  show pbernpoly n x *R f x = bernpoly n (x - of-int b') *R f x
    if x ∈ {real-of-int b'..b} - {real-of-int b'} for x
  proof -
    have x ≤ b x > real-of-int b' using that by auto
    with True have floor x = b' unfolding b'-def by (intro floor-unique; linarith)

    thus ?thesis by (simp add: pbernpoly-def frac-def)
  qed
qed (insert *, auto intro!: continuous-intros integrable-continuous-real
  continuous-on-subset[OF assms])
have (λt. pbernpoly n t *R f t) integrable-on ({a..a'} ∪ {a'..b'} ∪ {b'..b}) using
* A B C
  by (intro integrable-Un; (subst ivl-disj-un)?)
  (auto simp: ivl-disj-un max-def min-def)
also have {a..a'} ∪ {a'..b'} ∪ {b'..b} = {a..b} using * by auto
finally show ?thesis .
next
assume *: ¬ceiling a ≤ floor b
show ?thesis
proof (rule integrable-spike)
  show (λt. bernpoly n (t - floor a) *R f t) integrable-on {a..b} using *
  by (auto intro!: integrable-continuous-real continuous-intros assms)
next
show pbernpoly n x *R f x = bernpoly n (x - floor a) *R f x
  if x ∈ {a..b} - {} for x
proof -
  from * have **: b < floor a + 1
    unfolding ceiling-altdef by (auto split: if-splits simp: le-floor-iff)
  from that have x: x ≥ a x ≤ b by simp-all
  with ** have floor x = floor a by linarith

```

thus *?thesis* **by** (*simp add: pbernpoly-def frac-def*)
qed
qed *simp-all*
qed

lemma *EM-remainder'-bounded-linear*:

assumes *bounded-linear h*
assumes *continuous-on {a..b} f*
shows $EM\text{-remainder}'\ n\ (\lambda x. h\ (f\ x))\ a\ b = h\ (EM\text{-remainder}'\ n\ f\ a\ b)$
proof –
have $integral\ \{a..b\}\ (\lambda t. pbernpoly\ n\ t\ *_R\ h\ (f\ t)) =$
 $integral\ \{a..b\}\ (\lambda t. h\ (pbernpoly\ n\ t\ *_R\ f\ t))$ **using** *assms*
by (*simp add: linear-simps*)
also have $\dots = h\ (integral\ \{a..b\}\ (\lambda t. pbernpoly\ n\ t\ *_R\ f\ t))$
by (*subst integral-linear [OF - assms(1), symmetric]*)
(auto intro!: integrable-EM-remainder' assms(2) simp: o-def)
finally show *?thesis* **using** *assms(1)*
by (*simp add: EM-remainder'-def linear-simps*)
qed

lemma *EM-remainder-converges-of-real*:

assumes *EM-remainder-converges n f a continuous-on {of-int a..} f*
shows $EM\text{-remainder-converges}\ n\ (\lambda x. of\text{-real}\ (f\ x))\ a$
proof –
from *assms* **obtain** *L*
where $L: ((\lambda b. EM\text{-remainder}'\ n\ f\ (real\text{-of-int}\ a)\ (real\text{-of-int}\ b)) \longrightarrow L)$ *at-top*
by (*auto simp: EM-remainder-converges-def*)
have $((\lambda b. EM\text{-remainder}'\ n\ (\lambda x. of\text{-real}\ (f\ x))\ (of\text{-int}\ a)\ (of\text{-int}\ b)) \longrightarrow of\text{-real}\ L)$ *at-top*
proof (*rule Lim-transform-eventually*)
show $eventually\ (\lambda b. of\text{-real}\ (EM\text{-remainder}'\ n\ f\ (of\text{-int}\ a)\ (of\text{-int}\ b)) =$
 $EM\text{-remainder}'\ n\ (\lambda x. of\text{-real}\ (f\ x))\ (of\text{-int}\ a)\ (of\text{-int}\ b))$ *at-top*
using *eventually-ge-at-top[of a]*
by *eventually-elim*
(intro EM-remainder'-bounded-linear [OF bounded-linear-of-real, symmetric]
continuous-on-subset[OF assms(2)], auto)
qed (*intro tendsto-intros L*)
thus *?thesis* **unfolding** *EM-remainder-converges-def* **..**
qed

lemma *EM-remainder-converges-of-real-iff*:

fixes $f :: real \Rightarrow real$
assumes *continuous-on {of-int a..} f*
shows $EM\text{-remainder-converges}\ n\ (\lambda x. of\text{-real}\ (f\ x)) ::$
 $'a :: \{banach, real\text{-normed-algebra-1}, real\text{-inner}\} a \longleftrightarrow EM\text{-remainder-converges}$
 $n\ f\ a$
proof
assume $EM\text{-remainder-converges}\ n\ (\lambda x. of\text{-real}\ (f\ x)) :: 'a$ *a*
then obtain $L :: 'a$

where $L: ((\lambda b. EM\text{-remainder}' n (\lambda x. of\text{-real} (f x)) (of\text{-int} a) (of\text{-int} b)) \longrightarrow L)$ *at-top*
by (*auto simp: EM-remainder-converges-def*)
have $((\lambda b. EM\text{-remainder}' n f (of\text{-int} a) (of\text{-int} b)) \longrightarrow L \cdot 1)$ *at-top*
proof (*rule Lim-transform-eventually*)
show $eventually (\lambda b. EM\text{-remainder}' n (\lambda x. of\text{-real} (f x) :: 'a) (of\text{-int} a) (of\text{-int} b)) \cdot 1 =$
 $EM\text{-remainder}' n f (of\text{-int} a) (of\text{-int} b)$ *at-top* **using** *eventually-ge-at-top[of a]*
by *eventually-elim*
(subst EM-remainder'-bounded-linear [OF bounded-linear-of-real],
auto intro!: continuous-on-subset[OF assms])
qed (*intro tendsto-intros L*)
thus *EM-remainder-converges n f a* **unfolding** *EM-remainder-converges-def ..*
qed (*intro EM-remainder-converges-of-real assms*)

lemma *EM-remainder-of-real:*

assumes *continuous-on {a..} f*
shows $EM\text{-remainder} n (\lambda x. of\text{-real} (f x) :: 'a :: \{banach, real\text{-normed\text{-algebra}\text{-}1, real\text{-inner}\}) a =$
 $of\text{-real} (EM\text{-remainder} n f a)$
proof –
have *eq:* $EM\text{-remainder}' n (\lambda x. of\text{-real} (f x) :: 'a) (real\text{-of-int} a) =$
 $(\lambda x::int. of\text{-real} (EM\text{-remainder}' n f a x))$
by (*intro ext EM-remainder'-bounded-linear[OF bounded-linear-of-real]*
continuous-on-subset[OF assms]) *auto*
show *?thesis*
proof (*cases EM-remainder-converges n f a*)
case *False*
with *EM-remainder-converges-of-real-iff[OF assms, of n]* **show** *?thesis*
by (*auto simp: EM-remainder-def*)
next
case *True*
then obtain L **where** $L: ((\lambda x. EM\text{-remainder}' n f a (real\text{-of-int} x)) \longrightarrow L)$
at-top
by (*auto simp: EM-remainder-converges-def*)
have $L': ((\lambda x. EM\text{-remainder}' n (\lambda x. of\text{-real} (f x) :: 'a) a$
 $(real\text{-of-int} x)) \longrightarrow of\text{-real} L)$ *at-top* **unfolding** *eq* **by** (*intro*
tendsto-of-real L)
from $L L'$ *tendsto-Lim[OF - L] tendsto-Lim[OF - L']* **show** *?thesis*
by (*auto simp: EM-remainder-def EM-remainder-converges-def*)
qed
qed

lemma *EM-remainder'-cong:*

assumes $\bigwedge x. x \in \{a..b\} \implies f x = g x \ n = n' \ a = a' \ b = b'$
shows $EM\text{-remainder}' n f a b = EM\text{-remainder}' n' g a' b'$
proof –
have $integral \{a..b\} (\lambda t. pbernpoly n t *_R f t) = integral \{a'..b'\} (\lambda t. pbernpoly$

$n' t *_R g t$
unfolding *assms* **using** *assms* **by** (*intro integral-cong*) *auto*
with *assms* **show** *?thesis* **by** (*simp add: EM-remainder'-def*)
qed

lemma *EM-remainder-converges-cong*:

assumes $\bigwedge x. x \geq \text{of-int } a \implies f x = g x \ n = n' \ a = a'$
shows *EM-remainder-converges* $n \ f \ a = \text{EM-remainder-converges } n' \ g \ a'$
unfolding *EM-remainder-converges-def*
by (*subst EM-remainder'-cong[OF - refl refl refl, of - - f g]*) (*use assms in auto*)

lemma *EM-remainder-cong*:

assumes $\bigwedge x. x \geq \text{of-int } a \implies f x = g x \ n = n' \ a = a'$
shows *EM-remainder* $n \ f \ a = \text{EM-remainder } n' \ g \ a'$
proof –
have *: *EM-remainder-converges* $n \ f \ a = \text{EM-remainder-converges } n' \ g \ a'$
using *assms* **by** (*intro EM-remainder-converges-cong*) *auto*
show *?thesis* **unfolding** *EM-remainder-def*
by (*subst EM-remainder'-cong[OF - refl refl refl, of - - f g]*) (*use assms * in auto*)
qed

lemma *EM-remainder-converges-cnj*:

assumes *continuous-on* $\{a..\}$ *f* **and** *EM-remainder-converges* $n \ f \ a$
shows *EM-remainder-converges* $n \ (\lambda x. \text{cnj } (f x)) \ a$
proof –
interpret *bounded-linear* *cnj* **by** (*rule bounded-linear-cnj*)
obtain *L* **where** *L*: $((\lambda x. \text{EM-remainder}' \ n \ f \ (\text{real-of-int } a) \ (\text{real-of-int } x)) \longrightarrow L)$ *at-top*
using *assms* **unfolding** *EM-remainder-converges-def* **by** *blast*
note *tendsto-cnj* [*OF this*]
also have $(\lambda x. \text{cnj } (\text{EM-remainder}' \ n \ f \ (\text{real-of-int } a) \ (\text{real-of-int } x))) =$
 $(\lambda x. \text{EM-remainder}' \ n \ (\lambda x. \text{cnj } (f x)) \ (\text{real-of-int } a) \ (\text{real-of-int } x))$
by (*subst EM-remainder'-bounded-linear* [*OF bounded-linear-cnj*])
(rule continuous-on-subset [*OF assms(1)*], *auto*)
finally have *L'*: $(\dots \longrightarrow \text{cnj } L)$ *at-top* .
thus *EM-remainder-converges* $n \ (\lambda x. \text{cnj } (f x)) \ a$
by (*auto simp: EM-remainder-converges-def*)
qed

lemma *EM-remainder-converges-cnj-iff*:

assumes *continuous-on* $\{\text{of-int } a..\}$ *f*
shows *EM-remainder-converges* $n \ (\lambda x. \text{cnj } (f x)) \ a \iff \text{EM-remainder-converges } n \ f \ a$
proof
assume *EM-remainder-converges* $n \ (\lambda x. \text{cnj } (f x)) \ a$
hence *EM-remainder-converges* $n \ (\lambda x. \text{cnj } (\text{cnj } (f x))) \ a$
by (*rule EM-remainder-converges-cnj* [*rotated*]) (*auto intro: continuous-intros assms*)

thus *EM-remainder-converges* $n f a$ **by** *simp*
qed (*intro EM-remainder-converges-cnj assms*)

lemma *EM-remainder-cnj*:

assumes *continuous-on* $\{a..\}$ f
shows *EM-remainder* $n (\lambda x. \text{cnj } (f x)) a = \text{cnj } (\text{EM-remainder } n f a)$
proof (*cases EM-remainder-converges n f a*)
case *False*
hence $\neg \text{EM-remainder-converges } n (\lambda x. \text{cnj } (f x)) a$
by (*subst EM-remainder-converges-cnj-iff [OF assms]*)
with *False show ?thesis* **by** (*simp add: EM-remainder-def*)
next
case *True*
then obtain L **where** $L: ((\lambda x. \text{EM-remainder}' n f (\text{real-of-int } a) (\text{real-of-int } x))$
 $\longrightarrow L)$ *at-top*
unfolding *EM-remainder-converges-def* **by** *blast*
note *tendsto-cnj [OF this]*
also have $(\lambda x. \text{cnj } (\text{EM-remainder}' n f (\text{real-of-int } a) (\text{real-of-int } x))) =$
 $(\lambda x. \text{EM-remainder}' n (\lambda x. \text{cnj } (f x)) (\text{real-of-int } a) (\text{real-of-int } x))$
by (*subst EM-remainder'-bounded-linear [OF bounded-linear-cnj]*)
(rule continuous-on-subset [OF assms(1)], auto)
finally have $L': (\dots \longrightarrow \text{cnj } L)$ *at-top* .
moreover from *assms* **and** L **have** *EM-remainder* $n f a = L$
by (*intro EM-remainder-eqI*)
ultimately show *EM-remainder* $n (\lambda x. \text{cnj } (f x)) a = \text{cnj } (\text{EM-remainder } n f$
 $a)$
using L' **by** (*intro EM-remainder-eqI simp-all*)
qed

lemma *EM-remainder'-combine*:

fixes $f :: \text{real} \Rightarrow 'a :: \text{banach}$
assumes [*continuous-intros*]: *continuous-on* $\{a..c\}$ f
assumes $a \leq b \leq c$
shows *EM-remainder'* $n f a b + \text{EM-remainder}' n f b c = \text{EM-remainder}' n f$
 $a c$
proof –
have *integral* $\{a..b\} (\lambda t. \text{pbernpoly } n t *_R f t) + \text{integral } \{b..c\} (\lambda t. \text{pbernpoly } n$
 $t *_R f t) =$
 $\text{integral } \{a..c\} (\lambda t. \text{pbernpoly } n t *_R f t)$
by (*intro Henstock-Kurzweil-Integration.integral-combine assms integrable-EM-remainder'*)
from this [*symmetric*] **show** *?thesis* **by** (*simp add: EM-remainder'-def alge-*
bra-simps)
qed

lemma *uniformly-convergent-EM-remainder'*:

fixes $f :: 'a \Rightarrow \text{real} \Rightarrow 'b :: \{\text{banach}, \text{real-normed-algebra}\}$
assumes *deriv*: $\bigwedge y. a \leq y \implies (G \text{ has-real-derivative } g y)$ (*at y within* $\{a..\}$)
assumes *integrable*: $\bigwedge a' b y. y \in A \implies a \leq a' \implies a' \leq b \implies$
 $(\lambda t. \text{pbernpoly } n t *_R f y t) \text{ integrable-on } \{a'..b\}$

assumes *conv*: *convergent* ($\lambda y. G$ (*real* y))
assumes *bound*: *eventually* ($\lambda x. \forall y \in A. \text{norm } (f\ y\ x) \leq g\ x$) *at-top*
shows *uniformly-convergent-on* A ($\lambda b\ s. \text{EM-remainder}'\ n$ ($f\ s$) $a\ b$)
proof –
interpret *bounded-linear* $\lambda x::'b. ((- 1) \wedge \text{Suc } n / \text{fact } n) *_R x$
by (*rule bounded-linear-scaleR-right*)

from *bounded-pbernpoly* **obtain** C **where** $C: \bigwedge x. \text{norm } (\text{pbernpoly } n\ x) \leq C$ **by**
auto
from $C[\text{of } 0]$ **have** [*simp*]: $C \geq 0$ **by** *simp*

show *?thesis* **unfolding** *EM-remainder'-def*
proof (*intro* *uniformly-convergent-on* *uniformly-convergent-improper-integral'*)
fix x **assume** $x \geq a$
thus ($\lambda x. C * G\ x$) *has-real-derivative* $C * g\ x$ (*at* x *within* $\{a..\}$)
by (*intro* *DERIV-cmult* *deriv*)
next
fix $y\ a'\ b$ **assume** $y \in A\ a \leq a'\ a' \leq b$
thus ($\lambda t. \text{pbernpoly } n\ t *_R f\ y\ t$) *integrable-on* $\{a'..b\}$
by (*rule integrable*)
next
from *conv* **obtain** L **where** ($\lambda y. G$ (*real* y)) $\longrightarrow L$
by (*auto* *simp*: *convergent-def*)
from *tendsto-mult*[*OF* *tendsto-const*[*of* C] *this*]
show *convergent* ($\lambda y. C * G$ (*real* y))
by (*auto* *simp*: *convergent-def*)
next
show $\forall_F x$ *in* *at-top*. $\forall y \in A. \text{norm } (\text{pbernpoly } n\ x *_R f\ y\ x) \leq C * g\ x$
using C **unfolding** *norm-scaleR*
by (*intro* *eventually-mono*[*OF* *bound*] *ballI* *mult-mono*) *auto*
qed
qed

lemma *uniform-limit-EM-remainder*:
fixes $f :: 'a \Rightarrow \text{real} \Rightarrow 'b :: \{\text{banach,real-normed-algebra}\}$
assumes *deriv*: $\bigwedge y. a \leq y \implies (G \text{ has-real-derivative } g\ y)$ (*at* y *within* $\{a..\}$)
assumes *integrable*: $\bigwedge a'\ b\ y. y \in A \implies a \leq a' \implies a' \leq b \implies$
 $(\lambda t. \text{pbernpoly } n\ t *_R f\ y\ t) \text{ integrable-on } \{a'..b\}$
assumes *conv*: *convergent* ($\lambda y. G$ (*real* y))
assumes *bound*: *eventually* ($\lambda x. \forall y \in A. \text{norm } (f\ y\ x) \leq g\ x$) *at-top*
shows *uniform-limit* A ($\lambda b\ s. \text{EM-remainder}'\ n$ ($f\ s$) $a\ b$)
 $(\lambda s. \text{EM-remainder } n$ ($f\ s$) a) *sequentially*
proof –
have $*$: *uniformly-convergent-on* A ($\lambda b\ s. \text{EM-remainder}'\ n$ ($f\ s$) $a\ b$)
by (*rule* *uniformly-convergent-EM-remainder'*[*OF* *assms*])
also **have** *?this* \longleftrightarrow *?thesis*
unfolding *uniformly-convergent-uniform-limit-iff*
proof (*intro* *uniform-limit-cong* *refl* *always-eventually* *allI* *ballI*)
fix s **assume** $s \in A$

with * have **: convergent $(\lambda b. EM\text{-remainder}' n (f s) a b)$
by $(rule\ uniformly\ convergent\ imp\ convergent)$
show $lim (\lambda b. EM\text{-remainder}' n (f s) a b) = EM\text{-remainder } n (f s) a$
proof $(rule\ sym, rule\ EM\text{-remainder}\text{-eqI})$
have $((\lambda x. EM\text{-remainder}' n (f s) (real\text{-of}\text{-int } a) (real\ x)) \longrightarrow$
 $lim (\lambda x. EM\text{-remainder}' n (f s) (real\text{-of}\text{-int } a) (real\ x)))\ at\text{-top}$
(is $(- \longrightarrow ?L) -)$ **using ** unfolding convergent-LIMSEQ-iff by blast**
hence $((\lambda x. EM\text{-remainder}' n (f s) (real\text{-of}\text{-int } a) (real\ (nat\ x))) \longrightarrow ?L)$
at-top
by $(rule\ filterlim\ compose) (fact\ filterlim\ nat\ sequentially)$
thus $((\lambda x. EM\text{-remainder}' n (f s) (real\text{-of}\text{-int } a) (real\text{-of}\text{-int } x)) \longrightarrow ?L)$
at-top
by $(rule\ Lim\ transform\ eventually)$
 $(auto\ intro: eventually\ mono[OF\ eventually\ ge\ at\ top[of\ 0]])$
qed
qed
finally show
qed

lemma tendsto-EM-remainder:

fixes $f :: real \Rightarrow 'b :: \{banach, real\text{-normed}\text{-algebra}\}$
assumes $deriv: \bigwedge y. a \leq y \implies (G\ \text{has}\text{-real}\text{-derivative } g\ y) (at\ y\ within\ \{a..\})$
assumes $integrable: \bigwedge a' b. a \leq a' \implies a' \leq b \implies$
 $(\lambda t. pbernpoly\ n\ t\ *_R\ f\ t)\ integrable\text{-on } \{a'..b\}$
assumes $conv: convergent (\lambda y. G (real\ y))$
assumes $bound: eventually (\lambda x. norm (f\ x) \leq g\ x)\ at\text{-top}$
shows $filterlim (\lambda b. EM\text{-remainder}' n\ f\ a\ b)$
 $(nhds (EM\text{-remainder } n\ f\ a))\ sequentially$
proof –
have $uniform\ limit\ \{\}\ (\lambda b\ s. EM\text{-remainder}' n\ f\ a\ b)$
 $(\lambda s. EM\text{-remainder } n\ f\ a)\ sequentially$
using $assms\ by (intro\ uniform\ limit\ EM\text{-remainder}[where\ G = G\ and\ g =$
 $g])\ auto$
moreover $have () \in \{\}\ by\ simp$
ultimately show ?thesis by (rule tendsto-uniform-limitI)
qed

lemma EM-remainder-0 [simp]: $EM\text{-remainder } n (\lambda x. 0) a = 0$

by $(rule\ EM\text{-remainder}\text{-eqI}) (simp\ add: EM\text{-remainder}'\text{-def})$

lemma holomorphic-EM-remainder':

assumes $deriv:$
 $\bigwedge z\ t. z \in U \implies t \in \{a..x\} \implies$
 $((\lambda z. f\ z\ t)\ \text{has}\text{-field}\text{-derivative } f'\ z\ t) (at\ z\ within\ U)$
assumes $int: \bigwedge b\ c\ z\ e. a \leq b \implies c \leq x \implies z \in U \implies$
 $(\lambda t. of\text{-real } (bernpoly\ n\ (t - e)) * f\ z\ t)\ integrable\text{-on } \{b..c\}$
assumes $cont: continuous\text{-on } (U \times \{a..x\}) (\lambda(z, t). f'\ z\ t)$
assumes $convex\ U$
shows $(\lambda s. EM\text{-remainder}' n (f\ s) a\ x)\ holomorphic\text{-on } U$

