Irrational numbers from THE BOOK

Lawrence C. Paulson

March 17, 2025

Abstract

An elementary proof is formalised: that $\exp r$ is irrational for every nonzero rational number r. The mathematical development comes from the well-known volume *Proofs from THE BOOK* [1, pp. 51–2], by Aigner and Ziegler, who credit the idea to Hermite. The development illustrates a number of basic Isabelle techniques: the manipulation of summations, the calculation of quite complicated derivatives and the estimation of integrals. We also see how to import another AFP entry (Stirling's formula) [2].

As for the theorem itself, note that a much stronger and more general result (the Hermite–Lindemann–Weierstraß transcendence theorem) is already available in the AFP [3].

Contents

Acknowledgements The author was supported by the ERC Advanced Grant ALEXANDRIA (Project 742178) funded by the European Research Council.

1 Some irrational numbers

From Aigner and Ziegler, *Proofs from THE BOOK* (Springer, 2018), Chapter 8, pp. 50–51.

theory *Irrationals-From-THEBOOK* **imports** *Stirling-Formula.Stirling-Formula*

begin

1.1 Basic definitions and their consequences

definition *hf* **where** $hf \equiv \lambda n.\ \lambda x{::}real.\ (x\hat{}n * (1-x)\hat{}n)\ /\ fact\ n$

definition *cf* **where** $cf \equiv \lambda n\ i.\ if\ i < n\ then\ 0\ else\ (n\ choose\ (i-n)) * (-1)\hat{}(i-n)$

Mere knowledge that the coefficients are integers is not enough later on.

lemma *hf-int-poly*:
  fixes $x{::}real$
  shows $hf\ n = (\lambda x.\ (1\ /\ fact\ n) * (\sum i{=}0..2{*}n.\ real\text{-}of\text{-}int\ (cf\ n\ i) * x\hat{}i))$
proof
  fix x
  have *inj*: $inj\text{-}on\ ((+)n)\ \{..n\}$
    by (*auto simp*: *inj-on-def*)
  have [*simp*]: $((+)n)\ `\ \{..n\} = \{n..2{*}n\}$
    using *nat-le-iff-add* **by** *fastforce*
  have $(x\hat{}n * (-x + 1)\hat{}n) = x\ \hat{}\ n * (\sum k{\le}n.\ real\ (n\ choose\ k) * (-\ x)\ \hat{}\ k)$
    unfolding *binomial-ring* **by** *simp*
  also have $\ldots = x\ \hat{}\ n * (\sum k{\le}n.\ real\text{-}of\text{-}int\ ((n\ choose\ k) * (-1)\hat{}k) * x\ \hat{}\ k)$
    by (*simp add*: *mult.assoc flip*: *power-minus*)
  also have $\ldots = (\sum k{\le}n.\ real\text{-}of\text{-}int\ ((n\ choose\ k) * (-1)\hat{}k) * x\ \hat{}\ (n{+}k))$
    by (*simp add*: *sum-distrib-left mult-ac power-add*)
  also have $\ldots = (\sum i{=}n..2{*}n.\ real\text{-}of\text{-}int\ (cf\ n\ i) * x\hat{}i)$
    by (*simp add*: *sum.reindex* [*OF inj*, *simplified*] *cf-def*)
  finally have $hf\ n\ x = (1\ /\ fact\ n) * (\sum i = n..2 * n.\ real\text{-}of\text{-}int\ (cf\ n\ i) * x\hat{}i)$
    by (*simp add*: *hf-def*)
  moreover have $(\sum i = 0..{<}n.\ real\text{-}of\text{-}int\ (cf\ n\ i) * x\hat{}i) = 0$
    by (*simp add*: *cf-def*)
  ultimately show $hf\ n\ x = (1\ /\ fact\ n) * (\sum i = 0..2 * n.\ real\text{-}of\text{-}int\ (cf\ n\ i) * x\hat{}i)$
    using *sum.union-disjoint* [*of* $\{0..{<}n\}\ \{n..2{*}n\}\ \lambda i.\ real\text{-}of\text{-}int\ (cf\ n\ i) * x\hat{}i$]
    by (*simp add*: *ivl-disj-int-two*(7) *ivl-disj-un-two*(7) *mult-2*)
qed

Lemma (ii) in the text has strict inequalities, but that's more work and is less useful.

