The Transcendence of Certain Infinite Series

Angeliki Koutsoukou-Argyraki and Wenda Li

May 14, 2024

Abstract

We formalize the proofs of two transcendence criteria by J. Hančl and P. Rucki that assert the transcendence of the sums of certain infinite series built up by sequences that fulfil certain properties. Both proofs make use of Roth's celebrated theorem on diophantine approximations to algebraic numbers from 1955 which we implement as an assumption without having formalised its proof.

Contents

1 The transcendence of certain infinite series

theory *Transcendence-Series* **imports**
HOL−Analysis.Multivariate-Analysis
HOL−Computational-Algebra.Polynomial
Prime-Number-Theorem.Prime-Number-Theorem-Library
begin

We formalise the proofs of two transcendence criteria by J. Hančl and P. Rucki that assert the transcendence of the sums of certain infinite series built up by sequences that fulfil certain properties (Theorems 2.1 and 2.2 in [1], HanclRucki1 and HanclRucki2 here respectively). Both proofs make use of Roth's celebrated theorem on diophantine approximations to algebraic numbers from 1955 [2] which we assume and implement within the locale RothsTheorem.

A small mistake was detected in the original proof of Theorem 2.1, and the authors gave us a fix for the problem (by email). Our formalised proof incorporates this correction (see the Remark in the proof of HanclRucki1).

1.1 Misc

lemma *powr-less-inverse-iff*:
  fixes *x y z*::*real*
  assumes *x>0 y>0 z>0*
  shows *x powr y* < *z* ⟷ *x* < *z powr* (*inverse y*)
proof
  assume *x powr y* < *z*
  from *powr-less-mono2*[*OF - - this*,*of inverse y*]
  show *x* < *z powr inverse y*
    using *assms* **by** (*auto simp*:*powr-powr*)
next
  assume ∗:*x* < *z powr inverse y*
  from *powr-less-mono2*[*OF - - ∗*,*of y*] **show** *x powr y* < *z*
    using *assms* **by** (*auto simp*:*powr-powr*)
qed

lemma *powr-less-inverse-iff'*:
  fixes *x y z*::*real*
  assumes *x>0 y>0 z>0*
  shows *z*< *x powr y* ⟷ *z powr* (*inverse y*) < *x*
  using *powr-less-inverse-iff*[*symmetric*,*of - inverse y*] *assms* **by** *auto*

lemma *powr-less-eq-inverse-iff*:
  fixes *x y z*::*real*
  assumes *x>0 y>0 z>0*
  shows *x powr y* ≤ *z* ⟷ *x* ≤ *z powr* (*inverse y*)
  by (*meson assms not-less powr-less-inverse-iff'*)

lemma *powr-less-eq-inverse-iff'*:
  fixes *x y z*::*real*
  assumes *x>0 y>0 z>0*
  shows *z* ≤ *x powr y* ⟷ *z powr* (*inverse y*) ≤ *x*
  by (*simp add*: *assms powr-less-eq-inverse-iff*)

lemma *tendsto-PInfty-mono*:
  assumes (*ereal o f*) ⟶ ∞ $\forall_F$ *x in sequentially*. *f x* ≤ *g x*
  shows (*ereal o g*) ⟶ ∞
  using *assms* **unfolding** *comp-def tendsto-PInfty-eq-at-top*
  by (*elim filterlim-at-top-mono*, *simp*)

lemma *limsup-infinity-imp-Inf-many*:
  assumes *limsup f* = ∞
  shows (∀ *m*. ($\exists_\infty$*i*. *f i* > *ereal m*)) **unfolding** *INFM-nat*
proof (*clarify*,*rule ccontr*)
  fix *m k* **assume** ¬ (∃ *n*>*k*. *ereal m* < *f n*)
  then have ∀ *n*>*k*. *f n* ≤ *ereal m* **by** *auto*
  then have $\forall_F$ *n in sequentially*. *f n* ≤ *ereal m*
    using *eventually-at-top-dense* **by** *blast*
  then have *limsup f* ≤ *ereal m* **using** *Limsup-bounded* **by** *auto*

then show *False* **using** *assms* **by** *simp*
qed

lemma *snd-quotient-plus-leq*:
defines *de*≡(*snd o quotient-of*)
shows *de* (*x*+*y*) ≤ *de x* ∗ *de y*
proof −
obtain *x1 x2 y1 y2* **where** *xy*: *quotient-of x* = (*x1*,*x2*) *quotient-of y*=(*y1*,*y2*)
by (*meson surj-pair*)
have *x2*>*0* *y2*>*0* **using** *xy quotient-of-denom-pos* **by** *blast*+
then show *?thesis*
unfolding *de-def comp-def rat-plus-code xy*
apply (*auto split*:*prod.split simp*:*Rat.normalize-def Let-def*)
by (*smt div-by-1 gcd-pos-int int-div-less-self mult-eq-0-iff mult-sign-intros*(*1*))
qed

lemma *quotient-of-inj*: *inj quotient-of*
unfolding *inj-def* **by** (*simp add*: *quotient-of-inject*)

lemma *infinite-inj-imageE*:
assumes *infinite A inj-on f A f ‘ A* ⊆ *B*
shows *infinite B*
using *assms inj-on-finite* **by** *blast*

lemma *incseq-tendsto-limsup*:
fixes *f*::*nat* ⇒ *'a*::{*complete-linorder*,*linorder-topology*}
assumes *incseq f*
shows *f* ⟶ *limsup f*
using *LIMSEQ-SUP assms convergent-def convergent-ereal tendsto-Limsup*
trivial-limit-sequentially **by** *blast*

1.2 Main proofs

Since the proof of Roth's theorem has not been formalized yet, we formalize the statement in a locale and use it as an assumption.

locale *RothsTheorem* =
assumes *RothsTheorem*:∀ ξ κ. *algebraic* ξ ∧ ξ ∉ ℚ ∧ *infinite* {(*p*,*q*). *q*>*0* ∧
coprime p q ∧ |ξ − *of-int p* / *of-int q*| < *1* / *q powr* κ} ⟶ κ ≤ *2*

theorem (**in** *RothsTheorem*) *HanclRucki1*:
fixes *a b* ::*nat*⇒*int* **and** δ ::*real*
defines *aa*≡(λ*n*. *real-of-int* (*a n*)) **and** *bb*≡(λ*n*. *real-of-int* (*b n*))
assumes *a-pos*:∀ *k*. *a k* >*0* **and** *b-pos*:∀ *k*. *b k* >*0* **and** δ >*0*
and *limsup-infy*:*limsup* (λ *k*. *aa* (*k*+*1*)/(∏ *i* = *0*..*k*. *aa i*)*powr*(*2*+δ)∗(*1*/*bb*
(*k*+*1*))) = ∞
and *liminf-1*:*liminf* (λ*k*. *aa* (*k*+*1*) / *aa k* ∗ *bb k* / *bb* (*k*+*1*)) > *1*
shows ¬ *algebraic*(*suminf* (λ *k*. *bb k* / *aa k*))
proof −
have *summable*:*summable* (λ *k*. *bb k* / *aa k*)

```
proof (rule ratio-test-convergence)
  have [simp]:aa k>0 bb k>0 for k
    unfolding aa-def bb-def using a-pos b-pos by auto
  show ∀F n in sequentially. 0 < bb n / aa n
    by auto
  show 1 < liminf (λn. ereal (bb n / aa n / (bb (Suc n) / aa (Suc n))))
    using liminf-1 by (auto simp:algebra-simps)
qed
have [simp]: aa k>0 bb k>0 for k unfolding aa-def bb-def
  by (auto simp add: a-pos b-pos)
have ab-1:aa k≥1 bb k≥1 for k
  unfolding aa-def bb-def using a-pos b-pos
  by (auto simp add: int-one-le-iff-zero-less)

define B where B ≡ liminf (λx. ereal (aa (x + 1) / aa x ∗ bb x / bb (x + 1)))
define M where M ≡ (case B of ereal m ⇒ (m+1)/2 | - ⇒ 2)
have M > 1 M < B
  using liminf-1 unfolding M-def
  by (auto simp add: M-def B-def split: ereal.split)
```

