Irrational Rapidly Convergent Series

Angeliki Koutsoukou-Argyraki and Wenda Li

June 17, 2024

Abstract

We formalize with Isabelle/HOL a proof of a theorem by J. Hančl asserting the irrationality of the sum of a series consisting of rational numbers, built up by sequences that fulfill certain properties. Even though the criterion is a number theoretic result, the proof makes use only of analytical arguments. We also formalize a corollary of the theorem for a specific series fulfilling the assumptions of the theorem.

Contents

1 Main Theorem and Sketch of the Proof

We formalize the proof of the following theorem by J. Hančl (Theorem 3 in [1]) :

Theorem 1. *(Theorem 3 in [1]) Let $A \in \mathbb{R}$ with $A > 1$. Let $\{d_n\}_{n=1}^{\infty} \in \mathbb{R}$ with $d_n > 1$ for all $n \in \mathbb{N}$. Let $\{a_n\}_{n=1}^{\infty} \in \mathbb{Z}^+$, $\{b_n\}_{n=1}^{\infty} \in \mathbb{Z}^+$ such that :*

$$(1)\ \lim_{n\to\infty} a_n^{\frac{1}{2^n}} = A,$$

for all sufficiently large $n \in \mathbb{N}$:

$$(2)\ \frac{A}{a_n^{\frac{1}{2^n}}} > \prod_{j=n}^{\infty} d_j$$

and

$$(3)\ \lim_{n\to\infty} \frac{d_n^{2^n}}{b_n} = \infty.$$

Then the series $\alpha = \sum_{n=1}^{\infty} \frac{b_n}{a_n}$ *is an irrational number.*

The first step is to show that the series $\sum_{n=1}^{\infty} \frac{b_n}{a_n}$ converges to some $\alpha \in \mathbb{R}$. To show that $\alpha \in \mathbb{R} \setminus \mathbb{Q}$ we argue by proof by contradiction (to this end several auxiliary lemmas are firstly shown). In particular, assuming that $\alpha \in \mathbb{Q}$, i.e. that there exist $p, q \in \mathbb{Z}^+$ such that $\alpha = \frac{p}{q}$, we show that a quantity $\mathcal{A}(n) \geq 1$ for all $n \in \mathbb{N}$. At the same time, we find $n \in \mathbb{N}$ for which $\mathcal{A}(n) < 1$, yielding a contradiction from which we deduce the irrationality of the sum of the series.

For the proof see [1]. We note that the proof involves only elementary Analysis (criteria for convergence/divergence for sequences and series and several inequalities) and not any arithmetical/number theoretic arguments. Obviously for the formal proof we had to make many intermediate arguments explicit. Proofs of length of roughly 2 A4 pages in the original paper by J. Hančl were formalized in almost 1100 lines of code.

2 Corollary

We moreover formalize the following corollary that asserts the irrationality of the sum of an instance of a series that fulfills the assumptions of the theorem :

Corollary 1. *(Corollary 2 in [1]) Let* $A \in \mathbb{R}$ *with* $A > 1$*. Let* $\{a_n\}_{n=1}^{\infty} \in \mathbb{Z}^+$*,* $\{b_n\}_{n=1}^{\infty} \in \mathbb{Z}^+$ *such that :*

$$\lim_{n\to\infty} a_n^{\frac{1}{2^n}} = A$$

and for all sufficiently large $n \in \mathbb{N}$ *(in particular: for* $n \geq 6$*)*

$$a_n^{\frac{1}{2^n}} (1 + 4(2/3)^n) \leq A$$

and

$$b_n \leq 2^{(4/3)^{n-1}}.$$

Then the series $\sum_{n=1}^{\infty} \frac{b_n}{a_n}$ *is an irrational number.*

The above corollary is an immediate consequence of the theorem by setting $d_n = 1 + (2/3)^n$. For the formalized proof of the corollary one more auxiliary lemma was required.

3 Irrational Rapidly Convergent Series

theory *Irrationality-J-Hancl*
  imports *HOL−Analysis.Analysis HOL−Decision-Procs.Approximation*
begin

This is the formalisation of a proof by J. Hanl, in particular of the proof of his Theorem 3 in the paper: Irrational Rapidly Convergent Series, Rend. Sem. Mat. Univ. Padova, Vol 107 (2002).

The statement asserts the irrationality of the sum of a series consisting of rational numbers defined using sequences that fulfill certain properties. Even though the statement is number-theoretic, the proof uses only arguments from introductory Analysis.

We prove the central result (theorem Hancl3) by contradiction, by making use of some of the auxiliary lemmas. To this end, assuming that the sum is a rational number, for a quantity ALPHA(n) we show that ALPHA(n) ≥ 1 for all $n \in \mathbb{N}$. After that we show that we can find an $n \in \mathbb{N}$ for which ALPHA(n) < 1 which yields a contradiction and we thus conclude that the sum of the series is rational. We finally give an immediate application of theorem Hancl3 for a specific series (corollary Hancl3corollary, requiring lemma summable_ln_plus) which corresponds to Corollary 2 in the original paper by J. Hanl.

hide-const *floatarith.Max*

3.1 Misc

lemma *filterlim-sequentially-iff*:
  filterlim f F1 sequentially ⟷ *filterlim* (λ*x*. *f* (*x*+*k*)) *F1 sequentially*
  unfolding *filterlim-iff*
  by (*metis eventually-at-top-linorder eventually-sequentially-seg*)

lemma *filterlim-realpow-sequentially-at-top*:
  (*x*::*real*) > *1* ⟹ *filterlim* (*power x*) *at-top sequentially*
  apply (*rule LIMSEQ-divide-realpow-zero*[*THEN filterlim-inverse-at-top*,*of - 1*,*simplified*])
  by *auto*

lemma *filterlim-at-top-powr-real*:
  fixes *g*::′*b* ⇒ *real*
  assumes *filterlim f at-top F* (*g* ⟶ *g*′) *F g*′>*1*
  shows *LIM x F*. *g x powr f x* :> *at-top*
proof −
  have *LIM x F*. ((*g*′ + *1*) / *2*) *powr f x* :> *at-top*
  proof (*subst filterlim-at-top-gt*[*of - - 1*],*rule*+)
    fix *Z*::*real* **assume** *Z* > *1*
    have ∀$_F$ *x in F*. *ln Z* / *ln* ((*g*′ + *1*) / *2*) ≤ *f x*
      using *assms*(*1*) *filterlim-at-top* **by** *blast*
    then have ∀$_F$ *x in F*. *ln Z* ≤ *ln* (((*g*′ + *1*) / *2*) *powr f x*)

```
    proof (eventually-elim)
      case (elim x)
      then show ?case
      using ‹g'>1› by (auto simp:ln-powr divide-simps)
    qed
    then show ∀F x in F. Z ≤ ((g' + 1) / 2) powr f x
    proof (eventually-elim)
      case (elim x)
      then show ?case
        using ‹1 < Z› ‹g'>1› by auto
    qed
  qed
  moreover have ∀F x in F. ((g'+1)/2) powr f x ≤ g x powr f x
  proof −
    have ∀F x in F. g x > (g'+1)/2
      apply (rule order-tendstoD[OF assms(2)])
      using ‹g'>1› by auto
    moreover have ∀F x in F. f x>0
      using assms(1) filterlim-at-top-dense by blast
    ultimately show ?thesis
    proof eventually-elim
      case (elim x)
      then show ?case
        using ‹g'>1› by (auto intro!: powr-mono2)
    qed
  qed
  ultimately show ?thesis using filterlim-at-top-mono by fast
qed

lemma powrfinitesum:
  fixes a::real and s::nat assumes  s ≤ n
  shows  (∏j=s..(n::nat).(a powr (2^j)))  = a powr (∑j=s..(n::nat).(2^j))
  using ‹s ≤ n›
proof(induct n)
  case 0
  then show ?case by auto
next
  case (Suc n)
  have ?case when s≤n using Suc.hyps
     by (metis Suc.prems add.commute linorder-not-le powr-add prod.nat-ivl-Suc'
sum.cl-ivl-Suc that)
  moreover have ?case when s=Suc n
  proof−
    have (∏j = s..Suc n. a powr 2 ^ j) =(a powr 2 ^(Suc n))
      using ‹s=Suc n› by simp
     also have a powr 2 ^ Suc n = a powr sum (power 2) {s..Suc n} using that
by auto
     ultimately show (∏j = s..Suc n. a powr 2 ^ j) = a powr sum (power 2)
{s..Suc n}
```

    using ‹*s*≤*Suc n*› **by** *linarith*
  qed
  ultimately show *?case* **using** ‹*s*≤*Suc n*› **by** *linarith*
qed

