The Budan-Fourier Theorem and Counting Real Roots with Multiplicity

Wenda Li
April 19, 2020

Abstract

This entry is mainly about counting and approximating real roots (of a polynomial) with multiplicity. We have first formalised the Budan-Fourier theorem: given a polynomial with real coefficients, we can calculate sign variations on Fourier sequences to over-approximate the number of real roots (counting multiplicity) within an interval. When all roots are known to be real, the over-approximation becomes tight: we can utilise this theorem to count real roots exactly. It is also worth noting that Descartes’ rule of sign is a direct consequence of the Budan-Fourier theorem, and has been included in this entry. In addition, we have extended previous formalised Sturm’s theorem to count real roots with multiplicity, while the original Sturm’s theorem only counts distinct real roots. Compared to the Budan-Fourier theorem, our extended Sturm’s theorem always counts roots exactly but may suffer from greater computational cost.

Many problems in real algebraic geometry is about counting or approximating roots of a polynomial. Previous formalised results are mainly about counting distinct real roots (i.e. Sturm’s theorem in Isabelle/HOL [5, 2], HOL Light [4], PVS [9] and Coq [8]) and limited support for multiple real roots (i.e. Descartes’ rule of signs in Isabelle/HOL [3], HOL Light and Proof-Power1). In comparison, this entry provides more comprehensive support for reasoning about multiple real roots.

The main motivation of this entry is to cope with the roots-on-the-border issue when counting complex roots [7, 6], but the results here should be beneficial to other developments.

Our proof of the Budan-Fourier theorem mainly follows Theorem 2.35 in the book by Basu et al. [1] and that of the extended Sturm’s theorem is inspired by Theorem 10.5.6 in Rahman and Schmeisser’s book [10].

1 According to Freek Wiedijk’s “Formalising 100 Theorems” (http://www.cs.ru.nl/~freek/100/index.html)
1 Misc results for polynomials and sign variations

theory BF-Misc imports
  HOL-Computational-Algebra.Polynomial-Factorial
  HOL-Computational-Algebra.Fundamental-Theorem-Algebra
  Sturm-Tarski.Sturm-Tarski
begin

1.1 Induction on polynomial roots

lemma poly-root-induct-alt [case-names 0 no-proots root]:
  fixes p :: 'a :: idom poly
  assumes Q 0
  assumes \( \forall a. \text{poly} \ p \ a \neq 0 \) \implies Q p
  assumes \( \forall a \ p. \ Q \ p \implies Q (\llbracket -a, 1; \rrbracket * p) \)
  shows Q p
⟨proof⟩

1.2 Misc

lemma lead-coeff-pderiv:
  fixes p :: 'a::{comm-semiring-1,semiring-no-zero-divisors,semiring-char-0} poly
  shows lead-coeff (pderiv p) = of-nat (degree p) * lead-coeff p
⟨proof⟩

lemma gcd-degree-le-min:
  assumes p\neq0 q\neq0
  shows degree (gcd p q) \leq \min (degree p) (degree q)
⟨proof⟩

lemma lead-coeff-normalize-field:
  fixes p :: 'a::{field,semidom-divide-unit-factor} poly
  assumes p\neq0
  shows lead-coeff (normalize p) = 1
⟨proof⟩

lemma smult-normalize-field-eq:
  fixes p :: 'a::{field,semidom-divide-unit-factor} poly
  shows p = smult (lead-coeff p) (normalize p)
⟨proof⟩

lemma lead-coeff-gcd-field:
  fixes p q :: 'a::field-gcd poly
  assumes p\neq0 \lor q\neq0
  shows lead-coeff (gcd p q) = 1
⟨proof⟩

lemma poly-gcd-0-iff:
  poly (gcd p q) x = 0 \iff poly p x=0 \land poly q x=0
⟨proof⟩
lemma degree-eq-oneE:
  fixes p :: 'a::zero poly
  assumes degree p = 1
  obtains a b where p = [:a,b:] b≠0
⟨proof⟩

