The Euler–MacLaurin summation formula

Manuel Eberl

September 13, 2023

Abstract

The Euler–MacLaurin formula relates the value of a discrete sum $\sum_{i=c}^{b} f(i)$ 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 (λx. 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 . . . = 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 . . . = 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 ∉ R≤0 x ≥ 0
  shows z + of-real x ≠ 0
  using assms by (auto simp: add-eq-0-iff2)

lemma negligible-atLeastAtMostI: b ≤ a ⇒ negligible {a..(b::real)}
  by (cases b < a) auto

lemma integrable-on-negligible:
negligible $A \implies (f : 'n :: euclidean-space \Rightarrow 'a :: banach) \text{integrable-on } A$

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 equality1 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 $x : x \geq \text{real-of-int } a \text{ and } x < \text{real-of-int } b$ by simp-all

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

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

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

by simp

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' :: nat ⇒ (real ⇒ 'a :: banach) ⇒ real ⇒ real ⇒ 'a

where

$\text{EM-remainder'} n f a b = ((-1) ^ Suc n / \text{fact } n) \ast_R \text{integral } \{a..b\} (\lambda t. \text{pbernpoly } n t \ast_R f t)$

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 :: nat ⇒ (real ⇒ 'a :: banach) ⇒ int ⇒ bool

where

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

definition EM-remainder :: nat ⇒ (real ⇒ 'a :: banach) ⇒ int ⇒ 'a

where

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

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

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

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

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

lemma integrable-EM-remainder'\(\)
fixes \(a b :: \text{real} \) and \(f :: \text{real} \Rightarrow 'a :: \text{banach}\)
assumes \(\text{continuous-on} \ \{a..b\} \ f\)
shows \((\lambda t. \text{pbernpoly} n t \ast_R f t) \text{integrable-on} \ \{a..b\}\)
proof \((\text{cases } [a] \leq [b])\)
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 int_disj_un) ?)
(auto simp: int_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_alt_def 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: pbernopoly-def frac-def)
qed
qed simp-all

lemma EM-remainder'\ binder linear:
  assumes bounded-linear h
  assumes continuous-on {a..b} f
  shows  EM-remainder' n (λx. h (f x)) a b = h (EM-remainder' n f a b)
proof −
  have integral {a..b} (λt. pbernopoly n t *R h (f t)) =
    integral {a..b} (λt. h (pbernopoly n t *R f t)) using assms
    by (simp add: linear-simps)
  also have . . . = h (integral {a..b} (λt. pbernopoly n t *R f t))
    by (subst integral-linear [OF assms (1), symmetric])
  finally show ?thesis using assms by (simp add: EM-remainder'\ binder-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-remainder-converges n (λx. of-real (f x)) a
proof −
  from assms obtain L where L: ((λb. EM-remainder' n f (of-int a..) (of-int b)) −−→ L) at-top
    by (auto simp: EM-remainder-converges-def)
  have ((λb. EM-remainder' n (λx. of-real (f x)) (of-int a) (of-int b)) −−→ of-real L) at-top
    proof (rule Lim-transform-eventually)
      show eventually (λb. of-real (EM-remainder' n f (of-int a) (of-int b)) =
        EM-remainder' n (λx. of-real (f x)) (of-int a) (of-int b)) at-top
        using eventually-ge-at-top[of a]
      by eventually-elim
        (intro EM-remainder'\ binder-bounded-linear [OF bounded-linear-of-real, symmetric]
          continuous-on-subset[OF assms (2), auto])
    qed (intro tendssto-intros L)
thus ?thesis unfolding EM-remainder-converges-def ..
qed

lemma EM-remainder-converges-of-real-iff:
  fixes f :: real ⇒ real
  assumes continuous-on {of-int a..} f
  shows  EM-remainder-converges n (λx. of-real (f x) :: 'a :: {banach, real-normed-algebra-1, real-inner}) a −→ EM-remainder-converges n f a
proof
  assume EM-remainder-converges n (λx. of-real (f x) :: 'a) a
  then obtain L :: 'a
where \( L : (\lambda b. EM\text{-}remainder' n (\lambda x. of\text{-}real (f x)) (of\text{-}int a) (of\text{-}int b)) \rightarrow L \) at-top
by (auto simp: EM\text{-}remainder\text{-}converges-def)
have 
  \((\lambda b. EM\text{-}remainder' n f (of\text{-}int a) (of\text{-}int b)) \rightarrow L \cdot 1\) at-top
proof (rule Lim\text{-}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\text{-}remainder'\text{-}bounded\text{-}linear [OF bounded\text{-}linear\text{-}of\text{-}real],
auto intro: continuous-on\text{-}subset[OF assms])
qed (intro tendsto\text{-}intros L)
thus EM\text{-}remainder\text{-}converges n f a unfolding EM\text{-}remainder\text{-}converges-def ..
qed (intro EM\text{-}remainder\text{-}converges\text{-}of\text{-}real assms)

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

lemma EM\text{-}remainder'\text{-}cong:
  assumes \( \forall x. x \in \{a..b\} \Rightarrow f x = g x n = n' a = a' b = b' \)
  shows \( EM\text{-}remainder' n f a b = EM\text{-}remainder' n' g a' b' \)
proof –
  have \( \text{integral } \{a..b\} (\lambda t. \text{pbernpoly n t} \ast_R f t) = \text{integral } \{a'..b'\} (\lambda t. \text{pbernpoly} \)
lemma \( \text{EM-remainder-converges-cong} \):
assumes \( \forall x. x \geq \text{of-int}\ a \implies f x = g x n = n' a = a' \)
shows \( \text{EM-remainder-converges}\ n f a = \text{EM-remainder-converges}\ n' g a' \)
proof
  using assms by (intro \( \text{EM-remainder-converges-cong} \)) auto
qed

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

lemma \( \text{EM-remainder-converges-cnj} \):
assumes \( \text{continuous-on} \{ \text{of-int}\ a .. \}\ f \)
shows \( \text{EM-remainder-converges}\ n \ (\lambda x. \text{cnj}\ (f x)) a \)
proof
  interpret \( \text{bounded-linear}\ \text{cnj} \) by (rule \( \text{bounded-linear-cnj} \))
  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 \( \text{EM-remainder-converges-def} \) 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 \( \text{EM-remainder}'-bounded-linear [OF \text{bounded-linear-cnj}] \))
    (rule continuous-on-subset [OF assms(1)], auto)
  finally have \( L': (\ldots \longrightarrow \text{cnj}\ L)\) at-top .
  thus \( \text{EM-remainder-converges}\ n (\lambda x. \text{cnj}\ (f x)) a \)
    by (auto simp: \( \text{EM-remainder-converges-def} \))
qed

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

qed (intro \( \text{EM-remainder-converges-cnj} \) assms)

lemma \( \text{EM-remainder-cnj} \):
  assumes \( \text{continuous-on } \{a..\} \ f \)
  shows \( \text{EM-remainder } n \ (\lambda x. \text{cnj (f x)}) \ a = \text{cnj (EM-remainder } n \ f \ a) \)

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

lemma \( \text{EM-remainder }''\)-combine:
  fixes \( f :: \text{real} \Rightarrow \text{real} \Rightarrow \text{'a} \Rightarrow \text{'b} \)
  assumes \( \text{continuous-intros} : \text{continuous-on } \{a..c\} \ f \)
  assumes \( a \leq b \ b \leq c \)
  shows \( \text{EM-remainder }'' n \ f a b + \text{EM-remainder }'' n \ f b c = \text{EM-remainder }'' n \ f \ a \ c \)

proof
  have \( \text{integral } \{a..b\} (\lambda t. \text{pbernpoly } n \ t *_{\text{R}} f t) + \text{integral } \{b..c\} (\lambda t. \text{pbernpoly } n \ t *_{\text{R}} f t) = \)
    \( \text{integral } \{a..c\} (\lambda t. \text{pbernpoly } n \ t *_{\text{R}} f t) \)
    by (intro \( \text{Henstock-Kurzweil-Integration.integral-combine assms integrable-EM-remainder''} \))
  from this [symmetric] show ?thesis by (simp add: \( \text{EM-remainder }''\)-def algebra-simps)
qed

lemma uniformly-convergent-EM-remainder'::
  fixes \( f :: \{a \Rightarrow \text{real} \Rightarrow \text{'a} :: \{\text{banach}, \text{real-normed-algebra} \} \)
  assumes \( \text{deriv} : \lambda y. a \leq y \implies (G \text{ has-real-derivative } g \ y) \ (at \ y \ \text{within } \{a..\}) \)
  assumes \( \text{integrable} : \lambda a' \ y. y \in A \implies a \leq a' \implies a' \leq b \implies (\lambda t. \text{pbernpoly } n \ t *_{\text{R}} f y t) \text{integrable-on } \{a'..b\} \)
assumes conv: convergent (λy. G (real y))
assumes bound: eventually (λx. ∀y∈A. norm (f y x) ≤ g x) at-top
shows uniformly-convergent-on A (λb s. EM-remainder' n (f s) a b)
proof |
  interpret bounded-linear λx::'b. (λ y. Suc n / fact n) *R x
  by (rule bounded-linear-scaleR-right)

from bounded-pbernopoly obtain C where C: \∀ x. norm (pbernopoly n x) ≤ C by auto
from C[of 0] have [simp]; C ≥ 0 by simp