```

unfolding EM-remainder'-def scaleR-conv-of-real
proof (intro holomorphic-intros)
  have holo: ( $\lambda z. \text{integral } (\text{cbox } b \ c) (\lambda t. \text{of-real } (\text{bernpoly } n \ (t - e)) * f \ z \ t)$ )
holomorphic-on U
    if  $b \geq a \ c \leq x$  for  $b \ c \ e :: \text{real}$ 
proof (rule leibniz-rule-holomorphic)
  fix  $z \ t$  assume  $z \in U \ t \in \text{cbox } b \ c$ 
  thus ( $\lambda z. \text{complex-of-real } (\text{bernpoly } n \ (t - e)) * f \ z \ t$ ) has-field-derivative
    complex-of-real } (\text{bernpoly } n \ (t - e)) * f' \ z \ t (at z within U)
    using that by (intro DERIV-cmult deriv) auto
next
  fix  $z$  assume  $z \in U$ 
  thus ( $\lambda t. \text{complex-of-real } (\text{bernpoly } n \ (t - e)) * f \ z \ t$ ) integrable-on cbox b c
    using that int[of b c z] by auto
next
  have continuous-on ( $U \times \{b..c\}$ ) ( $\lambda(z, t). f' \ z \ t$ )
    using cont by (rule continuous-on-subset) (insert that, auto)
  thus continuous-on ( $U \times \text{cbox } b \ c$ ) ( $\lambda(z, t).$ 
    complex-of-real } (\text{bernpoly } n \ (t - e)) * f' \ z \ t)
    by (auto simp: case-prod-unfold intro!: continuous-intros)
qed fact+

consider  $a > x \mid a \leq x \text{ floor } x \leq a \mid a \leq x \text{ floor } x > a$  by force
hence ( $\lambda z. \text{integral } (\text{cbox } a \ x) (\lambda t. \text{of-real } (\text{pbernpoly } n \ t) * f \ z \ t)$ ) holomorphic-on
U
  (is ?f a x holomorphic-on -)
proof cases
  case 2
  have ( $\lambda z. \text{integral } (\text{cbox } a \ x) (\lambda t. \text{of-real } (\text{bernpoly } n \ (t - \text{of-int } \lfloor x \rfloor)) * f \ z \ t)$ )
    holomorphic-on U
    by (intro holo) auto
  also have ( $\lambda z. \text{integral } (\text{cbox } a \ x) (\lambda t. \text{of-real } (\text{bernpoly } n \ (t - \text{of-int } \lfloor x \rfloor)) * f$ 
z t)) = ?f a x
  proof (intro ext integral-cong, goal-cases)
  case (1  $z \ t$ )
  hence  $t \geq a \ t \leq x$  by auto
  hence floor t = floor x using 2 by linarith
  thus ?case by (simp add: pbernpoly-def frac-def)
qed
finally show ?thesis .
next
  case 3
  define  $N :: \text{int set}$  where  $N = \{\lceil a \rceil.. \lfloor x \rfloor\}$ 
  define  $A$  where  $A = \text{insert } \{a.. \text{of-int } \lceil a \rceil\} (\text{insert } \{\text{of-int } \lfloor x \rfloor..x\}$ 
    ( $\lambda n. \{\text{of-int } n.. \text{of-int } n + 1\}$ ) '  $N$ )
  {
  fix  $X$  assume  $X \in A$ 
  then consider  $X = \{a.. \text{of-int } \lceil a \rceil\} \mid X = \{\text{of-int } \lfloor x \rfloor..x\} \mid$ 
     $n$  where  $X = \{\text{of-int } n.. \text{of-int } n + 1\}$   $n \in N$  by (auto simp: A-def)
  }

```

```

} note A-cases = this

have division: A division-of {a..x}
proof (rule division-ofI)
  show finite A by (auto simp: A-def N-def)
  fix K assume K: K ∈ A
  from  $\exists$  have of-int  $\lceil a \rceil \leq x$ 
    using ceiling-le[of a floor x] by linarith
  moreover from  $\exists$  have of-int  $\lfloor x \rfloor \geq a$  by linarith
  ultimately show  $K \subseteq \{a..x\}$  using K  $\exists$  by (auto simp: A-def N-def)
linarith+
  from K show  $K \neq \{\}$  and  $\exists a b. K = \text{cbox } a b$  by (auto simp: A-def)
next
fix K1 K2 assume K: K1 ∈ A K2 ∈ A K1 ≠ K2
  have F1: interior {a.. $\lceil a \rceil$ } ∩ interior  $\{\lfloor x \rfloor..x\} = \{\}$  using  $\exists$  ceiling-le[of a
floor x]
  by (auto simp: min-def max-def)
  hence F2: interior  $\{\lfloor x \rfloor..x\} \cap \text{interior } \{a..\lceil a \rceil\} = \{\}$  by simp
  have F3: interior {a.. $\lceil a \rceil$ } ∩ interior {of-int n..of-int n+1} =  $\{\}$ 
    interior  $\{\lfloor x \rfloor..x\} \cap \text{interior } \{\text{of-int } n..\text{of-int } n+1\} = \{\}$ 
    interior {of-int n..of-int n+1} ∩ interior {a.. $\lceil a \rceil$ } =  $\{\}$ 
    interior {of-int n..of-int n+1} ∩ interior  $\{\lfloor x \rfloor..x\} = \{\}$  if  $n \in N$  for n
  using  $\exists$  ceiling-le[of a floor x] that by (auto simp: min-def max-def N-def)
  have F4: interior {real-of-int n..of-int n+1} ∩ interior {of-int m..of-int
m+1} =  $\{\}$ 
  if {real-of-int n..of-int n+1} ≠ {of-int m..of-int m+1} for m n
  proof -
    from that have  $n \neq m$  by auto
    thus ?thesis by simp
  qed
  from F1 F2 F3 F4 K show interior K1 ∩ interior K2 =  $\{\}$ 
  by (elim A-cases) (simp-all only: not-False-eq-True)
next
show  $\bigcup A = \{a..x\}$ 
proof (cases  $\lceil a \rceil = \lfloor x \rfloor$ )
  case True
  thus ?thesis using  $\exists$  by (auto simp: A-def N-def intro: order.trans) linarith+
next
  case False
  with  $\exists$  have *:  $\lceil a \rceil < \lfloor x \rfloor$  by linarith
  have  $\bigcup A = \{a..\text{of-int } \lceil a \rceil\} \cup (\bigcup_{n \in N. \{\text{of-int } n..\text{of-int } (n+1)\}}) \cup \{\text{of-int }
\lfloor x \rfloor..x\}$ 
  by (simp add: A-def Un-ac)
  also have  $(\bigcup_{n \in N. \{\text{of-int } n..\text{of-int } (n+1)\}}) = \{\text{of-int } \lceil a \rceil..\text{real-of-int } \lfloor x \rfloor\}$ 
  using * unfolding N-def by (intro Union-atLeastAtMost-real-of-int)
  also have  $\{a..\text{of-int } \lceil a \rceil\} \cup \dots = \{a..\text{real-of-int } \lfloor x \rfloor\}$ 
  using  $\exists$  * by (intro ivl-disj-un) auto
  also have  $\dots \cup \{\text{of-int } \lfloor x \rfloor..x\} = \{a..x\}$ 
  using  $\exists$  * by (intro ivl-disj-un) auto

```

```

    finally show ?thesis .
  qed
qed

have (λz. ∑ X∈A. integral X (λt. of-real (bernpoly n (t - [Inf X]))) * f z t))
  holomorphic-on U
proof (intro holomorphic-on-sum holo, goal-cases)
  case (1 X)
  from 1 and division have subset: X ⊆ {a..x} by (auto simp: division-of-def)
  from 1 obtain b c where [simp]: X = cbox b c b ≤ c by (auto simp: A-def)
  from subset have b ≥ a c ≤ x by auto
  hence (λx. integral (cbox b c) (λt. of-real (bernpoly n (t - [Inf {b..c}]))) * f
x t))
    holomorphic-on U by (intro holo) auto
  thus ?case by simp
qed
also have ?this ⟷ (λz. integral {a..x} (λt. of-real (pbernpoly n t) * f z t))
  holomorphic-on U
proof (intro holomorphic-cong refl, goal-cases)
  case (1 z)
  have ((λt. of-real (pbernpoly n t) * f z t) has-integral
    (∑ X∈A. integral X (λt. of-real (bernpoly n (t - [Inf X]))) * f z t)))
{a..x}
  using division
proof (rule has-integral-combine-division)
  fix X assume X: X ∈ A
  then obtain b c where X': X = {b..c} b ≤ c by (elim A-cases) auto
  from X and division have X ⊆ {a..x} by (auto simp: division-of-def)
  with X' have bc: b ≥ a c ≤ x by auto
  have ((λt. of-real (bernpoly n (t - of-int [Inf X]))) * f z t) has-integral
    integral X (λt. of-real (bernpoly n (t - of-int [Inf X]))) * f z t) X
  unfolding X' using ⟨z ∈ U⟩ bc by (intro integrable-integral int)
  also have ?this ⟷ ((λt. of-real (pbernpoly n t) * f z t) has-integral
    integral X (λt. of-real (bernpoly n (t - of-int [Inf X]))) * f z t) X
proof (rule has-integral-spike-eq[of {Sup X}], goal-cases)
  case (2 t)
  note t = this
  from ⟨X ∈ A⟩ have [t] = [Inf X]
  proof (cases rule: A-cases [consumes 1])
    case 1
    with t show ?thesis
    by (intro floor-unique) (auto simp: ceiling-altdef split: if-splits,
(linarith+)?)
  next
  case 2
  with t show ?thesis
  by (intro floor-unique) (auto simp: ceiling-altdef split: if-splits,
(linarith+)?)

```