1.3 More results about sign variations (i.e. changes

lemma changes-0[simp]:changes (0#xs) = changes xs
⟨proof⟩

lemma changes-Cons:changes (x#xs) = (if filter (λx. x≠0) xs = [] then
  0
  else if x* hd (filter (λx. x≠0) xs) < 0 then
    1 + changes xs
  else changes xs)
⟨proof⟩

lemma changes-filter-eq:
changes (filter (λx. x≠0) xs) = changes xs
⟨proof⟩

lemma changes-filter-empty:
assumes filter (λx. x≠0) xs = []
shows changes xs = 0 changes (a#xs) = 0 ⟨proof⟩

lemma changes-append:
assumes xs≠ [] ∧ ys≠ [] −→ (last xs = hd ys ∧ last xs≠0)
shows changes (xs@ys) = changes xs + changes ys
⟨proof⟩

lemma changes-drop-dup:
assumes xs≠ [] ys≠ [] −→ last xs=hd ys
shows changes (xs@ys) = changes (xs@ tl ys)
⟨proof⟩

lemma Im-poly-of-real:
Im (poly p (of-real x)) = poly (map-poly Im p) x
⟨proof⟩

lemma Re-poly-of-real:
Re (poly p (of-real x)) = poly (map-poly Re p) x
⟨proof⟩
1.4 More about \emph{map-poly and of-real}

\begin{enumerate}
\item \textbf{lemma} \texttt{of-real-poly-map-pCons} [\texttt{simp}]: \texttt{map-poly of-real (pCons a p) = pCons (of-real a)} (\texttt{map-poly of-real p})
\end{enumerate}

\begin{enumerate}
\item \textbf{lemma} \texttt{of-real-poly-map-plus} [\texttt{simp}]: \texttt{map-poly of-real (p + q) = map-poly of-real p} + \texttt{map-poly of-real q}
\end{enumerate}

\begin{enumerate}
\item \textbf{lemma} \texttt{of-real-poly-map-smult} [\texttt{simp}]: \texttt{map-poly of-real (smult s p) = smult (of-real s) (map-poly of-real p)}
\end{enumerate}

\begin{enumerate}
\item \textbf{lemma} \texttt{of-real-poly-map-mult} [\texttt{simp}]: \texttt{map-poly of-real (p * q) = map-poly of-real p * map-poly of-real q}
\end{enumerate}

\begin{enumerate}
\item \textbf{lemma} \texttt{of-real-poly-map-poly}: \texttt{of-real (poly p x) = poly (map-poly of-real p) (of-real x)}
\end{enumerate}

\begin{enumerate}
\item \textbf{lemma} \texttt{of-real-poly-map-power}: \texttt{map-poly of-real (p ^ n) = (map-poly of-real p) ^ n}
\end{enumerate}

\begin{enumerate}
\item \textbf{lemma} \texttt{of-real-poly-eq-iff} [\texttt{simp}]: \texttt{map-poly of-real p = map-poly of-real q} \iff \texttt{p = q}
\end{enumerate}

\begin{enumerate}
\item \textbf{lemma} \texttt{of-real-poly-eq-0-iff} [\texttt{simp}]: \texttt{map-poly of-real p = 0} \iff \texttt{p = 0}
\end{enumerate}

1.5 More about \emph{order}

\begin{enumerate}
\item \textbf{lemma} \texttt{order-multiplicity-eq}: \texttt{assumes p \neq 0} \texttt{shows order a p = multiplicity [:-a,1:] p}
\end{enumerate}

\begin{enumerate}
\item \textbf{lemma} \texttt{order-gcd}: \texttt{assumes p \neq 0 q \neq 0} \texttt{shows order x (gcd p q) = min (order x p) (order x q)}
\end{enumerate}

\begin{enumerate}
\item \textbf{lemma} \texttt{order-linear} [\texttt{simp}]: \texttt{order x [:-a,1:] = (if x=a then 1 else 0)}
\end{enumerate}

\begin{enumerate}
\item \textbf{lemma} \texttt{map-poly-order-of-real}:
\end{enumerate}
assumes \( p \neq 0 \)
shows order (of-real t) (map-poly of-real p) = order t p ⟨proof⟩

lemma order-pcompose:
assumes pcompose p q \neq 0
shows order x (pcompose p q) = order x (q - [:poly q x:]) * order (poly q x) p ⟨proof⟩

