A Proof from THE BOOK: The Partial Fraction Expansion of the Cotangent

Manuel Eberl

March 16, 2022

Abstract

In this article, I formalise a proof from THE BOOK [1, Chapter 23]; namely a formula that was called 'one of the most beautiful formulas involving elementary functions':

$$\pi \cot(\pi z) = \frac{1}{z} + \sum_{n=1}^{\infty} \left(\frac{1}{z+n} + \frac{1}{z-n} \right)$$

The proof uses Herglotz's trick to show the real case and analytic continuation for the complex case.

Contents

1 The Partial-Fraction Formula for the Cotangent Function

theory *Cotangent-PFD-Formula*
imports *HOL−Complex-Analysis.Complex-Analysis HOL−Real-Asymp.Real-Asymp*

begin

1.1 Auxiliary lemmas

The following variant of the comparison test for showing summability allows us to use a 'Big-O' estimate, which works well together with Isabelle's automation for real asymptotics.

lemma *summable-comparison-test-bigo*:
fixes $f :: nat \Rightarrow real$
assumes *summable* $(\lambda n.\ norm\ (g\ n))$ $f \in O(g)$
shows *summable f*
proof −
from ‹$f \in O(g)$› **obtain** C **where** C: *eventually* $(\lambda x.\ norm\ (f\ x) \le C * norm\ (g\ x))$ *at-top*
by (*auto elim*: *landau-o.bigE*)
thus *?thesis*
by (*rule summable-comparison-test-ev*) (*insert assms*, *auto intro*: *summable-mult*)
qed

lemma *uniformly-on-image*:
uniformly-on $(f$ ‘ $A)$ g = *filtercomap* $(\lambda h.\ h \circ f)$ (*uniformly-on* A $(g \circ f))$
unfolding *uniformly-on-def* **by** (*simp add*: *filtercomap-INF*)

lemma *uniform-limit-image*:
uniform-limit $(f$ ‘ $A)$ $g\ h\ F \longleftrightarrow$ *uniform-limit* A $(\lambda x\ y.\ g\ x\ (f\ y))$ $(\lambda x.\ h\ (f\ x))$ F
by (*simp add*: *uniformly-on-image filterlim-filtercomap-iff o-def*)

lemma *Ints-add-iff1* [*simp*]: $x \in \mathbb{Z} \Longrightarrow x + y \in \mathbb{Z} \longleftrightarrow y \in \mathbb{Z}$
by (*metis Ints-add Ints-diff add.commute add-diff-cancel-right'*)

lemma *Ints-add-iff2* [*simp*]: $y \in \mathbb{Z} \Longrightarrow x + y \in \mathbb{Z} \longleftrightarrow x \in \mathbb{Z}$
by (*metis Ints-add Ints-diff add-diff-cancel-right'*)

If a set is discrete (i.e. the difference between any two points is bounded from below), it has no limit points:

lemma *discrete-imp-not-islimpt*:
assumes e: $0 < e$
and d: $\forall x \in S.\ \forall y \in S.\ dist\ y\ x < e \longrightarrow y = x$
shows $\neg x$ *islimpt S*
proof
assume *x islimpt S*
hence *x islimpt* $S - \{x\}$

by (*meson islimpt-punctured*)
moreover from *assms* **have** *closed* (*S* − {*x*})
by (*intro discrete-imp-closed*) *auto*
ultimately show *False*
unfolding *closed-limpt* **by** *blast*
qed

In particular, the integers have no limit point:

lemma *Ints-not-limpt*: ¬((*x* :: *'a* :: *real-normed-algebra-1*) *islimpt* ℤ)
by (*rule discrete-imp-not-islimpt*[*of 1*]) (*auto elim*!: *Ints-cases simp*: *dist-of-int*)

The following lemma allows evaluating telescoping sums of the form

$$\sum_{n=0}^{\infty} (f(n) - f(n+k))$$

where $f(n) \longrightarrow 0$, i.e. where all terms except for the first k are cancelled by later summands.

lemma *sums-long-telescope*:
fixes *f* :: *nat* ⇒ *'a* :: {*topological-group-add*, *topological-comm-monoid-add*, *ab-group-add*}
assumes *lim*: *f* ⟶ *0*
shows (λ*n*. *f n* − *f* (*n* + *c*)) *sums* (∑ *k*<*c*. *f k*) (**is** - *sums ?S*)
proof −
thm *tendsto-diff*
have (λ*N*. *?S* − (∑ *n*<*c*. *f* (*N* + *n*))) ⟶ *?S* − *0*
by (*intro tendsto-intros tendsto-null-sum filterlim-compose*[*OF assms*]; *real-asymp*)
hence (λ*N*. *?S* − (∑ *n*<*c*. *f* (*N* + *n*))) ⟶ *?S*
by *simp*
moreover have *eventually* (λ*N*. *?S* − (∑ *n*<*c*. *f* (*N* + *n*)) = (∑ *n*<*N*. *f n* − *f* (*n* + *c*))) *sequentially*
using *eventually-ge-at-top*[*of c*]
proof *eventually-elim*
case (*elim N*)
have (∑ *n*<*N*. *f n* − *f* (*n* + *c*)) = (∑ *n*<*N*. *f n*) − (∑ *n*<*N*. *f* (*n* + *c*))
by (*simp only*: *sum-subtractf*)
also have (∑ *n*<*N*. *f n*) = (∑ *n*∈{..<*c*} ∪ {*c*..<*N*}. *f n*)
using *elim* **by** (*intro sum.cong*) *auto*
also have ... = (∑ *n*<*c*. *f n*) + (∑ *n*∈{*c*..<*N*}. *f n*)
by (*subst sum.union-disjoint*) *auto*
also have (∑ *n*<*N*. *f* (*n* + *c*)) = (∑ *n*∈{*c*..<*N*+*c*}. *f n*)
using *elim* **by** (*intro sum.reindex-bij-witness*[*of* - λ*n*. *n* − *c* λ*n*. *n* + *c*]) *auto*
also have ... = (∑ *n*∈{*c*..<*N*}∪{*N*..<*N*+*c*}. *f n*)
using *elim* **by** (*intro sum.cong*) *auto*
also have ... = (∑ *n*∈{*c*..<*N*}. *f n*) + (∑ *n*∈{*N*..<*N*+*c*}. *f n*)
by (*subst sum.union-disjoint*) *auto*
also have (∑ *n*∈{*N*..<*N*+*c*}. *f n*) = (∑ *n*<*c*. *f* (*N* + *n*))
by (*intro sum.reindex-bij-witness*[*of* - λ*n*. *n* + *N* λ*n*. *n* − *N*]) *auto*
finally show *?case*

by *simp*
qed
ultimately show *?thesis*
unfolding *sums-def* **by** (*rule Lim-transform-eventually*)
qed

