# Two theorems about the geometry of the critical points of a complex polynomial

Manuel Eberl

November 21, 2023

**Abstract**

This entry formalises two well-known results about the geometric relation between the *roots* of a complex polynomial and its *critical points*, i.e. the roots of its derivative.

The first of these is the *Gauß–Lucas Theorem*: The critical points of a complex polynomial lie inside the convex hull of its roots.

The second one is *Jensen's Theorem*: Every non-real critical point of a real polynomial lies inside a disc between two conjugate roots. These discs are called the *Jensen discs*: the Jensen disc of a pair of conjugate roots $a \pm bi$ is the smallest disc that contains both of them, i.e. the disc with centre $a$ and radius $b$.

# Contents

# 1 Missing Library Material

```
theory Polynomial_Crit_Geometry_Library
imports
  "HOL-Computational_Algebra.Computational_Algebra"
  "HOL-Library.FuncSet"
  "Formal_Puiseux_Series.Formal_Puiseux_Series"
begin
```

## 1.1 Multisets

```
lemma size_repeat_mset [simp]: "size (repeat_mset n A) = n * size A"
  by (induction n) auto

lemma count_image_mset_inj:
  "inj f ⟹ count (image_mset f A) (f x) = count A x"
  by (induction A) (auto dest!: injD)

lemma count_le_size: "count A x ≤ size A"
  by (induction A) auto

lemma image_mset_cong_simp:
  "M = M' ⟹ (⋀x. x ∈# M =simp=> f x = g x) ⟹ {#f x. x ∈# M#} = {#g
x. x ∈# M'#}"
  unfolding simp_implies_def by (auto intro: image_mset_cong)

lemma sum_mset_nonneg:
  fixes A :: "'a :: ordered_comm_monoid_add multiset"
  assumes "⋀x. x ∈# A ⟹ x ≥ 0"
  shows   "sum_mset A ≥ 0"
  using assms by (induction A) auto

lemma sum_mset_pos:
  fixes A :: "'a :: ordered_comm_monoid_add multiset"
  assumes "A ≠ {#}"
  assumes "⋀x. x ∈# A ⟹ x > 0"
  shows   "sum_mset A > 0"
proof -
  from assms obtain x where "x ∈# A"
    by auto
  hence "A = {#x#} + (A - {#x#})"
    by auto
  also have "sum_mset ... = x + sum_mset (A - {#x#})"
    by simp
  also have "... > 0"
  proof (rule add_pos_nonneg)
    show "x > 0"
      using ‹x ∈# A› assms by auto
    show "sum_mset (A - {#x#}) ≥ 0"
      using assms sum_mset_nonneg by (metis in_diffD order_less_imp_le)
```

```
  qed
  finally show ?thesis .
qed
```

## 1.2 Polynomials

```
lemma order_pos_iff: "p ≠ 0 ⟹ order x p > 0 ⟷ poly p x = 0"
  by (cases "order x p = 0") (auto simp: order_root order_0I)

lemma order_prod_mset:
  "0 ∉# P ⟹ order x (prod_mset P) = sum_mset (image_mset (λp. order
x p) P)"
  by (induction P) (auto simp: order_mult)

lemma order_prod:
  "(⋀x. x ∈ I ⟹ f x ≠ 0) ⟹ order x (prod f I) = (∑i∈I. order x
(f i))"
  by (induction I rule: infinite_finite_induct) (auto simp: order_mult)

lemma order_linear_factor:
  assumes "a ≠ 0 ∨ b ≠ 0"
  shows "order x [:a, b:] = (if b * x + a = 0 then 1 else 0)"
proof (cases "b * x + a = 0")
  case True
  have "order x [:a, b:] ≤ degree [:a, b:]"
    using assms by (intro order_degree) auto
  also have "... ≤ 1"
    by simp
  finally have "order x [:a, b:] ≤ 1" .
  moreover have "order x [:a, b:] > 0"
    using assms True by (subst order_pos_iff) (auto simp: algebra_simps)
  ultimately have "order x [:a, b:] = 1"
    by linarith
  with True show ?thesis
    by simp
qed (auto intro!: order_0I simp: algebra_simps)

lemma order_linear_factor' [simp]:
  assumes "a ≠ 0 ∨ b ≠ 0" "b * x + a = 0"
  shows "order x [:a, b:] = 1"
  using assms by (subst order_linear_factor) auto

lemma degree_prod_mset_eq: "0 ∉# P ⟹ degree (prod_mset P) = (∑p∈#P.
degree p)"
  for P :: "'a::idom poly multiset"
  by (induction P) (auto simp: degree_mult_eq)

lemma degree_prod_list_eq: "0 ∉ set ps ⟹ degree (prod_list ps) = (∑p←ps.
degree p)"
```

```
  for ps :: "'a::idom poly list"
  by (induction ps) (auto simp: degree_mult_eq)

lemma order_conv_multiplicity:
  assumes "p ≠ 0"
  shows   "order x p = multiplicity [:-x, 1:] p"
  using assms order[of p x] multiplicity_eqI by metis
```