Remark: In the original proof of Theorem 2.1 in [1] it was claimed in p.534 that from assumption (3) (i.e. *1 < liminf (λx. ereal (aa (x + 1) / aa x ∗ bb x / bb (x + 1)))*), we obtain that: $\forall A > 1\ \exists k_0\ \forall k > k_0\ \frac{1}{A}\frac{b_k}{a_k} > \frac{b_{k+1}}{a_{k+1}}$, however note the counterexample where $a_{k+1} = k(a_1 a_2 ... a_k)^{\lceil 2+\delta \rceil}$ if k is odd, and $a_{k+1} = 2a_k$ otherwise, with $b_k = 1$ for all k. In commmunication by email the authors suggested to replace the claim $\forall A > 1\ \exists k_0\ \forall k > k_0\ \frac{1}{A}\frac{b_k}{a_k} > \frac{b_{k+1}}{a_{k+1}}$ with $\exists A > 1\ \exists k_0\ \forall k > k_0\ \frac{1}{A}\frac{b_k}{a_k} > \frac{b_{k+1}}{a_{k+1}}$ which solves the problem and the proof proceeds as in the paper. The witness for $\exists A > 1$ is denoted by M here.

```
have bb-aa-event:∀F k in sequentially. (1/M)∗(bb k / aa k)> bb(k+1)/ aa (k+1)
  using less-LiminfD[OF ‹M < B›[unfolded B-def],simplified]
  by eventually-elim (use ‹M > 1› in ‹auto simp: field-simps›)

have bb-aa-p:∀F k in sequentially. ∀p. bb(k+p)/ aa (k+p) ≤ (1/M^p)∗(bb k /
aa k)
proof −
  obtain k0 where k0-ineq:
     ∀n≥k0. bb (n + 1) / aa (n + 1) < 1 / M ∗ (bb n / aa n)
    using bb-aa-event unfolding eventually-sequentially
    by auto
  have bb(k+p)/ aa (k+p) ≤ (1/M^p)∗(bb k / aa k) when k≥k0 for p k
  proof (induct p)
    case 0
    then show ?case by auto
  next
    case (Suc p)
    have  bb (k + Suc p) / aa (k + Suc p)  < 1 / M ∗ (bb (k+p) / aa (k+p))
```

```
      using k0-ineq[rule-format,of k+p] that by auto
    also have ... ≤ 1 / M ^(Suc p) ∗ (bb k / aa k)
      using Suc ‹M>1› by (auto simp add:field-simps)
    finally show ?case by auto
  qed
  then show ?thesis unfolding eventually-sequentially by auto
qed

define ξ where ξ = suminf (λ k. bb k / aa k)
have ξ-Inf-many:∃∞ k. |ξ − (∑k = 0..k. bb k / aa k)| < 1 / prod aa {0..k}
powr (2 + δ)
proof −
  have |ξ − (∑i = 0..k. bb i / aa i)| = |∑i. bb (i+(k+1)) / aa (i+(k+1))|
for k
    unfolding ξ-def
    apply (subst suminf-minus-initial-segment[of - k+1,OF summable])
    using atLeast0AtMost lessThan-Suc-atMost by auto

  moreover have ∃∞ k. |∑i. bb(i+(k+1))/ aa (i+(k+1))|
                        < 1 / prod aa {0..k} powr (2 + δ)
  proof −
    define P where P ≡ (λ i. ∀p. bb (i + 1 + p) / aa (i + 1 + p)
                                  ≤ 1 / M ^ p ∗ (bb (i + 1) / aa (i + 1)))
    define Q where Q ≡ (λ i. ereal (M / (M − 1))
              < ereal (aa (i + 1) / prod aa {0..i} powr (2 + δ) ∗ (1 / bb (i +
1))))
    have ∀∞ i. P i
      using bb-aa-p[THEN sequentially-offset, of 1] cofinite-eq-sequentially
      unfolding P-def by auto
    moreover have ∃∞i. Q i
     using limsup-infy[THEN limsup-infinity-imp-Inf-many,rule-format,of (M /
(M −1))]
      unfolding Q-def .
    moreover have |∑i. bb(i+(k+1))/ aa (i+(k+1))|
                        < 1 / prod aa {0..k} powr (2 + δ)
      when P k Q k for k
    proof −
      have summable-M:summable (λi. 1 / M ^ i)
        apply (rule summable-ratio-test[of 1/M])
        using ‹M>1› by auto

      have (∑i. bb (i + (k + 1)) / aa (i + (k + 1))) ≥ 0
        apply (rule suminf-nonneg)
       subgoal using summable-ignore-initial-segment[OF summable,of k+1] by
auto
        subgoal by (simp add: less-imp-le)
        done
      then have |∑i. bb (i + (k + 1)) / aa (i + (k + 1))|
                = (∑i. bb (i + (k + 1)) / aa (i + (k + 1)))
```

```
        by auto
      also have ... ≤ (∑ i. 1 / M ^ i * (bb (k + 1) / aa (k + 1)))
        apply (rule suminf-le)
        subgoal using that(1) unfolding P-def by (auto simp add:algebra-simps)
        subgoal using summable-ignore-initial-segment[OF summable,of k+1] by
auto
        subgoal using summable-mult2[OF summable-M,of bb (k + 1) / aa (k
+ 1)]
          by auto
        done
      also have ... = (bb (k + 1) / aa (k + 1)) * (∑ i. 1 / M ^ i)
        using suminf-mult2[OF summable-M,of bb (k + 1) / aa (k + 1)]
        by (auto simp:algebra-simps)
      also have ... = (bb (k + 1) / aa (k + 1)) * (∑ i. (1 / M) ^ i)
        using ‹M>1› by (auto simp: field-simps)
      also have ... = (bb (k + 1) / aa (k + 1)) * (M / (M −1))
        apply (subst suminf-geometric)
        using ‹M>1› by (auto simp: field-simps)
      also have ... < (bb (k + 1) / aa (k + 1)) * (aa (k + 1) /
                          prod aa {0..k} powr (2 + δ) * (1 / bb (k + 1)))
        apply (subst mult-less-cancel-left-pos)
        using that(2) unfolding Q-def by auto
      also have ... = 1/ prod aa {0..k} powr (2 + δ)
        using ab-1[of Suc k] by auto
      finally show ?thesis .
    qed
    ultimately show ?thesis by (smt INFM-conjI INFM-mono)
  qed
  ultimately show ?thesis by auto
qed

define pq where pq ≡ (λk. quotient-of (∑ i=0..k. of-int (b i) / of-int (a i)))
define p q where p ≡ fst ∘ pq and q = snd ∘ pq
have coprime-pq:coprime (p k) (q k)
    and q-pos:q k > 0 and pq-sum:p k / q k = (∑ i=0..k. b i / a i) for k
proof −
  have eq: quotient-of (∑ i=0..k. of-int (b i) / of-int (a i)) = (p k, q k)
    by (simp add: p-def q-def pq-def)
  from quotient-of-coprime[OF eq] show coprime (p k) (q k) .
  from quotient-of-denom-pos[OF eq] show q k > 0 .
  have (∑ i=0..k. b i / a i) = of-rat (∑ i=0..k. of-int (b i) / of-int (a i))
    by (simp add: of-rat-sum of-rat-divide)
  also have (∑ i=0..k. rat-of-int (b i) / rat-of-int (a i)) =
              rat-of-int (p k) / rat-of-int (q k)
    using quotient-of-div[OF eq] by simp
  finally show p k / q k = (∑ i=0..k. b i / a i) by (simp add:of-rat-divide)
qed

have ξ-Inf-many2:∃∞ k. |ξ − p k / q k| < 1 / q k powr (2 + δ)
```

```
    using ξ-Inf-many
  proof (elim INFM-mono)
    fix k assume asm:|ξ − (∑k = 0..k. bb k / aa k)| < 1 / prod aa {0..k} powr
(2 + δ)
    have |ξ − real-of-int (p k) / real-of-int (q k)|
            = |ξ − (∑k = 0..k. bb k / aa k)|
      using pq-sum unfolding aa-def bb-def by auto