1.2 Definition of auxiliary function

The following function is the infinite sum appearing on the right-hand side of the cotangent formula. It can be written either as

$$\sum_{n=1}^{\infty} \left(\frac{1}{x+n} + \frac{1}{x-n} \right)$$

or as

$$2x \sum_{n=1}^{\infty} \frac{1}{x^2 - n^2} \ .$$

definition *cot-pfd* :: *'a* :: {*real-normed-field*, *banach*} ⇒ *'a* **where**
cot-pfd x = ($\sum$ *n*. *2* ∗ *x* / (*x* ^ *2* − *of-nat* (*Suc n*) ^ *2*))

The sum in the definition of *cot-pfd* converges uniformly on compact sets. This implies, in particular, that *cot-pfd* is holomorphic (and thus also continuous).

lemma *uniform-limit-cot-pfd-complex*:
assumes *R* ≥ *0*
shows *uniform-limit* (*cball 0 R* :: *complex set*)
(λ*N x*. $\sum$ *n*<*N*. *2* ∗ *x* / (*x* ^ *2* − *of-nat* (*Suc n*) ^ *2*)) *cot-pfd sequentially*
unfolding *cot-pfd-def*
proof (*rule Weierstrass-m-test-ev*)
have *eventually* (λ*N*. *of-nat* (*N* + *1*) > *R*) *at-top*
by *real-asymp*
thus ∀$_F$ *N in sequentially*. ∀(*x*::*complex*)∈*cball 0 R*. *norm* (*2* ∗ *x* / (*x* ^ *2* − *of-nat* (*Suc N*) ^ *2*)) ≤
2 ∗ *R* / (*real* (*N* + *1*) ^ *2* − *R* ^ *2*)
proof *eventually-elim*
case (*elim N*)
show *?case*
proof *safe*
fix *x* :: *complex* **assume** *x*: *x* ∈ *cball 0 R*
have (*1* + *real N*)^2 − R^2 ≤ *norm* ((*1* + *of-nat N* :: *complex*) ^ *2*) − *norm* (*x* ^ *2*)
using *x* **by** (*auto intro*: *power-mono simp*: *norm-power simp flip*: *of-nat-Suc*)
also have ... ≤ *norm* (x^2 − (*1* + *of-nat N* :: *complex*)^2)
by (*metis norm-minus-commute norm-triangle-ineq2*)
finally show *norm* (*2* ∗ *x* / (x^2 − (*of-nat* (*Suc N*))^2)) ≤ *2* ∗ *R* / (*real* (*N* + *1*) ^ *2* − *R* ^ *2*)
unfolding *norm-mult norm-divide* **using** ‹*R* ≥ *0*› *x elim*

```
      by (intro mult-mono frac-le) (auto intro: power-strict-mono)
    qed
  qed
next
  show summable (λN. 2 * R / (real (N + 1) ^ 2 − R ^ 2))
  proof (rule summable-comparison-test-bigo)
    show (λN. 2 * R / (real (N + 1) ^ 2 − R ^ 2)) ∈ O(λN. 1 / real N ^ 2)
      by real-asymp
  next
    show summable (λn. norm (1 / real n ^ 2))
      using inverse-power-summable[of 2] by (simp add: field-simps)
  qed
qed

lemma sums-cot-pfd-complex:
  fixes x :: complex
  shows (λn. 2 * x / (x ^ 2 − of-nat (Suc n) ^ 2)) sums cot-pfd x
  using tendsto-uniform-limitI[OF uniform-limit-cot-pfd-complex[of norm x], of x]
  by (simp add: sums-def)

lemma sums-cot-pfd-complex':
  fixes x :: complex
  assumes x ∉ ℤ
  shows  (λn. 1 / (x + of-nat (Suc n)) + 1 / (x − of-nat (Suc n))) sums cot-pfd
x
proof −
  have (λn. 2 * x / (x ^ 2 − of-nat (Suc n) ^ 2)) sums cot-pfd x
    by (rule sums-cot-pfd-complex)
  also have (λn. 2 * x / (x ^ 2 − of-nat (Suc n) ^ 2)) =
            (λn. 1 / (x + of-nat (Suc n)) + 1 / (x − of-nat (Suc n))) (is ?lhs =
?rhs)
  proof
    fix n :: nat
    have neq1: x + of-nat (Suc n) ≠ 0
      using assms by (metis Ints-0 Ints-add-iff2 Ints-of-nat)
    have neq2: x − of-nat (Suc n) ≠ 0
      using assms by force
    have neq3: x ^ 2 − of-nat (Suc n) ^ 2 ≠ 0
    using assms by (metis Ints-of-nat eq-iff-diff-eq-0 minus-in-Ints-iff power2-eq-iff)
    show ?lhs n = ?rhs n using neq1 neq2 neq3
      by (simp add: divide-simps del: of-nat-Suc) (auto simp: power2-eq-square
algebra-simps)
  qed
  finally show ?thesis .
qed

lemma summable-cot-pfd-complex:
  fixes x :: complex
  shows summable (λn. 2 * x / (x ^ 2 − of-nat (Suc n) ^ 2))
```

using *sums-cot-pfd-complex*[*of x*] **by** (*simp add*: *sums-iff*)