## 1.3 Polynomials over algebraically closed fields

```
lemma irreducible_alg_closed_imp_degree_1:
  assumes "irreducible (p :: 'a :: alg_closed_field poly)"
  shows   "degree p = 1"
proof -
  have "¬(degree p > 1)"
    using assms alg_closed_imp_reducible by blast
  moreover from assms have "degree p ≠ 0"
    by (intro notI) auto
  ultimately show ?thesis
    by linarith
qed

lemma prime_poly_alg_closedE:
  assumes "prime (q :: 'a :: {alg_closed_field, field_gcd} poly)"
  obtains c where "q = [:-c, 1:]" "poly q c = 0"
proof -
  from assms have "degree q = 1"
    by (intro irreducible_alg_closed_imp_degree_1 prime_elem_imp_irreducible)
auto
  then obtain a b where q: "q = [:a, b:]"
    by (metis One_nat_def degree_pCons_eq_if nat.distinct(1) nat.inject
pCons_cases)
  have "unit_factor q = 1"
    using assms by auto
  thus ?thesis
    using that[of "-a"] q ‹degree q = 1›
    by (auto simp: unit_factor_poly_def one_pCons split: if_splits)
qed

lemma prime_factors_alg_closed_poly_bij_betw:
  assumes "p ≠ (0 :: 'a :: {alg_closed_field, field_gcd} poly)"
  shows "bij_betw (λx. [:-x, 1:]) {x. poly p x = 0} (prime_factors p)"
proof (rule bij_betwI[of _ _ _ "λq. -poly q 0"], goal_cases)
  case 1
  have [simp]: "p div [:1:] = p" for p :: "'a poly"
    by (simp add: pCons_one)
  show ?case using assms
    by (auto simp: in_prime_factors_iff dvd_iff_poly_eq_0 prime_def
          prime_elem_linear_field_poly normalize_poly_def one_pCons)
```

```
qed (auto simp: in_prime_factors_iff elim!: prime_poly_alg_closedE dvdE)

lemma alg_closed_imp_factorization':
  assumes "p ≠ (0 :: 'a :: alg_closed_field poly)"
  shows "p = smult (lead_coeff p) (∏x | poly p x = 0. [:-x, 1:] ^ order
x p)"
proof -
  obtain A where A: "size A = degree p" "p = smult (lead_coeff p) (∏x∈#A.
[:- x, 1:])"
    using alg_closed_imp_factorization[OF assms] by blast
  have "set_mset A = {x. poly p x = 0}" using assms
    by (subst A(2)) (auto simp flip: poly_hom.prod_mset_image simp: image_image)

  note A(2)
  also have "(∏x∈#A. [:- x, 1:]) =
               (∏x∈(λx. [:- x, 1:]) ` set_mset A. x ^ count {#[:- x,
1:]. x ∈# A#} x)"
    by (subst prod_mset_multiplicity) simp_all
  also have "set_mset A = {x. poly p x = 0}" using assms
    by (subst A(2)) (auto simp flip: poly_hom.prod_mset_image simp: image_image)
  also have "(∏x∈(λx. [:- x, 1:]) ` {x. poly p x = 0}. x ^ count {#[:-
x, 1:]. x ∈# A#} x) =
             (∏x | poly p x = 0. [:- x, 1:] ^ count {#[:- x, 1:]. x ∈#
A#} [:- x, 1:])"
    by (subst prod.reindex) (auto intro: inj_onI)
  also have "(λx. count {#[:- x, 1:]. x ∈# A#} [:- x, 1:]) = count A"
    by (subst count_image_mset_inj) (auto intro!: inj_onI)
  also have "count A = (λx. order x p)"
  proof
    fix x :: 'a
    have "order x p = order x (∏x∈#A. [:- x, 1:])"
      using assms by (subst A(2)) (auto simp: order_smult order_prod_mset)
    also have "... = (∑y∈#A. order x [:-y, 1:])"
      by (subst order_prod_mset) (auto simp: multiset.map_comp o_def)
    also have "image_mset (λy. order x [:-y, 1:]) A = image_mset (λy.
if y = x then 1 else 0) A"
      using order_power_n_n[of y 1 for y :: 'a]
      by (intro image_mset_cong) (auto simp: order_0I)
    also have "... = replicate_mset (count A x) 1 + replicate_mset (size
A - count A x) 0"
      by (induction A) (auto simp: add_ac Suc_diff_le count_le_size)
    also have "sum_mset ... = count A x"
      by simp
    finally show "count A x = order x p" ..
  qed
  finally show ?thesis .
qed
```