show ?thesis unfolding EM-remainder'-'def
proof (intro uniformly-convergent-on uniformly-convergent-improper-integral') |
  fix x assume x ≥ a
  thus (λx. 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 ∈ A a ≤ a' a' ≤ b
  thus (λt. pbernopoly n t *R f y t) integrable-on {a'.b} by (rule integrable)
next |
  from conv obtain L where (λy. G (real y)) ----→ L by (auto simp: convergent-def)
  from tendsto-mult[OF tendsto-const[of C] this] |
  show convergent (λy. C * G (real y)) by (intro DERIV-cmult deriv)
next |
  show ∀x in at-top. ∀y∈A. norm (pbernopoly n x *R f y x) ≤ 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 ⇒ real ⇒ 'b :: {banach,real-normed-algebra}
  assumes deriv: \∀ y. a ≤ y ⇒ (G has-real-derivative g y) (at y within {a..})
  assumes integrable: \∀ a' b y. y ∈ A ⇒ a ≤ a' ⇒ a' ≤ b ⇒ (λt. pbernopoly n t *R f y t) integrable-on {a'.b}
  assumes conv: convergent (λy. G (real y))
  assumes bound: eventually (λx. ∀y∈A. norm (f y x) ≤ g x) at-top
  shows uniform-limit A (λb s. EM-remainder' n (f s) a b)
    (λs. EM-remainder n (f s) a) sequentially
proof |
  have *: uniformly-convergent-on A (λb s. EM-remainder' n (f s) a b)
    by (rule uniformly-convergent-EM-remainder''[OF assms])
  also have ?this ⟷ ?thesis |
    unfolding uniformly-convergent-uniform-limit-iff
  proof (intro uniform-limit-cong refl always-eventually allI ballI)
    fix s assume s ∈ A
with * have **: convergent ($\lambda b$. EM-remainder' $n$ $(f\ s)\ a\ b$) by (rule uniformly-convergent-imp-convergent)
show $\lim (\lambda b$. EM-remainder' $n$ $(f\ s)\ a\ b$) = EM-remainder $n$ $(f\ s)\ a$
proof (rule sym, rule EM-remainder-eqI)
  have ($($\lambda z$. EM-remainder' $n$ $(f\ s)$ (real-of-int $a$) (real $x$))$ $\longrightarrow$ $\lim (\lambda z$. EM-remainder' $n$ $(f\ s)$ (real-of-int $a$) (real $x$))) at-top
  (is $(\ldots$ $\longrightarrow$ $?L)$ -) using ** unfolding convergent-LIMSEQ-iff by blast
hence (($\lambda z$. EM-remainder' $n$ $(f\ s)$ (real-of-int $a$) (real (nat $x$))) $\longrightarrow$ $?L$) at-top
by (rule filterlim-compose) (fact filterlim-nat-sequentially)
thus (($\lambda z$. EM-remainder' $n$ $(f\ s)$ (real-of-int $a$) (real-of-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 $\ldots$ .
qed

lemma tendsto-EM-remainder:
fixes $f :$ :: $\{\text{banach,real-normed-algebra}\}$
assumes deriv: $\forall y.\ a \leq y \implies (G\ \text{has-real-derivative}\ g\ y)$ (at $y$ within $\{a..\}$)
assumes integrable: $\forall a'\ b.\ a \leq a' \implies a' \leq b \implies ($($\lambda t$. bernpoly $n$ $t$ $*$ $f$ $t$)\ integrable-on\ $\{a'..b\}$
assumes conv: convergent ($\lambda y.\ G\ (\text{real}\ y)$)
assumes bound: eventually ($($\lambda x$. norm $(f\ x)$ $\leq$ $g\ x$)$)$ at-top
shows filterlim ($($\lambda b$. EM-remainder' $n$ $f$ $a$ $b$)$ (nhds (EM-remainder $n$ $f$ $a$))$ sequentially
proof -
  have uniform-limit $\{()\}$ ($\lambda b.\ EM-remainder'\ n\ f\ a\ b$)
  ($\lambda s.\ EM-remainder\ n\ f\ a$) sequentially
  using assms by (intro uniform-limit-EM-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-remainder $n$ ($\lambda x.\ 0$) $a = 0$
by (rule EM-remainder-eqI) (simp add: EM-remainder'-def)

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

proof (intro holomorphic-intros)

have holo: \( (\lambda z. \text{integral} \ (\text{cbox} \ b \ c) \ (\lambda t. \text{of-real} \ (\text{bernpoly} \ n \ (t - e)) \ast f \ z \ t)) \)
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 \( ((\lambda z. \text{complex-of-real} \ (\text{bernpoly} \ n \ (t - e)) \ast f \ z \ t)) \)
has-field-derivative \( \text{complex-of-real} \ (\text{bernpoly} \ n \ (t - e)) \ast f' \ z \ t) \) (at \( z \) within \( U \))
using that by (intro DERIV-cmult deriv) auto

next

fix \( z \) assume \( z \in \ U \)
thus \( (\lambda t. \text{complex-of-real} \ (\text{bernpoly} \ n \ (t - e)) \ast f \ z \ t) \)
integrable-on \( \text{cbox} \ b \ c \)
using that int \( _{\text{of} b c z} \) 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 \( (U \times \text{cbox} \ b \ c) \) \((\lambda(z,t). \text{complex-of-real} \ (\text{bernpoly} \ n \ (t - e)) \ast f' \ z \ t)\)
by (auto simp: case-prod-unfold intro: continuous-intros)

qed fact

consider \( a > x \ |\ a \leq x \ floor x \leq a \ |\ a \leq x \ floor x > a \) by force

hence \((\lambda z. \text{integral} \ (\text{cbox} \ a \ x) \ (\lambda t. \text{of-real} \ (\text{bernpoly} \ n \ t) \ast f \ z \ t)) \) holomorphic-on \( U \)

(is \( \text{?f a x holomorphic-on } -\))

proof cases

case 2

have \( (\lambda z. \text{integral} \ (\text{cbox} \ a \ x) \ (\lambda t. \text{of-real} \ (\text{bernpoly} \ n \ (t - \text{of-int} \ [x])) \ast f \ z \ t)) \)
holomorphic-on \( U \)
by (intro holo) auto

also have \( (\lambda z. \text{integral} \ (\text{cbox} \ a \ x) \ (\lambda t. \text{of-real} \ (\text{bernpoly} \ n \ (t - \text{of-int} \ [x])) \ast f \ z \ t)) = \text{?f a x} \)

proof (intro ext integral-cong, goal-cases)

case \((I \ z \ t)\)

hence \( t \geq a \ t \leq x \) by auto

hence \( floor t = floor x \) using 2 by linarith
thus \( \text{?case by \( (simp add: \text{bernpoly-def frac-def})\)}\)

Qed

finally show \( \text{?thesis} . \)

next

case 3

define \( N :: \text{int set} \) where \( N = \{\lfloor a\rfloor .. < \lfloor x\rfloor\} \)

define \( A \) where \( A = \text{insert} \ \{a..\text{of-int} \ [a]\} \ (\text{insert} \ \{\text{of-int} \ [x]..x\} \ ((\lambda n. \ \{\text{of-int} \ n..\text{of-int} \ n + 1\}) \ '(N)'))\)

\{

fix \( X \) assume \( X \in A \)
then consider \( X = \{a..\text{of-int} \ [a]\} \ |\ X = \{\text{of-int} \ [x]..x\} \ |

n \ where \ X = \{\text{of-int} \ n..\text{of-int} \ n + 1\} \ n \in N \) by (auto simp: \( A\)-def)

\}
\begin{itemize}
\item \textbf{note} \(A\)-cases = this
\end{itemize}

\textbf{have division:} A division-of \(\{a..x\}\)

\textbf{proof} (rule division-off)
\begin{itemize}
\item show \(\text{finite } A\) by (auto simp: A-def N-def)
\item fix \(K\) assume \(K \in A\)
\item from \(3\) have \(\text{of-int } [a] \leq x\)
\item using \texttt{ceiling-le}\(\text{[of a floor } x\text{]}\) by linarith
\item moreover from \(3\) have \(\text{of-int } [x] \geq a\) by linarith
\item ultimately show \(K \subseteq \{a..x\}\) using \(K 3\) by (auto simp: A-def N-def)
\end{itemize}

\texttt{linarith+}
\begin{itemize}
\item from \(K\) show \(K \neq \{\}\) and \(\exists a\ b.\ K = cbox a b\) by (auto simp: A-def)
\item next
\item fix \(K1 K2\) assume \(K1 \in A\ K2 \in A\ K1 \neq K2\)
\item have \(F1: \text{interior } \{a..[a]\} \cap \text{interior } \{[x]..x\} = \{\}\) using \(3\) ceiling-le\(\text{[of a floor } x\text{]}\)
\item by (auto simp: min-def max-def)
\item hence \(F2: \text{interior } \{[x]..x\} \cap \text{interior } \{a..[a]\} = \{\}\) by simp
\item have \(F3: \text{interior } \{a..[a]\} \cap \text{interior } \{\text{of-int } n..\text{of-int } n+1\} = \{\}\)
\item \text{interior } \{[x]..x\} \cap \text{interior } \{\text{of-int } n..\text{of-int } n+1\} = \{\}
\item \text{interior } \{\text{of-int } n..\text{of-int } n+1\} \cap \text{interior } \{[x]..x\} = \{\}\) if \(n \in N\) for \(n\)
\item using \(3\) ceiling-le\(\text{[of a floor } x\text{]}\) that by (auto simp: min-def max-def N-def)
\item have \(F4: \text{interior } \{\text{real-of-int } n..\text{of-int } n+1\} \cap \text{interior } \{\text{of-int } m..\text{of-int } m+1\} = \{\}\)
\item if \(\{\text{real-of-int } n..\text{of-int } n+1\} \neq \{\text{of-int } m..\text{of-int } m+1\}\) for \(m\ n\)
\item proof –
\item from that have \(n \neq m\) by auto
\item thus \(?\text{thesis by simp}\)
\item qed
\item from \(F1 F2 F3 F4 K\) show \(\text{interior } K1 \cap \text{interior } K2 = \{\}\)
\item by (elim \(A\)-cases) (simp-all only: not-False-eq-True)
\item next
\item show \(\bigcup A = \{a..x\}\)
\item proof (cases \([a] = [x]\))
\item case True
\item thus \(?\text{thesis using } 3\) by (auto simp: A-def N-def intro: order.trans) \texttt{linarith+}
\item next
\item case False
\item with \(3\) have \(\ast: [a] < [x]\) by linarith
\item have \(\bigcup A = \{a..\text{of-int } [a]\} \cup (\bigcup n \in N. \{\text{of-int } n..\text{of-int } (n+1)\}) \cup \{\text{of-int } [x]..x\}\)
\item by (simp add: A-def Un-ac)
\item also have \((\bigcup n \in N. \{\text{of-int } n..\text{of-int } (n+1)\}) = \{\text{of-int } [a]..\text{real-of-int } [x]\}\)
\item using \(\ast\) unfolding \(N\)-def by (intro Union-atLeastAtMost-real-of-int)
\item also have \(\{a..\text{of-int } [a]\} \cup \ldots = \{a..\text{real-of-int } [x]\}\)
\item using \(3\) by (intro ivl-disj-an) auto
\item also have \(\ldots \cup \{\text{of-int } [x]..x\} = \{a..x\}\)
\item using \(3\) by (intro ivl-disj-an) auto
\end{itemize}
finally show \(?thesis\).