lemma *summable-cot-pfd-real*:
fixes *x* :: *real*
shows *summable* ($\lambda n.$ *2* ∗ *x* / (*x* ^ *2* − *of-nat* (*Suc n*) ^ *2*))
proof −
have *summable* ($\lambda n.$ *complex-of-real* (*2* ∗ *x* / (*x* ^ *2* − *of-nat* (*Suc n*) ^ *2*)))
using *summable-cot-pfd-complex*[*of of-real x*] **by** *simp*
also have *?this* ⟷ *?thesis*
by (*rule summable-of-real-iff*)
finally show *?thesis* **.**
qed

lemma *sums-cot-pfd-real*:
fixes *x* :: *real*
shows ($\lambda n.$ *2* ∗ *x* / (*x* ^ *2* − *of-nat* (*Suc n*) ^ *2*)) *sums cot-pfd x*
using *summable-cot-pfd-real*[*of x*] **by** (*simp add*: *cot-pfd-def sums-iff*)

lemma *cot-pfd-complex-of-real* [*simp*]: *cot-pfd* (*complex-of-real x*) = *of-real* (*cot-pfd x*)
using *sums-of-real*[*OF sums-cot-pfd-real*[*of x*], **where** *?'a* = *complex*]
sums-cot-pfd-complex[*of of-real x*] *sums-unique2* **by** *auto*

lemma *uniform-limit-cot-pfd-real*:
assumes *R* ≥ *0*
shows *uniform-limit* (*cball 0 R* :: *real set*)
($\lambda N\ x.$ ∑ *n*<*N*. *2* ∗ *x* / (*x* ^ *2* − *of-nat* (*Suc n*) ^ *2*)) *cot-pfd sequentially*
proof −
have *uniform-limit* (*cball 0 R*)
($\lambda N\ x.$ *Re* (∑ *n*<*N*. *2* ∗ *x* / (*x* ^ *2* − *of-nat* (*Suc n*) ^ *2*))) ($\lambda x.$ *Re* (*cot-pfd x*)) *sequentially*
by (*intro uniform-limit-intros uniform-limit-cot-pfd-complex assms*)
hence *uniform-limit* (*of-real* ‘ *cball 0 R*)
($\lambda N\ x.$ *Re* (∑ *n*<*N*. *2* ∗ *x* / (*x* ^ *2* − *of-nat* (*Suc n*) ^ *2*))) ($\lambda x.$ *Re* (*cot-pfd x*)) *sequentially*
by (*rule uniform-limit-on-subset*) *auto*
thus *?thesis*
by (*simp add*: *uniform-limit-image*)
qed

1.3 Holomorphicity and continuity

lemma *holomorphic-on-cot-pfd* [*holomorphic-intros*]:
assumes *A* ⊆ −(ℤ−{*0*})
shows *cot-pfd holomorphic-on A*
proof −
have ∗: *open* (−(ℤ−{*0*}) :: *complex set*)
by (*intro open-Compl closed-subset-Ints*) *auto*
define *f* :: *nat* ⇒ *complex* ⇒ *complex*

```
    where f = (λN x. ∑n<N. 2 ∗ x / (x ^ 2 − of-nat (Suc n) ^ 2))
  have cot-pfd holomorphic-on −(ℤ−{0})
  proof (rule holomorphic-uniform-sequence[OF ∗])
    fix n :: nat
    have ∗∗: x^2 − (of-nat (Suc n))^2 ≠ 0 if x ∈ −(ℤ−{0}) for x :: complex and
n :: nat
    proof
      assume x^2 − (of-nat (Suc n))^2 = 0
      hence (of-nat (Suc n))^2 = x^2
        by algebra
      hence x = of-nat (Suc n) ∨ x = −of-nat (Suc n)
        by (subst (asm) eq-commute, subst (asm) power2-eq-iff) auto
      moreover have (of-nat (Suc n) :: complex) ∈ ℤ (−of-nat (Suc n) :: complex)
∈ ℤ
        by (intro Ints-minus Ints-of-nat)+
      ultimately show False using that
        by (auto simp del: of-nat-Suc)
    qed
    show f n holomorphic-on −(ℤ − {0})
      unfolding f-def by (intro holomorphic-intros ∗∗)
  next
    fix z :: complex assume z: z ∈ −(ℤ − {0})
    from ∗ z obtain r where r: r > 0 cball z r ⊆ −(ℤ−{0})
      using open-contains-cball by blast
    have uniform-limit (cball z r) f cot-pfd sequentially
      using uniform-limit-cot-pfd-complex[of norm z + r] unfolding f-def
    proof (rule uniform-limit-on-subset)
      show cball z r ⊆ cball 0 (norm z + r)
        unfolding cball-subset-cball-iff by (auto simp: dist-norm)
    qed (use ‹r > 0› in auto)
    with r show ∃d>0. cball z d ⊆ − (ℤ − {0}) ∧ uniform-limit (cball z d) f
cot-pfd sequentially
      by blast
  qed
  thus ?thesis
    by (rule holomorphic-on-subset) fact
qed

lemma continuous-on-cot-pfd-complex [continuous-intros]:
  assumes A ⊆ −(ℤ−{0})
  shows   continuous-on A (cot-pfd :: complex ⇒ complex)
  by (rule holomorphic-on-imp-continuous-on holomorphic-intros assms)+

lemma continuous-on-cot-pfd-real [continuous-intros]:
  assumes A ⊆ −(ℤ−{0})
  shows   continuous-on A (cot-pfd :: real ⇒ real)
proof −
  have continuous-on A (Re ∘ cot-pfd ∘ of-real)
    by (intro continuous-intros) (use assms in auto)
```

also have *Re* ∘ *cot-pfd* ∘ *of-real* = *cot-pfd*
  by *auto*
finally show *?thesis* **.**
qed

1.4 Functional equations

In this section, we will show three few functional equations for the function *cot-pfd*. The first one is trivial; the other two are a bit tedious and not very insightful, so I will not comment on them.