    also have ... < 1 / prod aa {0..k} powr (2 + δ)
      using asm by auto
    also have ... ≤ 1 / q k powr (2 + δ)
    proof −
      have q k ≤ prod aa {0..k}
      proof (induct k)
        case 0
        then show ?case unfolding q-def pq-def aa-def
          apply (simp add:rat-divide-code of-int-rat quotient-of-Fract)
          using ab-1[of 0,unfolded aa-def bb-def] unfolding Let-def normalize-def
          apply auto
        by (metis div-by-1 gcd-pos-int less-imp-le less-trans nonneg1-imp-zdiv-pos-iff

            not-less zdiv-mono2)
      next
        case (Suc k)
        define de where de ≡snd ∘ quotient-of
        have real-of-int (q (Suc k))
                = de (∑i=0..Suc k. of-int (b i) / of-int (a i))
          unfolding q-def pq-def de-def by simp
        also have ... = de ((∑i=0..k. of-int (b i) / of-int (a i))
                          + of-int (b (Suc k)) / of-int (a (Suc k)))
          by simp
        also have ... ≤ de (∑i=0..k. of-int (b i) / of-int (a i))
                          ∗ de (of-int (b (Suc k)) / of-int (a (Suc k)))
          using snd-quotient-plus-leq[folded de-def] by presburger
        also have ... = q k ∗ de (of-int (b (Suc k)) / of-int (a (Suc k)))
          unfolding q-def pq-def de-def by auto
        also have ... = q k ∗ snd (Rat.normalize (b (Suc k), a (Suc k)))
          by (simp add:rat-divide-code of-int-rat quotient-of-Fract de-def)
        also have ... ≤ q k ∗ aa (Suc k)
          using ab-1[of Suc k] q-pos[of k]
          unfolding normalize-def aa-def bb-def Let-def
          apply auto
          by (metis div-by-1 int-one-le-iff-zero-less less-trans
                nonneg1-imp-zdiv-pos-iff not-less zdiv-mono2 zero-less-one)
        also have ... ≤  prod aa {0..k} ∗ aa (Suc k)
          using Suc ab-1[of Suc k] by auto
        also have ... = prod aa {0..Suc k}
          by (simp add: prod.atLeast0-atMost-Suc)
        finally show ?case .
      qed
```

```
    then show ?thesis
      by (smt ‹0 < δ› frac-le of-int-0 of-int-le-iff powr-gt-zero
            powr-mono2 q-pos)
  qed
  finally show |ξ − real-of-int (p k) / real-of-int (q k)| < 1 / real-of-int (q k)
powr (2 + δ) .
qed

define pqs where pqs ≡{(p, q). q>0 ∧ coprime p q
    ∧ |ξ − real-of-int p / real-of-int q| < 1 / q powr (2 + δ)}
have ξ-infinite:infinite pqs
proof −
  define A where A ≡ {k. |ξ −  (p k) /  (q k)| < 1 / (q k) powr (2 + δ)}
  have ∃∞ k. |ξ − p k / q k| <  1 / q k powr (2 + δ)
    using ξ-Inf-many2 .
  then have infinite A
    unfolding Inf-many-def A-def by auto
  moreover have inj-on (λk. (p k, q k)) A
  proof −
    define g where g ≡ (λi. rat-of-int (b i) / rat-of-int (a i))
    define f where f ≡ (λk. ∑ i = 0..k. g i)
    have g-pos:g i>0 for i
      unfolding g-def by (simp add: a-pos b-pos)
    have strict-mono f unfolding strict-mono-def f-def
    proof safe
      fix x y::nat assume x < y
      then have sum g {0..y} − sum g {0..x} = sum g {x<..y}
        apply (subst  Groups-Big.sum-diff[symmetric])
        by (auto intro:arg-cong2[where f=sum])
      also have ... > 0
        apply (rule ordered-comm-monoid-add-class.sum-pos)
        using ‹x<y› g-pos by auto
      finally have sum g {0..y} − sum g {0..x} >0 .
      then show sum g {0..x} < sum g {0..y} by auto
    qed
    then have inj f using strict-mono-imp-inj-on by auto
    then have inj (quotient-of o f) by (simp add: inj-compose quotient-of-inj)
    then have inj (λk. (p k, q k))
      unfolding f-def p-def q-def pq-def comp-def
      apply (fold g-def)
      by auto
    then show ?thesis by (auto elim:subset-inj-on)
  qed
  moreover have (λk. (p k, q k)) ‘ A ⊆ pqs
    unfolding A-def pqs-def using coprime-pq q-pos by auto
  ultimately show ?thesis
    apply (elim infinite-inj-imageE)
    by auto
qed
```

```
  moreover have finite pqs if ξ ∈ ℚ
  proof −
    obtain m n where ξ-mn:ξ = (of-int m / of-int n) and coprime m n n>0
    proof −
      obtain m n where mn:|ξ| = (of-nat m / of-nat n) coprime m n n≠0
        using Rats-abs-nat-div-natE[OF ‹ξ ∈ ℚ› Rats-abs-nat-div-natE]
        by metis
      define m′ and n′::int
        where m′=(if ξ > 0 then nat m else −nat m) and n′=nat n
      then have ξ = (of-int m′ / of-int n′) coprime m′ n′ n′>0
        using mn by auto
      then show ?thesis using that by auto
    qed
     have pqs ⊆ {(m,n)} ∪ {x. x ∈pqs ∧ − |m| − 1 ≤fst x ∧ fst x ≤ |m| + 1 ∧
0<snd x ∧ snd x < n }
    proof (rule subsetI)
      fix x assume x ∈ pqs
      define p q where p ≡ fst x and q=snd x
      have q>0 coprime p q and pq-less:|ξ − p / q| < 1 / q powr (2 + δ)
        using ‹x∈pqs› unfolding p-def q-def pqs-def by auto
      have q-lt-n:q<n when m≠p ∨ n≠q
      proof −
       have m ∗ q ≠ n ∗ p using that ‹coprime m n› ‹coprime p q› ‹q>0› ‹n>0›
         by (metis eq-rat(1) fst-conv int-one-le-iff-zero-less mult.commute normal-
ize-stable
                not-one-le-zero quotient-of-Fract snd-conv)
        then have 1/(n∗q) ≤ |m/n − p/q|
          using ‹q>0› ‹n>0›
          apply (auto simp:field-simps)
       by (metis add-diff-cancel-left′ diff-diff-eq2 diff-zero less-irrefl not-le of-int-diff

                of-int-lessD of-int-mult)
        also have ... < 1 / q powr (2 + δ)
          using pq-less unfolding ξ-mn by auto
        also have ... ≤ 1 / q ^2
        proof −
          have real-of-int (q²) = q powr 2
            apply (subst powr-numeral)
            unfolding power2-eq-square using ‹q>0› by auto
          also have ... ≤ q powr (2 + δ)
            apply (rule powr-mono)
            using ‹q>0› ‹δ>0› by auto
          finally have real-of-int (q²) ≤ real-of-int q powr (2 + δ) .
          moreover have real-of-int q powr (2 + δ) > 0 using ‹0 < q› by auto
          ultimately show ?thesis by (auto simp:field-simps)
        qed
        finally have  1 /  (n ∗ q) < 1 / q² .
        then show ?thesis  using ‹q>0› ‹n>0›
          unfolding power2-eq-square by (auto simp:field-simps)
```