1.6 Polynomial roots / zeros

definition proots-within::'a::comm-semiring-0 poly ⇒ 'a set ⇒ 'a set where
proots-within p s = \{ x ∈ s. poly p x = 0 \}

abbreviation proots::'a::comm-semiring-0 poly ⇒ 'a set where
proots p ≡ proots-within p UNIV

lemma proots-def: proots p = \{ x. poly p x = 0 \} ⟨proof⟩

lemma proots-within-empty[simp]:
proots-within p \{} = \{\} ⟨proof⟩

lemma proots-within-0[simp]:
proots-within 0 s = s ⟨proof⟩

lemma proots-withinI[intro,simp]:
poly p x = 0 \implies x ∈ s \implies x ∈ proots-within p s ⟨proof⟩

lemma proots-within-iff[simp]:
x ∈ proots-within p s \iff poly p x = 0 ∧ x ∈ s ⟨proof⟩

lemma proots-within-union:
proots-within p A ∪ proots-within p B = proots-within p (A ∪ B) ⟨proof⟩

lemma proots-within-times:
fixes s::'a::{semiring-no-zero-divisors,comm-semiring-0} set
shows proots-within (p*q) s = proots-within p s ∪ proots-within q s ⟨proof⟩

lemma proots-within-gcd:
fixes s::'a::{factorial-ring-gcd,semiring-gcd-mult-normalize} set
shows proots-within (gcd p q) s = proots-within p s ∩ proots-within q s ⟨proof⟩

lemma proots-within-inter:
NO-MATCH UNIV s \implies proots-within p s = proots p ∩ s
lemma proots-within-proots[simp]:
  proots-within p s ⊆ proots p
⟨proof⟩

lemma finite-proots[simp]:
  fixes p :: 'a::idom poly
  shows p ≠ 0 ⇒ finite (proots-within p s)
⟨proof⟩

lemma proots-within-pCons-1-iff:
  fixes a :: 'a::idom
  shows proots-within [−a, 1:] s = (if a ∈ s then {a} else { })
        proots-within [a, −1:] s = (if a ∈ s then {a} else { })
⟨proof⟩

lemma proots-within-uminus[simp]:
  fixes p :: 'a::comm-ring poly
  shows proots-within (− p) s = proots-within p s
⟨proof⟩

lemma proots-within-smult:
  fixes a :: 'a::{semiring-no-zero-divisors, comm-semiring-0}
  assumes a ≠ 0
  shows proots-within (smult a p) s = proots-within p s
⟨proof⟩

1.7 Polynomial roots counting multiplicities.

definition proots-count::'a::idom poly ⇒ 'a set ⇒ nat where
  proots-count p s = (∑ r ∈ proots-within p s. order r p)

lemma proots-count-emtpy[simp]: proots-count p {} = 0
⟨proof⟩

lemma proots-count-times:
  fixes s :: 'a::idom set
  assumes p*q ≠ 0
  shows proots-count (p*q) s = proots-count p s + proots-count q s
⟨proof⟩

lemma proots-count-power-n-n:
  shows proots-count ([:− a, 1:] "n) s = (if a ∈ s ∧ n > 0 then n else 0)
⟨proof⟩

lemma degree-proots-count:
  fixes p :: complex poly
  shows degree p = proots-count p UNIV
lemma proots-count-smult:
  fixes a::'a::{semiring-no-zero-divisors,idom}
  assumes a\neq0
  shows proots-count (smult a p) s = proots-count p s
⟨proof⟩

lemma proots-count-pCons-1-iff:
  fixes a::'a::idom
  shows proots-count [:-a,1:] s = (if a\in s then 1 else 0)
⟨proof⟩