## 1.4 Complex polynomials and conjugation

```
lemma complex_poly_real_coeffsE:
  assumes "set (coeffs p) ⊆ ℝ"
  obtains p' where "p = map_poly complex_of_real p'"
proof (rule that)
  have "coeff p n ∈ ℝ" for n
    using assms by (metis Reals_0 coeff_in_coeffs in_mono le_degree zero_poly.rep_eq)
  thus "p = map_poly complex_of_real (map_poly Re p)"
    by (subst map_poly_map_poly) (auto simp: poly_eq_iff o_def coeff_map_poly)
qed

lemma order_map_poly_cnj:
  assumes "p ≠ 0"
  shows   "order x (map_poly cnj p) = order (cnj x) p"
proof -
  have "order x (map_poly cnj p) ≤ order (cnj x) p" if p: "p ≠ 0" for
p :: "complex poly" and x
  proof (rule order_max)
    interpret map_poly_idom_hom cnj
      by standard auto
    interpret field_hom cnj
      by standard auto
    have "[:-x, 1:] ^ order x (map_poly cnj p) dvd map_poly cnj p"
      using order[of "map_poly cnj p" x] p by simp
    also have "[:-x, 1:] ^ order x (map_poly cnj p) =
              map_poly cnj ([:-cnj x, 1:] ^ order x (map_poly cnj p))"
      by (simp add: hom_power)
    finally show "[:-cnj x, 1:] ^ order x (map_poly cnj p) dvd p"
      by (rule dvd_map_poly_hom_imp_dvd)
  qed fact+
  from this[of p x] and this[of "map_poly cnj p" "cnj x"] and assms show
?thesis
    by (simp add: map_poly_map_poly o_def)
qed
```

## 1.5 *n*-ary product rule for the derivative

```
lemma has_field_derivative_prod_mset [derivative_intros]:
  assumes "⋀x. x ∈# A ⟹ (f x has_field_derivative f' x) (at z)"
  shows   "((λu. ∏x∈#A. f x u) has_field_derivative (∑x∈#A. f' x *
(∏y∈#A-{#x#}. f y z))) (at z)"
  using assms
proof (induction A)
  case (add x A)
  note [derivative_intros] = add
  note [cong] = image_mset_cong_simp
  show ?case
    by (auto simp: field_simps multiset.map_comp o_def  intro!: derivative_eq_intros)
qed auto
```

```
lemma has_field_derivative_prod [derivative_intros]:
  assumes "⋀x. x ∈ A ⟹ (f x has_field_derivative f' x) (at z)"
  shows   "((λu. ∏x∈A. f x u) has_field_derivative (∑x∈A. f' x * (∏y∈A-{x}.
f y z))) (at z)"
  using assms
proof (cases "finite A")
  case [simp, intro]: True
  have "((λu. ∏x∈A. f x u) has_field_derivative
          (∑x∈A. f' x * (∏y∈#mset_set A-{#x#}. f y z))) (at z)"
    using has_field_derivative_prod_mset[of "mset_set A" f f' z] assms
    by (simp add: prod_unfold_prod_mset sum_unfold_sum_mset)
  also have "(∑x∈A. f' x * (∏y∈#mset_set A-{#x#}. f y z)) =
             (∑x∈A. f' x * (∏y∈#mset_set (A-{x}). f y z))"
    by (intro sum.cong) (auto simp: mset_set_Diff)
  finally show ?thesis
    by (simp add: prod_unfold_prod_mset)
qed auto

lemma has_field_derivative_prod_mset':
  assumes "⋀x. x ∈# A ⟹ f x z ≠ 0"
  assumes "⋀x. x ∈# A ⟹ (f x has_field_derivative f' x) (at z)"
  defines "P ≡ (λA u. ∏x∈#A. f x u)"
  shows   "(P A has_field_derivative (P A z * (∑x∈#A. f' x / f x z)))
(at z)"
  using assms
  by (auto intro!: derivative_eq_intros cong: image_mset_cong_simp
           simp:   sum_distrib_right mult_ac prod_mset_diff image_mset_Diff
multiset.map_comp o_def)

lemma has_field_derivative_prod':
  assumes "⋀x. x ∈ A ⟹ f x z ≠ 0"
  assumes "⋀x. x ∈ A ⟹ (f x has_field_derivative f' x) (at z)"
  defines "P ≡ (λA u. ∏x∈A. f x u)"
  shows   "(P A has_field_derivative (P A z * (∑x∈A. f' x / f x z)))
(at z)"
proof (cases "finite A")
  case True
  show ?thesis using assms True
    by (auto intro!: derivative_eq_intros
             simp: prod_diff1 sum_distrib_left sum_distrib_right mult_ac)
qed (auto simp: P_def)
```