qed
moreover have − |*m*| − *1* ≤ *p* ∧ *p* ≤ |*m*| + *1* **when** *m*≠*p* ∨ *n*≠*q*
proof −
define *qn* **where** *qn* ≡ *q*/*n*
have *0*<*qn* *qn*<*1* **unfolding** *qn-def* **using** *q-lt-n*[*OF* ‹*m*≠*p* ∨ *n*≠*q*›] ‹*q*>*0*› **by** *auto*

have |*m*/*n* − *p* / *q*| < *1* / *q* *powr* (*2* + δ) **using** *pq-less* **unfolding** *ξ-mn* **by** *simp*
then have |*p* / *q* − *m*/*n*| < *1* / *q* *powr* (*2* + δ) **by** *simp*
then have *m*/*n*− *1* / *q* *powr* (*2* + δ) < *p*/*q* ∧ *p*/*q* < *m*/*n* + *1* / *q* *powr* (*2* + δ)
unfolding *abs-diff-less-iff* **by** *auto*
then have *qn*∗*m*− *q* / *q* *powr* (*2* + δ) < *p* ∧ *p* < *qn*∗*m* + *q* / *q* *powr* (*2* + δ)
unfolding *qn-def* **using** ‹*q*>*0*› **by** (*auto* *simp*:*field-simps*)
moreover have − |*m*| − *1* ≤ *qn*∗*m*− *q* / *q* *powr* (*2* + δ)
proof −
have − |*m*| ≤ *qn*∗*m* **using** ‹*0*<*qn*› ‹*qn*<*1*›
apply (*cases* *m*≥*0*)
subgoal
apply *simp*
by (*meson less-eq-real-def mult-nonneg-nonneg neg-le-0-iff-le of-int-0-le-iff order-trans*)
subgoal by *simp*
done
moreover have − *1* ≤ − *q* / *q* *powr* (*2* + δ)
proof −
have *q* = *q* *powr* *1* **using** ‹*0* < *q*› **by** *auto*
also have ... ≤*q* *powr* (*2* + δ)
apply (*rule* *powr-mono*)
using ‹*q*>*0*› ‹δ>*0*› **by** *auto*
finally have *q* ≤ *q* *powr* (*2* + δ) **.**
then show *?thesis* **using** ‹*0* < *q*› **by** *auto*
qed
ultimately show *?thesis* **by** *auto*
qed
moreover have *qn*∗*m* + *q* / *q* *powr* (*2* + δ) ≤ |*m*| + *1*
proof −
have *qn*∗*m* ≤ |*m*| **using** ‹*0*<*qn*› ‹*qn*<*1*›
apply (*cases* *m*≥*0*)
subgoal by (*simp* *add*: *mult-left-le-one-le*)
subgoal by (*smt* *of-int-0-le-iff* *zero-le-mult-iff*)
done
moreover have *q* / *q* *powr* (*2* + δ) ≤ *1*
proof −
have *q* = *q* *powr* *1* **using** ‹*0* < *q*› **by** *auto*
also have ... ≤*q* *powr* (*2* + δ)
apply (*rule* *powr-mono*)

```
          using ‹q>0› ‹δ>0› by auto
        finally have q ≤ q powr (2 + δ) .
        then show ?thesis using ‹0 < q› by auto
      qed
      ultimately show ?thesis by auto
    qed
    ultimately show ?thesis by auto
  qed
  ultimately show x ∈ {(m, n)} ∪ {x ∈ pqs. − |m| − 1 ≤ fst x ∧ fst x ≤ |m|
+ 1
                    ∧ 0 < snd x ∧ snd x < n}
    using ‹x ∈ pqs› ‹q>0› unfolding p-def q-def by force
qed
moreover have finite {x. x ∈pqs ∧ − |m| − 1 ≤fst x ∧ fst x ≤ |m| + 1 ∧
0<snd x ∧ snd x < n }
proof −
  have finite ({− |m| − 1..|m| +1} × {0<..<n}) by blast
  moreover have {x. x ∈pqs ∧ − |m| − 1 ≤fst x ∧ fst x ≤ |m| + 1 ∧ 0<snd
x ∧ snd x < n } ⊆
          ({− |m| − 1..|m| +1} × {0<..<n})
    by auto
  ultimately show ?thesis
    apply (elim rev-finite-subset)
    by blast
qed
ultimately show ?thesis using finite-subset by auto
qed
ultimately show ?thesis
  apply (fold ξ-def)
  using RothsTheorem[rule-format,of ξ 2 + δ,folded pqs-def] ‹δ >0› by auto
qed


theorem (in RothsTheorem) HanclRucki2:
  fixes a b ::nat⇒int and δ ε ::real
  defines aa≡(λn. real-of-int (a n)) and bb≡(λn. real-of-int (b n))
  assumes a-pos:∀ k. a k >0 and b-pos:∀ k. b k >0 and δ >0
    and ε >0
    and limsup-infi:limsup (λ k.(aa (k+1)/(∏ i = 0..k. aa i)powr(2+(2/ε) + δ))
        ∗ (1/(bb (k+1)))) = ∞
    and ratio-large:∀ k. ( k ≥ t −→ (( aa(k+1)/bb( k+1)) ) powr(1/(1+ε))
          ≥ (( aa k/bb k) powr(1/(1+ε)))+1)
  shows ¬ algebraic(suminf (λ k. bb k /aa k))
proof−
  have aa-bb-pos[simp]:aa k>0 bb k>0 for k
    unfolding aa-def bb-def using a-pos b-pos by auto
  have summable:summable (λ k. bb k / aa k)
  proof −
    define c0 where c0≡(aa t/bb t) powr(1/(1+ε)) − t
```

```
have ab-k:(aa k / bb k) powr(1/(1+ε)) ≥ k + c0 when k≥t for k
  using that
proof (induct k)
  case 0
  then show ?case unfolding c0-def by simp
next
  case (Suc k)
  have ?case when ¬ t≤k
  proof −
    have t = Suc k using that Suc.prems by linarith
    then show ?thesis unfolding c0-def by auto
  qed
  moreover have ?case when t ≤ k
  proof −
    have (aa(k+1)/bb(k+1)) powr(1/(1+ε))
          ≥ ( aa k/bb k) powr(1/(1+ε))+1
      using ratio-large[rule-format,OF that] by blast
    then show ?thesis using Suc(1)[OF that] by simp
  qed
  ultimately show ?case by auto
qed
have summable (λk. 1 / (k + c0) powr (1+ε))
proof −
  have c0 + t > 0 unfolding c0-def
    using aa-bb-pos[of t] by (simp,linarith)
  then have summable (λk. 1 / (k + (c0+t)) powr (1+ε))
    using summable-hurwitz-zeta-real[of 1+ε c0+t]
    apply (subst (asm) powr-minus-divide)
    using ‹ε>0› by auto
  then show ?thesis
    apply (rule-tac summable-offset[of - t])
    by (auto simp:algebra-simps)
qed
moreover have bb k / aa k ≤ 1 / (k + c0) powr (1+ε) when k≥t for k
proof −
  have (aa t / bb t) powr (1 / (1 + ε)) > 0
    apply simp
    by (metis ‹⋀k. 0 < aa k› ‹⋀k. 0 < bb k› less-numeral-extra(3))
  then have k + c0 >0 unfolding c0-def using that by linarith
  then have aa k / bb k ≥ (k + c0) powr (1+ε)
    using ab-k[OF that]
    apply (subst (asm) powr-less-eq-inverse-iff')
    using ‹ε >0› by auto
  then have inverse (aa k / bb k) ≤ inverse ((k + c0) powr (1+ε))
    apply (elim le-imp-inverse-le)
    using ‹k + c0 >0› by auto
  then show ?thesis by (simp add: inverse-eq-divide)
qed
ultimately show ?thesis
```

```
    apply (elim summable-comparison-test'[where N=t])
    using aa-bb-pos by (simp add: less-eq-real-def)
  qed