cot-pfd is an odd function:

lemma *cot-pfd-complex-minus* [*simp*]: *cot-pfd* (−*x* :: *complex*) = −*cot-pfd x*
proof −
  have (λ*n*. *2* ∗ (−*x*) / ((−*x*) ^ *2* − *of-nat* (*Suc n*) ^ *2*)) =
      (λ*n*. − (*2* ∗ *x* / (*x* ^ *2* − *of-nat* (*Suc n*) ^ *2*)))
    by *simp*
  also have . . . *sums* −*cot-pfd x*
    by (*intro sums-minus sums-cot-pfd-complex*)
  finally show *?thesis*
    using *sums-cot-pfd-complex*[*of* −*x*] *sums-unique2* **by** *blast*
qed

lemma *cot-pfd-real-minus* [*simp*]: *cot-pfd* (−*x* :: *real*) = −*cot-pfd x*
  using *cot-pfd-complex-minus*[*of of-real x*]
  unfolding *of-real-minus* [*symmetric*] *cot-pfd-complex-of-real of-real-eq-iff* **.**

cot-pfd is periodic with period 1:

lemma *cot-pfd-plus-1-complex*:
  assumes *x* ∉ ℤ
  shows *cot-pfd* (*x* + *1* :: *complex*) = *cot-pfd x* − *1* / (*x* + *1*) + *1* / *x*
proof −
  have ∗: *x* ^ *2* ≠ *of-nat n* ^ *2* **if** *x* ∉ ℤ **for** *x* :: *complex* **and** *n*
    using *that* **by** (*metis Ints-of-nat minus-in-Ints-iff power2-eq-iff*)
  have ∗∗: *x* + *of-nat n* ≠ *0* **if** *x* ∉ ℤ **for** *x* :: *complex* **and** *n*
    using *that* **by** (*metis Ints-0 Ints-add-iff2 Ints-of-nat*)
  have [*simp*]: *x* ≠ *0*
    using *assms* **by** *auto*
  have [*simp*]: *x* + *1* ≠ *0*
    using *assms* **by** (*metis* ∗∗ *of-nat-1*)
  have [*simp*]: *x* + *2* ≠ *0*
    using ∗∗[*of x 2*] *assms* **by** *simp*

  have *lim*: (λ*n*. *1* / (*x* + *of-nat* (*Suc n*))) ⟶ *0*
   by (*intro tendsto-divide-0*[*OF tendsto-const*] *tendsto-add-filterlim-at-infinity*[*OF tendsto-const*]
       filterlim-compose[*OF tendsto-of-nat*] *filterlim-Suc*)
  have *sum1*: (λ*n*. *1* / (*x* + *of-nat* (*Suc n*)) − *1* / (*x* + *of-nat* (*Suc n* + *2*))) *sums*

```
      (∑n<2. 1 / (x + of-nat (Suc n)))
    using sums-long-telescope[OF lim, of 2] by (simp add: algebra-simps)

  have (λn. 2 * x / (x^2 − (of-nat (Suc n))^2) − 2 * (x + 1) / ((x + 1)^2 −
(of-nat (Suc (Suc n)))^2))
          sums (cot-pfd x − (cot-pfd (x + 1) − 2 * (x + 1) / ((x + 1)^2 − (of-nat
(Suc 0) ^ 2))))
    using sums-cot-pfd-complex[of x + 1]
    by (intro sums-diff sums-cot-pfd-complex, subst sums-Suc-iff) auto
  also have 2 * (x + 1) / ((x + 1)^2 − (of-nat (Suc 0) ^ 2)) = 2 * (x + 1) /
(x * (x + 2))
    by (simp add: algebra-simps power2-eq-square)
  also have (λn. 2 * x / (x^2 − (of-nat (Suc n))^2) −
                2 * (x + 1) / ((x + 1)^2 − (of-nat (Suc (Suc n)))^2)) =
            (λn. 1 / (x + of-nat (Suc n)) − 1 / (x + of-nat (Suc n + 2)))
    using *[of x] *[of x + 1] **[of x] **[of x + 1] assms
    apply (intro ext)
    apply (simp add: divide-simps del: of-nat-add of-nat-Suc)
    apply (simp add: algebra-simps power2-eq-square)
    done
  finally have sum2: (λn. 1 / (x + of-nat (Suc n)) − 1 / (x + of-nat (Suc n +
2))) sums
                        (cot-pfd x − cot-pfd (x + 1) + 2 * (x + 1) / (x * (x + 2)))
    by (simp add: algebra-simps)

  have cot-pfd x − cot-pfd (x + 1) + 2 * (x + 1) / (x * (x + 2)) =
        (∑n<2. 1 / (x + of-nat (Suc n)))
    using sum1 sum2 sums-unique2 by blast
  hence cot-pfd x − cot-pfd (x + 1) = −2 * (x + 1) / (x * (x + 2)) + 1 / (x +
1) + 1 / (x + 2)
    by (simp add: eval-nat-numeral divide-simps) algebra?
  also have ... = 1 / (x + 1) − 1 / x
    by (simp add: divide-simps) algebra?
  finally show ?thesis
    by algebra
qed

lemma cot-pfd-plus-1-real:
  assumes x ∉ ℤ
  shows   cot-pfd (x + 1 :: real) = cot-pfd x − 1 / (x + 1) + 1 / x
proof −
  have cot-pfd (complex-of-real (x + 1)) = cot-pfd (of-real x) − 1 / (of-real x +
1) + 1 / of-real x
    using cot-pfd-plus-1-complex[of x] assms by simp
  also have ... = complex-of-real (cot-pfd x − 1 / (x + 1) + 1 / x)
    by simp
  finally show ?thesis
    unfolding cot-pfd-complex-of-real of-real-eq-iff .
qed
```

cot-pfd satisfies the following functional equation:

$$2f(x) = f\left(\frac{x}{2}\right) + f\left(\frac{x+1}{2}\right) + \frac{2}{x+1}$$

