A simple proof that π is irrational

Manuel Eberl

March 17, 2025

Abstract

This entry provides a formalisation of Niven's famously short one-page proof that π is irrational. The proof uses only elementary algebra and analysis.

The intrinsic de Bruijn factor, i.e. the file size ratio between the gzipped Isabelle sources and a gzipped LaTeX version of the original paper's content, is roughly 4 despite the original paper's terse presentation.

Contents

1 A short proof of the irrationality of π

```
theory Pi_Irrational
imports
  "HOL-Analysis.Analysis"
  "Polynomial_Interpolation.Ring_Hom_Poly"
begin
```

1.1 Auxiliary material

```
lemma fact_dvd_pochhammer:
  assumes "m ≤ n + 1"
  shows   "fact m dvd pochhammer (int n - int m + 1) m"
proof -
  have "(real n gchoose m) * fact m = of_int (pochhammer (int n - int
m + 1) m)"
    by (simp add: gbinomial_pochhammer' pochhammer_of_int [symmetric])
  also have "(real n gchoose m) * fact m = of_int (int (n choose m) *
fact m)"
    by (simp add: binomial_gbinomial)
  finally have "int (n choose m) * fact m = pochhammer (int n - int m +
1) m"
    by (subst (asm) of_int_eq_iff)
  from this [symmetric] show ?thesis by simp
qed

lemma factor_dvd_higher_pderiv:
  fixes p :: "'a :: idom poly"
  assumes "p ^ n dvd q" "i < n"
  shows   "p dvd (pderiv ^^ i) q"
proof -
  from assms(1) obtain r where r: "q = p ^ n * r"
    by (elim dvdE)
  have "p dvd (pderiv ^^ i) (p ^ n * r)"
    using ‹n > i›
  proof (induction i arbitrary: r n)
    case (Suc i r n)
    have "p dvd (pderiv ^^ i) (p ^ n * pderiv r)"
      by (rule Suc.IH) (use Suc.prems in auto)
    moreover have "p dvd (pderiv ^^ i) (r * (p ^ (n - 1) * pderiv p))"
      using Suc.prems Suc.IH[of "n-1" "r * pderiv p"] by (simp add: algebra_simps)
    ultimately show ?case
      by (auto simp: le_imp_power_dvd pderiv_mult pderiv_power higher_pderiv_add
pderiv_smult
                  higher_pderiv_smult funpow_Suc_right
              simp flip: funpow.simps intro!: dvd_add dvd_smult)
  qed auto
  with r show ?thesis
    by simp
qed
```

```
lemma fact_dvd_higher_pderiv:
  "[:fact n :: int:] dvd (pderiv ^^ n) p"
proof -
  have "[:fact n:] dvd (pderiv ^^ n) (monom c k)" for c :: int and k
:: nat
    by (cases "n ≤ k + 1")
       (simp_all add: higher_pderiv_monom higher_pderiv_monom_eq_zero
          fact_dvd_pochhammer const_poly_dvd_iff)
  hence "[:fact n:] dvd (pderiv ^^ n) (∑k≤degree p. monom (coeff p k)
k)"
    by (simp_all add: higher_pderiv_sum dvd_sum)
  thus ?thesis by (simp add: poly_as_sum_of_monoms)
qed

lemma higher_pderiv_eq_0_iff:
  fixes p :: "'a::{comm_semiring_1,semiring_no_zero_divisors,semiring_char_0}
poly"
  shows   "(pderiv ^^ n) p = 0 ⟷ p = 0 ∨ n > degree p"
  by (cases n) (auto simp: pderiv_eq_0_iff degree_higher_pderiv)

lemma higher_pderiv_pcompose_linear:
  shows "(pderiv ^^ n) (pcompose p [:a, b:]) = smult (b ^ n) (pcompose
((pderiv ^^ n) p) [:a, b:])"
  by (induction n)
     (auto simp:  simp: pderiv_smult pderiv_pcompose algebra_simps pderiv_pCons)

lemma power_over_fact_tendsto_0:
  "(λn. (x :: real) ^ n / fact n) ⟶ 0"
  using summable_exp[of x] by (intro summable_LIMSEQ_zero) (simp add:
sums_iff field_simps)
```