qed

\[\text{have } (\lambda z. \sum_{x \in A} \text{integral } X (\lambda t. \text{of-real } (\text{bernpoly } n (t - \lfloor \text{Inf } X \rfloor)) * f z t))\]

\[\text{holomorphic-on } U\]

\[\text{proof } (\text{intro } \text{holomorphic-on-sum holo, goal-cases})\]

\[\text{case } (1 X)\]

\[\text{from } 1 \text{ and } \text{division have subset: } X \subseteq \{a..x\} \text{ by } (\text{auto simp: division-of-def})\]

\[\text{from } 1 \text{ obtain } b c \text{ where } [\text{simp}]: X = cbox b c b \leq c \text{ by } (\text{auto simp: A-def})\]

\[\text{from subset have } b \geq a c \leq x \text{ by } \text{auto}\]

\[\text{hence } (\lambda z. \text{integral } (cbox b c) (\lambda t. \text{of-real } (\text{bernpoly } n (t - \lfloor \text{Inf } \{b..c\}\rfloor)) * f x t))\]

\[\text{holomorphic-on } U \text{ by } (\text{intro holo}) \text{ auto}\]

\[\text{thus } \text{?case by simp}\]

\[\text{qed}\]

\[\text{also have } \text{?this } \longleftrightarrow (\lambda z. \text{integral } \{a..x\} (\lambda t. \text{of-real } (\text{bernpoly } n t) * f z t))\]

\[\text{holomorphic-on } U\]

\[\text{proof } (\text{intro } \text{holomorphic-cong refl, goal-cases})\]

\[\text{case } (1 z)\]

\[\text{have } ((\lambda t. \text{of-real } (\text{bernpoly } n t) * f z t) \text{ has-integral}\]

\[\sum_{x \in A} \text{integral } X (\lambda t. \text{of-real } (\text{bernpoly } n (t - \lfloor \text{Inf } X \rfloor)) * f z t))\]

\[\{a..x\}\]

\[\text{using } \text{division}\]

\[\text{proof } (\text{rule has-integral-combine-division})\]

\[\text{fix } X \text{ assume } X \in A\]

\[\text{then obtain } b c \text{ where } X' = \{b..c\} \text{ by } (\text{elim A-cases auto})\]

\[\text{from } X \text{ and } \text{division have } X \subseteq \{a..x\} \text{ by } (\text{auto simp: division-of-def})\]

\[\text{with } X' \text{ have } bc: b \geq a c \leq x \text{ by } \text{auto}\]

\[\text{have } ((\lambda t. \text{of-real } (\text{bernpoly } n (\lfloor \text{of-int } \text{Inf } X \rfloor)) * f z t) \text{ has-integral}\]

\[\text{integral } X (\lambda t. \text{of-real } (\text{bernpoly } n (t - \lfloor \text{of-int } \text{Inf } X \rfloor)) * f z t))\]

\[X\]

\[\text{unfolding } X' \text{ using } z \in U, bc \text{ by } (\text{intro integrable-integral-int})\]

\[\text{also have } \text{?this } \longleftrightarrow ((\lambda t. \text{of-real } (\text{bernpoly } n t) * f z t) \text{ has-integral}\]

\[\text{integral } X (\lambda t. \text{of-real } (\text{bernpoly } n (t - \lfloor \text{of-int } \text{Inf } X \rfloor)) * f z t))\]

\[X\]

\[\text{proof } (\text{rule has-integral-spike-eq[of } \{\text{Sup } X\}, \text{ goal-cases})\]

\[\text{case } (2 t)\]

\[\text{note } t = \text{this}\]

\[\text{from } (X \in A, \text{ have } [t] = \lfloor \text{Inf } X \rfloor)\]

\[\text{proof } (\text{cases rule: A-cases } [\text{consumes } 1])\]

\[\text{case } 1\]

\[\text{with } t \text{ show } \text{?thesis}\]

\[\text{by } (\text{intro floor-unique}) (\text{auto simp: ceiling-altdef split: if-splits, linarith+})?\]

\[\text{next}\]

\[\text{case } 2\]

\[\text{with } t \text{ show } \text{?thesis}\]

\[\text{by } (\text{intro floor-unique}) (\text{auto simp: ceiling-altdef split: if-splits, linarith+})?\]
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: pbernopoly-def frac-def)
  qed auto
finally show . . .
  qed
thus ?thesis by simp
qed auto
thus (\lambda z. integral \{a..x\} (\lambda t. of-real (pbernopoly n t) * f z t)) holomorphic-on U
by simp
qed

lemma
assumes deriv: \( \forall y. a \leq y \implies (G \text{ has-real-derivative } g y) \) (at \( y \) within \{a..\})
assumes deriv':
  \( \forall x. z \in U \implies x \geq a \implies t \in \{a..x\} \implies 
  ((\lambda z. f z t) \text{ has-field-derivative } f' z t) \) (at \( z \) within \( U \))
assumes cont: continuous-on \((U \times \{\text{of-int } a..\})\) \((\lambda(z, t). f' z t)\)
assumes int: \( \forall b c y. a \leq b \implies y \in U \implies 
  (\lambda t. \text{pbernopoly n t} * \text{of-real } (b z t)) \text{ integrable-on } \{b..c\}\)
assumes int': \( \forall a' b y. y \in U \implies a \leq a' \implies a' \leq b \implies 
  (\lambda t. \text{pbernopoly n t} * \text{of-real } (a'..b)) \text{ integrable-on } \{a'..b\}\)
assumes conv: convergent \((\lambda y. G (\text{real } y))\)
assumes bound: eventually \((\lambda y. \forall y \in U. \text{norm } (f y x) \leq g x)\) at-top
assumes open U
shows analytic-EM-remainder: \((\lambda s::\text{complex}. \text{EM-remainder } n (f s) a) \text{ analytic-on } U\)
  and holomorphic-EM-remainder: \((\lambda s::\text{complex}. \text{EM-remainder } n (f s) a) \text{ holomorphic-on } U\)
proof –
show \((\lambda s::\text{complex}. \text{EM-remainder } n (f s) a) \text{ analytic-on } U\)
unfolding analytic-on-def
proof
  fix z assume z \in U
  from \( \{z \in U\} \) and \( \langle \text{open } U \rangle \) obtain \( \varepsilon \) where \( \varepsilon: \varepsilon > 0 \) ball \( z \in U \)
  by (auto simp: open-contains-ball)
  have \((\lambda s. \text{EM-remainder } n (f s) a) \text{ holomorphic-on ball } z \in U\)
  proof (rule holomorphic-uniform-sequence)
    fix x :: nat
    show \((\lambda s. \text{EM-remainder'} n (f s) a x) \text{ holomorphic-on ball } z \in U\)
    proof (rule holomorphic-EM-remainder', goal-cases)
      fix s t assume s \in ball \( z \in U \) \( \{\text{real-of-int } a..\text{real } x\} \)
      thus \((\lambda z. f z t) \text{ has-field-derivative } f' s t) \text{ (at } s \text{ within ball } z \in U\)
        using \( \varepsilon \) by (intro DERIV-subset[OF deriv[of - x]])
  qed
  qed
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 \( f, f' :: \text{real} \Rightarrow 'a :: \text{banach} \)
  fixes \( a, b :: \text{int} \text{ and } I, S :: 'a \)
  fixes \( Y :: \text{real set} \)
  assumes \( a \leq b \)
  assumes fin: \( \text{finite } Y \)
  assumes cont: \( \text{continuous-on } \{\text{real-of-int } a..\text{real-of-int } b\} f \)
  assumes deriv [derivative-intros]:
  \( \forall x :: \text{real. } x \in \{a..b\} - Y \Rightarrow (f \text{ has-vector-derivative } f' x) \text{ (at } x) \)
  defines \( S \text{-def: } S \equiv (\sum i \in \{a..<b\}. f \, i) \text{ and } I \text{-def: } I \equiv \text{integral } \{a..b\} f \)
begin

lemma

next
  case (2 b c s e)
  with \( \varepsilon \) have \( s \in U \) by blast
  with 2 show \(?case using \varepsilon \text{ int[of } b s e c\} by (cases } a \leq x) \text{ auto} \)
next
  from cont show continuous-on \( \{\text{ball } z \varepsilon \times \{\text{real-of-int } a..\text{real } x\}\}) (\lambda z, t). f' z t) \)
  by (rule continuous-on-subset) (insert \( \varepsilon\), auto)
qed (auto)

next
  fix \( s \) assume \( s \in \text{ball } z \varepsilon \)
  have open \( \{\text{ball } z \varepsilon\} \) by simp
  with \( s \) obtain \( \delta \) where \( \delta > 0 \) \( \text{ball } s \delta \subseteq \text{ball } z \varepsilon\)
  unfolding open-contains-cball by blast
  moreover have bound': \( \text{eventually } (\lambda x. \forall y \in \text{ball } s \delta. \text{norm } (f y x) \leq g x) \)
  at-top
  by (intro eventually-mono [OF bound]) (insert \( \delta, \varepsilon\), auto)
  have uniform-limit \( \{\text{ball } s \delta\} (\lambda x s. \text{EM-remainder'} n (f s) (\text{real-of-int } a) (\text{real } x)) \)
  (\( \varepsilon \), \( s \), auto)
  ultimately show \( \exists \delta > 0. \text{ball } s \delta \subseteq \text{ball } z \varepsilon \wedge \text{uniform-limit } \{\text{ball } s \delta\} (\lambda x s. \text{EM-remainder'} n (f s) (\text{real-of-int } a) (\text{real } x)) \)
  (\( \lambda s. \text{EM-remainder } n (f s) a \) sequentially)
  by (rule uniform-limit-EM-remainder [OF deriv int'] conv bound') (insert \( \delta, \varepsilon\), auto)
  qed auto
with \( \varepsilon \) show \( \exists \varepsilon > 0. (\lambda s. \text{EM-remainder } n (f s) a) \text{ holomorphic-on } \text{ball } z \varepsilon \)
by blast
qed

thus \( (\lambda s :: \text{complex. } \text{EM-remainder } n (f s) a) \text{ holomorphic-on } U \)
  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.
**diff-sum-integral-has-integral-int:****

\((\lambda t. (\text{frac } t - 1/2) * \text{r} f' t) \text{ has-integral } (S - I - (f b - f a) / R 2)) [a..b] \)**

**proof (cases a = b)**

**case True**

thus \(\exists \text{thesis by (simp-all add: S-def I-def has-integral-refl)}\)