lemma *cot-pfd-funeq-complex*:
fixes *x* :: *complex*
assumes $x \notin \mathbb{Z}$
shows *2 ∗ cot-pfd x = cot-pfd (x / 2) + cot-pfd ((x + 1) / 2) + 2 / (x + 1)*
proof −
define *f* :: *complex ⇒ nat ⇒ complex* **where** *f = (λx n. 1 / (x + of-nat (Suc n)))*
define *g* :: *complex ⇒ nat ⇒ complex* **where** *g = (λx n. 1 / (x − of-nat (Suc n)))*
define *h* :: *complex ⇒ nat ⇒ complex* **where** *h = (λx n. 2 ∗ (f x (n + 1) + g x n))*

have *sums*: *(λn. f x n + g x n) sums cot-pfd x* **if** $x \notin \mathbb{Z}$ **for** *x*
unfolding *f-def g-def* **by** *(intro sums-cot-pfd-complex′ that)*

have *x / 2* $\notin \mathbb{Z}$
proof
assume *x / 2* $\in \mathbb{Z}$
hence *2 ∗ (x / 2)* $\in \mathbb{Z}$
by *(intro Ints-mult) auto*
thus *False* **using** *assms* **by** *simp*
qed
moreover have *(x + 1) / 2* $\notin \mathbb{Z}$
proof
assume *(x + 1) / 2* $\in \mathbb{Z}$
hence *2 ∗ ((x + 1) / 2) − 1* $\in \mathbb{Z}$
by *(intro Ints-mult Ints-diff) auto*
thus *False* **using** *assms* **by** *(simp add: field-simps)*
qed
ultimately have *(λn. (f (x / 2) n + g (x / 2) n) + (f ((x+1) / 2) n + g ((x+1) / 2) n)) sums*
(cot-pfd (x / 2) + cot-pfd ((x + 1) / 2))
by *(intro sums-add sums)*

also have *(λn. (f (x / 2) n + g (x / 2) n) + (f ((x+1) / 2) n + g ((x+1) / 2) n)) =*
(λn. h x (2 ∗ n) + h x (2 ∗ n + 1))
proof
fix *n* :: *nat*
have *(f (x / 2) n + g (x / 2) n) + (f ((x+1) / 2) n + g ((x+1) / 2) n) =*
(f (x / 2) n + f ((x+1) / 2) n) + (g (x / 2) n + g ((x+1) / 2) n)
by *algebra*
also have *f (x / 2) n + f ((x+1) / 2) n = 2 ∗ (f x (2 ∗ n + 1) + f x (2 ∗ n + 2))*

by (*simp add*: *f-def field-simps*)
also have $g\ (x\ /\ 2)\ n + g\ ((x+1)\ /\ 2)\ n = 2 * (g\ x\ (2 * n) + g\ x\ (2 * n + 1))$
by (*simp add*: *g-def field-simps*)
also have $2 * (f\ x\ (2 * n + 1) + f\ x\ (2 * n + 2)) + \ldots =$
$h\ x\ (2 * n) + h\ x\ (2 * n + 1)$
unfolding *h-def* **by** (*simp add*: *algebra-simps*)
finally show $(f\ (x\ /\ 2)\ n + g\ (x\ /\ 2)\ n) + (f\ ((x+1)\ /\ 2)\ n + g\ ((x+1)\ /\ 2)\ n) =$
$h\ x\ (2 * n) + h\ x\ (2 * n + 1)$ **.**
qed
finally have *sum1*:
$(\lambda n.\ h\ x\ (2 * n) + h\ x\ (2 * n + 1))$ *sums* (*cot-pfd* $(x\ /\ 2)$ + *cot-pfd* $((x + 1)\ /\ 2))$ **.**

have $f\ x \longrightarrow 0$ **unfolding** *f-def*
by (*intro tendsto-divide-0*[*OF tendsto-const*]
tendsto-add-filterlim-at-infinity[*OF tendsto-const*]
filterlim-compose[*OF tendsto-of-nat*] *filterlim-Suc*)
hence $(\lambda n.\ 2 * (f\ x\ n + g\ x\ n) + 2 * (f\ x\ (Suc\ n) - f\ x\ n))$ *sums* $(2 *$ *cot-pfd* $x + 2 * (0 - f\ x\ 0))$
by (*intro sums-add sums sums-mult telescope-sums assms*)
also have $(\lambda n.\ 2 * (f\ x\ n + g\ x\ n) + 2 * (f\ x\ (Suc\ n) - f\ x\ n)) = h\ x$
by (*simp add*: *h-def algebra-simps fun-eq-iff*)
finally have $*$: $h\ x$ *sums* $(2 *$ *cot-pfd* $x - 2 * f\ x\ 0)$
by *simp*

have $(\lambda n.\ sum\ (h\ x)\ \{n * 2..<n * 2 + 2\})$ *sums* $(2 *$ *cot-pfd* $x - 2 * f\ x\ 0)$
using *sums-group*[*OF* $*$, *of 2*] **by** *simp*
also have $(\lambda n.\ sum\ (h\ x)\ \{n*2..<n*2+2\}) = (\lambda n.\ h\ x\ (2 * n) + h\ x\ (2 * n + 1))$
by (*simp add*: *mult-ac*)
finally have *sum2*: $(\lambda n.\ h\ x\ (2 * n) + h\ x\ (2 * n + 1))$ *sums* $(2 *$ *cot-pfd* $x - 2 * f\ x\ 0)$ **.**

have *cot-pfd* $(x\ /\ 2)$ + *cot-pfd* $((x + 1)\ /\ 2) = 2 *$ *cot-pfd* $x - 2 * f\ x\ 0$
using *sum1 sum2 sums-unique2* **by** *blast*
also have $2 * f\ x\ 0 = 2\ /\ (x + 1)$
by (*simp add*: *f-def*)
finally show *?thesis* **by** *algebra*
qed

lemma *cot-pfd-funeq-real*:
fixes $x :: real$
assumes $x \notin \mathbb{Z}$
shows $2 *$ *cot-pfd* x = *cot-pfd* $(x\ /\ 2)$ + *cot-pfd* $((x + 1)\ /\ 2) + 2\ /\ (x + 1)$
proof −
have *complex-of-real* $(2 *$ *cot-pfd* $x) = 2 *$ *cot-pfd* (*complex-of-real* x)
by *simp*

also have ... = *complex-of-real* (*cot-pfd* (*x* / *2*) + *cot-pfd* ((*x* + *1*) / *2*) + *2* / (*x* + *1*))
 using *assms* **by** (*subst cot-pfd-funeq-complex*) (*auto simp flip*: *cot-pfd-complex-of-real*)
 finally show *?thesis*
  by (*simp only*: *of-real-eq-iff*)
qed