lemma
  assumes $0 \le x\ x \le 1$
  shows *hf-nonneg*: $0 \le hf\ n\ x$ **and** *hf-le-inverse-fact*: $hf\ n\ x \le 1/fact\ n$
  using *assms* **by** (*auto simp*: *hf-def divide-simps mult-le-one power-le-one*)

lemma *hf-differt* [*iff*]: *hf n differentiable at x*
  unfolding *hf-int-poly differentiable-def*
  by (*intro derivative-eq-intros exI* | *simp*)+

lemma *deriv-sum-int*:
  deriv ($\lambda x.$ $\sum$ *i=0..n. real-of-int* (*c i*) $*$ *x*^*i*) *x*
    = (*if n=0 then 0 else* ($\sum$ *i=0..n−1. of-int*((*i+1*) $*$ *c*(*Suc i*)) $*$ *x*^*i*))
  (**is** *deriv ?f x* = (*if n=0 then 0 else ?g*))
proof −
  have (*?f has-real-derivative ?g*) (*at x*) **if** *n* > *0*
  proof −
    have ($\sum$ *i* = *0..n. i* $*$ *x* ^ (*i* − *Suc 0*) $*$ (*c i*))
      = ($\sum$ *i* = *1..n.* (*real* (*i−1*) + *1*) $*$ *of-int* (*c i*) $*$ *x* ^ (*i−1*))
      using *that* **by** (*auto simp*: *sum.atLeast-Suc-atMost intro*!: *sum.cong*)
    also have ... = *sum* (($\lambda i.$ (*real i* + *1*) $*$ *c* (*Suc i*) $*$ *x*^*i*) $\circ$ ($\lambda n.$ *n−1*))
          {*1..Suc* (*n−1*)}
      using *that* **by** *simp*
    also have ... = *?g*
      by (*simp flip*: *sum.atLeast-atMost-pred-shift* [**where** *m=0*])
    finally have §: ($\sum$ *a* = *0..n. a* $*$ *x* ^ (*a* − *Suc 0*) $*$ (*c a*)) = *?g* **.**
    show *?thesis*
      by (*rule derivative-eq-intros* § | *simp*)+
  qed
  then show *?thesis*
    by (*force intro*: *DERIV-imp-deriv*)
qed

We calculate the coefficients of the *k*th derivative precisely.

lemma *hf-deriv-int-poly*:
  (*deriv*^^*k*) (*hf n*) = ($\lambda x.$ (*1/fact n*) $*$ ($\sum$ *i=0..2*$*$*n−k. of-int* (*int*($\prod$ {*i*<..*i+k*}) $*$ *cf n* (*i+k*)) $*$ *x*^*i*))
proof (*induction k*)
  case *0*
  show *?case*
    by (*simp add*: *hf-int-poly*)
next
  case (*Suc k*)
  define *F* **where** *F* ≡ $\lambda x.$ ($\sum$ *i* = *0..2*$*$*n* − *k. real-of-int* (*int*($\prod$ {*i*<..*i+k*}) $*$ *cf n* (*i+k*)) $*$ *x*^*i*)
  have *Fd*: *F field-differentiable at x* **for** *x*
    unfolding *field-differentiable-def F-def*
    by (*rule derivative-eq-intros exI* | *force*)+
  have [*simp*]: *prod int* {*i*<..*Suc* (*i* + *k*)} = (*1* + *int i*) $*$ *prod int* {*Suc i*<..*Suc* (*i* + *k*)} **for** *i*
    by (*metis Suc-le-mono atLeastSucAtMost-greaterThanAtMost le-add1 of-nat-Suc prod.head*)
  have *deriv* ($\lambda x.$ *F x* / *fact n*) *x*
    = ($\sum$ *i* = *0..2* $*$ *n* − *Suc k. of-int* (*int*($\prod$ {*i*<..*i*+ *Suc k*}) $*$ *cf n* (*Suc* (*i+k*))) $*$ *x*^*i*) / *fact n* **for** *x*

unfolding *deriv-cdivide-right* [*OF Fd*]
by (*fastforce simp add*: *F-def deriv-sum-int cf-def simp flip*: *of-int-mult intro*: *sum.cong*)
then show *?case*
by (*simp add*: *Suc F-def*)
qed