lemma proots-count-uminus[simp]:
  proots-count (− p) s = proots-count p s
⟨proof⟩

lemma card-proots-within-leq:
  assumes p\neq0
  shows proots-count p s \geq card (proots-within p s)
⟨proof⟩

lemma proots-count-0-imp-empty:
  assumes proots-count p s=0 p\neq0
  shows proots-within p s = {}
⟨proof⟩

lemma proots-count-leq-degree:
  assumes p\neq0
  shows proots-count p s \leq degree p
⟨proof⟩

lemma proots-count-union-disjoint:
  assumes A \cap B = {} p\neq0
  shows proots-count p (A \cup B) = proots-count p A + proots-count p B
⟨proof⟩

lemma proots-count-cong:
  assumes order-eq:\forall x\in s. order x p = order x q and p\neq0 and q\neq0
  shows proots-count p s = proots-count q s
⟨proof⟩

lemma proots-count-of-real:
  assumes p\neq0
  shows proots-count (map-poly of-real p) ((of-real::'a::{real-algebra-1,idom}) s) = proots-count p s
⟨proof⟩
lemma proots-pcompose:
  fixes p q::'a::field poly
  assumes p≠0 degree q=1
  shows proots-count (pcompose p q) s = proots-count p (poly q ' s)
⟨proof⟩

1.8 Composition of a polynomial and a rational function

definition fcompose::'a ::field poly ⇒ 'a poly ⇒ 'a poly ⇒ 'a poly where
fcompose p q r = fst (fold-coeffs (λa (c,d). (d*[:a:] + q * c,r*d)) p (0,1))

lemma fcompose-0 [simp]: fcompose 0 q r = 0
⟨proof⟩

lemma fcompose-const[simp]:fcompose [:a:] q r = [:a:]
⟨proof⟩

lemma fcompose-pCons:
fcompose (pCons a p) q1 q2 = smult a (q2^2(degree (pCons a p))) + q1 * fcompose p q1 q2
⟨proof⟩

lemma fcompose-uminus:
fcompose (−p) q r = − fcompose p q r
⟨proof⟩

lemma fcompose-add-less:
  assumes degree p1 > degree p2
  shows fcompose (p1+p2) q1 q2
        = fcompose p1 q1 q2 + q2^2(degree p1 − degree p2) * fcompose p2 q1 q2
⟨proof⟩

lemma fcompose-add-eq:
  assumes degree p1 = degree p2
  shows q2^2(degree p1 − degree (p1+p2)) * fcompose (p1+p2) q1 q2
        = fcompose p1 q1 q2 + fcompose p2 q1 q2
⟨proof⟩

lemma fcompose-add-const:
fcompose [:a:] + p) q1 q2 = smult a (q2 ^ degree p) + fcompose p q1 q2
⟨proof⟩

lemma fcompose-smult: fcompose (smult a p) q1 q2 = smult a (fcompose p q1 q2)
⟨proof⟩

lemma fcompose-mult: fcompose (p1*p2) q1 q2 = fcompose p1 q1 q2 * fcompose p2 q1 q2
⟨proof⟩
lemma fcompose-poly:
  assumes poly q2 x ≠ 0
  shows poly p (poly q1 x / poly q2 x) = poly (fcompose p q1 q2) x / poly (q2 ^ (degree p)) x
⟨proof⟩

lemma poly-fcompose:
  assumes poly q2 x ≠ 0
  shows poly (fcompose p q1 q2) x = poly p (poly q1 x / poly q2 x) * (poly q2 x) ^ (degree p)
⟨proof⟩

lemma poly-fcompose-0-denominator:
  assumes poly q2 x = 0
  shows poly (fcompose p q1 q2) x = poly q1 x ^ degree p * lead-coeff p
⟨proof⟩

lemma fcompose-0-denominator:
fixes p :: 'a :: field poly
assumes p ≠ 0 and q2 ≠ 0 and nzero: ∀ c. q1 ≠ smult c q2
  and infi: infinite (UNIV :: 'a set)
shows fcompose p q1 q2 ≠ 0 ⟨proof⟩