```

next
  case 3
  with t show ?thesis
    by (intro floor-unique) (auto simp: ceiling-altdef N-def split: if-splits)
  qed
  thus ?case by (simp add: pbernpoly-def frac-def)
  qed auto
  finally show ... .
  qed
  thus ?case by (simp add: has-integral-iff)
  qed
  finally show ?thesis by simp
  qed auto
  thus ( $\lambda z. \text{integral } \{a..x\} (\lambda t. \text{of-real } (pbernpoly\ n\ t) * f\ z\ t)$ ) holomorphic-on U
    by simp
  qed

```

lemma

```

assumes deriv:  $\bigwedge y. a \leq y \implies (G \text{ has-real-derivative } g\ y) \text{ (at } y \text{ within } \{a..\})$ 
assumes deriv':
   $\bigwedge z\ t\ x. z \in U \implies x \geq a \implies t \in \{a..x\} \implies$ 
    ( $\lambda z. f\ z\ t$ ) has-field-derivative  $f'\ z\ t$  (at z within U)
assumes cont: continuous-on (U  $\times$  {of-int a..}) ( $\lambda(z, t). f'\ z\ t$ )
assumes int:  $\bigwedge b\ c\ z\ e. a \leq b \implies z \in U \implies$ 
  ( $\lambda t. \text{of-real } (bernpoly\ n\ (t - e)) * f\ z\ t$ ) integrable-on {b..c}
assumes int':  $\bigwedge a'\ b\ y. y \in U \implies a \leq a' \implies a' \leq b \implies$ 
  ( $\lambda t. pbernpoly\ n\ t *_R f\ y\ t$ ) integrable-on {a'..b}
assumes conv: convergent ( $\lambda y. G\ (\text{real } y)$ )
assumes bound: eventually ( $\lambda x. \forall y \in U. \text{norm } (f\ y\ x) \leq g\ x$ ) at-top
assumes open U
  shows analytic-EM-remainder: ( $\lambda s::\text{complex}. \text{EM-remainder } n\ (f\ s)\ a$ ) ana-
    lytic-on U
  and holomorphic-EM-remainder: ( $\lambda s::\text{complex}. \text{EM-remainder } n\ (f\ s)\ a$ ) holo-
    morphic-on U
proof -
  show ( $\lambda s::\text{complex}. \text{EM-remainder } n\ (f\ s)\ a$ ) analytic-on U
  unfolding analytic-on-def
  proof
    fix z assume z  $\in U$ 
    from  $\langle z \in U \rangle$  and  $\langle \text{open } U \rangle$  obtain  $\varepsilon$  where  $\varepsilon : \varepsilon > 0$  ball z  $\varepsilon \subseteq U$ 
    by (auto simp: open-contains-ball)
    have ( $\lambda s. \text{EM-remainder } n\ (f\ s)\ a$ ) holomorphic-on ball z  $\varepsilon$ 
    proof (rule holomorphic-uniform-sequence)
      fix x :: nat
      show ( $\lambda s. \text{EM-remainder}'\ n\ (f\ s)\ a\ x$ ) holomorphic-on ball z  $\varepsilon$ 
      proof (rule holomorphic-EM-remainder', goal-cases)
        fix s t assume s  $\in$  ball z  $\varepsilon$  t  $\in$  {real-of-int a..real x}
        thus (( $\lambda z. f\ z\ t$ ) has-field-derivative  $f'\ s\ t$ ) (at s within ball z  $\varepsilon$ )
          using  $\varepsilon$  by (intro DERIV-subset[OF deriv'[of - x]]) auto
      qed
    qed
  qed

```

```

next
  case (2 b c s e)
  with  $\varepsilon$  have  $s \in U$  by blast
  with 2 show ?case using  $\varepsilon$  int[of b s e c] by (cases  $a \leq x$ ) auto
next
  from cont show continuous-on (ball z  $\varepsilon \times \{real-of-int\ a..real\ x\}$ ) ( $\lambda(z, t).$ 
f' z t)
    by (rule continuous-on-subset) (insert  $\varepsilon$ , auto)
  qed (auto)
next
  fix s assume  $s \in ball\ z\ \varepsilon$ 
  have open (ball z  $\varepsilon$ ) by simp
  with s obtain  $\delta$  where  $\delta > 0$  cball s  $\delta \subseteq ball\ z\ \varepsilon$ 
    unfolding open-contains-cball by blast
  moreover have bound': eventually ( $\lambda x. \forall y \in cball\ s\ \delta. norm\ (f\ y\ x) \leq g\ x$ )
at-top
    by (intro eventually-mono [OF bound']) (insert  $\delta\ \varepsilon$ , auto)
  have uniform-limit (cball s  $\delta$ ) ( $\lambda x\ s. EM-remainder'\ n\ (f\ s)\ (real-of-int\ a)$ 
(real x))
    ( $\lambda s. EM-remainder\ n\ (f\ s)\ a$ ) sequentially
    by (rule uniform-limit-EM-remainder'[OF deriv int' conv bound']) (insert  $\delta$ 
 $\varepsilon\ s$ , auto)
  ultimately show  $\exists \delta > 0. cball\ s\ \delta \subseteq ball\ z\ \varepsilon \wedge uniform-limit\ (cball\ s\ \delta)$ 
( $\lambda x\ s. EM-remainder'\ n\ (f\ s)\ (real-of-int\ a)\ (real\ x)$ )
( $\lambda s. EM-remainder\ n\ (f\ s)\ a$ ) sequentially by blast
  qed auto
  with  $\varepsilon$  show  $\exists \varepsilon > 0. (\lambda s. EM-remainder\ n\ (f\ s)\ a)$  holomorphic-on ball z  $\varepsilon$ 
by blast
qed
thus ( $\lambda s::complex. EM-remainder\ n\ (f\ s)\ a$ ) holomorphic-on U
by (rule analytic-imp-holomorphic)
qed

```

The following lemma is the first step in the proof of the Euler–MacLaurin formula: We show the relationship between the first-order remainder term and the difference of the integral and the sum.

```

context
  fixes  $f\ f' :: real \Rightarrow 'a :: banach$ 
  fixes  $a\ b :: int$  and  $I\ S :: 'a$ 
  fixes  $Y :: real\ set$ 
  assumes  $a \leq b$ 
  assumes fin: finite Y
  assumes cont: continuous-on {real-of-int a..real-of-int b} f
  assumes deriv [derivative-intros]:
     $\bigwedge x::real. x \in \{a..b\} - Y \implies (f\ has\ vector\ derivative\ f'\ x)\ (at\ x)$ 
  defines S-def:  $S \equiv (\sum i \in \{a <..b\}. f\ i)$  and I-def:  $I \equiv integral\ \{a..b\}\ f$ 
begin

lemma

```


diff-sum-integral-has-integral-int:
 $((\lambda t. (\text{frac } t - 1/2) *_R f' t) \text{ has-integral } (S - I - (f b - f a) /_R 2)) \{a..b\}$
proof (cases a = b)
case True
thus ?thesis **by** (simp-all add: S-def I-def has-integral-refl)
next
case False
with $\langle a \leq b \rangle$ **have** ab: $a < b$ **by** simp
let ?A = $(\lambda n. \{ \text{real-of-int } n.. \text{real-of-int } (n+1) \}) ' \{a..<b\}$
have division: ?A division-of {of-int a..of-int b}
using Union-atLeastAtMost-real-of-int[OF ab] **by** (simp add: division-of-def)
have cont' [continuous-intros]: continuous-on A f **if** $A \subseteq \{ \text{of-int } a.. \text{of-int } b \}$ **for**
A
using continuous-on-subset[OF cont that] .

define d **where** $d = (\lambda x. (f x + f (x + 1)) /_R 2 - \text{integral } \{x..x+1\} f)$
have $((\lambda t. (\text{frac } t - 1/2) *_R f' t) \text{ has-integral } d) \{ \text{of-int } i.. \text{of-int } (i+1) \}$
if $i \in \{a..<b\}$ **for** i
proof (rule has-integral-spike)
show $(\text{frac } x - 1 / 2) *_R f' x = (x - \text{of-int } i - 1 / 2) *_R f' x$
if $x \in \{ \text{of-int } i.. \text{of-int } (i + 1) \} - \{ \text{of-int } (i + 1) \}$ **for** x
proof -
have $x \geq \text{of-int } i < \text{of-int } (i + 1)$ **using** that **by** auto
hence floor x = of-int i **by** (subst floor-unique) auto
thus ?thesis **by** (simp add: frac-def)
qed
next
define h **where** $h = (\lambda x::\text{real}. (x - \text{of-int } i - 1 / 2) *_R f' x)$
define g **where** $g = (\lambda x::\text{real}. (x - \text{of-int } i - 1/2) *_R f x - \text{integral } \{ \text{of-int } i..x \} f)$
have *: $((\lambda x. \text{integral } \{ \text{real-of-int } i..x \} f) \text{ has-vector-derivative } f x) \text{ (at } x \text{ within } \{i..i+1\})$
if $x \in \{ \text{of-int } i <.. < \text{of-int } i + 1 \}$ **for** x **using** that i
by (intro integral-has-vector-derivative cont') auto
have $((\lambda x. \text{integral } \{ \text{real-of-int } i..x \} f) \text{ has-vector-derivative } f x) \text{ (at } x)$
if $x \in \{ \text{of-int } i <.. < \text{of-int } i + 1 \}$ **for** x
using that i at-within-interior[of x {of-int i..of-int (i + 1)}] *[of x] **by** simp
hence (h has-integral g (of-int (i + 1)) - g (of-int i)) {of-int i..of-int (i+1)}
unfolding g-def h-def **using** that
by (intro fundamental-theorem-of-calculus-interior-strong[OF fin])
(auto intro!: derivative-eq-intros continuous-intros indefinite-integral-continuous-1
integrable-continuous-real)
also **have** $g (\text{of-int } (i + 1)) - g (\text{of-int } i) = d i$
by (simp add: g-def scaleR-add-right [symmetric] d-def)
finally **show** (h has-integral d i) {of-int i..of-int (i + 1)} .
qed simp-all
hence *: $\bigwedge I. I \in ?A \implies ((\lambda x. (\text{frac } x - 1 / 2) *_R f' x) \text{ has-integral } d (\lfloor \text{Inf } I \rfloor))$
I
by (auto simp: add-ac)

have $((\lambda x :: \text{real. } (\text{frac } x - 1 / 2) *_{\mathbb{R}} f' x) \text{ has-integral } (\sum I \in ?A. d (\lfloor \text{Inf } I \rfloor)))$
 $(\bigcup ?A)$
by $(\text{intro has-integral-Union } * \text{ finite-image } I) (\text{force intro! : negligible-atLeastAtMostI pairwise } I) +$
also have $\bigcup ?A = \{\text{of-int } a.. \text{of-int } b\}$
by $(\text{intro Union-atLeastAtMost-real-of-int } ab)$
also have $(\sum I \in ?A. d (\lfloor \text{Inf } I \rfloor)) = (\sum i = a.. < b. d i)$
by $(\text{subst sum.reindex}) (\text{auto simp: inj-on-def})$
also have $\dots = (1 / 2) *_{\mathbb{R}} ((\sum i = a.. < b. f (\text{real-of-int } i)) +$
 $(\sum i = a.. < b. f (\text{real-of-int } (i + 1)))) -$
 $(\sum i = a.. < b. \text{integral } \{\text{real-of-int } i..1 + \text{real-of-int } i\} f)$
 $(\text{is } - = - *_{\mathbb{R}} (?S1 + ?S2) - ?S3)$
by $(\text{simp add: d-def algebra-simps sum.distrib sum-subtractf scaleR-sum-right})$
also have $?S1 = (\sum i = a.. b. f (\text{real-of-int } i)) - f b$
unfolding $S\text{-def using } ab \text{ by } (\text{subst sum-atLeastAtMost-int-last}) \text{ auto}$
also have $(\sum i = a.. b. f (\text{real-of-int } i)) = S + f a$
unfolding $S\text{-def using } ab \text{ by } (\text{subst sum-atLeastAtMost-int-head}) \text{ auto}$
also have $?S2 = S \text{ unfolding } S\text{-def}$
by $(\text{intro sum.reindex-bij-witness}[\text{of } - \lambda i. i-1 \lambda i. i+1]) \text{ auto}$
also have $(1 / 2) *_{\mathbb{R}} (S + f a - f b + S) =$
 $(1/2) *_{\mathbb{R}} S + (1/2) *_{\mathbb{R}} S - (f b - f a) /_{\mathbb{R}} 2$
by $(\text{simp add: algebra-simps})$
also have $(1/2) *_{\mathbb{R}} S + (1/2) *_{\mathbb{R}} S = S \text{ by } (\text{simp add: scaleR-add-right}$
 $[\text{symmetric}])$

also have $?S3 = (\sum I \in ?A. \text{integral } I f)$
by $(\text{subst sum.reindex}) (\text{auto simp: inj-on-def add-ac})$
also have $\dots = I \text{ unfolding } I\text{-def}$
by $(\text{intro integral-combine-division-topdown } [\text{symmetric}] \text{ division integrable-continuous-real}$
 $\text{continuous-intros}) \text{ simp-all}$
finally show $?thesis \text{ by } (\text{simp add: algebra-simps})$
qed

lemma *diff-sum-integral-has-integral-int'*:

$((\lambda t. \text{pbernpoly } 1 t *_{\mathbb{R}} f' t) \text{ has-integral } (S - I - (f b - f a) /_{\mathbb{R}} 2)) \{a..b\}$
using *diff-sum-integral-has-integral-int* **by** $(\text{simp add: pbernpoly-def bernpoly-def})$

lemma *EM-remainder'-Suc-0*: $EM\text{-remainder}' (Suc 0) f' a b = S - I - (f b - f a) /_{\mathbb{R}} 2$

using *diff-sum-integral-has-integral-int'* **by** $(\text{simp add: has-integral-iff } EM\text{-remainder}'\text{-def})$

end

Next, we show that the n -th-order remainder can be expressed in terms of the $n + 1$ -th-order remainder term. Iterating this essentially yields the Euler–MacLaurin formula.

context

fixes $f f' :: \text{real} \Rightarrow 'a :: \text{banach}$ **and** $a b :: \text{int}$ **and** $n :: \text{nat}$ **and** $A :: \text{real set}$

assumes ab : $a \leq b$ **and** n : $n > 0$
assumes fin : $finite\ A$
assumes $cont$: $continuous-on\ \{of-int\ a..of-int\ b\}\ f$
assumes $cont'$: $continuous-on\ \{of-int\ a..of-int\ b\}\ f'$
assumes $deriv$: $\bigwedge x. x \in \{of-int\ a<..
begin$

lemma $EM-remainder'-integral-conv-Suc$:

shows $integral\ \{a..b\}\ (\lambda t. pbernpoly\ n\ t\ *_R\ f\ t) =$
 $(bernoulli\ (Suc\ n)\ / \ real\ (Suc\ n))\ *_R\ (f\ b - f\ a) -$
 $integral\ \{a..b\}\ (\lambda t. pbernpoly\ (Suc\ n)\ t\ *_R\ f'\ t) /_R\ real\ (Suc\ n)$

unfolding $EM-remainder'-def$

proof –

let $?h = \lambda i. (pbernpoly\ (Suc\ n)\ (real-of-int\ i)\ / \ real\ (Suc\ n))\ *_R\ f\ (real-of-int\ i)$

define T **where** $T = integral\ \{a..b\}\ (\lambda t. (pbernpoly\ (Suc\ n)\ t\ / \ real\ (Suc\ n))\ *_R\ f'\ t)$

note $[derivative-intros] = has-field-derivative-pbernpoly-Suc'$

let $?A = real-of-int\ '\{a..b\} \cup A$

have $((\lambda t. pbernpoly\ n\ t\ *_R\ f\ t)\ has-integral\ (-T + (?h\ b - ?h\ a)))\ \{a..b\}$

proof $(rule\ integration-by-parts-interior-strong[OF\ bounded-bilinear-scaleR])$

from fin **show** $finite\ ?A$ **by** $simp$

from $\langle n > 0 \rangle$ **show** $continuous-on\ \{of-int\ a..of-int\ b\}\ (\lambda t. pbernpoly\ (Suc\ n)\ t\ / \ real\ (Suc\ n))$

by $(intro\ continuous-intros)\ auto$

show $continuous-on\ \{of-int\ a..of-int\ b\}\ f$ **by** $fact$

show $(f\ has-vector-derivative\ f'\ t)\ (at\ t)$ **if** $t \in \{of-int\ a<.. **for** $t$$

using $deriv[of\ t]$ **that** **by** $auto$

have $(\lambda t. pbernpoly\ (Suc\ n)\ t\ *_R\ f'\ t)\ integrable-on\ \{a..b\}$

by $(intro\ integrable-EM-remainder'\ cont')$

hence $(\lambda t. (1 / \ real\ (Suc\ n))\ *_R\ pbernpoly\ (Suc\ n)\ t\ *_R\ f'\ t)\ integrable-on\ \{a..b\}$

by $(rule\ integrable-cmul)$

also **have** $(\lambda t. (1 / \ real\ (Suc\ n))\ *_R\ pbernpoly\ (Suc\ n)\ t\ *_R\ f'\ t) =$

$(\lambda t. (pbernpoly\ (Suc\ n)\ t\ / \ real\ (Suc\ n))\ *_R\ f'\ t)$

by $(rule\ ext)\ (simp\ add:\ algebra-simps)$

finally **show** $((\lambda t. (pbernpoly\ (Suc\ n)\ t\ / \ real\ (Suc\ n))\ *_R\ f'\ t)$

$has-integral\ ?h\ b - ?h\ a - (-T + (?h\ b - ?h\ a)))\ \{a..b\}$

using $integrable-EM-remainder'[of\ a\ b\ f'\ Suc\ n]$

by $(simp\ add:\ has-integral-iff\ T-def)$

qed $(insert\ ab\ n,\ auto\ intro!\ derivative-eq-intros$

$simp:\ has-real-derivative-iff-has-vector-derivative\ [symmetric]\ not-le\ elim!:$

$Ints-cases)$

also **have** $?h\ b - ?h\ a = (bernoulli\ (Suc\ n)\ / \ real\ (Suc\ n))\ *_R\ (f\ b - f\ a)$

using n **by** $(simp\ add:\ algebra-simps\ bernoulli'-def)$

finally **have** $integral\ \{a..b\}\ (\lambda t. pbernpoly\ n\ t\ *_R\ f\ t) = \dots - T$

by $(simp\ add:\ has-integral-iff)$

also have $T = \text{integral } \{a..b\} (\lambda t. (1 / \text{real } (\text{Suc } n)) *_{\mathbb{R}} (\text{pbernpoly } (\text{Suc } n) t) *_{\mathbb{R}} f' t)$
by (*simp add: T-def*)
also have $\dots = \text{integral } \{a..b\} (\lambda t. \text{pbernpoly } (\text{Suc } n) t *_{\mathbb{R}} f' t) /_{\mathbb{R}} \text{real } (\text{Suc } n)$
by (*subst integral-cmul*) (*simp-all add: divide-simps*)
finally show *?thesis* .
qed

lemma *EM-remainder'-conv-Suc:*

$\text{EM-remainder}' n f a b =$
 $((-1) \wedge \text{Suc } n * \text{bernoulli } (\text{Suc } n) / \text{fact } (\text{Suc } n)) *_{\mathbb{R}} (f b - f a) +$
 $\text{EM-remainder}' (\text{Suc } n) f' a b$
by (*simp add: EM-remainder'-def EM-remainder'-integral-conv-Suc scaleR-diff-right*
scaleR-add-right field-simps del: of-nat-Suc)

end

context

fixes $f f' :: \text{real} \Rightarrow 'a :: \text{banach}$ **and** $a :: \text{int}$ **and** $n :: \text{nat}$ **and** $A :: \text{real set}$ **and** C

assumes $n: n > 0$

assumes *fin: finite A*

assumes *cont: continuous-on {of-int a..} f*

assumes *cont': continuous-on {of-int a..} f'*

assumes *lim: (f \longrightarrow C) at-top*

assumes *deriv: $\bigwedge x. x \in \{\text{of-int } a <..\} - A \implies (f \text{ has-vector-derivative } f' x) \text{ (at } x)$*

begin

lemma

shows *EM-remainder-converges-iff-Suc-converges:*

$\text{EM-remainder-converges } n f a \longleftrightarrow \text{EM-remainder-converges } (\text{Suc } n) f' a$

and *EM-remainder-conv-Suc:*

$\text{EM-remainder-converges } n f a \implies$

$\text{EM-remainder } n f a =$

$((-1) \wedge \text{Suc } n * \text{bernoulli } (\text{Suc } n) / \text{fact } (\text{Suc } n)) *_{\mathbb{R}} (C - f a) +$

$\text{EM-remainder } (\text{Suc } n) f' a$

proof (*rule iffI*)

define g **where** $g = (\lambda x. ((-1) \wedge \text{Suc } n * \text{bernoulli } (\text{Suc } n) / \text{fact } (\text{Suc } n)) *_{\mathbb{R}} (f x - f a))$

define G **where** $G = ((-1) \wedge \text{Suc } n * \text{bernoulli } (\text{Suc } n) / \text{fact } (\text{Suc } n)) *_{\mathbb{R}} (C - f a)$

have *limit-g: (g \longrightarrow G) at-top unfolding g-def G-def by (intro tendsto-intros lim)*

have $*$: *eventually* $(\lambda x. \text{EM-remainder}' n f (\text{real-of-int } a) (\text{real-of-int } x) =$

$g x + \text{EM-remainder}' (\text{Suc } n) f' (\text{real-of-int } a) (\text{real-of-int } x)) \text{ at-top}$

using *eventually-ge-at-top[of a]*