lemma *hf-deriv-0*: (*deriv*^^*k*) (*hf n*) *0* ∈ ℤ
proof (*cases n* ≤ *k*)
case *True*
then obtain *j* **where** (*fact k*::*real*) = *real-of-int j* ∗ *fact n*
by (*metis fact-dvd dvd-def mult.commute of-int-fact of-int-mult*)
moreover have *prod real* {*0*<..*k*} = *fact k*
by (*simp add*: *fact-prod atLeastSucAtMost-greaterThanAtMost*)
ultimately show *?thesis*
by (*simp add*: *hf-deriv-int-poly dvd-def*)
next
case *False*
then show *?thesis*
by (*simp add*: *hf-deriv-int-poly cf-def*)
qed

lemma *deriv-hf-minus*: *deriv* (*hf n*) = (λ*x*. − *deriv* (*hf n*) (*1*−*x*))
proof
fix *x*
have *hf n* = *hf n* ∘ (λ*x*. (*1*−*x*))
by (*simp add*: *fun-eq-iff hf-def mult.commute*)
then have *deriv* (*hf n*) *x* = *deriv* (*hf n* ∘ (λ*x*. (*1*−*x*))) *x*
by *fastforce*
also have . . . = *deriv* (*hf n*) (*1*−*x*) ∗ *deriv* ((−) *1*) *x*
by (*intro real-derivative-chain*) *auto*
finally show *deriv* (*hf n*) *x* = − *deriv* (*hf n*) (*1*−*x*)
by *simp*
qed

lemma *deriv-n-hf-diffr* [*iff*]: (*deriv*^^*k*) (*hf n*) *field-differentiable at x*
unfolding *field-differentiable-def hf-deriv-int-poly*
by (*rule derivative-eq-intros exI* | *force*)+

lemma *deriv-n-hf-minus*: (*deriv*^^*k*) (*hf n*) = (λ*x*. (−*1*)^*k* ∗ (*deriv*^^*k*) (*hf n*) (*1*−*x*))
proof (*induction k*)
case *0*
then show *?case*
by (*simp add*: *fun-eq-iff hf-def*)
next
case (*Suc k*)
have *o*: (λ*x*. (*deriv* ^^ *k*) (*hf n*) (*1*−*x*)) = (*deriv* ^^ *k*) (*hf n*) ∘ (−) *1*
by *auto*

show *?case*
proof
fix *x*
have [*simp*]: ((*deriv*^^*k*) (*hf n*) ∘ (−) *1*) *field-differentiable at x*
by (*force intro*: *field-differentiable-compose*)
have (*deriv* ^^ *Suc k*) (*hf n*) *x* = *deriv* (λ*x*. (−*1*) ^ *k* ∗ (*deriv* ^^ *k*) (*hf n*) (*1*−*x*)) *x*
by *simp* (*metis Suc*)
also have ... = (−*1*) ^ *k* ∗ *deriv* (λ*x*. (*deriv* ^^ *k*) (*hf n*) (*1*−*x*)) *x*
using *o* **by** *fastforce*
also have ... = (−*1*) ^ *Suc k* ∗ (*deriv* ^^ *Suc k*) (*hf n*) (*1*−*x*)
by (*subst o*, *subst deriv-chain*, *auto*)
finally show (*deriv* ^^ *Suc k*) (*hf n*) *x* = (−*1*) ^ *Suc k* ∗ (*deriv* ^^ *Suc k*) (*hf n*) (*1*−*x*) **.**
qed
qed

1.2 Towards the main result

lemma *hf-deriv-1*: (*deriv*^^*k*) (*hf n*) *1* ∈ ℤ
by (*smt* (*verit*, *best*) *Ints-1 Ints-minus Ints-mult Ints-power deriv-n-hf-minus hf-deriv-0*)

lemma *hf-deriv-eq-0*: *k* > *2*∗*n* ⟹ (*deriv*^^*k*) (*hf n*) = (λ*x*. *0*)
by (*force simp add*: *cf-def hf-deriv-int-poly*)