## 1.6 Facts about complex numbers

```
lemma Re_sum_mset: "Re (sum_mset X) = (∑x∈#X. Re x)"
  by (induction X) auto

lemma Im_sum_mset: "Im (sum_mset X) = (∑x∈#X. Im x)"
```

```
  by (induction X) auto

lemma Re_sum_mset': "Re (∑x∈#X. f x) = (∑x∈#X. Re (f x))"
  by (induction X) auto

lemma Im_sum_mset': "Im (∑x∈#X. f x) = (∑x∈#X. Im (f x))"
  by (induction X) auto

lemma inverse_complex_altdef: "inverse z = cnj z / norm z ^ 2"
  by (metis complex_div_cnj inverse_eq_divide mult_1)

end

theory Polynomial_Crit_Geometry
imports
  "HOL-Computational_Algebra.Computational_Algebra"
  "HOL-Analysis.Analysis"
  Polynomial_Crit_Geometry_Library
begin
```

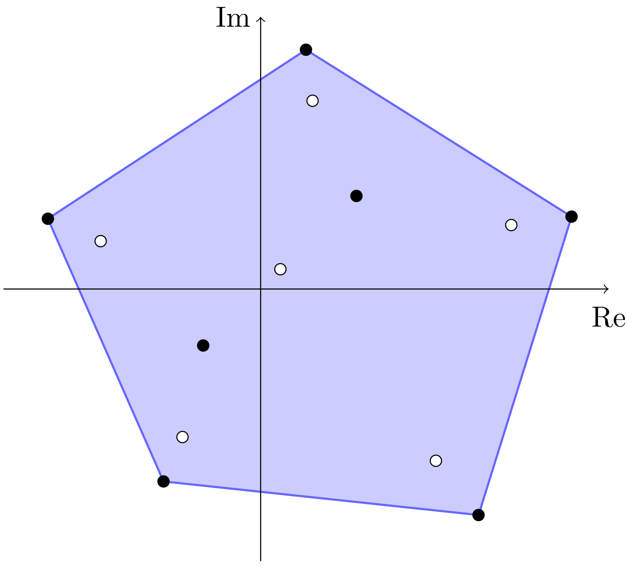


Figure 1: Example for the Gauß–Lucas Theorem: The roots (●) and critical points (○) of $x^7 - 2x^6 + x^5 + x^4 - (1+i)x^3 - 15ix^2 - 4(1-i)x - 7$.
The critical points all lie inside the convex hull of the roots (□).

## 2 The Gauß–Lucas Theorem

The following result is known as the *Gauß–Lucas Theorem*: The critical points of a non-constant complex polynomial lie inside the convex hull of its roots.

The proof is relatively straightforward by writing the polynomial in the form

$$p(x) = \prod_{i=1}^{n} (x - x_i)^{a_i} \ ,$$

from which we get the derivative

$$p'(x) = p(x) \cdot \sum_{i=1}^{n} \frac{a_i}{x - x_i} \ .$$

With some more calculations, one can then see that every root $x$ of $p'$ can be written as

$$x = \sum_{i=1}^{n} \frac{u_i}{U} \cdot x_i$$

where $u_i = \frac{a_i}{|x-x_i|^2}$ and $U = \sum_{i=1}^{n} u_i$.