```

proof eventually-elim
  case (elim b)
  thus ?case
  using EM-remainder'-conv-Suc[OF elim n fin continuous-on-subset[OF cont]
    continuous-on-subset[OF cont'] deriv] by (auto simp: g-def)
qed

{
  assume EM-remainder-converges n f a
  then obtain L
    where L: (( $\lambda b$ . EM-remainder' n f (real-of-int a) (real-of-int b))  $\longrightarrow$  L)
  at-top
    by (auto simp: EM-remainder-converges-def)
  have *: (( $\lambda b$ . EM-remainder' (Suc n) f' (real-of-int a) (real-of-int b))  $\longrightarrow$  L
  - G) at-top
  proof (rule Lim-transform-eventually)
    show  $\forall_F x$  in at-top. EM-remainder' n f (real-of-int a) (real-of-int x) - g x
  =
    EM-remainder' (Suc n) f' (real-of-int a) (real-of-int x)
    using * by (simp add: algebra-simps)
    show (( $\lambda x$ . EM-remainder' n f (real-of-int a) (real-of-int x) - g x)  $\longrightarrow$  L
  - G) at-top
    by (intro tendsto-intros filterlim-compose[OF limit-g] L)
  qed
  from * show EM-remainder-converges (Suc n) f' a unfolding EM-remainder-converges-def
  ..
  from * have EM-remainder (Suc n) f' a = L - G by (rule EM-remainder-eqI)
  moreover from L have EM-remainder n f a = L by (rule EM-remainder-eqI)
  ultimately show EM-remainder n f a = G + EM-remainder (Suc n) f' a by
  (simp add: G-def)
}
{
  assume EM-remainder-converges (Suc n) f' a
  then obtain L
    where L: (( $\lambda b$ . EM-remainder' (Suc n) f' (real-of-int a) (real-of-int b))  $\longrightarrow$ 
  L) at-top
    by (auto simp: EM-remainder-converges-def)
  have *: (( $\lambda b$ . EM-remainder' n f (real-of-int a) (real-of-int b))  $\longrightarrow$  G + L)
  at-top
  proof (rule Lim-transform-eventually)
    show  $\forall_F x$  in at-top. g x + EM-remainder' (Suc n) f' (real-of-int a) (real-of-int
  x) =
    EM-remainder' n f (real-of-int a) (real-of-int x)
    using * by (subst eq-commute)
    show (( $\lambda x$ . g x + EM-remainder' (Suc n) f' (real-of-int a) (real-of-int x))
   $\longrightarrow$  G + L) at-top
    by (intro tendsto-intros filterlim-compose[OF limit-g] L)
  qed
  thus EM-remainder-converges n f a unfolding EM-remainder-converges-def ..
}

```

```

}
qed

```

```

end

```

1.3 The conventional version of the Euler–MacLaurin formula

The following theorems are the classic Euler–MacLaurin formula that can be found, with slight variations, in many sources (e. g. [1, 2, 3]).

```

context

```

```

  fixes f :: real ⇒ 'a :: banach

```

```

  fixes fs :: nat ⇒ real ⇒ 'a

```

```

  fixes a b :: int assumes ab: a ≤ b

```

```

  fixes N :: nat assumes N: N > 0

```

```

  fixes Y :: real set assumes fin: finite Y

```

```

  assumes fs-0 [simp]: fs 0 = f

```

```

  assumes fs-cont [continuous-intros]:

```

```

    ∧k. k ≤ N ⇒ continuous-on {real-of-int a..real-of-int b} (fs k)

```

```

  assumes fs-deriv [derivative-intros]:

```

```

    ∧k x. k < N ⇒ x ∈ {a..b} − Y ⇒ (fs k has-vector-derivative fs (Suc k) x)
(at x)

```

```

begin

```

```

theorem euler-maclaurin-raw-strong-int:

```

```

  defines S ≡ (∑ i∈{a<..b}. f (of-int i))

```

```

  defines I ≡ integral {of-int a..of-int b} f

```

```

  defines c' ≡ λk. (bernoulli' (Suc k) / fact (Suc k)) *R (fs k b − fs k a)

```

```

  shows S − I = (∑ k<N. c' k) + EM-remainder' N (fs N) a b

```

```

proof −

```

```

  define c :: nat ⇒ 'a

```

```

  where c = (λk. ((−1) ^ (Suc k) * bernoulli (Suc k) / fact (Suc k)) *R (fs k b
− fs k a))

```

```

  have S − I = (∑ k<m. c k) + EM-remainder' m (fs m) a b if m ≥ 1 m ≤ N
for m

```

```

  using that

```

```

  proof (induction m rule: dec-induct)

```

```

    case base

```

```

    with ab fin fs-cont[of 0] show ?case using fs-deriv[of 0] N unfolding One-nat-def
    by (subst EM-remainder'-Suc-0[of - - Y f]) (simp-all add: algebra-simps S-def
I-def c-def)

```

```

  next

```

```

    case (step n)

```

```

    from step.premis have S − I = (∑ k<n. c k) + EM-remainder' n (fs n) a b

```

```

    by (intro step.IH) simp-all

```

```

    also have (∑ k<n. c k) = (∑ k<Suc n. c k) +

```

```

      (((−1) ^ n * bernoulli (Suc n) / fact (Suc n)) *R (fs n b − fs n a))

```

```

    (is - - + ?c) by (simp add: EM-remainder'-Suc-0 c-def)

```

also have $\dots + EM\text{-remainder}' n (fs\ n) a\ b = (\sum k < Suc\ n. c\ k) + (?c + EM\text{-remainder}' n (fs\ n) a\ b)$
by (*simp add: add.assoc*)
also from *step.premis step.hyps ab fin*
have $?c + EM\text{-remainder}' n (fs\ n) a\ b = EM\text{-remainder}' (Suc\ n) (fs\ (Suc\ n)) a\ b$
by (*subst EM-remainder'-conv-Suc [where A = Y]*)
(auto intro!: fs-deriv fs-cont)
finally show *?case .*
qed
from *this[of N] and N*
have $S - I = sum\ c\ \{..<N\} + EM\text{-remainder}' N (fs\ N) (real\text{-of-int}\ a)$
(real-of-int b) by simp
also have $sum\ c\ \{..<N\} = sum\ c'\ \{..<N\}$
proof (*intro sum.cong refl*)
fix $k :: nat$
show $c\ k = c'\ k$
by (*cases even k*)
(auto simp: c-def c'-def bernoulli'-def algebra-simps bernoulli-odd-eq-0)
qed
finally show *?thesis .*
qed
end

theorem *euler-maclaurin-strong-raw-nat:*

assumes $a \leq b\ 0 < N\ finite\ Y\ fs\ 0 = f$
 $(\bigwedge k. k \leq N \implies continuous\text{-on}\ \{real\ a..real\ b\}\ (fs\ k))$
 $(\bigwedge k\ x. k < N \implies x \in \{real\ a..real\ b\} - Y \implies$
 $(fs\ k\ has\ vector\ derivative\ fs\ (Suc\ k)\ x)\ (at\ x))$
shows $(\sum i \in \{a <..b\}. f\ (real\ i)) - integral\ \{real\ a..real\ b\}\ f =$
 $(\sum k < N. (bernoulli'\ (Suc\ k) / fact\ (Suc\ k)) *_R\ (fs\ k\ (real\ b) - fs\ k\ (real\ a))) +$
 $EM\text{-remainder}' N (fs\ N) (real\ a) (real\ b)$
proof –
have $(\sum i \in \{int\ a <..int\ b\}. f\ (real\text{-of-int}\ i)) -$
 $integral\ \{real\text{-of-int}\ (int\ a)..real\text{-of-int}\ (int\ b)\}\ f =$
 $(\sum k < N. (bernoulli'\ (Suc\ k) / fact\ (Suc\ k)) *_R$
 $(fs\ k\ (real\text{-of-int}\ (int\ b)) - fs\ k\ (real\text{-of-int}\ (int\ a)))) +$
 $EM\text{-remainder}' N (fs\ N) (real\text{-of-int}\ (int\ a)) (real\text{-of-int}\ (int\ b))$
using *assms* **by** (*intro euler-maclaurin-raw-strong-int [where Y = Y] assms*)
simp-all
also have $(\sum i \in \{int\ a <..int\ b\}. f\ (real\text{-of-int}\ i)) = (\sum i \in \{a <..b\}. f\ (real\ i))$
by (*intro sum.reindex-bij-witness [of - int nat] auto*)
finally show *?thesis* **by** *simp*
qed

1.4 The “Concrete Mathematics” version of the Euler–MacLaurin formula

As explained in *Concrete Mathematics* [3], the above form of the formula has some drawbacks: When applying it to determine the asymptotics of some concrete function, one is usually left with several different unwieldy constant terms that are difficult to get rid of.

There is no general way to determine what these constant terms are, but in concrete applications, they can often be determined or estimated by other means. We can therefore simply group all the constant terms into a single constant and have the user provide a proof of what it is.