lemma *summable-ratio-test-tendsto*:
  fixes *f* :: *nat* ⇒ *'a*::*banach*
  assumes *c* < *1* **and** ∀ *n*. *f n*≠*0* **and** (λ*n*. *norm* (*f* (*Suc n*)) / *norm* (*f n*)) ⟶ *c*
  shows *summable f*
proof −
  obtain *N* **where** *N-dist*:∀ *n*≥*N*. *dist* (*norm* (*f* (*Suc n*)) / *norm* (*f n*)) *c* < (*1*−*c*)/*2*
    using *assms* **unfolding** *tendsto-iff eventually-sequentially*
    by (*meson diff-gt-0-iff-gt zero-less-divide-iff zero-less-numeral*)
  have *norm* (*f* (*Suc n*)) / *norm* (*f n*) ≤ (*1*+*c*)/*2* **when** *n*≥*N* **for** *n*
    using *N-dist*[*rule-format*,*OF that*] ‹*c*<*1*›
    apply (*auto simp add*:*field-simps dist-norm*)
    by *argo*
  then have *norm* (*f* (*Suc n*)) ≤ (*1*+*c*)/*2* ∗ *norm* (*f n*) **when** *n*≥*N* **for** *n*
    using *that assms*(*2*)[*rule-format*, *of n*] **by** (*auto simp add*:*divide-simps*)
  moreover have (*1*+*c*)/*2* < *1* **using** ‹*c*<*1*› **by** *auto*
  ultimately show *?thesis*
    using *summable-ratio-test*[*of* - *N f*] **by** *blast*
qed

lemma *summable-ln-plus*:
  fixes *f*::*nat* ⇒ *real*
  assumes *summable f* ∀ *n*. *f n*>*0*
  shows *summable* (λ*n*. *ln* (*1*+*f n*))
proof (*rule summable-comparison-test-ev*[*OF* - *assms*(*1*)])
  have *ln* (*1* + *f n*) > *0* **for** *n* **by** (*simp add*: *assms*(*2*) *ln-gt-zero*)
  moreover have *ln* (*1* + *f n*) ≤ *f n* **for** *n*
    apply (*rule ln-add-one-self-le-self2*)
    using *assms*(*2*)[*rule-format*,*of n*] **by** *auto*
  ultimately show $\forall_F$ *n in sequentially*. *norm* (*ln* (*1* + *f n*)) ≤ *f n*
    by (*auto intro*!: *eventuallyI simp add*:*less-imp-le*)
qed

lemma *suminf-real-offset-le*:
  fixes *f* :: *nat* ⇒ *real*
  assumes *f*: ⋀*i*. *0* ≤ *f i* **and** *summable f*
  shows (∑*i*. *f* (*i* + *k*)) ≤ *suminf f*
proof −
  have (λ*n*. ∑*i*<*n*. *f* (*i* + *k*)) ⟶ (∑*i*. *f* (*i* + *k*))
    using *summable-sums*[*OF* ‹*summable f*›]
    by (*simp add*: *assms*(*2*) *summable-LIMSEQ summable-ignore-initial-segment*)
  moreover have (λ*n*. ∑*i*<*n*. *f i*) ⟶ (∑*i*. *f i*)
   using *summable-sums*[*OF* ‹*summable f*›] **by** (*simp add*: *sums-def atLeast0LessThan*

```
f)
  then have (λn. ∑ i<n + k. f i) ⟶ (∑ i. f i)
    by (rule LIMSEQ-ignore-initial-segment)
  ultimately show ?thesis
  proof (rule LIMSEQ-le, safe intro!: exI[of - k])
    fix n assume k ≤ n
    have (∑ i<n. f (i + k)) = (∑ i<n. (f ∘ (λi. i + k)) i)
      by simp
    also have ... = (∑ i∈(λi. i + k) ` {..<n}. f i)
      by (subst sum.reindex) auto
    also have ... ≤ sum f {..<n + k}
      by (intro sum-mono2) (auto simp: f)
    finally show (∑ i<n. f (i + k)) ≤ sum f {..<n + k} .
  qed
qed


lemma factt:
  fixes s n ::nat assumes s ≤ n
  shows (∑ i=s..n. 2^i) < (2^(n+1) :: real) using assms
proof (induct n)
  case 0
  show ?case by auto
next
  case (Suc n)
  have ?case when s=n+1 using that by auto
  moreover have ?case when s ≠ n+1
  proof –
    have(∑ i=s..(n+1). 2^i ) = (∑ i=s..n. 2^i ) + (2::real)^(n+1)
      using sum.cl-ivl-Suc ‹s ≤ Suc n ›
      by (auto simp add:add.commute)
    also have ... < (2) ^(n +1) + (2)^(n+1)
    proof –
      have s ≤n using that ‹s ≤ Suc n › by auto
      then show ?thesis
        using Suc.hyps ‹s ≤ n› by linarith
    qed
    also have ... = 2^(n+2) by simp
    finally show (∑ i=s..(Suc n). 2^i )< (2::real)^(Suc n+1) by auto
  qed
  ultimately show ?case by blast
qed
```

3.2 Auxiliary lemmas and the main proof

```
lemma showpre7:
  fixes a b ::nat⇒int and q p::int
  assumes q>0 and p>0and a: ∀ n. a n>0 and ∀ n. b n>0 and
```

```
    assumerational:(λ n. b (n+1) / a (n+1) )  sums (p/q)
  shows q∗((∏j=1..n. of-int( a j)))∗(suminf (λ(j::nat). (b (j+n+1)/ a (j+n+1
))))
        = ((∏j=1..n. of-int( a j)))∗(p −q∗ (∑j=1..n. b j / a j))
proof −
  define aa where aa=(∏j = 1..n. real-of-int (a j))
  define ff where ff=(λi. real-of-int (b (i+1)) / real-of-int (a (i+1)))

  have (∑j. ff (j+n)) =  (∑n. ff n) − sum ff {..<n}
    apply (rule suminf-minus-initial-segment)
    using assumerational unfolding ff-def by (simp add: sums-summable)
  also have ... = p/q − sum ff {..<n}
    using assumerational unfolding ff-def by (simp add: sums-iff)
  also have ... = p/q − (∑j=1..n. ff (j−1))
  proof −
    have sum ff {..<n} = (∑j=1..n. ff (j−1))
      apply (subst sum-bounds-lt-plus1[symmetric])
      by simp
    then show ?thesis unfolding ff-def by auto
  qed
  finally have (∑j. ff (j + n)) = p / q − (∑j = 1..n. ff (j − 1)) .
  then have q∗(∑j. ff (j + n)) = p − q∗(∑j = 1..n. ff (j − 1))
    using ‹q>0› by (auto simp add:field-simps)
  then have aa∗q∗(∑j. ff (j + n)) = aa∗(p − q∗(∑j = 1..n. ff (j − 1)))
    by auto
  then show ?thesis unfolding aa-def ff-def by auto
qed

lemma show7:
  fixes d::nat⇒real and a b::nat⇒int  and q p::int
  assumes q ≥1 and p ≥ 1 and a: ∀n. a n ≥ 1 and ∀n. b n ≥ 1
    and assumerational:(λ n. b (n+1) / a (n+1) ) sums (p/q)
  shows q∗((∏j=1..n. of-int( a j)))∗( suminf (λ (j::nat). (b (j+n+1)/ a (j+n+1
)))) ≥ 1
    (is ?L ≥ -)
proof−
  define LL where LL=?L
  define aa where aa=(∏j = 1..n. real-of-int (a j))
  define ff where ff=(λi. real-of-int (b (i+1)) / real-of-int (a (i+1)))

  have ?L > 0
  proof −
    have aa > 0
      unfolding aa-def using a
    by (induction n) (simp-all add: int-one-le-iff-zero-less prod-pos)
    moreover have (∑j. ff (j + n)) > 0
    proof (rule suminf-pos)
      have summable ff unfolding ff-def using assumerational
        using summable-def by blast
```