1.9 Bijection (bij-betw) and the number of polynomial roots

lemma proots-fcompose-bij-eq:
fixes p :: 'a :: field poly
assumes bij: bij-betw (λx. poly q1 x / poly q2 x) A B and p ≠ 0
  and nzero: ∀ x ∈ A. poly q2 x ≠ 0
  and max-deg: max (degree q1) (degree q2) ≤ 1
  and nconst: ∀ c. q1 ≠ smult c q2
  and infi: infinite (UNIV :: 'a set)
shows proots-count p B = proots-count (fcompose p q1 q2) A
⟨proof⟩

lemma proots-card-fcompose-bij-eq:
fixes p :: 'a :: field poly
assumes bij: bij-betw (λx. poly q1 x / poly q2 x) A B and p ≠ 0
  and nzero: ∀ x ∈ A. poly q2 x ≠ 0
  and max-deg: max (degree q1) (degree q2) ≤ 1
  and nconst: ∀ c. q1 ≠ smult c q2
  and infi: infinite (UNIV :: 'a set)
shows card (proots-within p B) = card (proots-within (fcompose p q1 q2) A)
⟨proof⟩
lemma proots-pcompose-bij-eq:
  fixes p::'a::idom poly
  assumes bij:bij-betw (λx. poly q x) A B and p≠0
    and q-deg: degree q = 1
  shows proots-count p B = proots-count (p ◦ p) q A ⟨proof⟩

lemma proots-card-pcompose-bij-eq:
  fixes p::'a::idom poly
  assumes bij:bij-betw (λx. poly q x) A B and p≠0
    and q-deg: degree q = 1
  shows card (proots-within p B) = card (proots-within (p ◦ p) q A) ⟨proof⟩
end

2 Budan-Fourier theorem

theory Budan-Fourier imports
  BF-Misc
begin

The Budan-Fourier theorem is a classic result in real algebraic geometry to over-approximate real roots of a polynomial (counting multiplicity) within an interval. When all roots of the polynomial are known to be real, the over-approximation becomes tight – the number of roots are counted exactly. Also note that Descartes’ rule of sign is a direct consequence of the Budan-Fourier theorem.


2.1 More results related to sign-r-pos

lemma sign-r-pos-nzero-right:
  assumes nzero:∀ x. c<x ∧ x≤d → poly p x ≠0 and c<d
  shows if sign-r-pos p c then poly p d>0 else poly p d<0
⟨proof⟩

lemma sign-r-pos-at-left:
  assumes p≠0
  shows if even (order c p) ↔ sign-r-pos p c then eventually (λx. poly p x>0)
    (at-left c)
    else eventually (λx. poly p x<0) (at-left c)
⟨proof⟩

lemma sign-r-pos-nzero-left:
  assumes nzero:∀ x. d≤x ∧ x<c → poly p x ≠0 and d<c
  shows if even (order c p) ↔ sign-r-pos p c then poly p d>0 else poly p d<0
⟨proof⟩
2.2 Fourier sequences

function pders::real poly ⇒ real poly list where
    pders p = (if p = 0 then [] else Cons p (pders (pderiv p)))
⟨proof⟩
termination
⟨proof⟩
declare pders.simps[simp del]
lemma set-pders-nzero:
    assumes p≠0 q∈set (pders p)
    shows q≠0
⟨proof⟩

2.3 Sign variations for Fourier sequences

definition changes-itv-der:: real ⇒ real ⇒ real poly ⇒ int where
    changes-itv-der a b p = (let ps = pders p in changes-poly-at ps a − changes-poly-at ps b)
definition changes-gt-der:: real ⇒ real poly ⇒ int where
    changes-gt-der a p = changes-poly-at (pders p) a
definition changes-le-der:: real ⇒ real poly ⇒ int where
    changes-le-der b p = (degree p − changes-poly-at (pders p) b)
lemma changes-poly-pos-inf-pders[simp]: changes-poly-pos-inf (pders p) = 0
⟨proof⟩
lemma changes-poly-neg-inf-pders[simp]: changes-poly-neg-inf (pders p) = degree p
⟨proof⟩
lemma pders-coefs-sgn-eq:map (λp. sgn(p poly p 0)) (pders p) = map sgn (coeffs p)
⟨proof⟩
lemma changes-poly-at-pders-0:changes-poly-at (pders p) 0 = changes (coeffs p)
⟨proof⟩