1.2 Main proof

```
locale pi_rational =
  fixes a b :: int
  assumes ab: "a / b = pi"
  assumes b: "b > 0"
begin

context
  fixes n :: nat
  assumes n: "n > 1"
begin

definition f :: "real poly" where
  "f = smult (1/fact n) ([:0, of_int a, -of_int b:] ^ n)"

lemma f_mirror: "f ∘p [:pi, -1:] = f"
```

```
  using b by (simp add: f_def pcompose_smult hom_distribs algebra_simps
flip: ab)

lemma degree_f [simp]: "degree f = 2 * n"
  using b by (simp add: f_def degree_mult_eq degree_power_eq)


definition F :: "real poly" where
  "F = (∑j≤n. (-1)^j * (pderiv ^^ (2*j)) f)"

lemma F_mirror: "F ∘p [:pi, -1:] = F"
proof -
  have "F ∘p [:pi, -1:] =
          (∑j≤n. (- 1) ^ j * (pderiv ^^ (2 * j)) f ∘p [:pi, -1:])"
    by (simp add: F_def hom_distribs)
  also have "... = (∑j≤n. (-1) ^ j * (pderiv ^^ (2 * j)) (f ∘p [:pi,
-1:]))"
    by (intro sum.cong) (auto simp: higher_pderiv_pcompose_linear)
  also have "... = F"
    by (simp add: f_mirror F_def)
  finally show ?thesis .
qed

lemma poly_F_pi: "poly F pi = poly F 0"
proof -
  have "poly F pi = poly (F ∘p [:pi, -1:]) 0"
    by (simp add: poly_pcompose)
  also have "... = poly F 0"
    by (subst F_mirror) auto
  finally show ?thesis .
qed

lemma F_int: "poly F 0 ∈ ℤ"
proof -
  have "poly ((pderiv ^^ j) f) 0 ∈ ℤ" for j
  proof (cases "j ≥ n")
    case False
    have "[:0, of_int a, -of_int b:] dvd (pderiv ^^ j) f"
      by (rule factor_dvd_higher_pderiv[of _ n])
         (use False in ‹auto simp: f_def dvd_smult›)
    hence "poly ((pderiv ^^ j) f) 0 = 0"
      by (auto elim!: dvdE simp flip: ab)
    thus ?thesis
      by simp
  next
    case True
    define f_aux where "f_aux = [:0, a, -b:] ^ n"
    have "[:fact n:] dvd [:fact j :: int:]"
      using True by (simp add: fact_dvd)
```

```
    also have "[:fact j:] dvd (pderiv ^^ j) f_aux"
      by (rule fact_dvd_higher_pderiv)
    finally obtain q where q: "(pderiv ^^ j) f_aux = smult (fact n) q"
      by (elim dvdE) auto

    have "f = smult (1 / fact n) (of_int_poly f_aux)"
      by (simp add: f_aux_def f_def hom_distribs)
    also have "(pderiv ^^ j) ... = of_int_poly q"
      by (simp add: q hom_distribs higher_pderiv_smult flip: of_int_hom.map_poly_higher_pde
    finally show ?thesis
      by simp
  qed
  thus "poly F 0 ∈ ℤ"
    unfolding F_def by (auto simp: poly_sum)
qed

lemma antideriv:
  "((λx. poly (pderiv F) x * sin x - poly F x * cos x)
     has_field_derivative (poly f x * sin x)) (at x within A)"
proof -
  have  "((λx. poly (pderiv F) x * sin x - poly F x * cos x)
           has_field_derivative (poly (pderiv (pderiv F) + F) x * sin
x)) (at x within A)"
    by (auto intro!: derivative_eq_intros simp: algebra_simps)
  also have "pderiv (pderiv F) + F = f"
  proof -
    have "pderiv (pderiv F) + F =
                 (∑j≤n. (-1) ^ j * (pderiv ^^ (2*j+2)) f) +
                 (∑j≤n. (-1) ^ j * (pderiv ^^ (2*j)) f)"
      by (simp add: F_def pderiv_sum pderiv_mult pderiv_add pderiv_power
pderiv_minus)
    also have "(∑j≤n. (-1) ^ j * (pderiv ^^ (2*j+2)) f) =
                (∑j∈{1..n+1}. (-1) ^ (j+1) * (pderiv ^^ (2*j)) f)"
      by (intro sum.reindex_bij_witness[of _ "λj. j-1" "λj. j+1"]) auto
    also have "... = (∑j∈{1..n}. (-1) ^ (j+1) * (pderiv ^^ (2*j)) f)"
      by (intro sum.mono_neutral_right) (auto simp: not_le higher_pderiv_eq_0_iff)
    also have "{1..n} = {..n} - {0}"
      by auto
    also have "(∑j∈... (-1) ^ (j+1) * (pderiv ^^ (2*j)) f) =
                (∑j≤n. (-1) ^ (j+1) * (pderiv ^^ (2*j)) f) + f"
      by (subst sum_diff) (use n in auto)
    finally show ?thesis
      by (simp add: sum_negf)
  qed
  finally show ?thesis .
qed

lemma bound: "pi / 2 * (a * pi) ^ n / fact n ≥ 1"
proof -
```

```
  define I where "I = (λx. poly (pderiv F) x * sin x - poly F x * cos
x)"
  have integral: "((λx. poly f x * sin x) has_integral (2 * poly F 0))