then show *summable* (λ*j*. *ff* (*j* + *n*)) **using** *summable-iff-shift*[*of ff n*] **by** *auto*
show ⋀*i*. *0* < *ff* (*i* + *n*) **unfolding** *ff-def* **using** *a assms*(*4*) *int-one-le-iff-zero-less* **by** *auto*
qed
ultimately show *?thesis* **unfolding** *aa-def ff-def* **using** ‹*q*≥*1*› **by** *auto*
qed
moreover have *?L* ∈ ℤ
proof −
have *?L* = *aa* ∗(*p* −*q*∗ (∑*j*=*1*..*n*. *b j* / *a j*))
unfolding *aa-def*
using *a assms assumerational int-one-le-iff-zero-less showpre7* **by** *force*
also have ... = *aa* ∗ *p* − *q* ∗ (∑*j*=*1*..*n*. *aa* ∗ *b j* / *a j*)
by (*auto simp add*:*algebra-simps sum-distrib-left*)
also have ... = *prod a* {*1*..*n*} ∗ *p* − *q* ∗ (∑*j* = *1*..*n*. *b j* ∗ *prod a* ({*1*..*n*} − {*j*}))
proof −
have (∑*j*=*1*..*n*. *aa* ∗ *b j* / *a j*) = (∑*j*=*1*..*n*. *b j* ∗ *prod a* ({*1*..*n*} − {*j*}))
unfolding *of-int-sum*
proof (*rule sum.cong*)
fix *j* **assume** *j* ∈ {*1*..*n*}
then have (∏*i* = *1*..*n*. *real-of-int* (*a i*)) = *a j* ∗ (∏*i*∈{*1*..*n*} − {*j*}. *real-of-int* (*a i*))
by (*meson finite-atLeastAtMost prod.remove*)
then have *aa* / *real-of-int* (*a j*) = *prod a* ({*1*..*n*} − {*j*})
unfolding *aa-def* **using** *a*[*rule-format*,*of j*] **by** (*auto simp add*:*field-simps*)
then show *aa* ∗ *b j* / *a j* = *b j* ∗ *prod a* ({*1*..*n*} − {*j*})
by (*metis mult.commute of-int-mult times-divide-eq-right*)
qed *simp*
moreover have *aa* ∗ *p* = (∏*j* = *1*..*n*. (*a j*)) ∗ *p*
unfolding *aa-def* **by** *auto*
ultimately show *?thesis* **by** *force*
qed
also have ... ∈ ℤ **using** *Ints-of-int* **by** *blast*
finally show *?thesis* **.**
qed
ultimately show *?thesis*
apply (*fold LL-def*)
by (*metis Ints-cases int-one-le-iff-zero-less not-less of-int-0-less-iff of-int-less-1-iff*)
qed

lemma *show8*:
fixes *d* ::*nat*⇒*real* **and** *a* :: *nat*⇒*int* **and** *s k*::*nat*
assumes *A* > *1* **and** *d*: ∀ *n*. *d n*> *1* **and** *a*:∀ *n*. *a n*>*0* **and** *s*>*0*
and *convergent-prod d*
and *assu2*: ∀ *n* ≥ *s*. *A* / *of-int* (*a n*) *powr* (*1* / *of-int* (*2*^*n*)) > (∏*j*. *d* (*n* + *j*))
shows ∀ *n*≥*s*. (∏*j*. *d* (*j*+*n*)) < *A* / (*MAX j*∈{*s*..*n*}. *of-int* (*a j*) *powr* (*1* / *of-int* (*2* ^*j*)))

```
proof (intro strip)
  fix n assume s ≤ n
  define sp where sp ≡ (λn. ∏j. d (j+n))
  define ff where ff ≡ (λ(j::nat). (real-of-int (a j)) powr(1 /of-int (2^j)))
  have sp i ≥ sp n when i≤n for i
  proof −
    have (∏j. d (j + i)) = (∏ia. d (ia + (n − i) + i)) ∗ (∏ia<n − i. d (ia +
i))
      proof (rule prodinf-split-initial-segment)
        show convergent-prod (λj. d (j + i))
          using ‹convergent-prod d› convergent-prod-iff-shift[of d i] by simp
        show ⋀j. j < n − i ⟹ d (j + i) ≠ 0
          by (metis d not-one-less-zero)
      qed
    then have sp i = sp n ∗ (∏j<n − i. d (i + j))
      unfolding sp-def using ‹n≥i› by (auto simp:algebra-simps)
    moreover have sp i>1 sp n>1
      unfolding sp-def using convergent-prod-iff-shift ‹convergent-prod d› d
      by (auto intro!:less-1-prodinf)
    moreover have (∏j<n − i. d (i + j)) ≥ 1
      using d less-imp-le by (auto intro: prod-ge-1)
    ultimately show ?thesis by auto
  qed
  moreover have ∀j≥s. A / ff j > sp j
    unfolding ff-def sp-def using assu2 by (auto simp:algebra-simps)
  ultimately have ∀j. s≤j ∧ j≤n ⟶ A / ff j > sp n by force
  then show sp n< A / Max (ff ‘ {s..n})
   by (metis (mono-tags, opaque-lifting) Max-in ‹n≥s› atLeastAtMost-iff empty-iff

        finite-atLeastAtMost finite-imageI imageE image-is-empty order-refl)
qed

lemma auxiliary1-9:
  fixes d ::nat⇒real and a::nat⇒int  and s m::nat
  assumes d: ∀n. d n> 1  and a: ∀n. a n>0 and s>0 and n ≥ m and  m ≥ s
    and auxifalse-assu: ∀n≥m. (of-int (a (n+1))) powr(1 /of-int (2^(n+1))) <
            (d (n+1))∗ (Max ((λ (j::nat). (of-int (a j)) powr(1 /of-int (2^j))) ‘
{s..n} ))
  shows (of-int (a (n+1))) powr(1 /of-int (2^(n+1))) <
     (∏j=(m+1)..(n+1). d j) ∗ (Max ((λ (j::nat). (of-int (a j)) powr(1 /of-int
(2^j))) ‘ {s..m}))
proof−
  define ff where ff ≡ λj. real-of-int (a j) powr (1 / of-int (2^j))
  have [simp]:ff j > 0 for j
   unfolding ff-def by (metis a less-numeral-extra(3) of-int-0-less-iff powr-gt-zero)

  have ff-asm:ff (n+1) < d (n+1) ∗ Max (ff ‘ {s..n}) when n≥m for n
    using auxifalse-assu that unfolding ff-def by simp
  from ‹n≥m›
```

```
  have Q: (Max( ff ` {s..n} )) ≤ (∏j=(m+1)..n. d j)∗ (Max (ff ` {s..m}))
  proof(induct n)
    case 0
    then show ?case using ‹m≥s› by simp
  next
    case (Suc n)
    have ?case when m=Suc n
      using that by auto
    moreover have ?case when m≠Suc n
    proof −
      have m ≤ n using that Suc(2) by simp
      then have IH: Max (ff ` {s..n}) ≤ prod d {m + 1..n} ∗ Max (ff ` {s..m})
        using Suc(1) by linarith
      have Max (ff ` {s..Suc n}) = Max (ff ` {s..n} ∪ {ff (Suc n)})
        using Suc.prems assms(5) atLeastAtMostSuc-conv by auto
      also have ... = max (Max (ff ` {s..n})) (ff (Suc n))
        using Suc.prems assms(5) max-def sup-assoc that by auto
      also have ... ≤ max (Max (ff ` {s..n})) (d (n+1) ∗ Max (ff ` {s..n}))
        using ‹m ≤ n› ff-asm by fastforce
      also have ... ≤ Max (ff ` {s..n}) ∗ max 1 (d (n+1))
      proof −
        have Max (ff ` {s..n}) ≥ 0
            by (metis (mono-tags, opaque-lifting) Max-in ‹⋀j. 0 < ff j› ‹m ≤ n›
assms(5)
            atLeastAtMost-iff empty-iff finite-atLeastAtMost finite-imageI imageE
            image-is-empty less-eq-real-def)
        then show ?thesis using max-mult-distrib-right
          by (simp add: max-mult-distrib-right mult.commute)
      qed
      also have ... = Max (ff ` {s..n}) ∗ d (n+1)
        by (metis d max.commute max.strict-order-iff)
      also have ... ≤  prod d {m + 1..n} ∗ Max (ff ` {s..m}) ∗ d (n+1)
        using IH d[rule-format, of n+1] by auto
      also have ... = prod d {m + 1..n+1} ∗ Max (ff ` {s..m})
        using ‹n≥m› by (simp add:prod.nat-ivl-Suc' algebra-simps)
      finally show ?case by simp
    qed
    ultimately show ?case by blast
  qed
  then have d (n+1) ∗ Max (ff ` {s..n} ) ≤ (∏j=(m+1)..(n+1). d j)∗ (Max (ff
` {s..m}))
    using ‹m≤n› d[rule-format,of Suc n] by (simp add:prod.nat-ivl-Suc')
  then show ?thesis using ff-asm[of n] ‹s≤m› ‹m≤n› unfolding ff-def by auto
qed

lemma show9:
  fixes d ::nat⇒real and a :: nat⇒int and s ::nat and A::real
  assumes  A > 1 and  d: ∀n. d n> 1 and a: ∀n. a n>0 and s>0
    and assu1: (( λ n. (of-int (a n)) powr(1 /of-int (2^n)))⟶ A) sequentially
```

```
    and convergent-prod d
    and 8: ∀n≥s. prodinf (λj. d( n+j))
              < A/(Max ((λ(j::nat). (of-int (a j)) powr(1 /of-int (2^j))) ` {s..n}))

  shows ∀m ≥s. ∃n≥m. ( (of-int (a (n+1))) powr(1 /of-int (2^(n+1))) ≥
          (d (n+1))∗ (Max ( ( λ (j::nat). (of-int (a j)) powr(1 /of-int (2^j))) `
{s..n} )))
proof (rule ccontr)
  define ff where ff ≡ (λj. real-of-int (a j) powr (1 / of-int (2^j)))
  assume assumptioncontra: ¬ (∀m ≥s. ∃n≥m. ff(n+1) ≥ d(n+1) ∗ Max (ff `
{s..n}))

  then obtain t where t≥s and
    ttt: ∀n≥t. ff (n+1) < d (n+1) ∗ Max (ff ` {s..n})
    by fastforce
  define B where B ≡ ∏j. d (t + 1 + j)
  have B>0 unfolding B-def
  proof (rule less-0-prodinf)
    show convergent-prod (λj. d (t + 1 + j))
      using convergent-prod-iff-shift[of d t+1] ‹convergent-prod d›
      by (auto simp: algebra-simps)
    show ⋀i. 0 < d (t + 1 + i)
      using d le-less-trans zero-le-one by blast
  qed
  have A ≤ B ∗ Max ( ff ` {s..t})
  proof (rule tendsto-le[of sequentially λn. (∏j=(t+1)..(n+1). d j) ∗ Max ( ff `
{s..t}) -
        λn. ff (n+1)])
    show (λn. ff (n + 1)) ⟶ A
      using assu1[folded ff-def] LIMSEQ-ignore-initial-segment by blast
    have (λn. prod d {t + 1..n + 1}) ⟶ B
    proof –
      have convergent-prod (λj. d (t + 1 + j))
          using ‹convergent-prod d› convergent-prod-iff-shift[of d t+1] by (simp
add:algebra-simps)
      then have (λn. ∏i≤n. d (t + 1 + i)) ⟶ B
        using B-def convergent-prod-LIMSEQ by blast
      then have (λn. ∏i∈{0..n}. d (i+(t + 1))) ⟶ B
        using atLeast0AtMost by (auto simp:algebra-simps)
      then have (λn. prod d {(t + 1)..n + (t + 1)}) ⟶ B
        apply (subst (asm) prod.shift-bounds-cl-nat-ivl[symmetric])
        by simp
      from seq-offset-neg[OF this,of t]
      show (λn. prod d {t + 1..n+1}) ⟶ B
        apply (elim Lim-transform)
        apply (rule LIMSEQ-I)
        apply (rule exI[where x=t+1])
        by auto
    qed
```