```

locale euler-maclaurin-int =
  fixes F f :: real ⇒ 'a :: banach
  fixes fs :: nat ⇒ real ⇒ 'a
  fixes a :: int
  fixes N :: nat assumes N: N > 0
  fixes C :: 'a
  fixes Y :: real set assumes fin: finite Y
  assumes fs-0 [simp]: fs 0 = f
  assumes fs-cont [continuous-intros]:
    ∧k. k ≤ N ⇒ continuous-on {real-of-int a..} (fs k)
  assumes fs-deriv [derivative-intros]:
    ∧k x. k < N ⇒ x ∈ {of-int a..} − Y ⇒ (fs k has-vector-derivative fs (Suc
k) x) (at x)
  assumes F-cont [continuous-intros]: continuous-on {of-int a..} F
  assumes F-deriv [derivative-intros]:
    ∧x. x ∈ {of-int a..} − Y ⇒ (F has-vector-derivative f x) (at x)
  assumes limit:
    ((λb. (∑ k=a..b. f k) − F (of-int b) −
      (∑ i<N. (bernoulli' (Suc i) / fact (Suc i)) *R fs i (of-int b))) → C)
  at-top
begin

context
  fixes C' T
  defines C' ≡ −f a + F a + C + (∑ k<N. (bernoulli' (Suc k) / fact (Suc k))
*_R (fs k (of-int a)))
  and T ≡ (λx. ∑ i<N. (bernoulli' (Suc i) / fact (Suc i)) *_R fs i x)
begin

lemma euler-maclaurin-strong-int-aux:
  assumes ab: a ≤ b
  defines S ≡ (∑ k=a..b. f (of-int k))
  shows S − F (of-int b) − T (of-int b) = EM-remainder' N (fs N) (of-int a)
(of-int b) + (C − C')
proof (cases a = b)
  case True
  thus ?thesis unfolding C'-def by (simp add: S-def EM-remainder'-def T-def)

```


next
case *False*
with *assms* **have** *ab: a < b* **by** *simp*
define *T'* **where** $T' = (\sum k < N. (\text{bernoulli}' (Suc\ k) / \text{fact} (Suc\ k)) *_{\mathbb{R}} (fs\ k$
(of-int a)))
have $(\sum i \in \{a <..b\}. f\ (of-int\ i)) - \text{integral}\ \{of-int\ a..of-int\ b\}\ f =$
 $(\sum k < N. (\text{bernoulli}' (Suc\ k) / \text{fact} (Suc\ k)) *_{\mathbb{R}} (fs\ k\ (of-int\ b) - fs\ k$
(of-int a))) +
 $EM\text{-remainder}'\ N\ (fs\ N)\ (of-int\ a)\ (of-int\ b)$ **using** *ab*
by (*intro euler-maclaurin-raw-strong-int [where Y = Y] N fin fs-0*
continuous-on-subset[OF fs-cont] fs-deriv) *auto*
also **have** (*f has-integral (F b - F a) {of-int a..of-int b}*) **using** *ab*
by (*intro fundamental-theorem-of-calculus-strong[OF fin]*)
(auto intro!: continuous-on-subset[OF F-cont] derivative-intros)
hence $\text{integral}\ \{of-int\ a..of-int\ b\}\ f = F\ (of-int\ b) - F\ (of-int\ a)$
by (*simp add: has-integral-iff*)
also **have** $(\sum k < N. (\text{bernoulli}' (Suc\ k) / \text{fact} (Suc\ k)) *_{\mathbb{R}} (fs\ k\ (of-int\ b) - fs\ k$
(of-int a))) =
 $T\ (of-int\ b) - T'$
by (*simp add: T-def T'-def algebra-simps sum-subtractf*)
also **have** $(\sum i \in \{a <..b\}. f\ (of-int\ i)) = S - f\ (of-int\ a)$
unfolding *S-def* **using** *ab* **by** (*subst sum-atLeastAtMost-int-head*) *auto*
finally **show** *?thesis* **by** (*simp add: algebra-simps C'-def T'-def*)
qed

lemma *EM-remainder-limit:*

assumes *ab: a ≤ b*
defines $D \equiv EM\text{-remainder}'\ N\ (fs\ N)\ (of-int\ a)\ (of-int\ b)$
shows $EM\text{-remainder}\ N\ (fs\ N)\ b = C' - D$
and *EM-remainder-converges: EM-remainder-converges N (fs N) b*
proof –
note *limit*
also **have** $((\lambda b. (\sum k = a..b. f\ (of-int\ k)) - F\ (of-int\ b) -$
 $(\sum i < N. (\text{bernoulli}' (Suc\ i) / \text{fact} (Suc\ i)) *_{\mathbb{R}} fs\ i\ (of-int\ b))) \longrightarrow$
C) at-top =
 $((\lambda b. (\sum k = a..b. f\ (of-int\ k)) - F\ (of-int\ b) - T\ (of-int\ b)) \longrightarrow C)$
at-top
unfolding *T-def ..*
also **have** $\text{eventually}\ (\lambda x. (\sum k = a..x. f\ k) - F\ (of-int\ x) - T\ (of-int\ x) =$
 $EM\text{-remainder}'\ N\ (fs\ N)\ (of-int\ a)\ (of-int\ x) + (C - C'))\ \text{at-top}$
(is eventually (λx. ?f x = ?g x) -)
using *eventually-gt-at-top[of b]*
by *eventually-elim (rule euler-maclaurin-strong-int-aux, insert ab, simp-all)*
hence $(?f \longrightarrow C)\ \text{at-top} \longleftrightarrow (?g \longrightarrow C)\ \text{at-top}$ **by** (*intro filterlim-cong refl*)
finally **have** $((\lambda x. ?g\ x - (C - C')) \longrightarrow (C - (C - C')))\ \text{at-top}$
by (*rule tendsto-diff[OF - tendsto-const]*)
hence $*$: $((\lambda x. EM\text{-remainder}'\ N\ (fs\ N)\ (of-int\ a)\ (of-int\ x)) \longrightarrow C')\ \text{at-top}$
by *simp*

```

have (( $\lambda x. EM\text{-remainder}' N (fs N) (of\text{-int } a) (of\text{-int } x) - D$ )  $\longrightarrow C' - D$ )
at-top
  by (intro tendsto-intros *)
also have eventually ( $\lambda x. EM\text{-remainder}' N (fs N) (of\text{-int } a) (of\text{-int } x) - D =$ 
   $EM\text{-remainder}' N (fs N) (of\text{-int } b) (of\text{-int } x)$ ) at-top
  (is eventually ( $\lambda x. ?f x = ?g x$ ) -) using eventually-ge-at-top[of b]
proof eventually-elim
  case (elim x)
  have  $EM\text{-remainder}' N (fs N) (of\text{-int } a) (of\text{-int } x) =$ 
   $D + EM\text{-remainder}' N (fs N) (of\text{-int } b) (of\text{-int } x)$ 
  using elim ab unfolding D-def
  by (intro EM-remainder'-combine [symmetric] continuous-on-subset[OF fs-cont])
auto
  thus ?case by simp
qed
  hence ( $?f \longrightarrow C' - D$ ) at-top  $\longleftrightarrow$  ( $?g \longrightarrow C' - D$ ) at-top by (intro
filterlim-cong refl)
  finally have *: ... .
from * show EM-remainder-converges N (fs N) b unfolding EM-remainder-converges-def
..
from * show EM-remainder N (fs N) b = C' - D
  by (rule EM-remainder-eqI)
qed

```

```

theorem euler-maclaurin-strong-int:
  assumes ab:  $a \leq b$ 
  defines  $S \equiv (\sum_{k=a..b}. f (of\text{-int } k))$ 
  shows  $S = F (of\text{-int } b) + C + T (of\text{-int } b) - EM\text{-remainder } N (fs N) b$ 
proof -
  have  $S = F (of\text{-int } b) + T (of\text{-int } b) + - (C' - EM\text{-remainder}' N (fs N) (of\text{-int } a) (of\text{-int } b)) + C$ 
  using euler-maclaurin-strong-int-aux[OF ab] by (simp add: algebra-simps S-def)
  also have  $C' - EM\text{-remainder}' N (fs N) (of\text{-int } a) (of\text{-int } b) = EM\text{-remainder } N (fs N) b$ 
  using ab by (rule EM-remainder-limit(1) [symmetric])
  finally show ?thesis by simp
qed

```

end
end

The following version of the formula removes all the terms where the associated Bernoulli numbers vanish.