{0..pi}"
  proof -
    have "((λx. poly f x * sin x) has_integral (I pi - I 0)) {0..pi}"
    proof (rule fundamental_theorem_of_calculus)
      show "(I has_vector_derivative (poly f x * sin x)) (at x within
{0..pi})" for x
        unfolding I_def using antideriv[of x] by (simp add: has_real_derivative_iff_has_vec
    qed auto
    also have "I pi - I 0 = poly F 0 + poly F pi"
      by (simp add: I_def)
    finally show ?thesis
      by (simp add: poly_F_pi)
  qed

  have nonneg: "poly f x * sin x ≥ 0" if x: "x ∈ {0..pi}" for x
    by (use x b in ‹auto simp: f_def sin_ge_zero divide_simps simp flip:
ab›)

  have bounds: "poly f x * sin x ∈ {0<..(a*pi)^n / fact n}" if x: "x ∈
{0<..<pi}" for x
  proof -
    have "poly f x > 0"
      using x b by (auto simp: f_def sin_gt_zero field_simps simp flip:
ab)

    have "poly f x * sin x ≤ poly f x * 1"
      using ‹poly f x > 0› by (intro mult_left_mono) auto
    also have "poly f x * 1 = (x * (pi - x) * b) ^ n / fact n"
      using b  by (simp add: f_def field_simps flip: ab power_mult_distrib)
    also have "... ≤ (pi * pi * b) ^ n / fact n"
      using x b by (intro divide_right_mono power_mono mult_mono) auto
    also have "pi * pi * b = a * pi"
      by (simp flip: ab)
    finally have "poly f x * sin x ≤ (a * pi) ^ n / fact n"
      by simp
    moreover have "poly f x * sin x > 0"
      using ‹poly f x > 0› x by (simp add: sin_gt_zero)
    ultimately show "poly f x * sin x ∈ {0<..(a*pi)^n / fact n}"
      by auto
  qed

  have "poly F 0 > 0"
  proof -
    have "integral {0..pi} (λx. poly f x * sin x) ≠ 0"
    proof (subst integral_eq_0_iff)
      have "poly f (pi/2) * sin (pi/2) ≠ 0" and "pi / 2 ∈ {0..pi}"
```

```
        using bounds[of "pi/2"] by auto
      thus "¬(∀x∈{0..pi}. poly f x * sin x = 0)"
        by blast
    qed (use nonneg in ‹auto intro!: continuous_intros›)
    hence "poly F 0 ≠ 0"
      using integral by (simp add: has_integral_iff)
    moreover have "2 * poly F 0 ≥ 0"
      by (rule has_integral_nonneg[OF integral]) (use nonneg in auto)
    ultimately show ?thesis
      by linarith
  qed
  moreover have "poly F 0 ∈ ℤ"
    using F_int by auto
  ultimately have "1 ≤ poly F 0"
    by (auto elim!: Ints_cases)

  also have "2 * poly F 0 ≤ pi * (a * pi) ^ n / fact n"
  proof (rule has_integral_le)
    have "((λ_. (a * pi) ^ n / fact n * 1) has_integral ((a * pi) ^ n
/ fact n) * pi) {0<..<pi}"
      by (rule has_integral_mult_right, subst has_integral_iff_emeasure_lborel)
auto
    thus "((λ_. (a * pi) ^ n / fact n) has_integral (pi * (a * pi) ^ n
/ fact n)) {0<..<pi}"
      by (simp add: algebra_simps)
  qed (use bounds integral in ‹auto simp: has_integral_Icc_iff_Ioo›)
  hence "poly F 0 ≤ pi / 2 * (a * pi) ^ n / fact n"
    by (simp add: field_simps)

  finally show ?thesis .
qed

end

lemma absurd: False
proof -
  have lim: "(λn. pi / 2 * ((a * pi) ^ n / fact n)) ⟶ (pi / 2 * 0)"
    by (rule tendsto_intros power_over_fact_tendsto_0)+
  have "eventually (λn. pi / 2 * (a * pi) ^ n / fact n < 1) at_top"
    using order_tendstoD(2)[OF lim, of 1] by (auto simp: mult_ac)
  hence "eventually (λn. n > 1 ∧ pi / 2 * (a * pi) ^ n / fact n < 1)
at_top"
    using eventually_gt_at_top[of 1] by eventually_elim auto
  then obtain n where "n > 1" "pi / 2 * (a * pi) ^ n / fact n < 1"
    using eventually_happens trivial_limit_sequentially by blast
  with bound[of n] show False
    by simp
qed
```

```
end


theorem pi_irrational: "pi ∉ ℚ"
proof
  assume "pi ∈ ℚ"
  then obtain a b :: int where ab: "b > 0" "pi = a / b"
    by (meson Rats_cases')
  interpret pi_rational a b
    by unfold_locales (use ab in auto)
  show False
    by (fact absurd)
qed

end
```

References

[1] I. Niven. A simple proof that π is irrational. *Bulletin of the American Mathematical Society*, 53(6):509, 1947.