1.5 The limit at 0

lemma *cot-pfd-real-tendsto-0*: *cot-pfd* −*0*→ (*0* :: *real*)
proof −
 have *filterlim cot-pfd* (*nhds 0*) (*at* (*0* :: *real*) *within ball 0 1*)
 proof (*rule swap-uniform-limit*)
  show *uniform-limit* (*ball 0 1*)
   ($\lambda N\ x.\ \sum n{<}N.\ 2 * x\ /\ (x^2 - (real\ (Suc\ n))^2)$) *cot-pfd sequentially*
  using *uniform-limit-cot-pfd-real*[*OF zero-le-one*] **by** (*rule uniform-limit-on-subset*) *auto*
  have (($\lambda x.\ 2 * x\ /\ (x^2 - (real\ (Suc\ n))^2)$) ⟶ *0*) (*at 0 within ball 0 1*) **for** *n*
  proof (*rule filterlim-mono*)
   show (($\lambda x.\ 2 * x\ /\ (x^2 - (real\ (Suc\ n))^2)$) ⟶ *0*) (*at 0*)
    by *real-asymp*
  qed (*auto simp*: *at-within-le-at*)
  thus $\forall_F$ *N in sequentially.*
   (($\lambda x.\ \sum n{<}N.\ 2 * x\ /\ (x^2 - (real\ (Suc\ n))^2)$) ⟶ *0*) (*at 0 within ball 0 1*)
   by (*intro always-eventually allI tendsto-null-sum*)
 qed *auto*
 thus *?thesis*
  by (*simp add*: *at-within-open-NO-MATCH*)
qed

1.6 Final result

To show the final result, we first prove the real case using Herglotz's trick, following the presentation in 'Proofs from THE BOOK'. [1, Chapter 23].

lemma *cot-pfd-formula-real*:
 assumes $x \notin \mathbb{Z}$
 shows *pi* ∗ *cot* (*pi* ∗ *x*) = *1* / *x* + *cot-pfd x*
proof −
 have *ev-not-int*: *eventually* ($\lambda x.\ r\ x \notin \mathbb{Z}$) (*at x*)
  if *filterlim r* (*at* (*r x*)) (*at x*) **for** *r* :: *real* ⇒ *real* **and** *x* :: *real*
 proof (*rule eventually-compose-filterlim*[*OF - that*])
  show *eventually* ($\lambda x.\ x \notin \mathbb{Z}$) (*at* (*r x*))
   using *Ints-not-limpt*[*of r x*] *islimpt-iff-eventually* **by** *blast*
 qed

We define the function $h(z)$ as the difference of the left-hand side and right-hand side. The left-hand side and right-hand side have singularities at the integers, but we will later see that these can be removed as *h* tends to *0* there.

```
define f :: real ⇒ real where f = (λx. pi ∗ cot (pi ∗ x))
define g :: real ⇒ real where g = (λx. 1 / x + cot-pfd x)
define h where h = (λx. if x ∈ ℤ then 0 else f x − g x)

have [simp]: h x = 0 if x ∈ ℤ for x
  using that by (simp add: h-def)
```

It is easy to see that the left-hand side and the right-hand side, and as a consequence also our function *h*, are odd and periodic with period 1.

```
have odd-h: h (−x) = −h x for x
  by (simp add: h-def minus-in-Ints-iff f-def g-def)
have per-f: f (x + 1) = f x for x
  by (simp add: f-def algebra-simps cot-def)
have per-g: g (x + 1) = g x if x ∉ ℤ for x
  using that by (simp add: g-def cot-pfd-plus-1-real)
interpret h: periodic-fun-simple′ h
  by standard (auto simp: h-def per-f per-g)
```

h tends to 0 at 0 (and thus at all the integers).

```
have h-lim: h −0→ 0
proof (rule Lim-transform-eventually)
  have eventually (λx. x ∉ ℤ) (at (0 :: real))
    by (rule ev-not-int) real-asymp
  thus eventually (λx::real. pi ∗ cot (pi ∗ x) − 1 / x − cot-pfd x = h x) (at 0)
    by eventually-elim (simp add: h-def f-def g-def)
next
  have (λx::real. pi ∗ cot (pi ∗ x) − 1 / x) −0→ 0
    unfolding cot-def by real-asymp
  hence (λx::real. pi ∗ cot (pi ∗ x) − 1 / x − cot-pfd x) −0→ 0 − 0
    by (intro tendsto-intros cot-pfd-real-tendsto-0)
  thus (λx. pi ∗ cot (pi ∗ x) − 1 / x − cot-pfd x) −0→ 0
    by simp
qed
```

This means that our *h* is in fact continuous everywhere:

```
have cont-h: continuous-on A h for A
proof −
  have isCont h x for x
  proof (cases x ∈ ℤ)
    case True
    then obtain n where [simp]: x = of-int n
      by (auto elim: Ints-cases)
    show ?thesis unfolding isCont-def
     by (subst at-to-0) (use h-lim in ‹simp add: filterlim-filtermap h.plus-of-int›)
  next
    case False
```

have *continuous-on* $(-\mathbb{Z})$ $(\lambda x.\ f\ x - g\ x)$
by (*auto simp*: *f-def g-def sin-times-pi-eq-0 mult.commute*[*of pi*] *intro*!: *continuous-intros*)
hence *isCont* $(\lambda x.\ f\ x - g\ x)\ x$
by (*rule continuous-on-interior*)
(*use False* **in** ‹*auto simp*: *interior-open open-Compl*[*OF closed-Ints*]›)
also have *eventually* $(\lambda y.\ y \in -\mathbb{Z})$ (*nhds x*)
using *False* **by** (*intro eventually-nhds-in-open*) *auto*
hence *eventually* $(\lambda x.\ f\ x - g\ x = h\ x)$ (*nhds x*)
by *eventually-elim* (*auto simp*: *h-def*)
hence *isCont* $(\lambda x.\ f\ x - g\ x)\ x \longleftrightarrow$ *isCont h x*
by (*rule isCont-cong*)
finally show *?thesis* **.**
qed
thus *?thesis*
by (*simp add*: *continuous-at-imp-continuous-on*)
qed
note [*continuous-intros*] = *continuous-on-compose2*[*OF cont-h*]