  have 7:∃∞ k. 1 / (M powr (ε/(1+ε)) * (∏ i = 0..k. aa i)
              powr(2+ δ * ε / (1+ε))) > (bb (k+1) / aa(k+1)) powr (ε / (1+ε))
    when M > 0 for M
  proof -
    define tt where tt ≡ (λi. prod aa {0..i} powr (2 + 2 / ε + δ))
    have tt-pos:tt i > 0 for i
      unfolding tt-def
      apply(subst powr-gt-zero,induct i)
      subgoal by (metis aa-bb-pos(1) order-less-irrefl prod-pos)
      subgoal by (metis ‹⋀k. 0 < aa k› order-less-irrefl prod-pos)
      done
    have ∃∞i. M < (aa (i + 1) / tt i * (1 / bb (i + 1)))
      using limsup-infinity-imp-Inf-many[OF limsup-infi,rule-format,of M]
      unfolding tt-def by auto
    then have ∃∞i. 1 / (M * tt i) > (bb (i+1) / aa (i+1))
      apply (elim INFM-mono)
      using ‹M>0› tt-pos by (auto simp:divide-simps algebra-simps)
    then have ∃∞i. (1 / (M * tt i)) powr (ε/(1+ε))
                  > (bb (i+1) / aa (i+1)) powr (ε/(1+ε))
      apply (elim INFM-mono powr-less-mono2[rotated 2])
      by (simp-all add: assms(6) pos-add-strict less-eq-real-def)
     moreover have tt i  powr (ε/(1+ε)) = prod aa {0..i} powr (2 + δ * ε /
(1+ε))
       for i
      unfolding tt-def
      apply (auto simp:powr-powr)
      using ‹ε>0› by (simp add:divide-simps,simp add:algebra-simps)
    ultimately show ?thesis
      apply (elim INFM-mono)
      by (smt nonzero-mult-div-cancel-left powr-divide powr-mult powr-one-eq-one
            that tt-pos zero-less-divide-iff)
  qed

  have 8:∀F k in sequentially. ∀s.  (( aa(k+s)/bb( k+s))) ≥
                (((( aa k/bb k) powr(1/(1+ε))) +s)powr(1+ε)
    using eventually-ge-at-top[of t]
  proof eventually-elim
    case (elim k)
    define ff where ff ≡ (λk. (aa k / bb k) powr (1 / (1 + ε)))
    have 11:ff k+s ≤ ff (k+s) for s
    proof (induct s)
      case 0
      then show ?case unfolding ff-def by auto
    next
      case (Suc s)
```

```
      then have ff k + Suc s ≤ ff (k+Suc s)
        using ratio-large[rule-format,of k+s] ‹ t ≤ k› unfolding ff-def by auto
      then show ?case by simp
    qed
    then have (ff k+s) powr (1+ε) ≤ ff (k+s) powr (1+ε) for s
      apply (rule powr-mono2[of 1+ε,rotated 2])
      unfolding ff-def using ‹ε>0› by auto
    then show ?case unfolding ff-def using ‹ε>0›
      apply (auto simp add:powr-powr)
      by (simp add: a-pos aa-def b-pos bb-def le-less)
  qed

  have 9: (∑r. 1/((z+real r)powr(1+ε))) ≤ 1/(ε *(z−1) powr ε)
      summable (λi. 1/((z+ real i)powr(1+ε)))
    when z>1 for z
  proof −
    define f where f ≡ (λr. 1/((z+ r)powr(1+ε)))
    have ((λx. f (x − 1)) has-integral ((z−1) powr − ε) / ε) {0..}
    proof −
      have ((λx. (z−1 + x) powr (− 1 − ε)) has-integral ((z−1) powr − ε) / ε)
{0<..}
        using powr-has-integral-at-top[of 0 z−1 − 1 − ε,simplified] ‹z>1› ‹ε>0›
        by auto
      then have ((λx. (z−1 + x) powr (− 1 − ε)) has-integral ((z−1) powr − ε)
/ ε) {0..}
        apply (subst (asm) has-integral-closure[symmetric])
        by (auto simp add: negligible-convex-frontier)
      then show ?thesis
        apply (rule has-integral-cong[THEN iffD1, rotated 1])
        unfolding f-def by (smt powr-minus-divide)
    qed
    moreover have ⋀x. 0 ≤ x ⟹ 0 ≤ f (x − 1) unfolding f-def by simp
    moreover have ⋀x y. 0 ≤ x ⟹ x ≤ y ⟹ f (y − 1) ≤ f (x − 1) unfolding
f-def
      by (smt assms(6) frac-le powr-mono2 powr-nonneg-iff that)
    ultimately have summable (λi. f (real i)) (∑ i. f (real i)) ≤ (z − 1) powr −
ε / ε
     using decreasing-sum-le-integral[of λx. f (x−1) ((z−1) powr − ε) / ε,simplified]
      by auto
    moreover have (z − 1) powr − ε / ε = 1/(ε *(z−1) powr ε)
      by (simp add: powr-minus-divide)
    ultimately show (∑ i. f (real i)) ≤ 1/(ε *(z−1) powr ε) by auto
    show summable (λi. f (real i)) using ‹summable (λi. f (real i))› .
  qed

  have u:(λk.( aa k / bb k)) ⟶ ∞
  proof −
    define ff where ff≡(λx. ereal (aa x / bb x))
    have limsup ff = ∞
```

```
proof -
  define tt where tt ≡ (λi. prod aa {0..i} powr (2 + 2 / ε + δ))
  have tt i ≥ 1 for i
    unfolding tt-def
    apply (rule ge-one-powr-ge-zero)
    subgoal
      apply (rule linordered-nonzero-semiring-class.prod-ge-1)
      by (simp add: a-pos aa-def int-one-le-iff-zero-less)
    subgoal by (simp add: ‹ε>0› ‹δ>0› less-imp-le)
    done
  then have limsup (λx.  (aa (x + 1) / tt x * (1 / bb (x + 1))))
             ≤ limsup (λx.  aa (x + 1) / bb (x + 1))
    apply (intro Limsup-mono eventuallyI)
    apply (auto simp add:field-simps order.order-iff-strict)
    by (metis aa-bb-pos(1) div-by-1 frac-less2 less-irrefl less-numeral-extra(1)
        not-le)
  also have ... = limsup (λx.  aa x / bb x)
    by (subst limsup-shift,simp)
  finally have limsup (λx. ereal (aa (x + 1) / tt x * (1 / bb (x + 1))))
                  ≤ limsup (λx. ereal (aa x / bb x)) .
  moreover have limsup (λx. ereal (aa (x + 1) / tt x * (1 / bb (x + 1))))
                   = ∞ using limsup-infi unfolding tt-def by auto
  ultimately show ?thesis
    unfolding ff-def using ereal-infty-less-eq2(1) by blast
qed
then have limsup (λk. ff (k+t)) =  ∞
  by (simp add: limsup-shift-k)
moreover have incseq (λk. ff (k+t))
proof (rule incseq-SucI)
  fix k::nat
  define gg where gg≡(λx. (aa x / bb x))
  have (gg (k+t)) powr (1 / (1 + ε)) + 1
           ≤ (gg (Suc (k+t))) powr (1 / (1 + ε))