**theorem** *pderiv_roots_in_convex_hull:*

```
  fixes p :: "complex poly"
  assumes "degree p ≠ 0"
  shows   "{z. poly (pderiv p) z = 0} ⊆ convex hull {z. poly p z = 0}"
proof safe
  fix z :: complex
  assume "poly (pderiv p) z = 0"
  show "z ∈ convex hull {z. poly p z = 0}"
  proof (cases "poly p z = 0")
    case True
    thus ?thesis by (simp add: hull_inc)
  next
    case False
    hence [simp]: "p ≠ 0" by auto
    define α where "α = lead_coeff p"
    have p_eq: "p = smult α (∏z | poly p z = 0. [:- z, 1:] ^ order z
p)"
      unfolding α_def by (rule alg_closed_imp_factorization') fact
    have poly_p: "poly p = (λw. α * (∏z | poly p z = 0. (w - z) ^ order
z p))"
      by (subst p_eq) (simp add: poly_prod fun_eq_iff)

    define S where "S = (∑w | poly p w = 0. of_nat (order w p) / (z
- w))"
    define u :: "complex ⇒ real" where "u = (λw. of_nat (order w p) /
norm (z - w) ^ 2)"
    define U where "U = (∑w | poly p w = 0. u w)"
    have u_pos: "u w > 0" if "poly p w = 0" for w
      using that False by (auto simp: u_def order_pos_iff intro!: divide_pos_pos)
    hence "U > 0" unfolding U_def
      using assms fundamental_theorem_of_algebra[of p] False
      by (intro sum_pos poly_roots_finite) (auto simp: constant_degree)

    note [derivative_intros del] = has_field_derivative_prod
    note [derivative_intros] = has_field_derivative_prod'
    have "(poly p has_field_derivative poly p z *
           (∑w | poly p w = 0. of_nat (order w p) *
              (z - w) ^ (order w p - 1) / (z - w) ^ order w p) ) (at
z)"
      (is "(_ has_field_derivative _ * ?S') _") using False
      by (subst (1 2) poly_p)
         (auto intro!: derivative_eq_intros simp: order_pos_iff mult_ac
power_diff S_def)
    also have "?S' = S" unfolding S_def
    proof (intro sum.cong refl, goal_cases)
      case (1 w)
      with False have "w ≠ z" and "order w p > 0"
        by (auto simp: order_pos_iff)
      thus ?case by (simp add: power_diff)
    qed
```

```
    finally have "(poly p has_field_derivative poly p z * S) (at z)" .
    hence "poly (pderiv p) z = poly p z * S"
      by (rule sym[OF DERIV_unique]) (auto intro: poly_DERIV)
    with ‹poly (pderiv p) z = 0› and ‹poly p z ≠ 0› have "S = 0" by
simp

    also have "S = (∑w | poly p w = 0. of_nat (order w p) * cnj z / norm
(z - w) ^ 2 -
                                        of_nat (order w p) * cnj w / norm
(z - w) ^ 2)"
      unfolding S_def by (intro sum.cong refl, subst complex_div_cnj)
                         (auto simp: diff_divide_distrib ring_distribs)
    also have "... = cnj z * (∑w | poly p w = 0. u w) - (∑w | poly p
w = 0. u w * cnj w)"
      by (simp add: sum_subtractf sum_distrib_left mult_ac u_def)
    finally have "cnj z * (∑w | poly p w = 0. of_real (u w)) =
                  (∑w | poly p w = 0. of_real (u w) * cnj w)" by simp
    from arg_cong[OF this, of cnj]
    have "z * of_real U = (∑w | poly p w = 0. of_real (u w) * w)"
      unfolding complex_cnj_mult by (simp add: U_def)
    hence "z = (∑w | poly p w = 0. of_real (u w) * w) / of_real U"
      using ‹U > 0› by (simp add: divide_simps)
    also have "... = (∑w | poly p w = 0. (u w / U) *R w)"
      by (subst sum_divide_distrib) (auto simp: scaleR_conv_of_real)
    finally have z_eq: "z = (∑w | poly p w = 0. (u w / U) *R w)" .

    show "z ∈ convex hull {z. poly p z = 0}"
    proof (subst z_eq, rule convex_sum)
      have "(∑i∈{w. poly p w = 0}. u i / U) = U / U"
        by (subst (2) U_def) (simp add: sum_divide_distrib)
      also have "... = 1" using ‹U > 0› by simp
      finally show "(∑i∈{w. poly p w = 0}. u i / U) = 1" .
    qed (insert ‹U > 0› u_pos,
          auto simp: hull_inc intro!: divide_nonneg_pos less_imp_le poly_roots_finite)
  qed
qed
```