Through the functional equations of the sine and cosine function, we can derive the following functional equation for f that holds for all non-integer reals:

have *eq-f*: $f\ x = (f\ (x\ /\ 2) + f\ ((x + 1)\ /\ 2))\ /\ 2$ **if** $x \notin \mathbb{Z}$ **for** x
proof −
have $x\ /\ 2 \notin \mathbb{Z}$
using *that* **by** (*metis Ints-add field-sum-of-halves*)
hence *nz1*: $sin\ (x/2 * pi) \neq 0$
by (*subst sin-times-pi-eq-0*) *auto*

have $(x + 1)\ /\ 2 \notin \mathbb{Z}$
proof
assume $(x + 1)\ /\ 2 \in \mathbb{Z}$
hence $2 * ((x + 1)\ /\ 2) - 1 \in \mathbb{Z}$
by (*intro Ints-mult Ints-diff*) *auto*
thus *False* **using** *that* **by** (*simp add*: *field-simps*)
qed
hence *nz2*: $sin\ ((x+1)/2 * pi) \neq 0$
by (*subst sin-times-pi-eq-0*) *auto*

have *nz3*: $sin\ (x * pi) \neq 0$
using *that* **by** (*subst sin-times-pi-eq-0*) *auto*

have *eq*: $sin\ (pi * x) = 2 * sin\ (pi * x\ /\ 2) * cos\ (pi * x\ /\ 2)$
$cos\ (pi * x) = (cos\ (pi * x\ /\ 2))^2 - (sin\ (pi * x\ /\ 2))^2$
using *sin-double*[*of pi* $*$ *x* / *2*] *cos-double*[*of pi* $*$ *x* / *2*] **by** *simp-all*
show *?thesis* **using** *nz1 nz2 nz3*
apply (*simp add*: *f-def cot-def field-simps*)
apply (*simp add*: *add-divide-distrib sin-add cos-add power2-eq-square eq algebra-simps*)

```
    done
  qed
```

The corresponding functional equation for *cot-pfd* that we have already shown leads to the same functional equation for g as we just showed for f:

```
  have eq-g: g x = (g (x / 2) + g ((x + 1) / 2)) / 2 if x ∉ ℤ for x
    using cot-pfd-funeq-real[OF that] by (simp add: g-def)
```

This then leads to the same functional equation for h, and because h is continuous everywhere, we can extend the validity of the equation to the full domain.

```
  have eq-h: h x = (h (x / 2) + h ((x + 1) / 2)) / 2 for x
  proof −
    have eventually (λx. x ∉ ℤ) (at x) eventually (λx. x / 2 ∉ ℤ) (at x)
         eventually (λx. (x + 1) / 2 ∉ ℤ) (at x)
      by (rule ev-not-int; real-asymp)+
    hence eventually (λx. h x − (h (x / 2) + h ((x + 1) / 2)) / 2 = 0) (at x)
    proof eventually-elim
      case (elim x)
      thus ?case using eq-f[of x] eq-g[of x]
        by (simp add: h-def field-simps)
    qed
    hence (λx. h x − (h (x / 2) + h ((x + 1) / 2)) / 2) −x→ 0
      by (simp add: tendsto-eventually)
    moreover have continuous-on UNIV (λx. h x − (h (x / 2) + h ((x + 1) /
2)) / 2)
      by (auto intro!: continuous-intros)
    ultimately have h x − (h (x / 2) + h ((x + 1) / 2)) / 2 = 0
      by (meson LIM-unique UNIV-I continuous-on-def)
    thus ?thesis
      by simp
  qed
```

Since h is periodic with period 1 and continuous, it must attain a global maximum h somewhere in the interval $[0, 1]$. Let's call this maximum m and let x_0 be some point in the interval $[0, 1]$ such that $h(x_0) = m$.

```
  define m where m = Sup (h ‘ {0..1})
  have m ∈ h ‘ {0..1}
    unfolding m-def
  proof (rule closed-contains-Sup)
    have compact (h ‘ {0..1})
      by (intro compact-continuous-image cont-h) auto
    thus bdd-above (h ‘ {0..1}) closed (h ‘ {0..1})
     by (auto intro: compact-imp-closed compact-imp-bounded bounded-imp-bdd-above)
  qed auto
  then obtain x0 where x0: x0 ∈ {0..1} h x0 = m
    by blast
```

have *h-le-m*: *h x* ≤ *m* **for** *x*
proof −
have *h x* = *h* (*frac x*)
unfolding *frac-def* **by** (*rule h.minus-of-int* [*symmetric*])
also have ... ≤ *m* **unfolding** *m-def*
proof (*rule cSup-upper*)
have *frac x* ∈ {*0..1*}
using *frac-lt-1*[*of x*] **by** *auto*
thus *h* (*frac x*) ∈ *h* ' {*0..1*}
by *blast*
next
have *compact* (*h* ' {*0..1*})
by (*intro compact-continuous-image cont-h*) *auto*
thus *bdd-above* (*h* ' {*0..1*})
by (*auto intro*: *compact-imp-bounded bounded-imp-bdd-above*)
qed
finally show *?thesis* **.**
qed

Through the functional equation for h, we can show that if h attains its maximum at some point x, it also attains it at $\frac{1}{2}x$. By iterating this, it attains the maximum at all points of the form $2^{-n}x_0$.