    using ratio-large[rule-format, of k+t,simplified] unfolding gg-def
    by auto
  then have (gg (k+t)) powr (1 / (1 + ε))
                  ≤ (gg (Suc (k+t))) powr (1 / (1 + ε))
    by auto
  then have gg (k+t)≤ gg (Suc (k+t))
    by (smt aa-bb-pos(1) aa-bb-pos(2) assms(6) divide-pos-pos gg-def
          powr-less-mono2)
  then show ff (k + t) ≤ ff (Suc k + t)
    unfolding gg-def ff-def by auto
qed
ultimately have (λk. ff (k+t)) ⟶ ∞ using incseq-tendsto-limsup
  by fastforce
then show ?thesis unfolding ff-def
  unfolding tendsto-def
  apply (subst eventually-sequentially-seg[symmetric,of - t])
```

```
    by simp
  qed

  define ξ where ξ = suminf (λ k. bb k / aa k)
  have 10:∀F k in sequentially. |ξ − (∑k = 0..k. bb k / aa k)|
                    < 2 / ε ∗ (bb (k+1) / aa (k+1)) powr (ε / (1+ε))
    using 8[THEN sequentially-offset,of 1] eventually-ge-at-top[of t]
          u[unfolded tendsto-PInfty,rule-format,THEN sequentially-offset
                    ,of (2 powr (1/ε) / (2 powr (1/ε) − 1)) powr (1+ε) 1]
  proof eventually-elim
    case (elim k)
    define tt where tt ≡ (aa (k + 1) / bb (k + 1)) powr (1 / (1 + ε))
    have tt>1
    proof −
      have (aa k / bb k) powr (1 / (1 + ε)) > 0
        by (metis a-pos aa-def b-pos bb-def divide-eq-0-iff less-irrefl
              of-int-eq-0-iff powr-gt-zero)
      then show ?thesis using ratio-large[rule-format,OF ‹k≥t›] unfolding tt-def
by argo
    qed
    have |ξ − (∑i = 0..k. bb i / aa i)| = |∑i. bb (i+(k+1)) / aa (i+(k+1))|
      unfolding ξ-def
      apply (subst suminf-minus-initial-segment[of - k+1,OF summable])
      using atLeast0AtMost lessThan-Suc-atMost by auto
    also have ... = (∑i. bb (i+(k+1)) / aa (i+(k+1)))
    proof −
      have (∑i. bb (i+(k+1)) / aa (i+(k+1))) > 0
        apply (rule suminf-pos)
        subgoal using summable[THEN summable-ignore-initial-segment,of k+1] .
        subgoal by (simp add: a-pos aa-def b-pos bb-def)
        done
      then show ?thesis by auto
    qed
    also have ... ≤ (∑i. 1 / (tt + i) powr (1 + ε))
    proof (rule suminf-le)
      define ff where ff ≡ (λk n. (aa (k+1) / bb (k+1)) powr (1 / (1 + ε)) +
real n)
      have bb (n + (k + 1)) / aa (n + (k + 1)) ≤ 1 / (ff k n) powr (1 + ε) for n
      proof −
        have ff k n powr (1 + ε) ≤ aa (k + n +1 ) / bb (k + n+1 )
          using elim(1)[rule-format,of n] unfolding ff-def by auto
        moreover have ff k n powr (1 + ε) > 0
          unfolding ff-def by (smt elim(2) of-nat-0-le-iff powr-gt-zero ratio-large)
        ultimately have 1 / ff k n powr (1 + ε) ≥ bb (k + (n + 1)) / aa (k +
(n + 1))
          apply (drule-tac le-imp-inverse-le)
          by (auto simp add: inverse-eq-divide)
        then show ?thesis by (auto simp:algebra-simps)
      qed
```

```
      then show ⋀n. bb (n + (k + 1)) / aa (n + (k + 1)) ≤ 1 / (tt + real n)
powr (1 + ε)
        unfolding ff-def tt-def by auto
      show summable (λi. bb (i + (k + 1)) / aa (i + (k + 1)))
        using summable[THEN summable-ignore-initial-segment,of k+1] .
      show summable (λx. 1 / (tt + real x) powr (1 + ε))
        using 9(2)[OF ‹tt>1›] .
    qed
    also have ... ≤ 1/(ε ∗(tt−1) powr ε)
      using 9[OF ‹tt>1›] by simp
    also have ... < 2 / (ε ∗tt powr ε)
    proof −
      have ((2 powr (1/ε) / (2 powr (1/ε) − 1)) powr (1 + ε)) < (aa (k+1) / bb
(k+1))
        using elim(3) by auto
      then have 2 powr (1/ε) / (2 powr (1/ε) − 1) < tt
        unfolding tt-def
        apply (drule-tac powr-less-mono2[rotated 2,where a=1/ (1 + ε)])
        using ‹ε>0› apply (auto simp:powr-powr )
        by (subst (asm) powr-one,auto simp add:field-simps)
      then have  tt < (tt−1) ∗ (2 powr (1/ε))
        using ‹ε>0› by (auto simp:divide-simps algebra-simps)
      then have tt powr ε < 2 ∗ (tt − 1) powr ε
        apply (drule-tac powr-less-mono2[rotated 2,where a=ε])
        using ‹ε>0› ‹tt>1› by (auto simp:powr-powr powr-mult)
      then show ?thesis
        using ‹ε>0› ‹tt>1› by (auto simp:divide-simps)
    qed
    also have ... =  2 / ε ∗ (bb (k + 1) / aa (k + 1)) powr (ε / (1 + ε))
      unfolding tt-def
      using ‹ε>0›
      by (auto simp:powr-powr divide-simps algebra-simps powr-divide less-imp-le)
    finally show ?case .
  qed

  define pq where pq ≡ (λk. quotient-of (∑i=0..k. of-int (b i) / of-int (a i)))
  define p q where p ≡ fst ∘ pq and q = snd ∘ pq
  have coprime-pq:coprime (p k) (q k)
      and q-pos:q k > 0 and pq-sum:p k / q k = (∑i=0..k. b i / a i) for k
  proof −
    have eq: quotient-of (∑i=0..k. of-int (b i) / of-int (a i)) = (p k, q k)
      by (simp add: p-def q-def pq-def)
    from quotient-of-coprime[OF eq] show coprime (p k) (q k) .
    from quotient-of-denom-pos[OF eq] show q k > 0 .
    have (∑i=0..k. b i / a i) = of-rat (∑i=0..k. of-int (b i) / of-int (a i))
      by (simp add: of-rat-sum of-rat-divide)
    also have (∑i=0..k. rat-of-int (b i) / rat-of-int (a i)) =
                rat-of-int (p k) / rat-of-int (q k)
      using quotient-of-div[OF eq] by simp
```

```
    finally show p k / q k = (∑ i=0..k. b i / a i) by (simp add:of-rat-divide)
  qed

  have ξ-Inf-many:∃∞ k. |ξ − p k / q k| <  1 / q k powr (2 + δ ∗ ε / (1 + ε))
  proof −
    have ∗:∃∞k. (bb (Suc k) / aa (Suc k)) powr (ε / (1 + ε))
            < ε / (2 ∗ prod aa {0..k} powr (2 + δ ∗ ε / (1 + ε)))