then show ($\lambda n.$ *prod d* {*t* + *1*..*n* + *1*} ∗ *Max* (*ff* ‘ {*s*..*t*})) ⟶ *B* ∗ *Max* (*ff* ‘ {*s*..*t*})
by (*auto intro*:*tendsto-eq-intros*)
have ∀$_F$ *n in sequentially*. (*ff* (*n*+*1*)) < (∏*j*=(*t*+*1*)..(*n*+*1*). *d j*) ∗ (*Max* (*ff* ‘ {*s*..*t*}))
unfolding *eventually-sequentially ff-def*
using *auxiliary1-9*[*OF d a* ‹*s*>*0*› - ‹*t*≥*s*› *ttt*[*unfolded ff-def*]]
by *blast*
then show ∀$_F$ *n in sequentially*. (*ff* (*n*+*1*)) ≤ (∏*j*=(*t*+*1*)..(*n*+*1*). *d j*) ∗ (*Max* (*ff* ‘ {*s*..*t*}))
by (*eventually-elim*,*simp*)
qed *simp*
also have ... ≤ *B* ∗ *Max* (*ff* ‘ {*s*..*t*+*1*})
proof −
have *Max* (*ff* ‘ {*s*..*t*}) ≤ *Max* (*ff* ‘ {*s*..*t* + *1*})
using ‹*t*≥*s*› **by** (*auto intro*: *Max-mono*)
then show *?thesis* **using** ‹*B*>*0*› **by** *auto*
qed
finally have *A* ≤ *B* ∗ *Max* (*ff* ‘ {*s*..*t* + *1*})
unfolding *B-def* **.**
moreover have *B* < *A* / *Max* (*ff* ‘ {*s*..*t* + *1*})
using *8*[*rule-format*, *of t*+*1*,*folded ff-def B-def*] ‹*s*≤*t*› **by** *auto*
moreover have *Max* (*ff* ‘ {*s*..*t*+*1*})>*0*
using ‹*A* ≤ *B* ∗ *Max* (*ff* ‘ {*s*..*t* + *1*})› ‹*B*>*0*› ‹*A*>*1*› *zero-less-mult-pos* [*of B Max* (*ff* ‘ {*s*..*Suc t*})]
by *fastforce*
ultimately show *False* **by** (*auto simp add*:*field-simps*)
qed

lemma *show10*:
fixes *d* ::*nat*⇒*real* **and** *a* ::*nat*⇒*int* **and** *s*::*nat*
assumes *d* [*rule-format*]: ∀ *n*. *d n*> *1*
and *a* [*rule-format*]: ∀ *n*. *a n*>*0* **and** *s*>*0*
and *9*: ∀ *m* ≥*s*. ∃ *n*≥*m*. *a* (*n*+*1*) *powr*(*1* / *of-int* (*2*^(*n*+*1*))) ≥
d (*n*+*1*) ∗ (*Max* ((λ*j*. (*of-int* (*a j*)) *powr*(*1* /*of-int* (*2*^*j*))) ‘ {*s*..*n*}))
shows ∀ *m*≥*s*. ∃ *n*≥*m*. *d* (*n*+*1*) *powr*(*2*^(*n*+*1*)) ∗ (∏*j*=*1*..*n*. *of-int*(*a j*)) ∗
(*1* / (∏*j*=*1*..*s*−*1*. *of-int*(*a j*))) ≤ *a* (*n*+*1*)
proof (*intro strip*)
fix *m* **assume** *s* ≤ *m*
from *9*[*rule-format*,*OF this*]
obtain *n* **where** *n*≥*m* **and** *asm-9*:((*of-int* (*a* (*n*+*1*))) *powr*(*1* /*of-int* (*2*^(*n*+*1*))) ≥
(*d* (*n*+*1*))∗ (*Max* ((λ (*j*::*nat*). (*of-int* (*a j*)) *powr*(*1* /*of-int* (*2*^*j*))) ‘ {*s*..*n*})))
by *auto*
with ‹*s*≤*m*› **have** *s*≤*n* **by** *auto*

define *M* **where** *M* ≡ λ*s*. *MAX j*∈{*s*..*n*}. *a j powr* (*1* / *real-of-int* (*2* ^ *j*))
have *prod*: (∏*j*=*1*..*n*. *real-of-int* (*a j*)) ∗ (*1* / (∏*j*=*1*..*s*−*1*. *of-int*(*a j*)))

```
        = (∏j=s..n. of-int( a j))
proof −
  define f where f= (λj. real-of-int( a j))
  have {Suc 0..n}= {Suc 0..s − Suc 0} ∪ {s..n} using ‹n≥s›  ‹s >0›
    by auto
  then have (∏j=1..n. f j) = (∏j=1..s−1. f j) ∗ (∏j=s..n. f j)
    apply (subst prod.union-disjoint[symmetric])
    by auto
  moreover have (∏j=1..s−1. f j) > 0
    by (metis a f-def of-int-0-less-iff prod-pos)
  then have (∏j=1..s−1. f j) ≠ 0 by argo
  ultimately show ?thesis unfolding f-def by auto
qed
then have d (n+1) powr 2 ^(n+1) ∗ (∏j = 1..n. of-int (a j)) ∗ (1 / (∏j =
1..s − 1. of-int (a j))) =
         d (n+1) powr 2 ^(n+1) ∗ (∏j = s..n. of-int (a j))
   by (metis mult.assoc prod)
also have
  ... ≤ ((d (n+1))powr(2^(n+1) ) ∗ (∏i=s..n. M s powr(2^i)) )
proof (rule mult-left-mono)
  show 0 ≤ (d (n + 1)) powr 2 ^(n + 1)
    by auto
  show (∏j = s..n. real-of-int (a j)) ≤ (∏i = s..n. M s powr 2 ^ i)
  proof (intro prod-mono conjI)
    fix i assume i: i ∈ {s..n}
    have a i = (a i powr (1 / real-of-int (2 ^ i))) powr 2 ^ i
      unfolding powr-powr by (simp add: a less-eq-real-def)
    also have  . . . ≤ M s powr(2^i)
      unfolding M-def using i by (force intro: powr-mono2)
    finally show a i ≤ M s powr 2 ^ i .
    show ⋀i. i ∈ {s..n} ⟹ 0 ≤ real-of-int (a i)
      by (meson a less-imp-le of-int-0-le-iff)
  qed
qed
also have ... = d(n+1) powr (2^(n+1)) ∗ M s powr (∑i=s..n. 2^i)
proof−
  have  d (n+1) powr (2^(n+1)) ≥ 1
     by (metis Transcendental.log-one d le-powr-iff zero-le-numeral zero-le-power
zero-less-one)
  moreover have (∏i=s..n. M s powr(2^i)) = M s powr (∑i=s..n. 2^i )
    using ‹s≤n› powrfinitesum by auto
  ultimately show ?thesis by auto
qed
also have ... ≤ d (n + 1) powr 2 ^(n + 1) ∗ M s powr(2^(n+1))
proof −
  have sum (power 2) {s..n} < (2::real) ^(n + 1) using factt ‹s≤n› by auto
  moreover have 1 ≤ M s
  proof −
     define S where S=(λ(j::nat). ( of-int( a j) powr(1 /real-of-int (2^j)) )) ‘
```