have *h-eq-m-iter-aux*: *h* (*x* / *2*) = *m* **if** *h x* = *m* **for** *x*
using *eq-h*[*of x*] *that h-le-m*[*of x* / *2*] *h-le-m*[*of* (*x* + *1*) / *2*] **by** *simp*
have *h-eq-m-iter*: *h* (*x0* / *2* ^ *n*) = *m* **for** *n*
proof (*induction n*)
case (*Suc n*)
have *h* (*x0* / *2* ^ *Suc n*) = *h* (*x0* / *2* ^ *n* / *2*)
by (*simp add*: *field-simps*)
also have ... = *m*
by (*rule h-eq-m-iter-aux*) (*use Suc.IH* **in** *auto*)
finally show *?case* **.**
qed (*use x0* **in** *auto*)

Since the sequence $n \mapsto 2^{-n}x_0$ tends to 0 and h is continuous, we derive *m* = *0*.

have (*λn. h* (*x0* / *2* ^ *n*)) ⟶ *h 0*
by (*rule continuous-on-tendsto-compose*[*OF cont-h*[*of UNIV*]]) (*force* | *real-asymp*)+
moreover from *h-eq-m-iter* **have** (*λn. h* (*x0* / *2* ^ *n*)) ⟶ *m*
by *simp*
ultimately have *m* = *h 0*
using *tendsto-unique* **by** *force*
hence *m* = *0*
by *simp*

Since h is odd, this means that h is identically zero everywhere, and our result follows.

have *h x* = *0*

using *h-le-m*[*of x*] *h-le-m*[*of* −*x*] ‹*m* = *0*› *odd-h*[*of x*] **by** *linarith*
thus *?thesis*
using *assms* **by** (*simp add*: *h-def f-def g-def*)
qed

We now lift the result from the domain $\mathbb{R}\backslash\mathbb{Z}$ to $\mathbb{C}\backslash\mathbb{Z}$. We do this by noting that $\mathbb{C}\backslash\mathbb{Z}$ is connected and the point $\frac{1}{2}$ is both in $\mathbb{C}\backslash\mathbb{Z}$ and a limit point of $\mathbb{R}\backslash\mathbb{Z}$.

lemma *one-half-limit-point-Reals-minus-Ints*: (*1* / *2* :: *complex*) *islimpt* ℝ − ℤ
proof (*rule islimptI*)
fix *T* :: *complex set*
assume *1* / *2* ∈ *T open T*
then obtain *r* **where** *r*: *r* > *0 ball* (*1* / *2*) *r* ⊆ *T*
using *open-contains-ball* **by** *blast*
define *y* **where** *y* = *1* / *2* + *min r* (*1* / *2*) / *2*
have *y* ∈ {*0*<..<*1*}
using *r* **by** (*auto simp*: *y-def*)
hence *complex-of-real y* ∈ ℝ − ℤ
by (*auto elim*!: *Ints-cases*)
moreover have *complex-of-real y* ≠ *1* / *2*
proof
assume *complex-of-real y* = *1* / *2*
also have *1* / *2* = *complex-of-real* (*1* / *2*)
by *simp*
finally have *y* = *1* / *2*
unfolding *of-real-eq-iff* **.**
with *r* **show** *False*
by (*auto simp*: *y-def*)
qed
moreover have *complex-of-real y* ∈ *ball* (*1* / *2*) *r*
using ‹*r* > *0*› **by** (*auto simp*: *y-def dist-norm*)
with *r* **have** *complex-of-real y* ∈ *T*
by *blast*
ultimately show ∃ *y*∈ℝ − ℤ. *y* ∈ *T* ∧ *y* ≠ *1* / *2*
by *blast*
qed

theorem *cot-pfd-formula-complex*:
fixes *z* :: *complex*
assumes *z* ∉ ℤ
shows *pi* ∗ *cot* (*pi* ∗ *z*) = *1* / *z* + *cot-pfd z*
proof −
let *?f* = λ*z*::*complex*. *pi* ∗ *cot* (*pi* ∗ *z*) − *1* / *z* − *cot-pfd z*
have *pi* ∗ *cot* (*pi* ∗ *z*) − *1* / *z* − *cot-pfd z* = *0*
proof (*rule analytic-continuation*[**where** *f* = *?f*])
show *?f holomorphic-on* −ℤ
unfolding *cot-def* **by** (*intro holomorphic-intros*) (*auto simp*: *sin-eq-0*)
next
show *open* (−ℤ :: *complex set*) *connected* (−ℤ :: *complex set*)

by (*auto intro*!: *path-connected-imp-connected path-connected-complement-countable countable-int*)
next
show $\mathbb{R} - \mathbb{Z} \subseteq (-\mathbb{Z} :: complex\ set)$
by *auto*
next
show $(1\ /\ 2 :: complex)$ *islimpt* $\mathbb{R} - \mathbb{Z}$
by (*rule one-half-limit-point-Reals-minus-Ints*)
next
show $1\ /\ (2 :: complex) \in -\mathbb{Z}$
using *fraction-not-in-ints*[*of 2 1*, **where** *?'a* = *complex*] **by** *auto*
next
show $z \in -\mathbb{Z}$
using *assms* **by** *simp*
next
show *?f z* = *0* **if** $z \in \mathbb{R} - \mathbb{Z}$ **for** *z*
proof −
have *complex-of-real pi* ∗ *cot* (*complex-of-real pi* ∗ *z*) − *1* / *z* − *cot-pfd z* = *complex-of-real* (*pi* ∗ *cot* (*pi* ∗ *Re z*) − *1* / *Re z* − *cot-pfd* (*Re z*))
using *that* **by** (*auto elim*!: *Reals-cases simp*: *cot-of-real*)
also have ... = *0*
by (*subst cot-pfd-formula-real*) (*use that* **in** ‹*auto elim*!: *Reals-cases*›)
finally show *?thesis* **.**
qed
qed
thus *?thesis*
by *algebra*
qed

end

References

[1] M. Aigner and G. M. Ziegler. *Proofs from THE BOOK*. Springer, 4th edition, 2009.