```

locale euler-maclaurin-int' =
  fixes  $F f :: real \Rightarrow 'a :: banach$ 
  fixes  $fs :: nat \Rightarrow real \Rightarrow 'a$ 
  fixes  $a :: int$ 
  fixes  $N :: nat$ 
  fixes  $C :: 'a$ 

```

fixes Y :: *real set* **assumes** *fin*: *finite* Y
assumes $fs-0$ [*simp*]: $fs\ 0 = f$
assumes $fs-cont$ [*continuous-intros*]:
 $\bigwedge k. k \leq 2*N+1 \implies \text{continuous-on } \{\text{real-of-int } a..\} (fs\ k)$
assumes $fs-deriv$ [*derivative-intros*]:
 $\bigwedge k\ x. k \leq 2*N \implies x \in \{\text{of-int } a..\} - Y \implies (fs\ k \text{ has-vector-derivative } fs\ (Suc\ k)\ x)\ (at\ x)$
assumes $F-cont$ [*continuous-intros*]: *continuous-on* $\{\text{of-int } a..\} F$
assumes $F-deriv$ [*derivative-intros*]:
 $\bigwedge x. x \in \{\text{of-int } a..\} - Y \implies (F \text{ has-vector-derivative } f\ x)\ (at\ x)$
assumes *limit*:
 $((\lambda b. (\sum k=a..b. f\ k) - F\ (\text{of-int } b) -$
 $(\sum i < 2*N+1. (\text{bernoulli}'\ (Suc\ i) / \text{fact}\ (Suc\ i)) *_{\mathbb{R}} fs\ i\ (\text{of-int } b)))) \longrightarrow$
 $C)$ *at-top*
begin

sublocale *euler-maclaurin-int* $F\ f\ fs\ a\ 2*N+1\ C\ Y$
by *standard* (*insert fin fs-0 fs-cont fs-deriv F-cont F-deriv limit, simp-all*)

theorem *euler-maclaurin-strong-int'*:
assumes $a \leq b$
shows $(\sum k=a..b. f\ (\text{of-int } k)) =$
 $F\ (\text{of-int } b) + C + (1 / 2) *_{\mathbb{R}} f\ (\text{of-int } b) +$
 $(\sum i=1..N. (\text{bernoulli}'\ (2*i) / \text{fact}\ (2*i)) *_{\mathbb{R}} fs\ (2*i-1)\ (\text{of-int } b)) -$
 $EM\text{-remainder}\ (2*N+1)\ (fs\ (2*N+1))\ b$

proof –
have $(\sum k=a..b. f\ (\text{real-of-int } k)) =$
 $F\ (\text{of-int } b) + C + (\sum i < 2*N+1. (\text{bernoulli}'\ (Suc\ i) / \text{fact}\ (Suc\ i))$
 $*_{\mathbb{R}} fs\ i\ (\text{of-int } b)) -$
 $EM\text{-remainder}\ (2*N+1)\ (fs\ (2*N+1))\ b$
by (*rule euler-maclaurin-strong-int*)
(simp-all only: lessThan-Suc-atMost Suc-eq-plus1 [symmetric] assms)
also have $\{.. < 2*N+1\} = \text{insert } 0\ \{1..2*N\}$ **by** *auto*
also have $(\sum i \in \dots. (\text{bernoulli}'\ (Suc\ i) / \text{fact}\ (Suc\ i)) *_{\mathbb{R}} fs\ i\ (\text{of-int } b)) =$
 $(1 / 2) *_{\mathbb{R}} f\ (\text{of-int } b) +$
 $(\sum i \in \{1..2*N\}. (\text{bernoulli}'\ (Suc\ i) / \text{fact}\ (Suc\ i)) *_{\mathbb{R}} fs\ i\ (\text{of-int } b))$
by (*subst sum.insert*) (*auto simp: assms bernoulli'-def*)
also have $(\sum i \in \{1..2*N\}. (\text{bernoulli}'\ (Suc\ i) / \text{fact}\ (Suc\ i)) *_{\mathbb{R}} fs\ i\ (\text{of-int } b))$
 $=$
 $(\sum i \in \{1..N\}. (\text{bernoulli}'\ (2*i) / \text{fact}\ (2*i)) *_{\mathbb{R}} fs\ (2*i-1)\ (\text{of-int } b))$
proof (*rule sym, rule sum.reindex-bij-witness-not-neutral*)
fix i **assume** $i \in \{1..2*N\} - \{i \in \{1..2*N\}. \text{even } i\}$
thus $2 * ((i + 1) \text{div } 2) - 1 = i (i + 1) \text{div } 2 \in \{1..N\} - \{\}$
by (*auto elim!: oddE*)
qed (*auto simp: bernoulli-odd-eq-0 bernoulli'-def algebra-simps*)
also have $\dots = (\sum i \in \{1..N\}. (\text{bernoulli}'\ (2*i) / \text{fact}\ (2*i)) *_{\mathbb{R}} fs\ (2*i-1)$
 $(\text{of-int } b))$
by (*intro sum.cong refl*) (*auto simp: bernoulli'-def*)
finally show *?thesis* **by** (*simp only: add-ac*)

qed

end

For convenience, we also offer a version where the sum ranges over natural numbers instead of integers.

lemma *sum-atLeastAtMost-of-int-nat-transfer*:

$(\sum_{k=int\ a..int\ b} f\ (of-int\ k)) = (\sum_{k=a..b} f\ (of-nat\ k))$
by (*intro sum.reindex-bij-witness*[*of - int nat*]) *auto*

lemma *euler-maclaurin-nat-int-transfer*:

fixes F **and** $f :: real \Rightarrow 'a :: real-normed-vector$

assumes $((\lambda b. (\sum_{k=a..b} f\ (real\ k)) - F\ (real\ b) - T\ (real\ b)) \longrightarrow C)$ *at-top*

shows $((\lambda b. (\sum_{k=int\ a..b} f\ (of-int\ k)) - F\ (of-int\ b) - T\ (of-int\ b)) \longrightarrow C)$ *at-top*

proof -

have $*$: $((\lambda b. (\sum_{k=int\ a..int\ b} f\ (of-int\ k)) - F\ (of-int\ (int\ b)) - T\ (of-int\ (int\ b)))$

$\longrightarrow C)$ *at-top* **using** *assms* **by** (*subst sum-atLeastAtMost-of-int-nat-transfer*)

simp

thus *?thesis* **by** (*rule filterlim-int-of-nat-at-topD*)

qed

locale *euler-maclaurin-nat* =

fixes $F\ f :: real \Rightarrow 'a :: banach$

fixes $fs :: nat \Rightarrow real \Rightarrow 'a$

fixes $a :: nat$

fixes $N :: nat$ **assumes** $N: N > 0$

fixes $C :: 'a$

fixes $Y :: real\ set$ **assumes** *fin*: *finite* Y

assumes *fs-0* [*simp*]: $fs\ 0 = f$

assumes *fs-cont* [*continuous-intros*]:

$\bigwedge k. k \leq N \implies continuous-on\ \{real\ a..\}\ (fs\ k)$

assumes *fs-deriv* [*derivative-intros*]:

$\bigwedge k\ x. k < N \implies x \in \{real\ a..\} - Y \implies (fs\ k\ has-vector-derivative\ fs\ (Suc\ k)\ x)$ (*at* x)

assumes *F-cont* [*continuous-intros*]: *continuous-on* $\{real\ a..\}\ F$

assumes *F-deriv* [*derivative-intros*]:

$\bigwedge x. x \in \{real\ a..\} - Y \implies (F\ has-vector-derivative\ f\ x)$ (*at* x)

assumes *limit*:

$((\lambda b. (\sum_{k=a..b} f\ k) - F\ (real\ b) -$

$(\sum_{i < N. (bernoulli'\ (Suc\ i) / fact\ (Suc\ i)) *_{\mathbb{R}} fs\ i\ (real\ b))) \longrightarrow C)$ *at-top*

begin

sublocale *euler-maclaurin-int* $F\ f\ fs\ int\ a\ N\ C\ Y$

by *standard* (*insert* N *fin* *fs-cont* *fs-deriv* *F-cont* *F-deriv*

euler-maclaurin-nat-int-transfer[*OF limit*], *simp-all*)

theorem *euler-maclaurin-strong-nat*:

assumes $ab: a \leq b$
defines $S \equiv (\sum k=a..b. f \text{ (real } k))$
shows $S = F \text{ (real } b) + C + (\sum i < N. (\text{bernoulli}' (Suc\ i) / \text{fact} (Suc\ i)) *_R fs\ i \text{ (real } b)) -$
 $EM\text{-remainder } N \text{ (fs } N) \text{ (int } b)$
using *euler-maclaurin-strong-int*[of int b]
by (*simp add: assms sum-atLeastAtMost-of-int-nat-transfer*)

end

locale *euler-maclaurin-nat'* =
fixes $F f :: \text{real} \Rightarrow 'a :: \text{banach}$
fixes $fs :: \text{nat} \Rightarrow \text{real} \Rightarrow 'a$
fixes $a :: \text{nat}$
fixes $N :: \text{nat}$
fixes $C :: 'a$
fixes $Y :: \text{real set}$ **assumes** *fin*: *finite* Y
assumes *fs-0* [*simp*]: $fs\ 0 = f$
assumes *fs-cont* [*continuous-intros*]:
 $\bigwedge k. k \leq 2*N+1 \implies \text{continuous-on } \{\text{real } a..\} (fs\ k)$
assumes *fs-deriv* [*derivative-intros*]:
 $\bigwedge k\ x. k \leq 2*N \implies x \in \{\text{real } a..\} - Y \implies (fs\ k \text{ has-vector-derivative } fs\ (Suc\ k)\ x) \text{ (at } x)$
assumes *F-cont* [*continuous-intros*]: *continuous-on* $\{\text{real } a..\} F$
assumes *F-deriv* [*derivative-intros*]:
 $\bigwedge x. x \in \{\text{real } a..\} - Y \implies (F \text{ has-vector-derivative } f\ x) \text{ (at } x)$
assumes *limit*:
 $((\lambda b. (\sum k=a..b. f\ k) - F \text{ (real } b) -$
 $(\sum i < 2*N+1. (\text{bernoulli}' (Suc\ i) / \text{fact} (Suc\ i)) *_R fs\ i \text{ (real } b))) \longrightarrow C)$
at-top
begin

sublocale *euler-maclaurin-int'* $F f fs \text{ int } a\ N\ C\ Y$
by *standard* (*insert fin fs-cont fs-deriv F-cont F-deriv*
euler-maclaurin-nat-int-transfer[*OF limit*], *simp-all*)

theorem *euler-maclaurin-strong-nat'*:
assumes $a \leq b$
shows $(\sum k=a..b. f \text{ (real } k)) =$
 $F \text{ (real } b) + C + (1 / 2) *_R f \text{ (real } b) +$
 $(\sum i=1..N. (\text{bernoulli} (2*i) / \text{fact} (2*i)) *_R fs (2*i-1) \text{ (real } b)) -$
 $EM\text{-remainder } (2*N+1) \text{ (fs } (2*N+1))\ b$
using *euler-maclaurin-strong-int'*[of b]
by (*simp add: assms sum-atLeastAtMost-of-int-nat-transfer*)

end

1.5 Bounds on the remainder term

The following theorems provide some simple means to bound the remainder terms. In practice, better bounds can often be obtained e. g. for the n -th remainder term by expanding it to the sum of the first discarded term in the expansion and the $n + 1$ -th remainder term.

lemma

fixes $f :: \text{real} \Rightarrow 'a :: \{\text{real-normed-field}, \text{banach}\}$
and $g\ g' :: \text{real} \Rightarrow \text{real}$
assumes $\text{fin}: \text{finite } Y$
assumes $\text{pbernpoly-bound}: \forall x. |\text{pbernpoly } n\ x| \leq D$
assumes $\text{cont-f}: \text{continuous-on } \{a..\} f$
assumes $\text{cont-g}: \text{continuous-on } \{a..\} g$
assumes $\text{cont-g}': \text{continuous-on } \{a..\} g'$
assumes $\text{limit-g}: (g \longrightarrow C) \text{ at-top}$
assumes $\text{f-bound}: \bigwedge x. x \geq a \implies \text{norm } (f\ x) \leq g'\ x$
assumes $\text{deriv}: \bigwedge x. x \in \{a..\} - Y \implies (g \text{ has-field-derivative } g'\ x) \text{ (at } x)$
shows $\text{norm-EM-remainder-le-strong-int}: \forall x. \text{of-int } x \geq a \longrightarrow \text{norm } (\text{EM-remainder } n\ f\ x) \leq D / \text{fact } n * (C - g\ x)$
and $\text{norm-EM-remainder-le-strong-nat}: \forall x. \text{real } x \geq a \longrightarrow \text{norm } (\text{EM-remainder } n\ f\ (\text{int } x)) \leq D / \text{fact } n * (C - g\ x)$
proof –
from pbernpoly-bound **have** $D: \text{norm } (\text{pbernpoly } n\ x) \leq D$ **for** x **by** auto
from $\text{this}[\text{of } 0]$ **have** $D\text{-nonneg}: D \geq 0$ **by** simp
define D' **where** $D' = D / \text{fact } n$
from $D\text{-nonneg}$ **have** $D'\text{-nonneg}: D' \geq 0$ **by** $(\text{simp add: } D'\text{-def})$
have $\text{bound}: \text{norm } (\text{EM-remainder}'\ n\ f\ x\ y) \leq D' * (g\ y - g\ x)$
if $xy: x \geq a\ x \leq y$ **for** $x\ y :: \text{real}$
proof –
have $\text{norm } (\text{EM-remainder}'\ n\ f\ x\ y) = \text{norm } (\text{integral } \{x..y\} (\lambda t. \text{pbernpoly } n\ t *_{\mathbb{R}} f\ t)) / \text{fact } n$
by $(\text{simp add: } \text{EM-remainder}'\text{-def})$