{*s*..*n* }
    have *finite S* **unfolding** *S-def* **by** *auto*
    moreover have *S*≠{} **unfolding** *S-def* **using** ‹*s*≤*n*› **by** *auto*
    moreover have ∃ *x*∈*S*. *x*≥*1*
    proof−
      have *a s powr* (*1* / (*2*^*s*)) ≥ *1*
      proof (*rule ge-one-powr-ge-zero*)
        show *1* ≤ *real-of-int* (*a s*)
          by (*simp add*: *a int-one-le-iff-zero-less*)
      qed *auto*
      moreover have *of-int*(*a s*) *powr*(*1* /*real-of-int* (*2*^*s*)) ∈ *S*
        unfolding *S-def*
        using ‹*s*≤*n*› **by** *auto*
      ultimately show *?thesis* **by** *auto*
    qed
    ultimately show *?thesis*
      using *Max-ge-iff*[*of S 1*] **unfolding** *S-def M-def* **by** *blast*
  qed
  moreover have *0* ≤ (*d* (*n* + *1*)) *powr 2* ^(*n* + *1*) **by** *auto*
  ultimately show *?thesis*
    by (*simp add*: *mult-left-mono powr-mono M-def*)
qed

also have ... = (*d* (*n*+*1*) ∗ *M s*) *powr*(*2*^(*n*+*1*))
proof −
  have *d* (*n* + *1*) ≥ *0* **using** *d*[*of n*+*1*] **by** *argo*
  moreover have *M s* ≥ *0*
    using ‹*s*≤*n*› **by** (*auto simp*: *M-def Max-ge-iff*)
  ultimately show *?thesis*
    unfolding *M-def* **using** *powr-mult* **by** *auto*
qed
also have ... ≤ (*real-of-int* (*a* (*n* + *1*)) *powr* (*1* / *real-of-int* (*2* ^(*n* + *1*)))) *powr 2* ^(*n* + *1*)
proof (*rule powr-mono2*)
  have *M s* ≥ *0*
    using ‹*s*≤*n*› **by** (*auto simp*: *M-def Max-ge-iff*)
  moreover have *d* (*n* + *1*) ≥*0*
    using *d*[*of n*+*1*] **by** *argo*
  ultimately show *0* ≤ (*d* (*n* + *1*)) ∗ *M s* **by** *auto*
  show (*d* (*n* + *1*)) ∗ *M s* ≤ *real-of-int* (*a* (*n* + *1*)) *powr* (*1* / *real-of-int* (*2* ^(*n* + *1*)))
    using *M-def asm-9* **by** *presburger*
qed *simp*
also have ... = (*of-int* (*a* (*n*+*1*)))
  by (*simp add*: *a less-eq-real-def pos-add-strict powr-powr*)
finally show ∃ *n*≥*m*. *d* (*n* + *1*) *powr 2* ^(*n* + *1*) ∗ (∏*j* = *1*..*n*. *real-of-int* (*a j*)) ∗
        (*1* / (∏*j* = *1*..*s* − *1*. *real-of-int* (*a j*)))
        ≤ *real-of-int* (*a* (*n* + *1*)) **using** ‹*n*≥*m*› ‹*m*≥*s*›

```
    by force
qed

lemma lasttoshow:
  fixes d ::nat⇒real  and a b ::nat⇒int  and q::int and s::nat
  assumes d: ∀ n. d n> 1
     and a:∀ n. a n>0 and s>0 and q>0
    and A > 1 and b:∀ n. b n>0 and 9:
    ∀ m≥s. ∃ n≥m. ((of-int (a (n+1))) powr(1 /of-int (2^(n+1))) ≥
         (d (n+1))∗ (Max ((λ(j::nat). (of-int (a j)) powr(1 /of-int (2^j))) ‘ {s..n}
)))
    and assu3: filterlim( λ n. (d n)^(2^n)/ b n) at-top sequentially
    and 5: ∀F n in sequentially. (∑ j. (b (n + j)) /  (a (n + j))) ≤ 2 ∗ b n / a n
  shows ∃ n. q∗((∏ j=1..n. real-of-int(a j))) ∗ suminf (λ(j::nat). (b (j+n+1)/ a
(j+n+1)))<1
proof−
  define as where as=(∏ j = 1..s − 1. real-of-int (a j))
  obtain n where n≥s and n-def1:real-of-int q ∗ as ∗ 2
      ∗ real-of-int (b (n + 1)) / d (n + 1) powr 2 ^ (n + 1) < 1
     and n-def2:d (n + 1) powr 2 ^ (n + 1) ∗ (∏ j = 1..n. real-of-int (a j)) ∗ (1
/ as)
        ≤ real-of-int (a (n + 1))
    and n-def3:(∑ j. (b (n+1 + j)) /  (a (n+1 + j))) ≤ 2 ∗ b (n+1) / a (n+1)
  proof −
    have ∗:(λn. real-of-int (b n) / d n ^ 2 ^ n) ⟶ 0
      using tendsto-inverse-0-at-top[OF assu3] by auto
    then have (λn. real-of-int (b n) / d n powr 2 ^ n) ⟶ 0
    proof −
      have d n ^ 2 ^ n = d n powr (of-nat (2 ^ n)) for n
        by (metis d le-less-trans powr-realpow zero-le-one)
      then show ?thesis using ∗ by auto
    qed
    from tendsto-mult-right-zero[OF this,of q ∗ as ∗ 2]
    have (λn. q ∗ as ∗ 2 ∗ b n / d n powr 2 ^ n) ⟶ 0
      by auto
    then have ∀F n in sequentially. q ∗ as ∗ 2 ∗ b n / d n powr 2 ^ n < 1
      by (elim order-tendstoD) simp
    then have ∀F n in sequentially. q ∗ as ∗ 2 ∗ b n / d n powr 2 ^ n < 1
                ∧ (∑ j. (b (n + j)) /  (a (n + j))) ≤ 2 ∗ b n / a n
      using 5 by eventually-elim auto
    then obtain N where N-def:∀ n≥N. q ∗ as ∗ 2 ∗ b n / d n powr 2 ^ n < 1
          ∧ (∑ j. (b (n + j)) /  (a (n + j))) ≤ 2 ∗ b n / a n
      unfolding eventually-sequentially by auto
    obtain n where n≥s and n≥N and n-def:d (n + 1) powr 2 ^ (n + 1)
               ∗ (∏ j = 1..n. of-int (a j)) ∗ (1 / as) ≤ real-of-int (a (n + 1))
     using show10[OF d a ‹s>0› 9,folded as-def,rule-format,of max s N] by auto
    with N-def[rule-format, of n+1] that[of n]  show ?thesis by auto
  qed
```

define *pa* **where** $pa \equiv (\prod j = 1..n.\ real\text{-}of\text{-}int\ (a\ j))$
define *dn* **where** $dn \equiv d\ (n + 1)\ powr\ 2\ \hat{}\ (n + 1)$
have [*simp*]:$dn > 0\ as > 0$
subgoal unfolding *dn-def* **by** (*metis d not-le numeral-One powr-gt-zero zero-le-numeral*)
subgoal unfolding *as-def* **by** (*simp add*: *a prod-pos*)
done
have [*simp*]:$pa>0$
unfolding *pa-def* **using** *a* **by** (*simp add*: *prod-pos*)