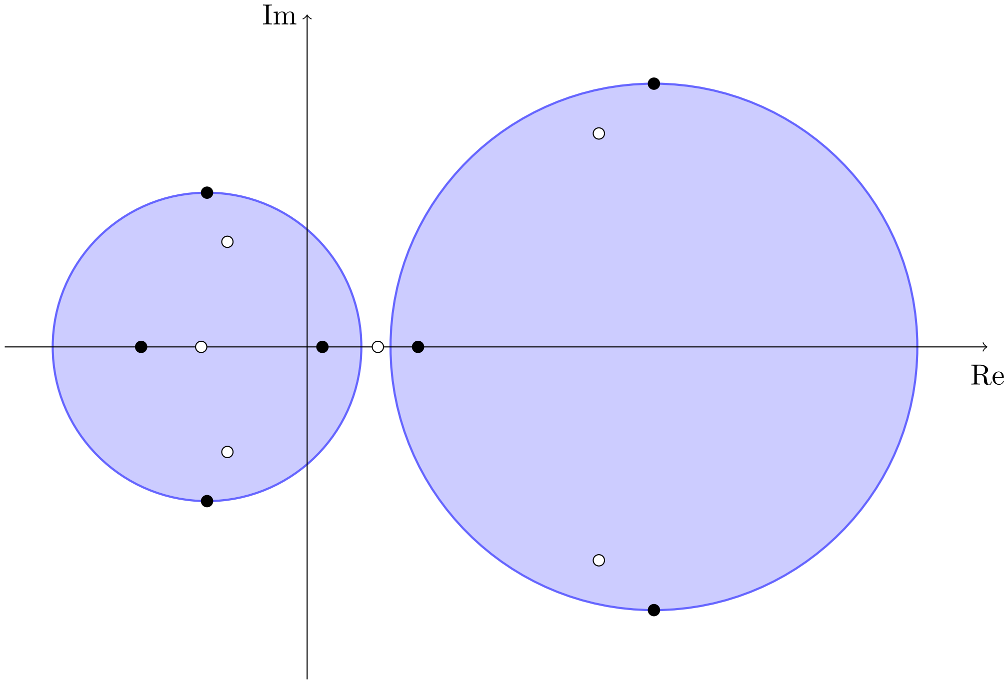


Figure 2: Example for Jensen's Theorem: The roots (•) and critical points (○) of the polynomial $x^7 - 3x^6 + 2x^5 + 8x^4 + 10x^3 - 10x + 1$.
It can be seen that all the non-real critical points lie inside a Jensen disc (○), whereas there can be real critical points that do *not* lie inside a Jensen disc.

## 3 Jensen's Theorem

For each root *w* of a real polynomial *p*, the Jensen disc of *w* is the smallest disc containing both $w$ and $\overline{w}$, i.e. the disc with centre $\mathrm{Re}(w)$ and radius $|\mathrm{Im}(w)|$.