**next**

**case False**

with \(a \leq b\) have \(a < b\) by simp

let \(\mathcal{A} = (\lambda n. \{\text{real-of-int } n..\text{real-of-int } (n+1)\}) \cdot \{a..<b\}\)

**have division: \(\mathcal{A} \text{ division-of } \{\text{of-int } a..\text{of-int } b\}\)**

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 \((\lambda t. (\text{frac } t - 1/2) * R f' t) \text{ has-integral } (\lambda i. \{\text{of-int } i..\text{of-int } (i+1)\})\)**

**if i: \(i \in \{a..<b\}\) 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)\} \text{ for } x\)**

**proof -**

**have \(x \geq \text{of-int } i \leq \text{of-int } (i + 1)\) using that by auto**

**hence floor x = \text{of-int } i \text{ by (subst floor-unique) auto}**

**thus \(\exists \text{thesis by (simp add: frac-def)}\)**

**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 \(\ast: (\lambda x. \text{integral } \{\text{real-of-int } i..x\} f) \text{ has-vector-derivative } f x\) \(\text{at } x \text{ within } \{i..i+1\}\)**

**if x \in \{\text{of-int } i..<\text{of-int } i + 1\} \text{ for } x \text{ 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\} \text{ for } x\)**

**using that i \text{ at-within-interior}[of x \{\text{of-int } i..\text{of-int } (i + 1)\}] \ast[of x] \text{ by simp}**

**hence \(h \text{ has-integral } g \{\text{of-int } i + 1\} = g \{\text{of-int } i\} \text{ has-vector-derivative } f x\) \(\text{at } x\)**

**unfolding \(g\text{-def h-def using that}\)**

**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 \(h \text{ has-integral } d i\) \(\{\text{of-int } i..\text{of-int } (i + 1)\}\).

**qed simp-all**

hence \(\forall I. I \subseteq A \Rightarrow ((\lambda x. (\text{frac } x - 1 / 2) * R f' x) \text{ has-integral } (\lfloor \text{Inf } I \rfloor))\)

**I**

by (auto simp: add-ac)
have ((\lambda x::real. (frac x - 1 / 2) *_R f' x) has-integral (∑I∈?A. d (∑i=..<b. d i)))
(∪ A)
by (intro has-integral-Union * finite-imageI) (force intro!: negligible-atLeastAtMostI pairwise)+
also have ∪ A = {of-int a..of-int b}
  by (intro Union-atLeastAtMost-real-of-int ab)
also have (∑I∈?A. d (∑i=a..<b. d i)) = (∑i=..<b. f (real-of-int i) +
  (∑i=a..<b. f (real-of-int (i + 1))))
(is - = - *_R (?S1 + ?S2) - ?S3)
  by (simp add: d-def algebra-simps sum.distrib sum.subtractf scaleR-sum-right)
also have ?S1 = (∑i = a..b. f (real-of-int i) - f b
unfolding S-def using ab by (subst sum-atLeastAtMost-int-last) auto
also have (∑i = a..b. f (real-of-int i)) = S + f a
unfolding S-def using ab by (subst sum-atLeastAtMost-int-head) auto
also have ?S2 = S unfolding S-def
  by (intro sum.reindex-bij-witness[of - λi. i-1 λi. i+1]) auto
also have (1 / 2) *_R (S + f a - f b + S) =
  (1/2) *_R S + (1/2) *_R S - (f b - f a) / R 2
  by (simp add: algebra-simps)
also have (1/2) *_R S + (1/2) *_R S = S by (simp add: scaleR-add-right [symmetric])
also have ?S3 = (∑I∈?A. integral I f)
  by (subst sum.reindex) (auto simp: inj-on-def add-ac)
also have .. = I unfolding I-def
  by (intro integral-combine-division-topdown [symmetric] division integrable-continuous-real
continuous-intros) simp-all
finally show ?thesis by (simp add: algebra-simps)
qed

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

lemma EM-remainder'-Suc-0: EM-remainder' (Suc 0) f' a b = S - I - (f b - f a) / R 2
  using diff-sum-integral-has-int' by (simp add: has-integral-iff EM-remainder'-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' :: real \Rightarrow 'a :: banach and a b :: int and n :: nat and A :: real set
assumes \( ab \): \( a \leq b \) and \( n \): \( n > 0 \)
assumes \( \text{fin} \): \( \text{finite } A \)
assumes \( \text{cont} \): \( \text{continuous-on } \{ \text{of-int } a \ldots \text{of-int } b \} f \)
assumes \( \text{cont'} \): \( \text{continuous-on } \{ \text{of-int } a \ldots \text{of-int } b \} f' \)
assumes \( \text{deriv} \): \( \forall x. \ x \in \{ \text{of-int } a < \ldots < \text{of-int } b \} \rightarrow A \implies (f \text{ has-vector-derivative } f' \ x) \) (at \( x \))
begin

lemma \( \text{EM-remainder'}-\text{integral-conv-Suc} \):
shows \( \int \{ a..b \} (\lambda t. \text{pbernopoly } n \ t *_R f t) =
(\text{bernoulli } (\text{Suc } n) / \text{real } (\text{Suc } n)) *_R (f b - f a) -
\int \{ a..b \} (\lambda t. \text{pbernopoly } (\text{Suc } n) t *_R f' t) /_R \text{real } (\text{Suc } n) \)
unfolding \( \text{EM-remainder'}-\text{def} \)
proof –
let \( \lambda t. \text{pbernopoly } (\text{Suc } n) (\text{real-of-int } i) / \text{real } (\text{Suc } n)) *_R f (\text{real-of-int } i) \)
define \( T \) where \( T = \int \{ a..b \} (\lambda t. \text{pbernopoly } (\text{Suc } n) t / \text{real } (\text{Suc } n)) *_R f' t) \)
\note [derivative-intros] = \has-\field-derivative-pbernopoly-Suc'
let \( ?A = \text{real-of-int } \{ a..b \} \cup A \)
have \( \int \{ a..b \} (\lambda t. \text{pbernopoly } n \ t *_R f t) \text{ has-integral } (- T + (a b - a h)) \) \{ a..b \}
proof (\text{rule integration-by-parts-interior-strong }[\text{OF bounded-bilinear-scaleR}])
  from \( \text{fin} \) show \( \text{finite } ?A \) by \text{simp}
from \( n > 0 \) \show continuous-on \( \{ \text{of-int } a..\text{of-int } b \} (\lambda t. \text{pbernopoly } (\text{Suc } n) t / \text{real } (\text{Suc } n)) \)
by (\text{intro continuous-intros} \text{auto})
show continuous-on \( \{ \text{of-int } a..\text{of-int } b \} f \) by \text{fact}
show \( (f \text{ has-vector-derivative } f' t) \) (at \( t \)) \if \( t \in \{ \text{of-int } a..<\text{of-int } b \} - ?A \) for \( t \)
  using \( \text{deriv}[f t] \) that by \text{auto}
have \( (\lambda t. \text{pbernopoly } (\text{Suc } n) t *_R f' t) \text{ integrable-on } \{ a..b \} \)
by (\text{intro integrable-EM-remainder'} \text{cont'})
hence \( (\lambda t. (t / \text{real } (\text{Suc } n)) *_R \text{pbernopoly } (\text{Suc } n) t *_R f' t) \text{ integrable-on } \{ a..b \} \)
by (\text{rule integrable-cmul})
also have \( (\lambda t. (t / \text{real } (\text{Suc } n)) *_R \text{pbernopoly } (\text{Suc } n) t *_R f' t) =
(\lambda t. \text{pbernopoly } (\text{Suc } n) t / \text{real } (\text{Suc } n)) *_R f' t) \)
by (\text{rule ext}) (\text{simp add: algebra-simps})
finally show \( ((\lambda t. \text{pbernopoly } (\text{Suc } n) t / \text{real } (\text{Suc } n)) *_R f' t)
\text{ has-integral } ?h b - ?h a - (- T + (a b - a h)) \) \{ a..b \}
using \( \text{integrable-EM-remainder'}[\text{of } a b f' \text{ Suc } n] \)
by (\text{simp add: has-integral-iff } T-\text{def})
\text{qed} (\text{insert } ab n, \text{auto intro!: derivative-eq-intros})
\text{simp: has-real-derivative-iff-has-vector-derivative [symmetric] not-le elim!:
\Ints-cases)}
also have \( ?h b - ?h a = (\text{bernoulli } (\text{Suc } n) / \text{real } (\text{Suc } n)) *_R (f b - f a) \)
using \( n \) by (\text{simp add: algebra-simps bernoulli'-def})
finally have \( \int \{ a..b \} (\lambda t. \text{pbernopoly } n t *_R f t) = \ldots - T \)
by (\text{simp add: has-integral-iff})

19
also have \( T = \text{integral} \{a..b\} (\lambda t. \{1 / \text{real} (Suc n)\} *_{\mathbb{R}} (\text{pbernopoly} (Suc n) \, t)) *_{\mathbb{R}} f' \, t) \)

by \((\text{simp add: } T\text{-def})\)
also have \( \ldots = \text{integral} \{a..b\} (\lambda t. \text{pbernopoly} (Suc n) \, t *_{\mathbb{R}} f' \, t) /_{\mathbb{R}} \text{real} (Suc n) \)

by \((\text{subst integral-cmul}) \text{(simp-all add: divide-simps)}\)
finally show \(!\text{thesis}!\).

qed

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

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 \( \text{fin}: \text{finite} \, A \)
assumes \( \text{cont}: \text{continuous-on} \, \{\text{of-int} \, a..\} \, f \)
assumes \( \text{cont'}: \text{continuous-on} \, \{\text{of-int} \, a..\} \, f' \)
assumes \( \text{lim}: (f \longleftarrow C) \, \text{at-top} \)
assumes \( \text{deriv}: \forall x. x \in \{\text{of-int} \, a<..\} - A \Longrightarrow (f \, \text{has-vector-derivative} \, f' \, x) \, \text{(at} \, x) \)

begin

lemma shows \( EM\text{-remainder-converges-iff-Suc-converges}: \)
\( EM\text{-remainder-converges} \, n \, f \, a \, b \iff EM\text{-remainder-converges} \, (Suc \, n) \, f' \, a \)
and \( EM\text{-remainder-conv-Suc}: \)
\( EM\text{-remainder-converges} \, n \, f \, a \Longrightarrow \)
\( EM\text{-remainder} \, n \, f \, a = \)
\( \{(-1) \, ^{\, \text{Suc} \, n} \, * \, \text{bernoulli} \, (Suc \, n) \, / \, \text{fact} \, (Suc \, n)\} \, *_{\mathbb{R}} \, (C \, - \, f \, a) + \)
\( EM\text{-remainder} \, (Suc \, n) \, f' \, a \)
proof \( (\text{rule iffI})\)