have *K*: $q* (\prod j=1..n.\ real\text{-}of\text{-}int\ (a\ j)) * suminf\ (\lambda\ (j::nat).\ (b\ (j+n+1)/\ a\ (j+n+1)))$
$\leq q* (\prod j=1..n.\ real\text{-}of\text{-}int\ (a\ j)) *2* (b\ (n+1))/(a(\ n+1))$
apply (*fold pa-def*)
using *mult-left-mono*[*OF n-def3*,*of real-of-int q* ∗ *pa*]
‹$n\geq s$› ‹$pa>0$› ‹$q>0$› **by** (*auto simp add*:*algebra-simps*)
also have *KK*:... $\leq 2*q* (\prod j=1..n.\ real\text{-}of\text{-}int\ (a\ j)) * (b(n+1))*$
$((\prod j=1..s-1.\ real\text{-}of\text{-}int(\ a\ j))/((d\ (n+1))powr(2\hat{}(n+1)) * (\prod j=1..n.\ real\text{-}of\text{-}int\ (\ a\ j))))$
proof −
have $dn * pa * (1\ /\ as) \leq real\text{-}of\text{-}int\ (a\ (n + 1))$
using *n-def2* **unfolding** *dn-def pa-def* **.**
then show *?thesis*
apply (*fold pa-def dn-def as-def*)
using ‹$pa>0$› ‹$q>0$› *a*[*rule-format*, *of Suc n*] *b*[*rule-format*, *of Suc n*]
by (*auto simp add*: *field-simps*)
qed
also have *KKK*: ... $= q * (\prod j=1..(s-1).\ real\text{-}of\text{-}int(a\ j)) * 2 * b\ (n+1)\ /\ d\ (n+1)\ powr\ 2\hat{}(n+1)$
apply (*fold as-def pa-def dn-def*)
apply *simp*
using ‹$0 < pa$› **by** *blast*
also have *KKKK*: ... < 1 **using** *n-def1* **unfolding** *as-def* **by** *simp*
finally show *?thesis* **by** *auto*
qed