We now show that if *p* is a real polynomial, every non-real critical point of *p* lies inside a Jensen disc of one of its non-real roots.

```
definition jensen_disc :: "complex ⇒ complex set" where
  "jensen_disc w = cball (of_real (Re w)) |Im w|"

theorem pderiv_root_in_jensen_disc:
  fixes p :: "complex poly"
  assumes "set (coeffs p) ⊆ ℝ" and "degree p ≠ 0"
  assumes "poly (pderiv p) z = 0" and "z ∉ ℝ"
  shows   "∃w. w ∉ ℝ ∧ poly p w = 0 ∧ z ∈ jensen_disc w"
proof (rule ccontr)
  have real_coeffs: "coeff p n ∈ ℝ" for n
```

```
    using assms(1) by (metis Reals_0 coeff_0 coeff_in_coeffs le_degree
subsetD)
  define d where "d = (λx. dist z (Re x) ^ 2 - Im x ^ 2)"

  assume *: "¬(∃w. w ∉ ℝ ∧ poly p w = 0 ∧ z ∈ jensen_disc w)"
  have d_pos: "d w > 0" if "poly p w = 0" "w ∉ ℝ" for w
  proof -
    have "dist z (Re w) > |Im w|"
      using * that unfolding d_def jensen_disc_def by (auto simp: dist_commute)
    hence "dist z (Re w) ^ 2 > |Im w| ^ 2"
      by (intro power_strict_mono) auto
    thus ?thesis
      by (simp add: d_def)
  qed

  have "poly p z ≠ 0"
    using d_pos[of z] assms by (auto simp: d_def dist_norm cmod_power2)
  hence [simp]: "p ≠ 0" by auto
  define α where "α = lead_coeff p"
  have [simp]: "α ≠ 0"
    using assms(4) by (auto simp: α_def)
  obtain A where p_eq: "p = smult α (∏x∈#A. [:-x, 1:])"
    unfolding α_def using alg_closed_imp_factorization[of p] by auto
  have poly_p: "poly p = (λw. α * (∏z∈#A. w - z))"
    by (subst p_eq) (simp add: poly_prod_mset fun_eq_iff)
  have [simp]: "poly p z = 0 ⟷ z ∈# A" for z
    by (auto simp: poly_p α_def)

  define Apos where "Apos = filter_mset (λw. Im w > 0) A"
  define Aneg where "Aneg = filter_mset (λw. Im w < 0) A"
  define A0 where "A0 = filter_mset (λw. Im w = 0) A"
  have "A = Apos + Aneg + A0"
    unfolding Apos_def Aneg_def A0_def by (induction A) auto

  have count_A: "count A w = order w p" for w
  proof -
    have "0 ∉# {#[:- x, 1:]. x ∈# A#}"
      by auto
    hence "order w p = (∑x∈#A. order w [:- x, 1:])"
      by (simp add: p_eq order_smult order_prod_mset multiset.map_comp
o_def)
    also have "... = (∑x∈#A. if w = x then 1 else 0)"
      by (simp add: order_linear_factor)
    also have "... = count A w"
      by (induction A) auto
    finally show ?thesis ..
  qed

  have "Aneg = image_mset cnj Apos"
```

```
  proof (rule multiset_eqI)
    fix x :: complex
    have "order (cnj x) (map_poly cnj p) = order x p"
      by (subst order_map_poly_cnj) auto
    also have "map_poly cnj p = p"
      using assms(1) by (metis Reals_cnj_iff map_poly_idI' subsetD)
    finally have [simp]: "order (cnj x) p = order x p" .

    have "count (image_mset cnj Apos) (cnj (cnj x)) = count Apos (cnj
x)"
      by (subst count_image_mset_inj) (auto simp: inj_on_def)
    also have "... = count Aneg x"
      by (simp add: Apos_def Aneg_def count_A)
    finally show "count Aneg x = count (image_mset cnj Apos) x"
      by simp
  qed

  have [simp]: "cnj x ∈# A ⟷ x ∈# A" for x
  proof -
    have "cnj x ∈# A ⟷ poly p (cnj x) = 0"
      by simp
    also have "poly p (cnj x) = cnj (poly (map_poly cnj p) x)"
      by simp
    also have "map_poly cnj p = p"
      using real_coeffs by (intro poly_eqI) (auto simp: coeff_map_poly
Reals_cnj_iff)
    finally show ?thesis
      by simp
  qed

  define N where "N = (λx. norm ((z - x) * (z - cnj x)))"
  have N_pos: "N x > 0" if "x ∈# A" for x
    using that ‹poly p z ≠ 0› by (auto simp: N_def)
  have N_nonneg: "N x ≥ 0" and [simp]: "N x ≠ 0" if "x ∈# A" for x
    using N_pos[OF that] by simp_all
```

We show that (∑x∈#A. 1 / (z - x)) = 0 (which is relatively obvious) and then that the imaginary part of this sum is positive, which is a contradiction.

```
  define S where "S = (∑x∈#A. 1 / (z - x))"
  note [derivative_intros del] = has_field_derivative_prod_mset
  note [derivative_intros] = has_field_derivative_prod_mset'
  have "(poly p has_field_derivative poly p z * S) (at z)"
    using ‹poly p z ≠ 0› unfolding S_def
    by (subst (1 2) poly_p)
       (auto intro!: derivative_eq_intros simp: order_pos_iff mult_ac
          power_diff multiset.map_comp o_def)
    hence "poly (pderiv p) z = poly p z * S"
    by (rule sym[OF DERIV_unique]) (auto intro: poly_DERIV)
  with ‹poly (pderiv p) z = 0› and ‹poly p z ≠ 0› have "S = 0" by simp
```