      using 7[of (2 / ε) powr ((1+ε)/ ε)] ‹ε>0›
      by (auto simp:powr-powr)
    have ∗∗:∀∞k. |ξ − (∑ k = 0..k. bb k / aa k)|
              < 2 / ε ∗ (bb (k + 1) / aa (k + 1)) powr (ε / (1 + ε))
      using 10[unfolded cofinite-eq-sequentially[symmetric]] .
    from INFM-conjI[OF ∗ ∗∗] show ?thesis
    proof (elim INFM-mono)
      fix k assume asm:(bb (Suc k) / aa (Suc k)) powr (ε / (1 + ε))
                        < ε / (2 ∗ prod aa {0..k} powr (2 + δ ∗ ε / (1 + ε))) ∧
                         |ξ − (∑ k = 0..k. bb k / aa k)|
                       < 2 / ε ∗ (bb (k + 1) / aa (k + 1)) powr (ε / (1 + ε))
      have |ξ − real-of-int (p k) / real-of-int (q k)|
              = |ξ − (∑ k = 0..k. bb k / aa k)|
        using pq-sum unfolding aa-def bb-def by auto
      also have ... < (2 / ε) ∗ (bb (k+1) / aa (k+1)) powr (ε / (1+ε))
        using asm by auto
      also have ... <  (2 / ε) ∗ (ε / (2 ∗ prod aa {0..k} powr (2 + δ ∗ ε / (1 +
ε))))
        apply (rule mult-strict-left-mono)
        using asm ‹ε>0› by auto
      also have ... = 1/ prod aa {0..k} powr (2 + δ ∗ ε / (1 + ε))
        using ‹ε>0› by auto
      also have ... ≤ 1 / q k powr (2 + δ ∗ ε / (1 + ε))
      proof −
        have q k ≤ prod aa {0..k}
        proof (induct k)
          case 0
          then show ?case unfolding q-def pq-def aa-def
            apply (simp add:rat-divide-code of-int-rat quotient-of-Fract)
            using aa-bb-pos[of 0,unfolded aa-def bb-def] unfolding Let-def normal-
ize-def
            apply auto
            by (metis div-by-1 less-imp-le less-trans nonneg1-imp-zdiv-pos-iff
                not-less zdiv-mono2)
        next
          case (Suc k)
          define de where de ≡ snd ∘ quotient-of
          have real-of-int (q (Suc k))
                  = de (∑ i=0..Suc k. of-int (b i) / of-int (a i))
            unfolding q-def pq-def de-def by simp
          also have ... = de ((∑ i=0..k. of-int (b i) / of-int (a i))
                            + of-int (b (Suc k)) / of-int (a (Suc k)))
```

```
        by simp
      also have ... ≤ de (∑ i=0..k. of-int (b i) / of-int (a i))
                      ∗ de (of-int (b (Suc k)) / of-int (a (Suc k)))
        using snd-quotient-plus-leq[folded de-def] by presburger
      also have ... = q k ∗ de (of-int (b (Suc k)) / of-int (a (Suc k)))
        unfolding q-def pq-def de-def by auto
      also have ... = q k ∗ snd (Rat.normalize (b (Suc k), a (Suc k)))
        by (simp add:rat-divide-code of-int-rat quotient-of-Fract de-def)
      also have ... ≤ q k ∗ aa (Suc k)
        using aa-bb-pos[of Suc k] q-pos[of k]
        unfolding normalize-def aa-def bb-def Let-def
        apply auto
        by (metis div-by-1 int-one-le-iff-zero-less less-trans
              nonneg1-imp-zdiv-pos-iff not-less zdiv-mono2 zero-less-one)
      also have ... ≤  prod aa {0..k} ∗ aa (Suc k)
        using Suc aa-bb-pos[of Suc k] by auto
      also have ... = prod aa {0..Suc k}
        by (simp add: prod.atLeast0-atMost-Suc)
      finally show ?case .
    qed
    then show ?thesis
      apply (rule-tac divide-left-mono)
      apply (rule powr-mono2)
      using ‹δ>0› ‹ε>0› q-pos[of k]
       apply (auto simp:powr-mult[symmetric])
      by (metis aa-bb-pos(1) less-irrefl)
  qed
  finally show |ξ −  p k / q k| < 1 /  q k powr (2 + δ ∗ ε / (1 + ε)) .
 qed
qed

 define pqs where pqs ≡ {(p, q). q>0 ∧ coprime p q ∧ |ξ − real-of-int p /
real-of-int q|
              < 1 / q powr (2 + δ ∗ ε / (1 + ε))}
have ξ-infinite:infinite pqs
proof −
  define A where A ≡ {k. |ξ −  (p k) /  (q k)| < 1 / (q k) powr (2 + δ ∗ ε /
(1 + ε))}
  note ξ-Inf-many
  then have infinite A
    unfolding Inf-many-def A-def by auto
  moreover have inj-on (λk. (p k, q k)) A
  proof −
    define g where g ≡ (λi. rat-of-int (b i) / rat-of-int (a i))
    define f where f ≡ (λk. ∑ i = 0..k. g i)
    have g-pos:g i>0 for i
      unfolding g-def by (simp add: a-pos b-pos)
    have strict-mono f unfolding strict-mono-def f-def
    proof safe
```

```
      fix x y::nat assume x < y
      then have sum g {0..y} − sum g {0..x} = sum g {x<..y}
        apply (subst  Groups-Big.sum-diff [symmetric])
        by (auto intro:arg-cong2[where f=sum])
      also have ... > 0
        apply (rule ordered-comm-monoid-add-class.sum-pos)
        using ‹x<y› g-pos by auto
      finally have sum g {0..y} − sum g {0..x} >0 .
      then show sum g {0..x} < sum g {0..y} by auto
    qed
    then have inj f using strict-mono-imp-inj-on by auto
    then have inj (quotient-of o f) by (simp add: inj-compose quotient-of-inj)
    then have inj (λk. (p k, q k))
      unfolding f-def p-def q-def pq-def comp-def
      apply (fold g-def)
      by auto
    then show ?thesis by (auto elim:subset-inj-on)
  qed
  moreover have (λk. (p k, q k)) ‘ A ⊆ pqs
    unfolding A-def pqs-def using coprime-pq q-pos by auto
  ultimately show ?thesis
    apply (elim infinite-inj-imageE)
    by auto
qed
moreover have finite pqs if ξ ∈ ℚ
proof −
  obtain m n where ξ-mn:ξ = (of-int m / of-int n) and coprime m n n>0
  proof −
    obtain m n where mn:|ξ| = (of-nat m / of-nat n) coprime m n n≠0
      using Rats-abs-nat-div-natE[OF ‹ξ ∈ ℚ› Rats-abs-nat-div-natE]
      by metis
    define m′ and n′::int
      where m′=(if ξ > 0 then nat m else −nat m) and n′=nat n