lemma
fixes *d* ::*nat*⇒*real* **and** *a b* ::*nat*⇒*int* **and** *A*::*real*
assumes $A > 1$ **and** *d*: $\forall n.\ d\ n> 1$ **and** *a*: $\forall n.\ a\ n>0$ **and** *b*:$\forall n.\ b\ n>0$
and *assu1*: $((\ \lambda\ n.\ (of\text{-}int\ (a\ n))\ powr(1\ /of\text{-}int\ (2\hat{}n))) \longrightarrow A)$ *sequentially*
and *assu3*: *filterlim* $(\ \lambda\ n.\ (d\ n)\hat{}(2\hat{}n)/\ b\ n)$ *at-top sequentially*
and *convergent-prod d*
shows *issummable*: *summable* $(\lambda j.\ b\ j\ /\ a\ j)$
and *show5*: $\forall_F\ n$ *in sequentially*. $(\sum j.\ (b\ (n + j))\ /\ (a\ (n + j))) \leq 2 * b\ n\ /\ a\ n$
proof−
define *c* **where** $c = (\lambda j.\ b\ j\ /\ a\ j)$
have *c-pos*:$c\ j>0$ **for** *j*
using *a b* **unfolding** *c-def* **by** *simp*
have *c-ratio-tendsto*:$(\lambda n.\ c\ (n+1)\ /\ c\ n\ ) \longrightarrow 0$

```
  proof -
    define nn where nn ≡ (λn. (2::int) ^ (Suc n))
    define ff where ff ≡ (λn. (a n / a (Suc n)) powr(1 /nn n)*(d(Suc n)))
    have nn-pos:nn n>0 and ff-pos:ff n>0 for n
      subgoal unfolding nn-def by simp
      subgoal unfolding ff-def
        using d[rule-format, of Suc n] a[rule-format, of n] a[rule-format, of Suc n]
        by auto
      done
    have ff-tendsto:ff ⟶ 1 / sqrt A
    proof -
       have (of-int (a n)) powr(1 / (nn n)) = sqrt(of-int (a n) powr(1 /of-int
(2^n))) for n
        unfolding nn-def using a
        by (simp add: powr-half-sqrt [symmetric] powr-powr ac-simps)
      moreover have (( λ n. sqrt(of-int (a n) powr(1 /of-int (2^n))))⟶ sqrt
A) sequentially
        using assu1 tendsto-real-sqrt by blast
      ultimately have (( λ n. (of-int (a n)) powr(1 /of-int (nn n)))⟶ sqrt A)
sequentially
        by auto
     from tendsto-divide[OF this assu1[THEN LIMSEQ-ignore-initial-segment[where
k=1]]]
      have (λn. (a n / a (Suc n)) powr(1 /nn n)) ⟶ 1/sqrt A
        using ‹A>1› a unfolding nn-def
      by (auto simp add:powr-divide less-imp-le inverse-eq-divide sqrt-divide-self-eq)
      moreover have (λn. d (Suc n))⟶ 1
        apply (rule convergent-prod-imp-LIMSEQ)
        using convergent-prod-iff-shift[of d 1] ‹convergent-prod d› by auto
      ultimately show ?thesis
        unfolding ff-def by (auto intro:tendsto-eq-intros)
    qed
    have (λn. (ff n) powr nn n) ⟶ 0
    proof -
      define aa where aa=(1+1/sqrt A) / 2
      have eventually (λn. ff n<aa) sequentially
        apply (rule order-tendstoD[OF ff-tendsto])
        unfolding aa-def using ‹A>1› by (auto simp add:field-simps)
      moreover have (λn. aa powr nn n) ⟶ 0
      proof -
        have (λy. aa ^ (nat ∘ nn) y) ⟶ 0
          apply (rule tendsto-power-zero)
          subgoal unfolding nn-def comp-def
            apply (rule filterlim-subseq)
            by (auto intro:strict-monoI)
          subgoal unfolding aa-def using ‹A>1› by auto
          done
        then show ?thesis
        proof (elim filterlim-mono-eventually)
```

```
        have aa>0 unfolding aa-def using ‹A>1›
          by (auto simp add:field-simps pos-add-strict)
        then show ∀F x in sequentially. aa ^(nat ∘ nn) x = aa powr real-of-int
(nn x)
          by (auto simp: powr-int order.strict-implies-order[OF nn-pos])
      qed auto
    qed
    ultimately show ?thesis
      apply (elim metric-tendsto-imp-tendsto)
      apply (auto intro!:powr-mono2 elim!:eventually-mono)
      using nn-pos ff-pos by (meson le-cases not-le)+
  qed
  then have (λn. (d (Suc n))^(nat (nn n)) * (a n / a (Suc n))) ⟶ 0
  proof (elim filterlim-mono-eventually)
    show ∀F x in sequentially. ff x powr (nn x) = d (Suc x) ^ nat (nn x) * (a x
/ a (Suc x))
      apply (rule eventuallyI)
      subgoal for x
        unfolding ff-def
        using a[rule-format,of x] a[rule-format,of Suc x] d[rule-format,of Suc x]
nn-pos[of x]
        apply (auto simp add:field-simps powr-divide powr-powr powr-mult )
        by (simp add: powr-int)
      done
  qed auto
  moreover have (λn. b (Suc n) / (d (Suc n))^(nat (nn n))) ⟶ 0
  using tendsto-inverse-0-at-top[OF assu3,THEN LIMSEQ-ignore-initial-segment[where
k=1]]
  unfolding nn-def by (auto simp add:field-simps nat-mult-distrib nat-power-eq)
  ultimately have (λn. b (Suc n) * (a n / a (Suc n))) ⟶ 0
    apply −
    subgoal premises asm
      using tendsto-mult[OF asm,simplified]
      apply (elim filterlim-mono-eventually)
        using d by (auto simp add:algebra-simps,metis (mono-tags, lifting) al-
ways-eventually
          not-one-less-zero)
    done
  then have (λn. (b (Suc n) / b n) * (a n / a (Suc n))) ⟶ 0
    apply (elim Lim-transform-bound[rotated])
    apply (rule eventuallyI)
    subgoal for x using a[rule-format, of x] a[rule-format, of Suc x]
        b[rule-format, of x] b[rule-format, of Suc x]
      by (auto simp add:field-simps)
    done
  then show ?thesis unfolding c-def by (auto simp add:algebra-simps)
qed
from c-ratio-tendsto
have (λn. norm (b (Suc n) / a (Suc n)) / norm (b n / a n)) ⟶ 0
```

```
    unfolding c-def
   using a b by (force simp add:divide-simps abs-of-pos intro: Lim-transform-eventually)
  from summable-ratio-test-tendsto[OF - - this] a b
  show summable c unfolding c-def
    by (metis c-def c-pos less-irrefl zero-less-one)
  have ∀F n in sequentially. (∑j. c (n + j)) ≤ 2 * c n
  proof -
    obtain N where N-ratio:∀n≥N. c (n + 1) / c n < 1 / 2
    proof -
      have eventually (λn. c (n+1) / c n < 1/2) sequentially
        using c-ratio-tendsto[unfolded tendsto-iff,rule-format, of 1/2,simplified]
        apply eventually-elim
        subgoal for n using c-pos[of n] c-pos[of Suc n] by auto
        done
      then show ?thesis using that unfolding eventually-sequentially by auto
    qed
    have (∑j. c (j + n)) ≤ 2 * c n when n≥N for n
    proof -
      have (∑j<m. c (j + n)) ≤ 2*c n * (1 - 1 / 2^(m+1)) for m
      proof (induct m)
        case 0
        then show ?case using c-pos[of n] by simp
      next
        case (Suc m)
        have (∑j<Suc m. c (j + n)) = c n + (∑i<m. c (Suc i + n))
          unfolding sum.lessThan-Suc-shift by simp
        also have ... ≤ c n + (∑i<m. c (i + n) / 2)
        proof -
          have c (Suc i + n) ≤ c (i + n) / 2 for i
            using N-ratio[rule-format,of i+n] ‹n≥N› c-pos[of i+n] by simp
          then show ?thesis by (auto intro:sum-mono)
        qed
        also have ... = c n + (∑i<m. c (i + n)) / 2
          unfolding sum-divide-distrib by simp
        also have ... ≤ c n + c n * (1 - 1 / 2 ^(m + 1))
          using Suc by auto
        also have ... = 2 * c n * (1 - 1 / 2 ^(Suc m + 1))
          by (auto simp add:field-simps)
        finally show ?case .
      qed
      then have (∑j<m. c (j + n)) ≤ 2*c n for m
        using c-pos[of n]
      by (smt divide-le-eq-1-pos divide-pos-pos nonzero-mult-div-cancel-left zero-less-power)
      moreover have summable (λj. c (j + n))
        using ‹summable c› by (simp add: summable-iff-shift)
       ultimately show ?thesis using suminf-le-const[of λj. c (j+n) 2*c n] by
auto
    qed
   then show ?thesis unfolding eventually-sequentially by (auto simp add:algebra-simps)
```

```
  qed
  then show ∀F n in sequentially. (∑j. (b (n + j)) / (a (n + j))) ≤ 2 ∗ b n /
a n
    unfolding c-def by simp
qed


theorem Hancl3:
  fixes d ::nat⇒real and  a b :: nat⇒int
  assumes A > 1 and d: ∀n. d n > 1 and a: ∀n. a n>0 and b: ∀n. b n > 0
and s>0
    and assu1: (λn. (a n) powr(1 / of-int(2^n))) ⟶ A
    and assu2: ∀n ≥ s. A / (a n) powr (1 / of-int(2^n)) > (∏j. d (n+j))
    and assu3: LIM n sequentially. d n ^ 2 ^ n / b n :> at-top
    and convergent-prod d
  shows (∑n. b n / a n) ∉ ℚ
proof (rule ccontr)
  assume asm: ¬ ((∑n. b n / a n) ∉ ℚ)
  have ab-sum: summable (λj. b j / a j)
    using issummable[OF ‹A>1› d a b assu1 assu3 ‹convergent-prod d›] .
  obtain p q ::int where q>0 and pq-def: (λn. b (n+1) / a (n+1)) sums (p/q)
  proof −
    from asm have (∑n. b n / a n) ∈ ℚ by auto
    then have (∑n. b (n+1) / a (n+1)) ∈ ℚ
      apply (subst suminf-minus-initial-segment[OF ab-sum,of 1])
      by auto
    then obtain p' q' ::int where q'≠0 and pq-def: (λ n. b (n+1) / a (n+1) )
sums (p'/q')
      unfolding Rats-eq-int-div-int
     using summable-ignore-initial-segment[OF ab-sum,of 1,THEN summable-sums]
      by force
    define p q where p ≡ (if q'<0 then −p' else p') and q ≡ (if q'<0 then −q'
else q')
    have p'/q' = p/q q>0
      using ‹q'≠0› unfolding p-def q-def by auto
    then show ?thesis using that[of q p] pq-def by auto
  qed

  define ALPHA where
      ALPHA = (λn. of-int q ∗ (∏j=1..n. of-int(a j)) ∗ (∑j. (b (j+n+1)/a
(j+n+1))))
  have ALPHA n ≥ 1 for n
  proof −
    have (∑n. b (n+1) / a (n+1)) > 0
    proof (rule suminf-pos)
      show summable (λn. b (n + 1) / real-of-int (a (n + 1)))
        using summable-ignore-initial-segment[OF ab-sum,of 1] by auto
      show ⋀n. 0 < b (n + 1) / a (n + 1)
        using a b by simp
```

```
    qed
    then have p/q > 0
      using pq-def sums-unique by force
    then have q≥1 p≥1 using ‹q>0› by (auto simp add:divide-simps)
    moreover have ∀ n. 1 ≤ a n ∀ n. 1 ≤ b n using a b
      by (auto simp add: int-one-le-iff-zero-less)
    ultimately show ?thesis unfolding ALPHA-def
      using show7[OF - - - - pq-def] by auto
  qed
  moreover have ∃ n. ALPHA n < 1 unfolding ALPHA-def
  proof (rule lasttoshow[OF d a ‹s>0› ‹q>0› ‹A>1› b - assu3])
    show ∀F n in sequentially. (∑ j. b (n+j) / a (n+j)) ≤ (2 ∗ b n) / a n
      using show5[OF ‹A>1› d a b assu1 assu3 ‹convergent-prod d›] by simp
    show ∀ m≥s. ∃ n≥m. d (n+1) ∗ (MAX j∈{s..n}. a j powr (1 / of-int (2 ^ j)))
                  ≤ a (n+1) powr (1 / of-int (2 ^ (n+1)))
      apply (rule show9[OF ‹A>1› d a ‹s>0› assu1 ‹convergent-prod d›])
        using show8[OF ‹A>1› d a ‹s>0› ‹convergent-prod d› assu2] by (simp
add:algebra-simps)
  qed
  ultimately show False using not-le by blast
qed

corollary Hancl3corollary:
  fixes A::real and  a b :: nat⇒int
  assumes A > 1 and a: ∀ n. a n>0 and b: ∀ n. b n>0
    and assu1: (λn. (a n) powr(1 / of-int(2^n))) ⟶ A
    and asscor2: ∀ n ≥ 6. a n powr(1 / of-int (2^n)) ∗ (1 + 4∗(2/3)^n) ≤ A
                          ∧ b n ≤ 2 powr (4/3)^(n−1)
  shows (∑ n. b n / a n) ∉ ℚ
proof−
  define d::nat⇒real where d= (λ n. 1+(2/3)^(n+1))
  have dgt1:∀ n. 1 < d n  unfolding d-def by auto
  moreover have convergent-prod d
    unfolding d-def
   by (simp add: abs-convergent-prod-imp-convergent-prod summable-imp-abs-convergent-prod)
  moreover have ∀ n≥6. (∏ j. d (n+j)) < A / a n powr (1 / of-int (2 ^ n))
  proof (intro strip)
    fix n::nat assume 6 ≤ n
    have d-sum:summable (λj. ln (d j)) unfolding d-def
      by (auto intro: summable-ln-plus)

    have (∑ j. ln (d (n + j))) < ln (1+4 ∗ (2/3) ^ n)
    proof −
      define c::real where c = (2/3) ^ n
      have 0<c c<1/8
      proof −
        have c = (2/3)^6 ∗ (2/3) ^(n−6)
          unfolding c-def using ‹n≥6›
          by (metis le-add-diff-inverse power-add)
```

```
      also have ... ≤ (2/3)^6 by (auto intro:power-le-one)
      also have ... < 1/8 by (auto simp add:field-simps)
      finally show c < 1/8 .
    qed (simp add:c-def)

    have (∑j. ln (d (n + j))) ≤ (∑j. (2/3) ^(n + j + 1))
    proof (rule suminf-le)
      show ⋀j. ln (d (n + j)) ≤ (2/3) ^(n + j + 1)
        unfolding d-def
     by (metis divide-pos-pos less-eq-real-def ln-add-one-self-le-self zero-less-numeral
zero-less-power)
      show summable (λj. ln (d (n + j)))
        using summable-ignore-initial-segment[OF d-sum]
        by (force simp add: algebra-simps)
      show summable (λj. (2 / 3::real) ^(n + j + 1))
       using summable-geometric[THEN summable-ignore-initial-segment,of 2/3
n+1]
        by (auto simp add:algebra-simps)
    qed
    also have ... = (∑j. (2/3)^(n+1)*(2/3) ^j)
      by (auto simp add:algebra-simps power-add)
    also have ... = (2/3)^(n+1) * (∑j. (2/3) ^j)
      by (force intro!: summable-geometric suminf-mult)
    also have ... = 2 * c
      unfolding c-def
      by (simp add: suminf-geometric)
    also have ... < 4 * c − (4 * c)^2
      using ‹0<c› ‹c<1/8›
    by (sos ((((A<0 * A<1) * R<1) + ((A<=0 * R<1) * (R<1/16 * [1]^2)))))
    also have ... ≤ ln (1 + 4 * c)
      apply (rule ln-one-plus-pos-lower-bound)
      using ‹0<c› ‹c<1/8› by auto
    finally show ?thesis unfolding c-def by simp
  qed
  then have exp (∑j. ln (d (n + j))) < 1 + 4 * (2/3) ^n
    by (smt (z3) divide-pos-pos ln-exp ln-ge-iff zero-less-power)
  moreover have exp (∑j. ln (d (n + j))) = (∏j. d (n + j))
  proof (subst exp-suminf-prodinf-real [symmetric])
    show ⋀k. 0 ≤ ln (d (n + k))
      using dgt1 by (simp add: less-imp-le)
    show abs-convergent-prod (λna. exp (ln (d (n + na))))
    proof (subst exp-ln)
      show ⋀j. 0 < d (n + j)
        using dgt1 le-less-trans zero-le-one by blast
      show abs-convergent-prod (λj. d (n + j))
        unfolding abs-convergent-prod-def
        using ‹convergent-prod d›
        by (simp add: dgt1 convergent-prod-iff-shift less-imp-le algebra-simps)
    qed
```