also have $(\lambda t. D * g'\ t) \text{ integrable-on } \{x..y\}$ **using** xy
by $(\text{intro } \text{integrable-continuous-real continuous-intros continuous-on-subset}[\text{OF cont-g}'])$
 auto
hence $\text{norm } (\text{integral } \{x..y\} (\lambda t. \text{pbernpoly } n\ t *_{\mathbb{R}} f\ t)) \leq \text{integral } \{x..y\} (\lambda t. D * g'\ t)$ **using** $D\ D\text{-nonneg } xy$
by $(\text{intro } \text{integral-norm-bound-integral integrable-EM-remainder}'\ \text{continuous-on-subset}[\text{OF cont-f}]) (\text{auto intro!: } \text{mult-mono } f\text{-bound})$
also have $\dots = D * \text{integral } \{x..y\} g'$ **by** simp
also have $(g' \text{ has-integral } (g\ y - g\ x)) \{x..y\}$ **using** xy
by $(\text{intro } \text{fundamental-theorem-of-calculus-strong}[\text{OF fin}] \text{ continuous-on-subset}[\text{OF cont-g}])$
 $(\text{auto simp: } \text{has-real-derivative-iff-has-vector-derivative } [\text{symmetric}] \text{ intro!: } \text{deriv})$

hence $\text{integral } \{x..y\} g' = g y - g x$ **by** (*simp add: has-integral-iff*)
finally show $?thesis$ **by** (*simp add: D'-def divide-simps*)
qed

have $\text{lim: } ((\lambda y. \text{EM-remainder}' n f x (\text{of-int } y)) \longrightarrow \text{EM-remainder } n f x)$
at-top
if $x \geq a$ **for** $x :: \text{int}$
proof –
have $(\lambda n. g (\text{real } n)) \longrightarrow C$
by (*rule filterlim-compose[OF limit-g filterlim-real-sequentially]*)
hence *Cauchy*: $\text{Cauchy } (\lambda n. g (\text{real } n))$ **using** *convergent-eq-Cauchy* **by** *blast*
have *Cauchy* $(\lambda y. \text{EM-remainder}' n f x (\text{int } y))$
proof (*rule CauchyI', goal-cases*)
case $(1 \ \varepsilon)$
define ε' **where** $\varepsilon' = (\text{if } D' = 0 \text{ then } 1 \text{ else } \varepsilon / (2 * D'))$
from $\langle \varepsilon > 0 \rangle$ *D'-nonneg* **have** $\varepsilon': \varepsilon' > 0$ **by** (*simp add: \varepsilon'-def divide-simps*)
from *CauchyD*[*OF Cauchy this*] **obtain** M
where $M: \bigwedge m n. m \geq M \implies n \geq M \implies \text{norm } (g (\text{real } m) - g (\text{real } n))$
 $< \varepsilon'$ **by** *blast*
show *?case*
proof (*intro CauchyI' exI[of - max (max 0 M) (nat x)] allI impI, goal-cases*)
case $(1 \ k \ l)$
have $\text{EM-remainder}' n f x k + \text{EM-remainder}' n f k l = \text{EM-remainder}' n$
 $f x l$
using $1 \ x$ **by** (*intro EM-remainder'-combine continuous-on-subset[OF cont-f]] auto*)
hence $\text{EM-remainder}' n f x l - \text{EM-remainder}' n f x k = \text{EM-remainder}'$
 $n f k l$
by (*simp add: algebra-simps*)
also from $1 \ x$ **have** $\text{norm } \dots \leq D' * (g l - g k)$ **by** (*intro bound*) *auto*
also have $g l - g k \leq \text{norm } (g l - g k)$ **by** *simp*
also from 1 **have** $\dots \leq \varepsilon'$ **using** M [*of l k*] **by** *auto*
also from $\langle \varepsilon > 0 \rangle$ **have** $D' * \varepsilon' \leq \varepsilon / 2$ **by** (*simp add: \varepsilon'-def*)
also from $\langle \varepsilon > 0 \rangle$ **have** $\dots < \varepsilon$ **by** *simp*
finally show *?case* **by** (*simp add: D'-nonneg mult-left-mono dist-norm norm-minus-commute*)
qed
qed
then obtain L **where** $(\lambda y. \text{EM-remainder}' n f x (\text{int } y)) \longrightarrow L$
by (*auto simp: Cauchy-convergent-iff convergent-def*)
from *filterlim-int-of-nat-at-topD*[*OF this*]
have $((\lambda y. \text{EM-remainder}' n f x (\text{of-int } y)) \longrightarrow L)$ *at-top* **by** *simp*
moreover from *this* **have** $\text{EM-remainder } n f x = L$ **by** (*rule EM-remainder-eqI*)
ultimately show $(\lambda y. \text{EM-remainder}' n f x (\text{of-int } y)) \longrightarrow \text{EM-remainder}$
 $n f x$ *at-top*
by *simp*
qed

have $*$: $\text{norm } (\text{EM-remainder } n f x) \leq D' * (C - g x)$ **if** $x: x \geq a$ **for** $x :: \text{int}$

proof (*rule tendsto-le*)
show $((\lambda y. D' * (g \text{ (of-int } y) - g \text{ (of-int } x))) \longrightarrow D' * (C - g \text{ (of-int } x)))$
at-top
by (*intro tendsto-intros filterlim-compose[OF limit-g]*)
show $((\lambda y. \text{norm } (EM\text{-remainder}' n f x \text{ (of-int } y))) \longrightarrow \text{norm } (EM\text{-remainder } n f x))$ *at-top*
using x **by** (*intro tendsto-norm lim*)
show *eventually* $(\lambda y. \text{norm } (EM\text{-remainder}' n f \text{ (of-int } x) \text{ (of-int } y)) \leq D' * (g \text{ (of-int } y) - g \text{ (of-int } x)))$ *at-top*
using *eventually-ge-at-top*[*of x*] **by** *eventually-elim* (*rule bound, insert x, simp-all*)
qed *simp-all*
thus $\forall x. \text{of-int } x \geq a \longrightarrow \text{norm } (EM\text{-remainder } n f x) \leq D' * (C - g x)$ **by** *blast*

have $\text{norm } (EM\text{-remainder } n f x) \leq D' * (C - g x)$ **if** $x: x \geq a$ **for** $x :: \text{nat}$
using x **by** *simp*
thus $\forall x. \text{real } x \geq a \longrightarrow \text{norm } (EM\text{-remainder } n f \text{ (int } x)) \leq D' * (C - g x)$ **by** *blast*
qed

lemma

fixes $f :: \text{real} \Rightarrow 'a :: \{\text{real-normed-field, banach}\}$
and $g g' :: \text{real} \Rightarrow \text{real}$
assumes *fin*: *finite Y*
assumes *pbernpoly-bound*: $\forall x. |pbernpoly n x| \leq D$
assumes *cont-f*: *continuous-on {a..} f*
assumes *cont-g*: *continuous-on {a..} g*
assumes *cont-g'*: *continuous-on {a..} g'*
assumes *limit-g*: $(g \longrightarrow 0)$ *at-top*
assumes *f-bound*: $\bigwedge x. x \geq a \implies \text{norm } (f x) \leq g' x$
assumes *deriv*: $\bigwedge x. x \in \{a..\} - Y \implies (g \text{ has-field-derivative } -g' x)$ (*at x*)
shows *norm-EM-remainder-le-strong-int'*:
 $\forall x. \text{of-int } x \geq a \longrightarrow \text{norm } (EM\text{-remainder } n f x) \leq D / \text{fact } n * g x$
and *norm-EM-remainder-le-strong-nat'*:
 $\forall x. \text{real } x \geq a \longrightarrow \text{norm } (EM\text{-remainder } n f \text{ (int } x)) \leq D / \text{fact } n * g x$

proof –
have $\forall x. \text{of-int } x \geq a \longrightarrow \text{norm } (EM\text{-remainder } n f x) \leq D / \text{fact } n * (0 - (-g x))$ **using** *assms*
by (*intro norm-EM-remainder-le-strong-int[OF fin pbernpoly-bound - - cont-g']*)
(auto intro! continuous-intros tendsto-eq-intros derivative-eq-intros)
thus $\forall x. \text{of-int } x \geq a \longrightarrow \text{norm } (EM\text{-remainder } n f x) \leq D / \text{fact } n * g x$ **by** *auto*

next
have $\forall x. \text{real } x \geq a \longrightarrow \text{norm } (EM\text{-remainder } n f \text{ (int } x)) \leq D / \text{fact } n * (0 - (-g x))$ **using** *assms*
by (*intro norm-EM-remainder-le-strong-nat[OF fin pbernpoly-bound - - cont-g']*)
(auto intro! continuous-intros tendsto-eq-intros derivative-eq-intros)
thus $\forall x. \text{real } x \geq a \longrightarrow \text{norm } (EM\text{-remainder } n f \text{ (int } x)) \leq D / \text{fact } n * g x$

by auto
qed

lemma norm-EM-remainder'-le:

fixes $f :: \text{real} \Rightarrow 'a :: \{\text{real-normed-field, banach}\}$
and $g g' :: \text{real} \Rightarrow \text{real}$
assumes $\text{cont-f: continuous-on } \{a..\} f$
assumes $\text{cont-g': continuous-on } \{a..\} g'$
assumes $\text{f-bigo: eventually } (\lambda x. \text{norm } (f x) \leq g' x) \text{ at-top}$
assumes $\text{deriv: eventually } (\lambda x. (g \text{ has-field-derivative } g' x) (at x)) \text{ at-top}$
obtains $C D$ where
eventually $(\lambda x. \text{norm } (EM\text{-remainder}' n f a x) \leq C + D * g x) \text{ at-top}$
proof -
note $\text{cont} = \text{continuous-on-subset}[OF \text{cont-f}] \text{continuous-on-subset}[OF \text{cont-g}']$
from $\text{bounded-pbernpoly}[of n]$ obtain D where $D: \bigwedge x. \text{norm } (pbernpoly n x) \leq D$
by blast
from $\text{this}[of 0]$ have $D\text{-nonneg: } D \geq 0$ by simp
from $\text{eventually-conj}[OF \text{f-bigo eventually-conj}[OF \text{deriv eventually-ge-at-top}[of a]]]$
obtain $x0$ where $x0$:
 $x0 \geq a \wedge x. x \geq x0 \implies \text{norm } (f x) \leq g' x$
 $\bigwedge x. x \geq x0 \implies (g \text{ has-field-derivative } g' x) (at x)$
by (auto simp: eventually-at-top-linorder)
define C where $C = (\text{norm } (\text{integral } \{a..x0\} (\lambda t. pbernpoly n t *_R f t))) - D * g x0) / \text{fact } n$
have eventually $(\lambda x. \text{norm } (EM\text{-remainder}' n f a x) \leq C + D / \text{fact } n * g x) \text{ at-top}$
using eventually-ge-at-top[of $x0$]
proof eventually-elim
case (elim x)
have $\text{integral } \{a..x\} (\lambda t. pbernpoly n t *_R f t) =$
 $\text{integral } \{a..x0\} (\lambda t. pbernpoly n t *_R f t) +$
 $\text{integral } \{x0..x\} (\lambda t. pbernpoly n t *_R f t)$ (is - = ?I1 + ?I2) using elim
 $x0(1)$
by (intro Henstock-Kurzweil-Integration.integral-combine [symmetric] integrable-EM-remainder' cont) auto
also have $\text{norm } \dots \leq \text{norm } ?I1 + \text{norm } ?I2$ by (rule norm-triangle-ineq)
also have $\text{norm } ?I2 \leq \text{integral } \{x0..x\} (\lambda t. D * g' t)$
using $x0 D D\text{-nonneg}$
by (intro integral-norm-bound-integral integrable-EM-remainder')
(auto intro!: integrable-continuous-real continuous-intros cont mult-mono)
also have $\dots = D * \text{integral } \{x0..x\} g'$ by simp
also from elim have $(g' \text{ has-integral } (g x - g x0)) \{x0..x\}$
by (intro fundamental-theorem-of-calculus)
(auto intro!: has-field-derivative-at-within[OF $x0(3)$]
simp: has-real-derivative-iff-has-vector-derivative [symmetric])
hence $\text{integral } \{x0..x\} g' = g x - g x0$ by (simp add: has-integral-iff)
finally have $\text{norm } (\text{integral } \{a..x\} (\lambda t. pbernpoly n t *_R f t)) \leq \text{norm } ?I1 +$