The case for positive integers

lemma *exp-nat-irrational*:
assumes *s* > *0* **shows** *exp* (*real-of-int s*) ∉ ℚ
proof
assume *exp* (*real-of-int s*) ∈ ℚ
then obtain *a b* **where** *ab*: *a* > *0 b* > *0 coprime a b* **and** *exp-s*: *exp s* = *of-int a* / *of-int b*
by (*smt* (*verit*) *Rats-cases′ divide-nonpos-pos exp-gt-zero of-int-0-less-iff*)
define *n* **where** *n* ≡ *nat* (*max* (*a*^*2*) (*3* ∗ *s*^*3*))
then have *ns3*: *s*^*3* ≤ *real n* / *3*
by *linarith*
have *n* > *0*
using ‹*a* > *0*› **by** (*simp add*: *n-def max.strict-coboundedI1*)
then have *s* ^(*2*∗*n*+*1*) ≤ *s* ^(*3*∗*n*)
using ‹*a* > *0*› *assms* **by** (*intro power-increasing*) *auto*
also have ... = *real-of-int*(*s*^*3*) ^ *n*
by (*simp add*: *power-mult*)
also have ... ≤ (*n* / *3*) ^ *n*
using *assms ns3* **by** (*simp add*: *power-mono*)
also have ... ≤ (*n* / *exp 1*) ^ *n*
using *exp-le* ‹*n* > *0*› **by** (*auto simp*: *divide-simps*)
finally have *s-le*: *s* ^(*2*∗*n*+*1*) ≤ (*n* / *exp 1*) ^ *n*
by *presburger*
have *a-less*: *a* < *sqrt* (*2*∗*pi*∗*n*)

```
proof −
  have 2∗pi > 1
    using pi-ge-two by linarith
  have a ≤ sqrt n
    using ‹0 < n› n-def of-nat-nat real-le-rsqrt by fastforce
  also have . . . < sqrt (2∗pi∗n)
    by (simp add: ‹0 < n› ‹1 < 2 ∗ pi›)
  finally show ?thesis .
qed
have sqrt (2∗pi∗n) ∗ (n / exp 1) ^ n > a ∗ s ^ (2∗n+1)
  using mult-strict-right-mono [OF a-less] mult-left-mono [OF s-le]
 by (smt (verit, best) s-le ab(1) assms of-int-1 of-int-le-iff of-int-mult zero-less-power)
then have n: fact n > a ∗ s ^ (2∗n+1)
  using fact-bounds(1) by (smt (verit, best) ‹0 < n› of-int-fact of-int-less-iff)
define F where F ≡ λx. ∑ i≤2∗n. (−1)^i ∗ s^(2∗n−i) ∗ (deriv^^i) (hf n) x
have Fder: (F has-real-derivative −s ∗ F x + s^(2∗n+1) ∗ hf n x) (at x) for x
proof −
  have ∗: sum f {..n+n} = sum f {..<n+n} if f (n+n) = 0 for f::nat ⇒ real
    by (smt (verit, best) lessThan-Suc-atMost sum.lessThan-Suc that)
  have [simp]: (deriv ((deriv ^^ (n+n)) (hf n)) x) = 0
    using hf-deriv-eq-0 [where k= Suc(n+n)] by simp
  have §: (∑ k≤n+n. (−1) ^ k ∗ ((deriv ^^ Suc k) (hf n) x ∗ of-int s ^ (n+n −
k)))
        + s ∗ (∑ j=0..n+n. (−1) ^ j ∗ ((deriv ^^ j) (hf n) x ∗ of-int s ^ (n+n
− j)))
        = s ∗ (hf n x ∗ of-int s ^ (n+n))
    using ‹n>0›
    apply (subst sum-Suc-reindex)
    apply (simp add: algebra-simps atLeast0AtMost)
    apply (force simp add: ∗ mult.left-commute [of of-int s] minus-nat.diff-Suc
sum-distrib-left
           simp flip: sum.distrib intro: comm-monoid-add-class.sum.neutral split:
nat.split-asm)
    done
  show ?thesis
    unfolding F-def
    apply (rule derivative-eq-intros field-differentiable-derivI | simp)+
    using § by (simp add: algebra-simps atLeast0AtMost eval-nat-numeral)
qed
have F01-Ints: F 0 ∈ ℤ F 1 ∈ ℤ
  by (simp-all add: F-def hf-deriv-0 hf-deriv-1 Ints-sum)
define sF where sF ≡ λx. exp (of-int s ∗ x) ∗ F x
define sF' where sF' ≡ λx. of-int s ^ Suc(2∗n) ∗ (exp (of-int s ∗ x) ∗ hf n x)
have sF-der: (sF has-real-derivative sF' x) (at x) for x
  unfolding sF-def sF'-def
  by (rule refl Fder derivative-eq-intros | force simp: algebra-simps)+