    then have ξ = (of-int m′ / of-int n′) coprime m′ n′ n′>0
      using mn by auto
    then show ?thesis using that by auto
  qed
   have pqs ⊆ {(m,n)} ∪ {x. x ∈pqs ∧ − |m| − 1 ≤fst x ∧ fst x ≤ |m| + 1 ∧
0<snd x ∧ snd x < n }
  proof (rule subsetI)
    fix x assume x ∈ pqs
    define p q where p ≡ fst x and q=snd x
    have q>0 coprime p q and pq-less:|ξ − p / q|
        < 1 / q powr (2 + δ ∗ ε / (1 + ε))
      using ‹x∈pqs› unfolding p-def q-def pqs-def by auto
    have q-lt-n:q<n when m≠p ∨ n≠q
    proof −
     have m ∗ q ≠ n ∗ p using that ‹coprime m n› ‹coprime p q› ‹q>0› ‹n>0›
       by (metis eq-rat(1) fst-conv int-one-le-iff-zero-less mult.commute normal-
```

```
ize-stable
        not-one-le-zero quotient-of-Fract snd-conv)
    then have 1/(n*q) ≤ |m/n − p/q|
      using ‹q>0› ‹n>0›
      apply (auto simp:field-simps)
    by (metis add-diff-cancel-left' diff-diff-eq2 diff-zero less-irrefl not-le of-int-diff

        of-int-lessD of-int-mult)
    also have ... < 1 / q powr (2 + δ ∗ ε / (1 + ε))
      using pq-less unfolding ξ-mn by auto
    also have ... ≤ 1 / q ^2
    proof −
      have real-of-int (q^2) = q powr 2
        apply (subst powr-numeral)
        unfolding power2-eq-square using ‹q>0› by auto
      also have ... ≤ q powr (2 + δ ∗ ε / (1 + ε))
        apply (rule powr-mono)
        using ‹q>0› ‹δ>0› ‹ε>0› by auto
      finally have real-of-int (q^2)
              ≤ real-of-int q powr (2 + δ ∗ ε / (1 + ε)) .
      moreover have real-of-int q powr (2 + δ ∗ ε / (1 + ε)) > 0 using ‹0 <
q› by auto
      ultimately show ?thesis by (auto simp:field-simps)
    qed
    finally have 1 / (n ∗ q) < 1 / q^2 .
    then show ?thesis using ‹q>0› ‹n>0›
      unfolding power2-eq-square by (auto simp:field-simps)
  qed
  moreover have − |m| − 1 ≤ p ∧ p ≤ |m| + 1 when m≠p ∨ n≠q
  proof −
    define qn where qn ≡ q/n
    have 0<qn qn<1 unfolding qn-def using q-lt-n[OF ‹m≠p ∨ n≠q›] ‹q>0›
by auto

    have |m/n − p / q| < 1 / q powr (2 + δ ∗ ε / (1 + ε))
      using pq-less unfolding ξ-mn by simp
    then have |p / q − m/n| < 1 / q powr (2 + δ ∗ ε / (1 + ε)) by simp
    then have m/n− 1 / q powr (2 + δ ∗ ε / (1 + ε))
        < p/q ∧ p/q < m/n + 1 / q powr (2 + δ ∗ ε / (1 + ε))
      unfolding abs-diff-less-iff by auto
    then have qn∗m− q / q powr (2 + δ ∗ ε / (1 + ε)) < p
        ∧ p < qn∗m + q / q powr (2 + δ ∗ ε / (1 + ε))
      unfolding qn-def using ‹q>0› by (auto simp:field-simps)
    moreover have − |m| − 1 ≤ qn∗m− q / q powr (2 + δ ∗ ε / (1 + ε))
    proof −
      have − |m| ≤ qn∗m using ‹0<qn› ‹qn<1›
        apply (simp add: abs-if)
        by (smt (verit, best) mult-nonneg-nonneg of-int-nonneg)
      moreover have − 1 ≤ − q / q powr (2 + δ ∗ ε / (1 + ε))
```

```
        proof -
          have q = q powr 1 using ‹0 < q› by auto
          also have ... ≤q powr (2 + δ ∗ ε / (1 + ε))
            apply (rule powr-mono)
            using ‹q>0› ‹δ>0› ‹ε>0› by auto
          finally have q ≤ q powr (2 + δ ∗ ε / (1 + ε)) .
          then show ?thesis using ‹0 < q› by auto
        qed
        ultimately show ?thesis by auto
      qed
      moreover have qn∗m + q / q powr (2 + δ ∗ ε / (1 + ε)) ≤ |m| + 1
      proof -
        have qn∗m ≤ |m| using ‹0<qn› ‹qn<1›
          apply (simp add: abs-if mult-left-le-one-le)
          by (meson less-eq-real-def mult-pos-neg neg-0-less-iff-less of-int-less-0-iff
order-trans)
        moreover have q / q powr (2 + δ ∗ ε / (1 + ε)) ≤ 1
        proof -
          have q = q powr 1 using ‹0 < q› by auto
          also have ... ≤q powr (2 + δ ∗ ε / (1 + ε))
            apply (rule powr-mono)
            using ‹q>0› ‹δ>0› ‹ε>0› by auto
          finally have q ≤ q powr (2 + δ ∗ ε / (1 + ε)) .
          then show ?thesis using ‹0 < q› by auto
        qed
        ultimately show ?thesis by auto
      qed
      ultimately show ?thesis by auto
    qed
    ultimately show x ∈ {(m, n)} ∪ {x ∈ pqs. − |m| − 1 ≤ fst x ∧ fst x ≤ |m|
+ 1
                         ∧ 0 < snd x ∧ snd x < n}
      using ‹x ∈ pqs› ‹q>0› unfolding p-def q-def by force
  qed
  moreover have finite {x. x ∈pqs ∧ − |m| − 1 ≤fst x ∧ fst x ≤ |m| + 1 ∧
0<snd x ∧ snd x < n }
  proof -
    have finite ({− |m| − 1..|m| +1} × {0<..<n}) by blast
    moreover have {x. x ∈pqs ∧ − |m| − 1 ≤fst x ∧ fst x ≤ |m| + 1 ∧ 0<snd
x ∧ snd x < n } ⊆
          ({− |m| − 1..|m| +1} × {0<..<n})
      by auto
    ultimately show ?thesis
      using finite-subset by blast
  qed
  ultimately show ?thesis using finite-subset by auto
qed
ultimately show ?thesis
  apply (fold ξ-def)
```

using *RothsTheorem*[*rule-format*,*of* ξ *2* + δ ∗ ε / (*1* + ε),*folded pqs-def*]
‹δ >*0*› ‹ε>*0*›
apply (*auto simp*:*divide-simps*)
by (*meson mult-le-0-iff not-less*)
qed

1.3 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. Thanks to Iosif Pinelis for his clarification on MathOverflow regarding the summmability of the series in *RothsTheorem.HanclRucki2* https://mathoverflow.net/questions/323069/why-is-this-series-summable and to Manuel Eberl for his helpful comments.

end

References

[1] J. Hančl and P. Rucki. The transcendence of certain infinite series. *Rocky Mountain Journal of Mathematics*, 35(2):531–537, 2005.

[2] K. F. Roth. Rational approximations to algebraic numbers. *Mathematika*, 2(3):1–20, 1955.