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

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

have \( \text{limit-g} \, (g \longleftarrow G) \, \text{at-top} \, \text{unfolding} \, g\text{-def } G\text{-def} \, \text{by} \, \text{(intro tendsto-intros lim)} \)

have \( \ast: \text{eventually} \, (\lambda x. \, EM\text{-remainder}^{-1} \, n \, f \, (\text{real-of-int} \, a) \, (\text{real-of-int} \, x) = \)
\( g \, x + EM\text{-remainder}^{-1} \, (Suc \, n) \, f' \, (\text{real-of-int} \, a) \, (\text{real-of-int} \, x)) \, \text{at-top} \)
using \( \text{eventually-ge-at-top[of a]} \)
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: ((λb. EM-remainder' n f (real-of-int a) (real-of-int b)) −→ L)
  at-top
    by (auto simp: EM-remainder-converges-def)
  have *: ((λb. EM-remainder' (Suc n) f' (real-of-int a) (real-of-int b)) −→ L
       − G) at-top
    proof (rule Lim-transform-eventually)
      show ∀ 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 ((λx. EM-remainder' n f (real-of-int a) (real-of-int x) − g x) −→ 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: ((λb. EM-remainder' (Suc n) f' (real-of-int a) (real-of-int b)) −→ L)
  at-top
    by (auto simp: EM-remainder-converges-def)
  have *: ((λb. EM-remainder' n f (real-of-int a) (real-of-int b)) −→ G + L)
  at-top
    proof (rule Lim-transform-eventually)
      show ∀ 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 ((λx. g x + EM-remainder' (Suc n) f' (real-of-int a) (real-of-int x)) −→ 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 ..
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 $f :: \text{real} \Rightarrow \text{banach}$
fixes $a :: \text{nat} \Rightarrow \text{real} \Rightarrow \text{a}$
fixes $N :: \text{nat}$
assumes $N \geq 0$
fixes $Y :: \text{real set}$
assumes $\text{finite } Y$
assumes $\text{fs-0 } [\text{simp}]: \text{fs 0 } = f$
assumes $\text{fs-cont } [\text{continuous-intros}]: \forall k. k \leq N \Rightarrow \text{continuous-on } \{\text{real-of-int } a .. \text{real-of-int } b\} (fs k)$
assumes $\text{fs-deriv } [\text{derivative-intros}]: \forall k x. k < N \Rightarrow x \in \{a .. b\} - Y \Rightarrow (fs k \text{ has-vector-derivative } fs (Suc k) x)$

begin

defines $S \equiv (\sum_{i \in \{a < .. b\}} f (\text{of-int } i))$
defines $I \equiv \text{integral } \{\text{of-int } a .. \text{of-int } b\} f$
defines $c' \equiv \lambda k. (\text{bernoulli}^' (Suc k) / \text{fact} (Suc k)) * R (fs k b - fs k a)$
shows $S - I = (\sum_{k < N} c' k) + \text{EM-remainder}' N (fs N) a b$
proof –
define $c :: \text{nat} \Rightarrow 'a$
where $c = (\lambda k. (1 - 1) ^ k * \text{bernoulli} (Suc k) / \text{fact} (Suc k)) * R (fs k b - fs k a)$
have $S - I = (\sum_{k < m} c k) + \text{EM-remainder}' m (fs m) a b$ if $m \geq 1 m \leq N$
for $m$
using that
proof ( induction $m$ rule: dec-induct)
case base
with $ab \text{ fin } fs-\text{cont}[of 0]$ show $\text{case using } fs-\text{deriv}[of 0] N$ unfolding One-nat-def
by ( subst $EM-\text{remainder}'-Suc-0[\text{of - } Y f]$ ) ( simp-all add: algebra-simps S-def I-def c-def)
next
case ( step $n$)
from step.prems have $S - I = (\sum_{k < n} c k) + \text{EM-remainder}' n (fs n) a b$
by ( intro step.IH) simp-all
also have $(\sum_{k < n} c k) = (\sum_{k < Suc n} c k) + ((1 - 1) ^ n * \text{bernoulli} (Suc n) / \text{fact} (Suc n)) * R (fs n b - fs n a))$
(is - = + ?c) by ( simp add: EM-remainder'-Suc-0 c-def)
also have \(\ldots + EM\text{-}remainder' n (fs n) a b = (\sum k < Suc n. c k) + (\exists c + EM\text{-}remainder' n (fs n) a b)\)
  by (simp add: add.assoc)
also from step.prems step.hyps ab fin
  have \(\exists c + EM\text{-}remainder' n (fs n) a b = EM\text{-}remainder' (Suc n) (fs (Suc n)) a b\)
  by (subst EM\text{-}remainder'-cone-Suc [where \(A = Y\)])
    (auto intro: fs-deriv fs-cont)
finally show \(?case .\)
qed

from this[of \(N\)] and \(N\)
  have \(S - I = \sum c \{..<N\} + EM\text{-}remainder' N (fs N) (real-of-int a) (real-of-int b)\) by simp
also have \(\sum c \{..<N\} = \sum c' \{..<N\}\)
proof (intro sum.cong refl)
  fix \(k :: \text{nat}\)
  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 \(a \le b \le N\) finite \(Y\) \(fs 0 = f\)
  \((\forall k. k \le N \implies \text{continuous-on} \{\text{real } a..\text{real } b\} (fs k))\)
  \((\forall k x. k < N \implies x \in \{\text{real } a..\text{real } b\} - Y \implies \text{(fs k has-vector-derivative fs (Suc k) x) (at x)})\)
shows \((\sum i \in \{\text{int } a<..\text{int } b\}. f (\text{real-of-int } i)) - \text{integral} \{\text{real } a..\text{real } b\} f = \)
  \((\sum k < N. (\text{bernoulli'} (Suc k) / \text{fact} (Suc k)) \ast_R (fs k (\text{real-of-int } b) - fs k (\text{real-of-int } a))) + \)
  \(EM\text{-}remainder' N (fs N) (\text{real-of-int } a) (\text{real-of-int } b)\)
proof
  have \((\sum i \in \{\text{int } a<..\text{int } b\}. f (\text{real-of-int } i)) - \text{integral} \{\text{real } a..\text{real } b\} f = \)
    \((\sum k < N. (\text{bernoulli'} (Suc k) / \text{fact} (Suc k)) \ast_R (fs k (\text{real-of-int } b) - fs k (\text{real-of-int } a))) + \)
    \(EM\text{-}remainder' N (fs N) (\text{real-of-int } a) (\text{real-of-int } b)\)
    using assms by (intro euler-maclaurin-raw-strong-int [where \(Y = Y\)] assms)
  simp-all
  also have \((\sum i \in \{\text{int } a<..\text{int } b\}. f (\text{real-of-int } i)) = \sum i \in \{a<..b\}. f (real i))\)
    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]: \( \forall k. k \leq N \implies \text{continuous-on} \{\text{real-of-int} \ a..\} (fs k) \)
assumes fs-deriv [derivative-intros]: \( \forall k x. k < N \implies x \in \{\text{of-int} \ a..\} - Y \implies (fs k \ \text{has-vector-derivative} \ fs \ (\text{Suc} k) \ x) \ (at \ x) \)
assumes F-cont [continuous-intros]: \( \text{continuous-on} \{\text{of-int} \ a..\} F \)
assumes F-deriv [derivative-intros]: \( \forall x. x \in \{\text{of-int} \ a..\} - Y \implies (F \ \text{has-vector-derivative} \ f x) \ (at \ x) \)
assumes limit: 
(\( \lambda b. (\sum_{k=a..b} f k) - F \ (\text{of-int} \ b) - (\sum_{i<N} (\text{bernoulli}' (\text{Suc} i) \ / \ \text{fact} \ (\text{Suc} i)) \ *_R \ fs \ i \ (\text{of-int} \ b)) \) \longrightarrow \ C)
next
case False
with assms have ab: a < b by simp
  define T' where T' = (∑ k< N. (bernoulli' (Suc k) / fact (Suc k)) *R (fs k (of-int a)))
  have (∑ i∈{a..b}. (f (of-int i)) = (∑ k< N. (bernoulli' (Suc k) / fact (Suc k)) *R (fs k (of-int b) - fs k (of-int a))) +
    EM-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 integral {of-int a..of-int b} f = F (of-int b) - F (of-int a)
    by (simp add: has-integral-iff)
  also have (∑ k< N. (bernoulli' (Suc k) / fact (Suc k)) *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 (∑ i∈{a..b}. (f (of-int i)) = S - f (of-int a)
    unfolding S-def using ab by (subst subst-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 ≡ EM-remainder' N (fs N) (of-int a) (of-int b)
  shows EM-remainder N (fs N) b = C' - D
  and EM-remainder-converges: EM-remainder-converges N (fs N) b
proof -
  note limit
  also have ((λb. (∑ k = a..b. (f (of-int k))) - F (of-int b) -
     (∑ i< N. (bernoulli' (Suc i) / fact (Suc i)) *R fs i (of-int b))) ----> C) at-top =
    ((λb. (∑ k = a..b. (f (of-int k))) - F (of-int b) - T (of-int b)) ----> C)
    at-top
    unfolding T-def ..
  also have eventually (λx. (∑ k=a..x. f k) - F (of-int x) - T (of-int x) =
    EM-remainder' N (fs N) (of-int a) (of-int x) + (C - C')) 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 ----> C) at-top ↔ (?g ----> C) at-top by (intro filterlim-cong refl)
  finally have ((λx. ?g x - (C - C')) ----> (C - (C - C'))) at-top
    by (rule tendsto-diff[OF OF - tendsto-const])
  hence: (λx. EM-remainder' N (fs N) (of-int a) (of-int x) ----> C') at-top
    by simp