For determining *Im S*, we decompose the sum into real roots and pairs of conjugate and merge the sum of each pair of conjugate roots.

```
  have "Im S = (∑x∈#Apos. Im (1 / (z - x))) + (∑x∈#Aneg. Im (1 / (z
- x))) + (∑x∈#A0. Im (1 / (z - x)))"
    by (simp add: S_def ‹A = Apos + Aneg + A0› Im_sum_mset')
  also have "Aneg = image_mset cnj Apos"
    by fact
  also have "(∑x∈#.... Im (1 / (z - x))) = (∑x∈#Apos. Im (1 / (z -
cnj x)))"
    by (simp add: multiset.map_comp o_def)
  also have "(∑x∈#Apos. Im (1 / (z - x))) + (∑x∈#Apos. Im (1 / (z -
cnj x))) =
            (∑x∈#Apos. Im (1 / (z - x) + 1 / (z - cnj x)))"
    by (subst sum_mset.distrib [symmetric]) simp_all
  also have "image_mset (λx. Im (1 / (z - x) + 1 / (z - cnj x))) Apos
=
             image_mset (λx. - 2 * Im z * d x / N x ^ 2) Apos"
  proof (intro image_mset_cong, goal_cases)
    case (1 x)
    have "1 / (z - x) + 1 / (z - cnj x) = (2 * z - (x + cnj x)) * inverse
((z - x) * (z - cnj x))"
      using ‹poly p z ≠ 0› 1
      by (auto simp: divide_simps Apos_def complex_is_Real_iff simp flip:
Reals_cnj_iff)
    also have "x + cnj x = 2 * Re x"
      by (subst complex_add_cnj) auto
    also have "inverse ((z - x) * (z - cnj x)) = (cnj z - cnj x) * (cnj
z - x) / N x ^ 2"
      by (subst inverse_complex_altdef) (simp_all add: N_def)
    also have "Im ((2 * z - complex_of_real (2 * Re x)) * ((cnj z - cnj
x) * (cnj z - x) / N x ^ 2)) =
                 (-2 * Im z * (Im z ^ 2 - Im x ^ 2 + (Re x - Re z) ^ 2))
/ N x ^ 2"
      by (simp add: algebra_simps power2_eq_square)
    also have "Im z ^ 2 - Im x ^ 2 + (Re x - Re z) ^ 2 = d x"
      unfolding dist_norm cmod_power2 d_def by (simp add: power2_eq_square
algebra_simps)
    finally show ?case .
  qed
  also have "sum_mset ... = -Im z * (∑x∈#Apos.  2 * d x / N x ^ 2)"
    by (subst sum_mset_distrib_left) (simp_all add: multiset.map_comp
o_def mult_ac)
  also have "image_mset (λx. Im (1 / (z - x))) A0 = image_mset (λx. -Im
z / N x) A0"
  proof (intro image_mset_cong, goal_cases)
    case (1 x)
    have [simp]: "Im x = 0"
      using 1 by (auto simp: A0_def)
    have [simp]: "cnj x = x"
```

```
      by (auto simp: complex_eq_iff)
    show "Im (1 / (z - x)) = -Im z / N x"
      by (simp add: Im_divide N_def cmod_power2 norm_power flip: power2_eq_square)
  qed
  also have "sum_mset ... = -Im z * (∑x∈#A0. 1 / N x)"
    by (simp add: sum_mset_distrib_left multiset.map_comp o_def)
  also have "-Im z * (∑x∈#Apos.  2 * d x / N x ^ 2) + ... =
             -Im z * ((∑x∈#Apos.  2 * d x / N x ^ 2) + (∑x∈#A0. 1 /
N x))"
    by algebra
  also have "Im S = 0"
    using ‹S = 0› by simp
  finally have "((∑x∈#Apos. 2 * d x / N x ^ 2) + (∑x∈#A0. 1 / N x))
= 0"
    using ‹z ∉ ℝ› by (simp add: complex_is_Real_iff)

  moreover have "((∑x∈#Apos.  2 * d x / N x ^ 2) + (∑x∈#A0. 1 / N
x)) > 0"
  proof -
    have "A ≠ {#}"
      using ‹degree p ≠ 0› p_eq by fastforce
    hence "Apos ≠ {#} ∨ A0 ≠ {#}"
      using ‹Aneg = image_mset cnj Apos› ‹A = Apos + Aneg + A0› by auto
    thus ?thesis
    proof
      assume "Apos ≠ {#}"
      hence "(∑x∈#Apos.  2 * d x / N x ^ 2) > 0"
        by (intro sum_mset_pos)
           (auto intro!: mult_pos_pos divide_pos_pos d_pos simp: Apos_def
complex_is_Real_iff)
      thus ?thesis
        by (intro add_pos_nonneg sum_mset_nonneg) (auto intro!: N_nonneg
simp: A0_def)
    next
      assume "A0 ≠ {#}"
      hence "(∑x∈#A0. 1 / N x) > 0"
        by (intro sum_mset_pos) (auto intro!: divide_pos_pos N_pos simp:
A0_def)
      thus ?thesis
        by (intro add_nonneg_pos sum_mset_nonneg)
           (auto intro!: N_pos less_imp_le[OF d_pos] mult_nonneg_nonneg
divide_nonneg_pos
                 simp: Apos_def complex_is_Real_iff)
    qed
  qed

  ultimately show False
    by simp
qed
```

**end**

## References

[1] F. Enescu. Math 4444/6444 Polynomials 2, Lecture Notes, Lecture 9. https://math.gsu.edu/fenescu/fall2010/lec9-polyn-2010.pdf, 2010.

[2] M. Marden. *Geometry of Polynomials*. American Mathematical Society Mathematical Surveys. American Mathematical Society, 1966.