```

D * (g x - g x0)
  by simp-all
  thus norm (EM-remainder' n f a x) ≤ C + D / fact n * g x
    by (simp add: EM-remainder'-def field-simps C-def)
qed
thus ?thesis by (rule that)
qed

```

1.6 Application to harmonic numbers

As a first application, we can apply the machinery we have developed to the harmonic numbers.

definition *harm-remainder* :: nat ⇒ nat ⇒ real **where**

```

harm-remainder N n = EM-remainder (2*N+1) (λx. -fact (2*N+1) / x ^
(2*N+2)) (int n)

```

lemma *harm-expansion*:

assumes *n*: $n > 0$ **and** *N*: $N > 0$

shows $\text{harm } n = \ln n + \text{euler-mascheroni} + 1 / (2*n) -$
 $(\sum_{i=1..N}. \text{bernoulli } (2*i) / ((2*i) * n ^ (2*i))) - \text{harm-remainder}$

N n

proof –

define *fs* **where** $fs = (\lambda k x. (-1) ^ k * \text{fact } k / x ^ (\text{Suc } k) :: \text{real})$

interpret *euler-maclaurin-nat'* $\ln \lambda x. 1/x$ *fs* 1 *N* *euler-mascheroni* {}

proof

fix *k x* **assume** $k \leq 2*N$ $x \in \{\text{real } 1..\} - \{\}$

thus (*fs k* *has-vector-derivative fs* (*Suc k*) *x*) (*at x*)

by (*cases k = 0*)

(*auto intro!*: *derivative-eq-intros*

simp: fs-def has-real-derivative-iff-has-vector-derivative [symmetric]
field-simps power-diff)

next

have $(\lambda b. \text{harm } b - \ln (\text{real } b) -$

$(\sum_{i < 2*N+1}. \text{bernoulli}' (\text{Suc } i) * (-1) ^ i / (\text{real } (\text{Suc } i) * (\text{real } b ^ \text{Suc } i))))$

$\longrightarrow (\text{euler-mascheroni} - (\sum_{i < 2*N+1}. 0))$

by (*intro tendsto-diff euler-mascheroni-LIMSEQ tendsto-sum*

real-tendsto-divide-at-top[OF tendsto-const]

filterlim-tendsto-pos-mult-at-top[OF tendsto-const] filterlim-pow-at-top

filterlim-real-sequentially) *auto*

thus $(\lambda b. (\sum_{k=1..b}. 1 / \text{real } k) - \ln (\text{real } b) -$

$(\sum_{i < 2*N+1}. (\text{bernoulli}' (\text{Suc } i) / \text{fact } (\text{Suc } i)) *_{\mathbb{R}} fs i (\text{real } b))) \longrightarrow$
euler-mascheroni

by (*simp add: harm-def divide-simps fs-def*)

qed (*insert n N, auto intro!*: *continuous-intros derivative-eq-intros*

simp: fs-def has-real-derivative-iff-has-vector-derivative [symmetric])

have $\text{harm } n = (\sum_{k=1..n}. 1 / \text{real } k)$ **by** (*simp add: harm-def divide-simps*)

also have $\dots = \ln (\text{real } n) + \text{euler-mascheroni} + (1/2) *_{\mathbb{R}} (1 / \text{real } n) +$

$$\left(\sum_{i=1..N}. (\text{bernoulli } (2*i) / \text{fact } (2*i)) *_{\mathbb{R}} \text{fs } (2*i-1) (\text{real } n) \right)$$
 –

$$\text{EM-remainder } (2*N+1) (\text{fs } (2*N+1)) (\text{int } n) \text{ using } n \ N$$
 using n by (intro euler-maclaurin-strong-nat') simp-all
 also have $\left(\sum_{i=1..N}. (\text{bernoulli } (2*i) / \text{fact } (2*i)) *_{\mathbb{R}} (\text{fs } (2*i-1) (\text{real } n)) \right)$
 =

$$\left(\sum_{i=1..N}. -(\text{bernoulli } (2*i) / (\text{real } (2*i) * \text{real } n ^{(2*i)})) \right)$$
 by (intro sum.cong refl)
 (simp-all add: fs-def divide-simps fact-reduce del: of-nat-Suc power-Suc)
 also have ... = $-(\sum_{i=1..N}. \text{bernoulli } (2*i) / (\text{real } (2*i) * \text{real } n ^{(2*i)}))$
 by (simp add: sum-negf)
 finally show ?thesis unfolding fs-def by (simp add: harm-remainder-def)
 qed

lemma of-nat-ge-1-iff: of-nat $x \geq (1 :: 'a :: \text{linordered-semidom}) \iff x \geq 1$
 using of-nat-le-iff[of 1 x] by (simp del: of-nat-le-iff)

lemma harm-remainder-bound:

fixes $N :: \text{nat}$
 assumes $N: N > 0$
 shows $\exists C. \forall n \geq 1. \text{norm } (\text{harm-remainder } N \ n) \leq C / \text{real } n ^{(2*N+1)}$
proof –
 from bounded-pbernpoly[of $2*N+1$] obtain D where $D: \forall x. |pbernpoly (2*N+1) x| \leq D$ by auto
 have $\forall x. 1 \leq \text{real } x \implies \text{norm } (\text{harm-remainder } N \ x) \leq D / \text{fact } (2*N+1) * (\text{fact } (2*N) / x ^{(2*N+1)})$
 unfolding harm-remainder-def of-int-of-nat-eq
proof (rule norm-EM-remainder-le-strong-nat'[of {}])
 fix $x :: \text{real}$ assume $x: x \geq 1$
 show $\text{norm } (-\text{fact } (2*N+1) / x ^{(2 * N + 2)}) \leq \text{fact } (2*N+1) / x ^{(2*N+2)}$
 using x by simp
next
 show $((\lambda x::\text{real}. \text{fact } (2 * N) / x ^{(2 * N + 1)}) \longrightarrow 0) \text{ at-top}$
 by (intro real-tendsto-divide-at-top[OF tendsto-const] filterlim-pow-at-top filterlim-ident) simp-all
qed (insert $N \ D$, auto intro!: derivative-eq-intros continuous-intros simp: field-simps power-diff)
 hence $\forall x. 1 \leq x \implies \text{norm } (\text{harm-remainder } N \ x) \leq D / (2*N+1) / \text{real } x ^{(2*N+1)}$ by simp
 thus ?thesis by blast
qed

1.7 Application to sums of inverse squares

In the same vein, we can derive the asymptotics of the partial sum of inverse squares.

lemma sum-inverse-squares-expansion:

assumes $n: n > 0$ **and** $N: N > 0$
shows $(\sum_{k=1..n}. 1 / \text{real } k^2) =$
 $\pi^2 / 6 - 1 / \text{real } n + 1 / (2 * \text{real } n^2) -$
 $(\sum_{i=1..N}. \text{bernoulli } (2*i) / n^{(2*i+1)}) -$
 $\text{EM-remainder } (2*N+1) (\lambda x. -\text{fact } (2*N+2) / x^{(2*N+3)})$
(int n)
proof –
have $\exists: \exists = \text{Suc } (\text{Suc } (\text{Suc } 0))$ **by** *(simp add: eval-nat-numeral)*
define fs **where** $fs = (\lambda k x. (-1)^k * \text{fact } (\text{Suc } k) / x^{(k+2)} :: \text{real})$
interpret *euler-maclaurin-nat'* $\lambda x. -1/x \lambda x. 1/x^2 fs 1 N \pi^2/6 \{\}$
proof
fix $k x$ **assume** $k \leq 2*N$ $x \in \{\text{real } 1..\}$ – $\{\}$
thus *(fs k has-vector-derivative fs (Suc k) x) (at x)*
by *(cases k = 0)*
(auto intro!: derivative-eq-intros
simp: fs-def has-real-derivative-iff-has-vector-derivative [symmetric]
field-simps power-diff)
next
from *inverse-squares-sums*
have $(\lambda n. \sum_{k < n}. 1 / \text{real } (\text{Suc } k)^2) \longrightarrow \pi^2 / 6$ **by** *(simp add: sums-def)*
also have $(\lambda n. \sum_{k < n}. 1 / \text{real } (\text{Suc } k)^2) = (\lambda n. \sum_{k=1..n}. 1 / \text{real } k^2)$
by *(intro ext sum.reindex-bij-witness[of - \lambda n. n - 1 Suc]) auto*
finally have $(\lambda b. (\sum_{k=1..b}. 1 / \text{real } k^2) + 1 / \text{real } b -$
 $(\sum_{i < 2*N+1}. \text{bernoulli}' (\text{Suc } i) * (-1)^i / (\text{real } b^{(i+2)})))$
 $\longrightarrow (\pi^2/6 + 0 - (\sum_{i < 2*N+1}. 0))$
by *(intro tendsto-diff tendsto-add real-tendsto-divide-at-top[OF tendsto-const]*

filterlim-tendsto-pos-mult-at-top[OF tendsto-const] filterlim-pow-at-top
filterlim-real-sequentially tendsto-sum) auto
thus $(\lambda b. (\sum_{k=1..b}. 1 / \text{real } k^2) - (-1 / \text{real } b) -$
 $(\sum_{i < 2*N+1}. (\text{bernoulli}' (\text{Suc } i) / \text{fact } (\text{Suc } i)) *_{\mathbb{R}} fs i (\text{real } b))) \longrightarrow$
 $\pi^2/6$
by *(simp add: harm-def field-simps fs-def del: power-Suc of-nat-Suc)*
qed *(insert n N, auto intro!: continuous-intros derivative-eq-intros*
simp: fs-def has-real-derivative-iff-has-vector-derivative [symmetric] power2-eq-square)

have $(\sum_{k=1..n}. 1 / \text{real } k^2) = -1 / \text{real } n + \pi^2/6 + (1/2) *_{\mathbb{R}} (1 / \text{real } n^2) +$
 $(\sum_{i=1..N}. (\text{bernoulli } (2*i) / \text{fact } (2*i)) *_{\mathbb{R}} fs (2*i-1) (\text{real } n)) -$
 $\text{EM-remainder } (2*N+1) (fs (2*N+1))$ *(int n) using n N*
using n **by** *(intro euler-maclaurin-strong-nat') simp-all*
also have $(\sum_{i=1..N}. (\text{bernoulli } (2*i) / \text{fact } (2*i)) *_{\mathbb{R}} (fs (2*i-1) (\text{real } n)))$
 $=$
 $(\sum_{i=1..N}. -(\text{bernoulli } (2*i) / (\text{real } n^{(2*i+1)})))$
by *(intro sum.cong refl)*
(simp-all add: fs-def divide-simps fact-reduce del: of-nat-Suc power-Suc)
also have $\dots = -(\sum_{i=1..N}. \text{bernoulli } (2*i) / \text{real } n^{(2*i+1)})$

by (*simp add: sum-negf*)
finally show *?thesis unfolding fs-def by (simp add: fs-def 3)*
qed

lemma *sum-inverse-squares-remainder-bound:*

fixes $N :: nat$
assumes $N: N > 0$
defines $R \equiv (\lambda n. EM\text{-remainder } (2*N+1) (\lambda x. -fact (2*N+2) / x ^{(2*N+3)})$
(int n)
shows $\exists C. \forall n \geq 1. norm (R n) \leq C / real n ^{(2*N+2)}$
proof –
have $3: 3 = Suc (Suc (Suc 0))$ **by** *simp*
from *bounded-pbernpoly[of 2*N+1]* **obtain** D **where** $D: \forall x. |pbernpoly (2*N+1) x| \leq D$ **by** *auto*
have $\forall x. 1 \leq real x \longrightarrow norm (R x) \leq D / fact (2*N+1) * (fact (2*N+1) / x ^{(2*N+2)})$
unfolding *R-def of-int-of-nat-eq*
proof (*rule norm-EM-remainder-le-strong-nat'[of {}]*)
fix $x :: real$ **assume** $x: x \geq 1$
show $norm (-fact (2*N+2) / x ^{(2*N+3)}) \leq fact (2*N+2) / x ^{(2*N+3)}$
using x **by** *simp*
next
show $((\lambda x::real. fact (2*N+1) / x ^{(2*N+2)}) \longrightarrow 0)$ *at-top*
by (*intro real-tendsto-divide-at-top[OF tendsto-const] filterlim-pow-at-top filterlim-ident*)
simp-all
qed (*insert N D, auto intro!: derivative-eq-intros continuous-intros simp: field-simps power-diff 3*)
hence $\forall x \geq 1. norm (R x) \leq D / real x ^{(2 * N + 2)}$ **by** *simp*
thus *?thesis* **by** *blast*
qed

end

2 Connection of Euler–MacLaurin summation to Landau symbols

theory *Euler-MacLaurin-Landau*

imports

Euler-MacLaurin

Landau-Symbols.Landau-More

begin

2.1 O -bound for the remainder term

Landau symbols allow us to state the bounds on the remainder terms from the Euler–MacLaurin formula a bit more nicely.

lemma

fixes $f :: \text{real} \Rightarrow 'a :: \{\text{real-normed-field, banach}\}$
and $g\ g' :: \text{real} \Rightarrow \text{real}$
assumes $\text{fin}: \text{finite } Y$
assumes $\text{cont-f}: \text{continuous-on } \{a..\} f$
assumes $\text{cont-g}: \text{continuous-on } \{a..\} g$
assumes $\text{cont-g}': \text{continuous-on } \{a..\} g'$
assumes $\text{limit-g}: (g \longrightarrow 0) \text{ at-top}$
assumes $\text{f-bound}: \bigwedge x. x \geq a \implies \text{norm } (f\ x) \leq g'\ x$
assumes $\text{deriv}: \bigwedge x. x \in \{a..\} - Y \implies (g \text{ has-field-derivative } -g'\ x) \text{ (at } x)$
shows $\text{EM-remainder-strong-bigo-int}: (\lambda x::\text{int}. \text{norm } (\text{EM-remainder } n\ f\ x)) \in O(g)$
and $\text{EM-remainder-strong-bigo-nat}: (\lambda x::\text{nat}. \text{norm } (\text{EM-remainder } n\ f\ x)) \in O(g)$
proof –
from $\text{bounded-pbernpoly}[of\ n]$ **obtain** D **where** $D: \forall x. |pbernpoly\ n\ x| \leq D$ **by** auto
from $\text{norm-EM-remainder-le-strong-int}'[OF\ \text{fin}\ D\ \text{assms}(2-)]$
have $*$: $\bigwedge x. x \geq a \longrightarrow \text{norm } (\text{EM-remainder } n\ f\ x) \leq D / \text{fact } n * g\ x$ **by** auto
have $**$: $\text{eventually } (\lambda x::\text{int}. \text{norm } (\text{EM-remainder } n\ f\ x) \leq \text{abs } (D / \text{fact } n) * \text{abs } (g\ x)) \text{ at-top}$
using $\text{eventually-ge-at-top}[of\ \text{ceiling } a]$
proof eventually-elim
case $(\text{elim } x)$
with $*[of\ x]$ **have** $\text{norm } (\text{EM-remainder } n\ f\ x) \leq D / \text{fact } n * g\ x$ **by** $(\text{simp add: ceiling-le-iff})$
also have $\dots \leq \text{abs } (D / \text{fact } n * g\ x)$ **by** $(\text{rule abs-ge-self})$
also have $\dots = \text{abs } (D / \text{fact } n) * \text{abs } (g\ x)$ **by** $(\text{simp add: abs-mult})$
finally show $?case$.
qed
thus $(\lambda x::\text{int}. \text{norm } (\text{EM-remainder } n\ f\ x)) \in O(g)$
by $(\text{intro bigoI}[of\ -\ \text{abs } D / \text{fact } n]) (\text{auto elim!: eventually-mono})$
hence $(\lambda x::\text{nat}. \text{norm } (\text{EM-remainder } n\ f\ (\text{int } x))) \in O(\lambda x. g\ (\text{of-int } (\text{int } x)))$
by $(\text{rule landau-o.big.compose}) (\text{fact filterlim-int-sequentially})$
thus $(\lambda x::\text{nat}. \text{norm } (\text{EM-remainder } n\ f\ x)) \in O(g)$ **by** simp
qed

2.2 Asymptotic expansion of the harmonic numbers

We can now show the asymptotic expansion

$$H_n = \ln n + \gamma + \frac{1}{2n} - \sum_{i=1}^m \frac{B_{2i}}{2i} n^{-2i} + O(n^{-2m-2})$$

lemma $\text{harm-remainder-bigo}$:

assumes $N > 0$

shows $\text{harm-remainder } N \in O(\lambda n. 1 / \text{real } n \wedge (2 * N + 1))$

proof –

from $\text{harm-remainder-bound}[OF\ \text{assms}]$

obtain C **where** $\forall n \geq 1. \text{norm} (\text{harm-remainder } N \ n) \leq C / \text{real } n^{\wedge}(2 * N + 1)$..
thus *?thesis*
by (*intro bigoI[of - C] eventually-mono[OF eventually-ge-at-top[of 1]]*) *auto*
qed

lemma *harm-expansion-bigo*:

fixes $N :: \text{nat}$
defines $T \equiv \lambda n. \ln \ n + \text{euler-mascheroni} + 1 / (2 * n) -$
 $(\sum i=1..N. \text{bernoulli } (2*i) / ((2*i) * n^{\wedge}(2*i)))$
defines $S \equiv (\lambda n. \text{bernoulli } (2*(\text{Suc } N)) / ((2*\text{Suc } N) * \text{real } n^{\wedge}(2*\text{Suc } N)))$
shows $(\lambda n. \text{harm } n - T \ n) \in O(\lambda n. 1 / \text{real } n^{\wedge}(2 * N + 2))$
proof –
have $(\lambda n. \text{harm } n - T \ n) \in \Theta(\lambda n. -S \ n - \text{harm-remainder } (\text{Suc } N) \ n)$
by (*intro bigthetaI-cong eventually-mono[OF eventually-gt-at-top[of 0::nat]]*)
(auto simp: T-def harm-expansion[of - Suc N] S-def)
also have $(\lambda n. -S \ n - \text{harm-remainder } (\text{Suc } N) \ n) \in O(\lambda n. 1 / \text{real } n^{\wedge}(2 * N + 2))$
proof (*intro sum-in-bigo*)
show $(\lambda x. -S \ x) \in O(\lambda n. 1 / \text{real } n^{\wedge}(2 * N + 2))$ **unfolding** *S-def*
by (*rule landau-o.big.compose[OF - filterlim-real-sequentially]*) *simp*
have $\text{harm-remainder } (\text{Suc } N) \in O(\lambda n. 1 / \text{real } n^{\wedge}(2 * \text{Suc } N + 1))$
by (*rule harm-remainder-bigo*) *simp-all*
also have $(\lambda n. 1 / \text{real } n^{\wedge}(2 * \text{Suc } N + 1)) \in O(\lambda n. 1 / \text{real } n^{\wedge}(2 * N + 2))$
by (*rule landau-o.big.compose[OF - filterlim-real-sequentially]*) *simp*
finally show $\text{harm-remainder } (\text{Suc } N) \in \dots$.
qed
finally show *?thesis* .
qed

lemma *harm-expansion-bigo-simple1*:

$(\lambda n. \text{harm } n - (\ln \ n + \text{euler-mascheroni} + 1 / (2 * n))) \in O(\lambda n. 1 / n^{\wedge} 2)$
using *harm-expansion-bigo[of 0]* **by** (*simp add: power2-eq-square*)

lemma *harm-expansion-bigo-simple2*:

$(\lambda n. \text{harm } n - (\ln \ n + \text{euler-mascheroni})) \in O(\lambda n. 1 / n)$

proof –

have $(\lambda n. \text{harm } n - (\ln \ n + \text{euler-mascheroni} + 1 / (2 * n)) + 1 / (2 * n)) \in O(\lambda n. 1 / n)$

proof (*rule sum-in-bigo*)

have $(\lambda n. \text{harm } n - (\ln \ n + \text{euler-mascheroni} + 1 / (2 * n))) \in O(\lambda n. 1 / \text{real } n^{\wedge} 2)$

using *harm-expansion-bigo-simple1* **by** *simp*

also have $(\lambda n. 1 / \text{real } n^{\wedge} 2) \in O(\lambda n. 1 / \text{real } n)$

by (*rule landau-o.big.compose[OF - filterlim-real-sequentially]*) *simp-all*

finally show $(\lambda n. \text{harm } n - (\ln \ n + \text{euler-mascheroni} + 1 / (2 * n))) \in O(\lambda n. 1 / n)$ **by** *simp*

qed *simp-all*

thus *?thesis* **by** (*simp add: algebra-simps*)
qed

lemma *harm-expansion-bigo-simple'*:
harm = o ($\lambda n. \ln n + \text{euler-mascheroni} + 1 / (2 * n)$) *+ o* ($O(\lambda n. 1 / n ^ 2)$)
using *harm-expansion-bigo-simple1*
by (*subst set-minus-plus [symmetric]*) (*simp-all add: fun-diff-def*)

2.3 Asymptotic expansion of the sum of inverse squares

Similarly to before, we show

$$\sum_{i=1}^n \frac{1}{i^2} = \frac{\pi^2}{6} - \frac{1}{n} + \frac{1}{2n^2} - \sum_{i=1}^m B_{2i} n^{-2i-1} + O(n^{-2m-3})$$

context

fixes $R :: \text{nat} \Rightarrow \text{nat} \Rightarrow \text{real}$

defines $R \equiv (\lambda N n. \text{EM-remainder } (2*N+1) (\lambda x. -\text{fact } (2*N+2) / x ^ (2*N+3))$
(int n))

begin

lemma *sum-inverse-squares-remainder-bigo*:

assumes $N > 0$

shows $R N \in O(\lambda n. 1 / \text{real } n ^ (2 * N + 2))$

proof –

from *sum-inverse-squares-remainder-bound[OF assms]*

obtain C

where $\forall n \geq 1. \text{norm } (\text{EM-remainder } (2 * N + 1) (\lambda x. -\text{fact } (2 * N + 2) /$
 $x ^ (2 * N + 3)) (\text{int } n))$

$\leq C / \text{real } n ^ (2 * N + 2) ..$

thus *?thesis*

by (*intro bigoI[of - C] eventually-mono[OF eventually-ge-at-top[of 1]]*) (*auto simp: R-def*)

qed

lemma *sum-inverse-squares-expansion-bigo*:

fixes $N :: \text{nat}$

defines $T \equiv \lambda n. \pi ^ 2 / 6 - 1 / n + 1 / (2*n ^ 2) -$

$(\sum_{i=1..N. \text{bernoulli } (2*i) / (n ^ (2*i+1)))$

defines $S \equiv (\lambda n. \text{bernoulli } (2*(\text{Suc } N)) / (\text{real } n ^ (2*N+3)))$

shows $(\lambda n. (\sum_{i=1..n. 1 / \text{real } i ^ 2) - T n) \in O(\lambda n. 1 / \text{real } n ^ (2 * N +$
 $3))$

proof –

have $\exists: \exists = \text{Suc } (\text{Suc } (\text{Suc } 0))$ **by** *simp*

have $(\lambda n. (\sum_{i=1..n. 1 / \text{real } i ^ 2) - T n) \in \Theta(\lambda n. -S n - R (\text{Suc } N) n)$

unfolding *R-def*

by (*intro bigthetaI-cong eventually-mono[OF eventually-gt-at-top[of 0::nat]]*)

(auto simp: T-def sum-inverse-squares-expansion[of - Suc N] S-def \exists

simp del: One-nat-def)

also have $(\lambda n. -S n - R (Suc N) n) \in O(\lambda n. 1 / \text{real } n^{\wedge}(2 * N + 3))$
proof (*intro sum-in-bigo*)
show $(\lambda x. -S x) \in O(\lambda n. 1 / \text{real } n^{\wedge}(2 * N + 3))$ **unfolding** *S-def*
by (*rule landau-o.big.compose[OF - filterlim-real-sequentially]*) *simp*
have $R (Suc N) \in O(\lambda n. 1 / \text{real } n^{\wedge}(2 * Suc N + 2))$
by (*rule sum-inverse-squares-remainder-bigo*) *simp-all*
also have $2 * Suc N + 2 = 2 * N + 4$ **by** *simp*
also have $(\lambda n. 1 / \text{real } n^{\wedge}(2 * N + 4)) \in O(\lambda n. 1 / \text{real } n^{\wedge}(2 * N + 3))$
by (*rule landau-o.big.compose[OF - filterlim-real-sequentially]*) *simp*
finally show $R (Suc N) \in \dots$
qed
finally show *?thesis* .
qed

lemma *sum-inverse-squares-expansion-bigo-simple*:
 $(\lambda n. (\sum i=1..n. 1 / \text{real } i^{\wedge} 2) - (\pi^{\wedge} 2 / 6 - 1 / n + 1 / (2 * n^{\wedge} 2))) \in O(\lambda n. 1 / n^{\wedge} 3)$
using *sum-inverse-squares-expansion-bigo[of 0]* **by** (*simp add: power2-eq-square*)

lemma *sum-inverse-squares-expansion-bigo-simple'*:
 $(\lambda n. (\sum i=1..n. 1 / \text{real } i^{\wedge} 2)) = o(\lambda n. \pi^{\wedge} 2 / 6 - 1 / n + 1 / (2 * n^{\wedge} 2))$
 $+ o O(\lambda n. 1 / n^{\wedge} 3)$
using *sum-inverse-squares-expansion-bigo-simple*
by (*subst set-minus-plus [symmetric]*) (*simp-all add: fun-diff-def*)

end

end

References

- [1] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover, New York, tenth printing edition, 1964.
- [2] T. M. Apostol. *An Elementary View of Euler's Summation Formula*. *The American Mathematical Monthly*, 106(5):409–418, 1999.
- [3] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Addison–Wesley, Boston, MA, USA, 2nd edition, 1994.