25
have \((\lambda x. \text{EM-remainder}' \ N \ (fs \ N) \ (\text{of-int} \ a) \ (\text{of-int} \ x) - D) \rightarrow C' - D)\) at-top
   by (intro tendsto-intros *)
also have eventually \((\lambda x. \text{EM-remainder}' \ N \ (fs \ N) \ (\text{of-int} \ a) \ (\text{of-int} \ x) - D = EM-remainder' \ N \ (fs \ N) \ (\text{of-int} \ b) \ (\text{of-int} \ x))\) at-top
   (is eventually \((\lambda x. \ ?f v = \ ?g v)\) - using eventually-ge-at-top[of b]
proof eventually-elim
   case (elim x)
   have \(\text{EM-remainder}' \ N \ (fs \ N) \ (\text{of-int} \ a) \ (\text{of-int} \ x) = \ C + EM-remainder' \ N \ (fs \ N) \ (\text{of-int} \ b) \ (\text{of-int} \ x)\)
     using elim \(ab\) unfolding \(D\)-def
   by (intro EM-remainder'-combine [symmetric] continuous-on-subset[OF fs-cont])
auto thus \(?\text{case} \ by \ simp\)
qed

hence \((\ ?f \rightarrow C' - D) \ at-top \leftrightarrow (\ ?g \rightarrow C' - D) \ at-top\) by (intro filterlim-cong refl)
finally have *: \ldots .
from * show \(\text{EM-remainder-converges} \ N \ (fs \ N) \ b\) unfolding \(\text{EM-remainder-converges-def}\) ..
from * show \(\text{EM-remainder} \ N \ (fs \ N) \ b = C' - D\)
   by (rule EM-remainder-eqI)
qed

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

end

end

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

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

begin

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

27
For convenience, we also offer a version where the sum ranges over natural numbers instead of integers.

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

\[
\sum_{k=\text{int } a.. \text{int } b} f (\text{of-int } k) = \sum_{k=\text{a..b}} f (\text{of-nat } k)
\]

by (intro sum.reindex-bij-witness[of - int nat]) auto

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

fixes F and f :: real

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

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

**Proof**

have \(\ast\): \((\lambda b. (\sum_{k=\text{int } a.. \text{int } b} f (\text{real } k)) - F (\text{of-int } (\text{int } b))\) \(\longrightarrow C)\) at-top using assms by (subst sum-atLeastAtMost-of-int-nat-transfer)

simp

thus \(\ast\)thesis by (rule filterlim-int-of-nat-at-topD)

qed

**Locale** euler-maclaurin-nat =

fixes F f :: real ⇒ ′a :: real

fixes a :: ′a

fixes N :: ′a assumes \(N > 0)\)

fixes C :: ′a

fixes Y :: real set assumes \(\text{fin: finite } Y)\)

assumes F-cont [continuous-intros]: \(\forall k. k \leq N \implies \text{continuous-on } \{ \text{real } a.. \} (f k)\)

assumes F-deriv [derivative-intros]: \(\forall x. x \in \{ \text{real } a.. \} - Y \implies (f k \text{ has-vector-derivative } f x) (at x)\)

assumes limit:

\((\lambda b. (\sum_{k=\text{a..b}} f k) - F (\text{real } b) - (\sum_{i<N. (\text{bernoulli'} (\text{Suc } i) / \text{fact } (\text{Suc } i)) *_{\text{R}} f s i (\text{real } b)) \longrightarrow C)\) 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 \( S = F (\text{real } b) + C + (\sum_{i<N} (\text{bernoulli}' (\text{Suc } i) / \text{fact (Suc } i)) *_{R} fs i (\text{real } b)) - \text{EM-remainder } N (fs N) (\text{int } b) \)
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: \( \text{simp}: fs 0 = f \)
assumes fs-cont: \( \text{continuous-intros}; \forall k. \text{real } k \leq 2*N+1 \Rightarrow \text{continuous-on } \{\text{real } a..\} (fs k) \)
assumes fs-deriv: \( \text{derivative-intros}; \forall k. x. \text{real } k \leq 2*N \Rightarrow x \in \{\text{real } a..\} - Y \Rightarrow (fs k \text{ has-vector-derivative } fs (\text{Suc } k) x) (\text{at } x) \)
assumes F-cont: \( \text{continuous-intros}; \text{continuous-on } \{\text{real } a..\} F \)
assumes F-deriv: \( \text{derivative-intros}; \forall x. x \in \{\text{real } a..\} - Y \Rightarrow (F \text{ has-vector-derivative } f x) (\text{at } x) \)
assumes limit:
\[ ((\lambda b. (\sum_{k=a..b} f k)) - F \text{ (real } b) - (\sum_{i<2*N+1} (\text{bernoulli}' (\text{Suc } i) / \text{fact (Suc } i)) *_{R} fs (2*i-1) \text{ (real } b)) - \text{EM-remainder } (2*N+1) (fs (2*N+1)) b) \]
at-top
begin

sublocale euler-maclaurin-int' = euler-maclaurin-nat' a N C Y
by standard (insert fin fs-cont fs-deriv F-cont F-deriv euler-maclaurin-nat-int-transfer[of \( \text{int } b \)], simp-all)

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

```plaintext
fixes $f :: \text{real} \Rightarrow 'a :: \{\text{real-normed-field, banach}\}$
and $g, g' :: \text{real} \Rightarrow \text{real}$
assumes fin: finite $Y$
assumes pbernopoly-bound: $\forall x. |\text{pbernopoly } n x| \leq D$
assumes cont-f: continuous-on $\{a..\} f$
assumes cont-g: continuous-on $\{a..\} g$
assumes cont-g': continuous-on $\{a..\} g'$
assumes limit-g: $(g \longrightarrow C) \text{ at-top}$
assumes f-bound: $\forall x. x \geq a \Rightarrow \text{norm } (f x) \leq g' x$
assumes deriv: $\forall x. x \in \{a..\} - Y \Rightarrow (g \text{ has-field-derivative } g' x) \text{ (at } x)$
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**

```plaintext
from pbernopoly-bound have $D: \text{norm } (\text{pbernopoly } n x) \leq D \text{ for } x \text{ by auto}$
from this[of 0] have $D\text{-nonneg: } D \geq 0 \text{ by simp}$
define $D' \text{ where } D' = D / \text{fact } n$
from $D\text{-nonneg} \text{ have } D'\text{-nonneg: } D' \geq 0 \text{ by (simp add: } D'\text{-def})$