let ?N = b ∗ integral {0..1} sF'
have sF'-integral: (sF' has-integral sF 1 − sF 0) {0..1}
 by (smt (verit) fundamental-theorem-of-calculus has-real-derivative-iff-has-vector-derivative
```

```
                has-vector-derivative-at-within sF-der)
  then have ?N = a * F 1 − b * F 0
    using ‹b > 0› by (simp add: integral-unique exp-s sF-def algebra-simps)
  also have ... ∈ ℤ
    using hf-deriv-1 by (simp add: F01-Ints)
  finally have N-Ints: ?N ∈ ℤ .
  have sF′ (1/2) > 0 and ge0: ⋀x. x ∈ {0..1} ⟹ 0 ≤ sF′ x
    using assms by (auto simp: sF′-def hf-def)
  moreover have continuous-on {0..1} sF′
    unfolding sF′-def hf-def by (intro continuous-intros) auto
  ultimately have False if (sF′ has-integral 0) {0..1}
    using has-integral-0-cbox-imp-0 [of 0 1 sF′ 1/2] that by auto
  then have integral {0..1} sF′ > 0
    by (metis ge0 has-integral-nonneg integral-unique order-le-less sF′-integral)
  then have 0 < ?N
    by (simp add: ‹b > 0›)
  have integral {0..1} sF′ = of-int s ^ Suc(2*n) * integral {0..1} (λx. exp (s*x)
* hf n x)
    unfolding sF′-def by force
  also have ... ≤ of-int s ^ Suc(2*n) * (exp s * (1 / fact n))
  proof (rule mult-left-mono)
    have integral {0..1} (λx. exp (s*x) * hf n x) ≤ integral {0..1} (λx::real. exp s
* (1/fact n))
    proof (intro mult-mono integral-le)
      show (λx. exp (s*x) * hf n x) integrable-on {0..1}
        using ‹0 < ?N› not-integrable-integral sF′-def by fastforce
    qed (use assms hf-nonneg hf-le-inverse-fact in auto)
    also have ... = exp s * (1 / fact n)
      by simp
    finally show integral {0..1} (λx. exp (s*x) * hf n x) ≤ exp s * (1 / fact n) .
  qed (use assms in auto)
  finally have ?N ≤ b * of-int s ^ Suc(2*n) * exp s * (1 / fact n)
    using ‹b > 0› by (simp add: sF′-def mult-ac divide-simps)
  also have ... < 1
    using n apply (simp add: field-simps exp-s)
    by (metis of-int-fact of-int-less-iff of-int-mult of-int-power)
  finally show False
    using ‹0 < ?N› Ints-cases N-Ints by force
qed

theorem exp-irrational:
  fixes q::real assumes q ∈ ℚ q ≠ 0 shows exp q ∉ ℚ
proof
  assume q: exp q ∈ ℚ
  obtain s t where s ≠ 0 t > 0 q = of-int s / of-int t
    by (metis Rats-cases′ assms div-0 of-int-0)
  then have (exp q) ^ (nat t) = exp s
    by (smt (verit, best) exp-divide-power-eq of-nat-nat zero-less-nat-eq)
```

moreover have *exp q* ^ (*nat t*) ∈ ℚ
by (*simp add*: *q*)
ultimately show *False*
by (*smt* (*verit*, *del-insts*) *Rats-inverse* ‹*s* ≠ *0*› *exp-minus exp-nat-irrational of-int-of-nat*)
qed

corollary *ln-irrational*:
fixes *q*::*real* **assumes** *q* ∈ ℚ *q* > *0* *q* ≠ *1* **shows** *ln q* ∉ ℚ
using *assms exp-irrational* [*of ln q*] *exp-ln-iff* [*of q*] **by** *force*

end

References

[1] M. Aigner and G. M. Ziegler. *Proofs from THE BOOK*. Springer, 6th edition, 2018.

[2] M. Eberl. Stirling's formula. *Archive of Formal Proofs*, Sept. 2016. https://isa-afp.org/entries/Stirling_Formula.html, Formal proof development.

[3] M. Eberl. The Hermite–Lindemann–Weierstraß transcendence theorem. *Archive of Formal Proofs*, Mar. 2021. https://isa-afp.org/entries/Hermite_Lindemann.html, Formal proof development.