show ($\prod j.$ *exp* (*ln* (*d* (*n* + *j*)))) = ($\prod j.$ *d* (*n* + *j*))
by (*meson dgt1 exp-ln not-less not-one-less-zero order-trans*)
qed
ultimately have ($\prod j.$ *d* (*n* + *j*)) < *1* + *4* ∗ (*2*/*3*) ^ *n*
by *simp*
also have ... ≤ *A* / (*a n*) *powr* (*1* / *of-int* (*2* ^ *n*))
proof −
have *a n powr* (*1* / *real-of-int* (*2* ^ *n*)) > *0*
using *a*[*rule-format*,*of n*] **by** *auto*
then show *?thesis* **using** *asscor2*[*rule-format*,*OF* ‹*6*≤*n*›]
by (*auto simp add*:*field-simps*)
qed
finally show ($\prod j.$ *d* (*n* + *j*)) < *A* / *real-of-int* (*a n*) *powr* (*1* / *of-int* (*2* ^ *n*))
.
qed
moreover have *LIM n sequentially. d n* ^ *2* ^ *n* / *real-of-int* (*b n*) :> *at-top*
proof −
have *LIM n sequentially. d n* ^ *2* ^ *n* / *2 powr*((*4*/*3*)^(*n*−*1*)) :> *at-top*
proof −
define *n1* **where** *n1* ≡ (λ*n*. (*2*::*real*) ∗ (*3*/*2*)^(*n*−*1*))
define *n2* **where** *n2* ≡ (λ*n*. ((*4*::*real*)/*3*)^(*n*−*1*))
have *LIM n sequentially.* (((*1*+(*8*/*9*)/(*n1 n*)) *powr* (*n1 n*))/*2*) *powr* (*n2 n*)
:> *at-top*
proof (*rule filterlim-at-top-powr-real*[**where** *g*′=*exp* (*8*/*9*) / *2*])
define *e1* **where** *e1* = *exp* (*8*/*9*) / (*2*::*real*)
show *e1*>*1* **unfolding** *e1-def* **by** (*approximation 4*)
show (λ*n*. ((*1*+(*8*/*9*)/(*n1 n*)) *powr* (*n1 n*))/*2*) ⟶ *e1*
proof −
have (λ*n*. (*1*+(*8*/*9*)/(*n1 n*)) *powr* (*n1 n*)) ⟶ *exp* (*8*/*9*)
apply (*rule filterlim-compose*[*OF tendsto-exp-limit-at-top*])
unfolding *n1-def*
by (*auto intro*!: *filterlim-tendsto-pos-mult-at-top*
filterlim-realpow-sequentially-at-top
simp:*filterlim-sequentially-iff*[*of* λ*x*. (*3* / *2*) ^ (*x* − *Suc 0*) - *1*])
then show *?thesis* **unfolding** *e1-def*
by (*intro tendsto-intros*,*auto*)
qed
show *filterlim n2 at-top sequentially*
unfolding *n2-def*
apply (*subst filterlim-sequentially-iff*[*of* λ*n*. (*4* / *3*) ^ (*n* − *1*) - *1*])
by (*auto intro*:*filterlim-realpow-sequentially-at-top*)
qed
moreover have $\forall_F$ *n in sequentially.* (((*1*+(*8*/*9*)/(*n1 n*)) *powr* (*n1 n*))/*2*)
powr (*n2 n*)
= *d n* ^ *2* ^ *n* / *2 powr*((*4*/*3*)^(*n*−*1*))
proof (*rule eventually-sequentiallyI*)
fix *k*::*nat* **assume** *k* ≥ *1*
have ((*1* + *8* / *9* / *n1 k*) *powr n1 k* / *2*) *powr n2 k*
= (((*1* + *8* / *9* / *n1 k*) *powr n1 k*) *powr n2 k*) / *2 powr* (*4* / *3*) ^ (*k* −

```
1)
        by (simp add: n1-def n2-def powr-divide)
      also have ... = (1 + 8 / 9 / n1 k) powr (n1 k * n2 k) / 2 powr (4 / 3) ^
(k − 1)
        by (simp add: powr-powr)
      also have ... = (1 + 8 / 9 / n1 k) powr (2 ^ k) / 2 powr (4 / 3) ^(k −
1)
      proof −
        have n1 k * n2 k = 2 ^ k
          unfolding n1-def n2-def
          using ‹k≥1› by (simp add: mult-ac flip:power-mult-distrib power-Suc)
        then show ?thesis by simp
      qed
      also have ... = (1 + 8 / 9 / n1 k) ^(2 ^ k) / 2 powr (4 / 3) ^(k − 1)
        unfolding n1-def
          by (smt (verit, best) powr-realpow divide-pos-pos numeral-plus-numeral
numeral-plus-one of-nat-numeral of-nat-power semiring-norm(2) zero-less-power)
      also have ... = d k ^ 2 ^ k / 2 powr (4 / 3) ^(k − 1)
      proof −
        have **: 8 / 9 / n1 k = (2/3) ^(k+1)
          unfolding n1-def using ‹k≥1›
          by (simp add: divide-simps split: nat-diff-split)
        then show ?thesis
          unfolding d-def by presburger
      qed
      finally show ((1 + 8 / 9 / n1 k) powr n1 k / 2) powr n2 k
                = d k ^ 2 ^ k / 2 powr (4 / 3) ^(k − 1) .
    qed
    ultimately show ?thesis using filterlim-cong by fast
  qed
  moreover have ∀F n in sequentially. d n ^ 2 ^ n / 2 powr((4/3)^(n−1))
      ≤ d n ^ 2 ^ n / real-of-int (b n)
    using eventually-sequentiallyI[of 6]
    by (smt (verit, best) asscor2 b dgt1 frac-le of-int-0-less-iff zero-le-power)
  ultimately show ?thesis by (auto elim: filterlim-at-top-mono)
 qed
 ultimately show ?thesis using Hancl3[OF ‹A>1› - a b - assu1,of d 6] by force
qed

end
```

4 Acknowledgements

A. K.-A. and W.L. were supported by the ERC Advanced Grant ALEXANDRIA (Project 742178) funded by the European Research Council and led by Professor Lawrence Paulson at the University of Cambridge, UK.

References

[1] J. Hančl. Irrational rapidly convergent series. *Rendiconti del Seminario Matematico della Università di Padova*, 107:225–231, 2002.