have $\text{bound: } \text{norm } (\text{EM-remainder'} n f x y) \leq D' * (g y - g x)$
if $x y: x \geq a x \leq y \text{ for } x y :: \text{real}$
proof 
  have $\text{norm } (\text{EM-remainder'} n f x y) = \text{norm } (\text{integral } \{x..y\} (\lambda t. \text{pbernopoly } n t * R f t)) / \text{fact } n$
  by (simp add: EM-remainder'-def)
  also have $(\lambda t. D * g' t) \text{ integrable-on } \{x..y\} \text{ using } x y$
  by (intro integrable-continuous-real-continuous-intros continuous-on-subset[OF cont-g'])
auto
hence $\text{norm } (\text{integral } \{x..y\} (\lambda t. \text{pbernopoly } n t * R f t)) \leq$
  $\text{integral } \{x..y\} (\lambda t. D * g' t) \text{ using } D \text{ D-nonneg } x y$
  by (intro integral-norm-bound-integral integrable-EM-remainder' continuous-on-subset[OF cont-f']) (auto intro: mult-mono f-bound)
also have $\text{.. } = D * \text{ integral } \{x..y\} g' \text{ by simp}$
also have $(g' \text{ has-integral } (g y - g x)) \{x..y\} \text{ using } x y$
by (intro fundamental-theorem-of-calculus-strong[OF fin] continuous-on-subset[OF cont-g'])
(auto simp: has-real-derivative-iff-has-vector-derivative [symmetric] intro: deriv)
```

30
hence 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 \(\lim ((\lambda y. \text{EM-remainder'} \ n \ f \ x \ (\text{of-int} \ y)) \longmapsto \text{EM-remainder} \ n \ f \ x)\)
at-top
if \(x \geq a\) for \(x :: \text{int}\)
proof
have \((\lambda n. g \ (\text{real} \ n)) \longmapsto C\)
  by (rule filterlim-compose[OF limit-g filterlim-real-sequentially])
hence Cauchy: 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 \(\epsilon\))
define \(\epsilon'\) where \(\epsilon' = (\text{if } D' = 0 \text{ then } 1 \text{ else } \epsilon / (2 * D'))\)
from \(\epsilon > 0\) \(\text{D'}-\text{nonneg}\) have \(\epsilon'; \epsilon' > 0\) by (simp add: \(\epsilon'\)-def divide-simps)
from Cauchy\(D[\text{OF Cauchy this}]\) obtain \(M\)
  where \(M: \bigwedge m \ n. m \geq M \implies n \geq M \implies \text{norm} \ (g \ (\text{real} \ m) - g \ (\text{real} \ n)) < \epsilon'\) by blast
show \(?case\)
proof (intro CauchyI' exI[of - max (max 0 \(M\)) (nat \(x\))] allI implI, 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 \(\text{EM-remainder'}-\text{combine continuous-on-subset[OF cont-f]}\)) 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 \(\text{norm} \ldots \leq D' \ast (g \ l - g \ k)\) by (intro bound) auto
    also have \(g \ l - g \ k \leq \text{norm} \ (g \ l - g \ k)\) by simp
    also from \(1 \) have \(\ldots \leq \epsilon'\) using \(M[\text{of } l \ k]\) by auto
    also from \(\epsilon > 0\) have \(D' \ast \epsilon' \leq \epsilon / 2\) by (simp add: \(\epsilon'\)-def)
    also from \(\epsilon > 0\) have \(\ldots < \epsilon\) by simp
  finally show \(?case\) by (simp add: \(D'\)-nonneg mult-left-mono dist-norm norm-minus-commute)
qed
qed
then obtain \(L\) where \((\lambda y. \text{EM-remainder'} \ n \ f \ x \ (\text{int} \ y)) \longmapsto L\)
  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)) \longmapsto L) at-top by simp
moreover from this have \(\text{EM-remainder} \ n \ f \ x = L\) by (rule \(\text{EM-remainder-eqI}\))
ultimately show ((\lambda y. \text{EM-remainder'} \ n \ f \ x \ (\text{of-int} \ y)) \longmapsto \text{EM-remainder} \ n \ f \ x) at-top
  by simp
qed

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

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

lemma
  fixes \(f::\text{real } \Rightarrow \{\text{real-normed-field, banach}\}\)
    and \(g g'::\text{real } \Rightarrow \text{real}\)
  assumes \(\text{fin:: } \{\text{finite } Y\}\)
  assumes \(\text{pbernopoly-bound:: } \forall x. |\text{pbernopoly } n x| \leq D\)
  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 \(\text{limit-g:: } (g \rightarrow 0)\) at-top
  assumes \(\text{f-bound:: } \forall x. x \geq a \Rightarrow \text{norm } (f x) \leq g' x\)
  assumes \(\text{deriv:: } \forall x. x \in \{a.\} - Y \Rightarrow (g \text{ has-field-derivative } g' x) \text{ (at } x)\)
  shows \(\text{norm-EM-remainder-le-strong-int': }\)
    \(\forall x. \text{of-int } x \geq a \rightarrow \text{norm (EM-remainder } n f x) \leq D / \text{fact } n * g x\)
  and \(\text{norm-EM-remainder-le-strong-nat': }\)
    \(\forall x. \text{real } x \geq a \rightarrow \text{norm (EM-remainder } n f (\text{int } x)) \leq D / \text{fact } n * g x\)
proof
  have \(\forall x. \text{of-int } x \geq a \rightarrow \text{norm (EM-remainder } n f x) \leq D / \text{fact } n * (0 - (\text{- } g x))\) using \(\text{assms}\)
    by (intro \text{norm-EM-remainder-le-strong-int}[OF \text{fin pbernopoly-bound - - cont-g'}])
      (auto intro: \text{continuous-intros tendsto-eq-intros derivative-eq-intros})
  thus \(\forall x. \text{of-int } x \geq a \rightarrow \text{norm (EM-remainder } n f (\text{int } x)) \leq D / \text{fact } n * g x\) by auto
next
  have \(\forall x. \text{real } x \geq a \rightarrow \text{norm (EM-remainder } n f (\text{int } x)) \leq D / \text{fact } n * (0 - (\text{- } g x))\) using \(\text{assms}\)
    by (intro \text{norm-EM-remainder-le-strong-nat}[OF \text{fin pbernopoly-bound - - cont-g'}])
      (auto intro: \text{continuous-intros tendsto-eq-intros derivative-eq-intros})
  thus \(\forall x. \text{real } x \geq a \rightarrow \text{norm (EM-remainder } n f (\text{int } x)) \leq D / \text{fact } n * g x\)
by auto 

qed

lemma norm-EM-remainder'-le: 
  fixes f :: real ⇒ 'a :: {real-normed-field, banach} 
  and g g' :: real ⇒ real 
  assumes cont-f: continuous-on {a..} f 
  assumes cont-g': continuous-on {a..} g' 
  assumes f-bigo: eventually (λx. norm (f x) ≤ g' x) at-top 
  assumes deriv: eventually (λx. (g has-field-derivative g' x) (at x)) at-top 
  obtains C D where 
    eventually (λx. norm (EM-remainder' n f a x) ≤ C + D * g x) at-top 
proof – 
  note cont = continuous-on-subset[OF cont-f] continuous-on-subset[OF cont-g'] 
  from bounded-bernpoly[of n] obtain D where D: \( \forall x. \text{norm \( \text{bernpoly \( n \) \( x \) \)}} \leq D \) by blast 
  from this[of 0] have D-nonneg: D ≥ 0 by simp 
  from eventually-conj[OF f-bigo eventually-conj[OF deriv eventually-ge-at-top[of a]]] 
    obtain x0 where x0: 
      \( x0 ≥ a \ \land \ x ≥ x0 \Longrightarrow \text{norm \( f x \) ≤ g' x} \) 
      \( \land x. x ≥ x0 \Longrightarrow (g \text{ has-field-derivative g' x}) \atop x \) 
    by (auto simp: eventually-at-top-linorder) 
  define C where C = (norm (integral {a..x0} (λt. bernpoly n t *R f t)) - D * g x0) / fact n 
  have eventually (λx. norm (EM-remainder' n f a x) ≤ C + D / fact n * g x) at-top 
    using eventually-ge-at-top[of x0] 
  proof eventually-elim 
    case (elim x) 
    have integral {a..x} (λt. bernpoly n t *R f t) = 
      integral {a..x0} (λt. bernpoly n t *R f t) + 
      integral {x0..x} (λt. bernpoly 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 norm .. ≤ norm ?I1 + norm ?I2 by (rule norm-triangle-ineq) 
    also have norm ?I2 ≤ integral {x0..x} (λt. D * g' t) 
      using x0 D-nonneg 
      by (intro integral-norm-bound-integral integrable-EM-remainder') 
        (auto intro!: integrable-continuous-real continuous-intros cont mult-mono) 
    also have .. = D * 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 integral {x0..x} g' = g x - g x0 by (simp add: has-integral-iff) 
    finally have norm (integral {a..x} (λt. bernpoly n t *R f t)) ≤ norm ?I1 +
\[ D \ast (g x - g x_0) \]

by simp-all

thus \[
\| \text{EM-remainder'} n f a x \| \leq C + D / \text{fact } n * g x
\]

by (simp add: EM-remainder'-def field-simps C-def)

qed

thus \( \text{thesis by (rule that)} \)

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

\[
\text{harm-remainder } N \ n = \text{EM-remainder } (2 \ast N + 1) (\lambda x. -\text{fact } (2 \ast N + 1) / x ^ ((2 \ast N + 2)) (\text{int } n)
\]

**lemma** harm-expansion:

assumes \( n > 0 \) and \( N > 0 \)

shows \( \text{harm } n = \ln n + \text{euler-mascheroni} + 1 / (2 \ast n) - \left( \sum_{i=1..N} \text{bernoulli'} (2 \ast i) / [(2 \ast i) * n ^ (2 \ast i)] \right) - \text{harm-remainder} \)

proof

fix \( k x \) assume \( k \leq 2 \ast N x \in \{\text{real }1..\} - {}\)

thus \( \lambda b. \text{harm } b - \ln \text{real } b - \left( \sum_{i=1..b} \text{bernoulli'} (2 \ast i) * (-1) ^ i / (\text{real } (2 \ast i)) \right) \)

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 \( \lambda b. \text{harm } b - \ln \text{real } b - \left( \sum_{i=1..b} \text{bernoulli'} (2 \ast i) * (-1) ^ i / (\text{real } (2 \ast i)) \right) \)

by (intro tendsto-diff euler-mascheroni-LIMSEQ tendsto-sam

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 \( \lambda b. \left( \sum_{i=1..b} 1 / \text{real } b \right) - \ln \text{real } b - \left( \sum_{i=1..b} \text{bernoulli'} (2 \ast i) / \text{fact } (2 \ast i) \ast R \text{fs } i \text{ real } b \right) \)

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 \( \text{harm } n = \left( \sum_{k=1..n} 1 / \text{real } k \right) \)

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

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

34
\(\sum_{i=1..N} (\text{bernoulli } (2*i) \div \text{fact } (2*i)) \ast_R \text{fs } (2*i-1) \) (real n))

EM-remainder \( (2*N+1) \) (fs \( (2*N+1) \)) (int n) using n N

also have \(\sum_{i=1..N} (\text{bernoulli } (2*i) \div \text{fact } (2*i)) \ast_R \text{fs } (2*i-1) \) (real n))
by (intro sum.simp cong refl)
(simp-all add: fs-def divide-simps fact-reduce del: of-natSuc power-Suc)
also have \(\ldots = - (\sum_{i=1..N} \text{bernoulli } (2*i) \div (\text{real } (2*i) \ast \text{real } n \sim (2*i)))\)
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 \(\geq (1 :: 'a :: linordered-semidom) \longrightarrow x \geq 1\)
using of-nat-le-iff[of 1 x] by (simp del: of-nat-le-iff)

lemma harm-remainder-bound:

fixes N :: nat

assumes N: N > 0

shows \(\exists C. \forall n \geq 1. \text{norm } (\text{harm-remainder } N n) \leq C / \text{real } n \sim (2*N+1)\)

proof –
from bounded-bernpoly[of \(2*N+1\)] obtain D where D: \(\forall x. \text{bernopoly } (2*N+1) x| \leq D \) by auto

have \(\forall x. 1 \leq \text{real } x \longrightarrow \text{norm } (\text{harm-remainder } N x) \leq D / \text{fact } (2*N+1) \ast (\text{fact } (2*N) / x \sim (2*N+1))\)
unfolding harm-remainder-def of-int-of-nat-eq
proof (rule norm-EM-remainder-le-strong-nat[of {}])
fix x :: real assume x: x \(\geq 1\)

show \(\text{norm } (\text{-fact } (2*N+1) / x \sim (2*N+2)) \leq \text{fact } (2*N+1) / x \sim (2*N+2)\)
using x by simp
next
show \((\lambda x::\text{real}. \text{fact } (2*N) / x \sim (2*N+1)) \longrightarrow 0\) 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)

hence \(\forall x. 1 \leq x \longrightarrow \text{norm } (\text{harm-remainder } N x) \leq D / (2*N+1) \ast \text{real } x \sim (2*N+1) \) 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 $n \colon n > 0$ and $N \colon N > 0$
shows $(\sum_{k=1..n} 1 \div real k ^ 2) = pi ^ 2 / 6 - 1 / real n + 1 / (2 * real n ^ 2) - 
(\sum_{i=1..N} bernoulli (2*i) / n ^ (2*i+1)) - 
\text{EM-remainder} (2*N+1) (\lambda x. \text{fact} (2*N+2) / x ^ (2*N+3))$

$(int n)$
proof
have 3 = Suc (Suc (Suc 0)) by (simp add: eval-nat-numeral)
define $fs$ where $fs = (\lambda k x. (-1) ^ k * \text{fact} (Suc k) / x ^ (k+2) :: real)$
interpret euler-maclaurin-nat' $\lambda x. -1/x \lambda x. 1/x^2$ $fs$ 1 $N$ pi^2/6 \{ \}
proof
  fix $k \ x$ assume $k \leq 2*N \ x \in \{real 1..\} - {}$
  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 $(\lambda n. \sum_{k < n} 1 \div real (Suc k) ^ 2) \longrightarrow pi^2 / 6$ by (simp add: sums-def)
  also have $(\lambda n. \sum_{k < n} 1 \div real (Suc k) ^ 2) = (\lambda n. \sum_{k=1..n} 1 \div real k ^ 2)$
  by (intro ext sum.reindex-bij-witness[of - $\lambda n. n - 1$ Suc])
finally have $(\lambda b. (\sum_{k=1..b} 1 \div real k ^ 2) + 1 \div real b - 
(\sum_{i<2*N+1} bernoulli' (Suc i) * (-1) ^ i / (real b ^ (i+2)))) 
\longrightarrow (pi^2 / 6 + 0 - (\sum_{i<2*N+1} 0))$
  by (intro tendssto-diff tendssto-add real-tendssto-divide-at-top[OF tendssto-const]
    filterlim-tendssto-pos-mult-at-top[OF tendssto-const] filterlim-pow-at-top
    filterlim-real-sequentially tendssto-sum) auto
thus $(\lambda b. (\sum_{k=1..b} 1 \div real k ^ 2) - (-1 \div real b) - 
(\sum_{i<2*N+1} bernoulli' (Suc i) / (fact (Suc i)) * _R fs i (real b))) 
\longrightarrow pi^2 / 6$
  by (simp add: harm-def field-simps fs-def del: power-Suc of-nat-Suc)
qed (intro n $N$, auto intro!: continuous-intros derivative-eq-intros
  simp: fs-def has-real-derivative-iff-has-vector-derivative [symmetric] power2-eq-square)

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

36
by (simp add: sum-negf)
finally show \( \text{thesis unfolding fs-def by (simp add: fs-def 3)} \)
qed

lemma sum-inverse-squares-remainder-bound:
  fixes \( N :: \text{nat} \)
  assumes \( N : N > 0 \)
  defines \( R \equiv (\lambda n. \text{EM-remainder } (2 \times N + 1) (\lambda x. - \text{fact } (2 \times N + 2) / x ^ {2 \times N + 3})) \)
  shows \( \exists C. \forall n \geq 1. \text{norm } (R n) \leq C / \text{real } n ^ {(2 \times N + 2)} \)
proof
  have \( 3 : 3 = \text{Suc (Suc (Suc 0))} \) by simp
  from \text{bounded-bernopoly[of } 2 \times N + 1] \ \text{obtain } D \ \text{where } D: \forall x. \text{bernopoly } (2 \times N + 1) \)
  \( x \leq D \) by auto
  have \( \forall x. 1 \leq x \rightarrow \text{norm } (R x) \leq D / \text{fact } (2 \times N + 1) * (\text{fact } (2 \times N + 1) / x ^ {(2 \times N + 2)}) \)
    unfolding \( R \)-def \ \text{of-int-of-nat-eq} \)
proof (rule \text{norm-EM-remainder-te-strong-nat[of \{}])
    fix \( x :: \text{real} \)
    assume \( x : x \geq 1 \)
    show \( \text{norm } (\text{fact } (2 \times N + 2) / x ^ {(2 \times N + 3)}) \leq \text{fact } (2 \times N + 2) / x ^ {(2 \times N + 3)} \)
      using \( x \) by simp
next
  show \( ((\lambda x::\text{real}. \text{fact } (2 \times N + 1) / x ^ {(2 \times N + 2)}) \rightarrow 0) \) \text{at-top}
    by (intro \text{real-tendsto-divide-at-top[}OF tendsto-const]\ \text{filterlim-pow-at-top filterlim-ident})
  simp-all
  qed (insert \( N \) \( D \), auto intro!: \text{derivative-eq-intros continuous-intros simp: field-simps power-diff 3}
  hence \( \forall x \geq 1. \text{norm } (R x) \leq D / \text{real } x ^ {2 \times N + 2} \) by simp
  thus \( \text{thesis by blast} \)
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 $f :: real \Rightarrow 'a :: \{real-normed-field, banach\}$
and $g, g' :: real \Rightarrow real$
assumes fin: finite $Y$
assumes cont-$f$: continuous-on $\{a..\} f$
assumes cont-$g$: continuous-on $\{a..\} g$
assumes cont-$g'$: continuous-on $\{a..\} g'$
assumes limit-$g$: $(g \xrightarrow{\text{at-top}} 0)$
assumes $f$-bound: $\forall x. x \geq a \Rightarrow \text{norm}(f x) \leq g' x$
assumes deriv: $\forall x. x \in \{a..\} - Y \Rightarrow (g \text{ has-field-derivative} - g' x \text{ (at $x$)})$
shows EM-remainder-strong-bigo-int: $(\lambda x::\text{int. norm} (\text{EM-remainder} n f x)) \in O(g)$
and EM-remainder-strong-bigo-nat: $(\lambda x::\text{nat. norm} (\text{EM-remainder} n f x)) \in O(g)$
proof –
from bounded-phbernopoly[of $n$] obtain $D$ where $\forall x. \text{phbernopoly} n x \leq D$ by auto
from norm-EM-remainder-le-strong-int[of $n$ assms]
have $\ast$: $(\lambda x. x \geq a \Rightarrow \text{norm} (\text{EM-remainder} n f x) \leq D / \text{fact} n \times g' x)$ by auto
have $\ast\ast$: eventually $(\lambda x::\text{int. norm} (\text{EM-remainder} n f x) \leq \text{abs} (D / \text{fact} n) \times \text{abs} (g x)) \text{ (at-top)}$
using eventually-ge-at-top[of ceiling $a$]
proof eventually-elim
case (elim $x$)
with $\ast[of x]$ have $(\lambda x::\text{int. norm} (\text{EM-remainder} n f x) \leq D / \text{fact} n \times g' x)$ by (simp add: ceiling-le-iff)
also have $\ldots \leq \text{abs} (D / \text{fact} n \times g x)$ by (rule $\ast\ast\ast$)
also have $\ldots = \text{abs} (D / \text{fact} n) \times \text{abs} (g x)$ by (simp add: $\ast\ast\ast$)
finally show $\ast\ast\ast$
qed
thus $(\lambda x::\text{int. norm} (\text{EM-remainder} n f x)) \in O(g)$
by (intro $\ast\ast\ast$) (auto elim: eventually-mono)
	hence $(\lambda x::\text{nat. norm} (\text{EM-remainder} n f \text{ (int $x$)})) \in O(\lambda x. g \text{ (of-int (int $x$))))$
by (rule landau-o.big.compose) (fact filterlim-int-sequentially)
thus $(\lambda x::\text{nat. norm} (\text{EM-remainder} n f x)) \in O(g)$ 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 harm-remainder-bigo:
assumes $N > 0$
shows harm-remainder $N \in O(\lambda n. \ln n + \gamma + (2 * N + 1))$
proof –
from harm-remainder-bound[of $n$ assms]
obtain $C$ where $\forall n \geq 1. \ norm \ (harm\text{-}remainder \ N \ n) \leq C / real \ n ^ 2$ (2 * N + 1) ..

thus ?thesis
by (intro bigo[of - C] eventually-mono[OF eventually-ge-at-top[of 1]]) auto
qed

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

lemma harm-expansion-bigo-simple1:
$(\lambda n. harm \ n − (ln \ n + euler\text{-}mascheroni + 1 / (2 * n))) \in O(\lambda n. 1 / n ^ 2)$
using harm-expansion-bigo[of 0] by (simp add: power2-eq-square)

lemma harm-expansion-bigo-simple2:
$(\lambda n. harm \ n − (ln \ n + euler\text{-}mascheroni)) \in O(\lambda n. 1 / n)$
proof
have $(\lambda n. harm \ n − (ln \ n + euler\text{-}mascheroni + 1 / (2 * n)) + 1 / (2 * n))$
\in O(\lambda n. 1 / n)
proof (rule sum-in-bigo)
have $(\lambda n. harm \ n − (ln \ n + euler\text{-}mascheroni + 1 / (2 * n))) \in O(\lambda n. 1 / real \ n ^ 2)$
using harm-expansion-bigo-simple1 by simp
also have $(\lambda n. 1 / real \ n ^ 2) \in O(\lambda n. 1 / real \ n)$
by (rule landau-o.big.compose[OF filterlim-real-sequentially]) simp-all
finally show $(\lambda n. harm \ n − (ln \ n + euler\text{-}mascheroni + 1 / (2 * n))) \in O(\lambda n. 1 / n)$ by simp
qed simp-all
thus \( \text{thesis by (simp add: algebra-simps)} \)

qed

**lemma** harm-expansion-bigo-simple:

\[
\text{harm} = o (\lambda n. \ln n + \text{euler-mascheroni} + 1 / (2 * n)) + o O(\lambda n. 1 / n \sim 2)
\]

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

**(int n)**

begin

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

**assumes** \( N > 0 \)

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

**proof**

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

**obtain** \( C \)

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

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

**thus** \( \text{thesis} \)

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

qed

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

**fixes** \( N :: \text{nat} \)

**defines** \( T \equiv (\lambda n. \pi^2 / 6 - 1 / n + 1 / (2 * n \sim 2) - (\sum_{i=1..N. \text{beroulli} (2*i) / (n \sim (2*i+1))}) \)

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

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

**proof**

**have** \( 3 = \text{Suc (Suc 0)} \) by simp

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

**unfolding** \( R-def \)

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

(auto simp: T-def sum-inverse-squares-expansion[of - Suc N] S-def 3 simp del: One-nat-def)
also have \((\lambda n. -S n - R (Suc N) n) \in O(\lambda n. \frac{1}{n^3})\)

proof (intro sum-in-bigo)

show \((\lambda x. -S x) \in O(\lambda n. \frac{1}{n^3})\) unfolding S-def

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

have \(R (Suc N) \in O(\lambda n. \frac{1}{n^3})\)

by (rule sum-inverse-squares-remainder-bigo) simp-all

also have \(2 * Suc N + 2 = 2 * N + 4\) by simp

also have \((\lambda n. \frac{1}{n^3}) \in O(\lambda n. \frac{1}{n^3})\)

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

to show \(R (Suc N) \in \ldots\).

\(\text{qed}\)

finally show \(\text{thesis} \).

\(\text{qed}\)

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

\((\lambda n. (\sum_{i=1..n.} \frac{1}{i^2}) - (\pi^2 / 6 - 1 / n + 1 / (2 \cdot n^2))) \in O(\lambda n. \frac{1}{n^3})\)

using sum-inverse-squares-expansion-bigo[of 0] by (simp add: power2-eq-square)

lemma sum-inverse-squares-expansion-bigo-simple':

\((\lambda n. (\sum_{i=1..n.} \frac{1}{i^2}) =o (\lambda n. \pi^2 / 6 - 1 / n + 1 / (2 \cdot n^2)) +o O(\lambda n. \frac{1}{n^3}))\)

using sum-inverse-squares-expansion-bigo-simple

by (subst set-minus-plus [symmetric]) (simp-all add: fun-diff-def)

end

end

References