2.4 Budan-Fourier theorem

lemma budan-fourier-aux-right:
    assumes c<d2 and p≠0
    assumes ∀x. c<x∧ x≤d2 → (∀q∈set (pders p). poly q x≠0)
    shows changes-itv-der c d2 p=0
⟨proof⟩
lemma budan-fourier-aux-left:
assumes $d_1 < c$ and $p \neq 0$
assumes $\forall x. \ d_1 \leq x \land x < c \longrightarrow (\forall q \in \text{set \ (pders \ p)}. \ \text{poly} \ q \ x \neq 0)$
shows changes-itv-der $d_1 \ c \ p \geq \text{order} \ c \ p \land \text{even} \ (\text{changes-itv-der} \ d_1 \ c \ p - \text{order} \ c \ p)$
⟨proof⟩

lemma budan-fourier-aux-left:
assumes $d_1 < c$ and $p \neq 0$
assumes nzero: $\forall x. \ d_1 < x \land x < c \longrightarrow (\forall q \in \text{set \ (pders \ p)}. \ \text{poly} \ q \ x \neq 0)$
shows changes-itv-der $d_1 \ c \ p \geq \text{order} \ c \ p \ \text{even} \ (\text{changes-itv-der} \ d_1 \ c \ p - \text{order} \ c \ p)$
⟨proof⟩

theorem budan-fourier-interval:
assumes $a < b$ \(p \neq 0\)
shows changes-itv-der $a \ b \ p \geq \text{proots-count} \ p \ \{x. \ a < x \land x \leq b\} \land$
even \ (changes-itv-der $a \ b \ p$ - \text{proots-count} \ p \ \{x. \ a < x \land x \leq b\})
⟨proof⟩

theorem budan-fourier-gt:
assumes $p \neq 0$
shows changes-gt-der $a \ p \geq \text{proots-count} \ p \ \{x. \ a < x\} \land$
even \ (changes-gt-der $a \ p$ - \text{proots-count} \ p \ \{x. \ a < x\})
⟨proof⟩

Descartes’ rule of signs is a direct consequence of the Budan-Fourier theorem

theorem descartes-sign:
fixes $p :: \text{real poly}$
assumes $p \neq 0$
shows changes $(\text{coeffs} \ p) \geq \text{proots-count} \ p \ \{x. \ 0 < x\} \land$
even \ (changes $(\text{coeffs} \ p)$ - \text{proots-count} \ p \ \{x. \ 0 < x\})
⟨proof⟩

theorem budan-fourier-le:
assumes $p \neq 0$
shows changes-le-der $b \ p \geq \text{proots-count} \ p \ \{x. \ x \leq b\} \land$
even \ (changes-le-der $b \ p$ - \text{proots-count} \ p \ \{x. \ x \leq b\})
⟨proof⟩

2.5 Count exactly when all roots are real

definition all-roots-real :: \text{real poly} \Rightarrow \text{bool} \ where
all-roots-real $p = (\forall r \in \text{roots} \ (\text{map-poly \ of-real} \ p)). \ \text{Im} \ r = 0$

lemma all-roots-real-mult [simp]:
all-roots-real $(p \cdot q) \longleftrightarrow \text{all-roots-real} \ p \land \text{all-roots-real} \ q$
⟨proof⟩

12
lemma all-roots-real-const-iff:
  assumes all-real:all-roots-real p
  shows degree p≠0 ↔ (∃ x. poly p x=0) (proof)

lemma all-roots-real-degree:
  assumes all-roots-real p
  shows proots-count p UNIV = degree p (proof)

lemma all-roots-roots-mobius:
  fixes a b::real
  assumes all-roots-real p and a<b
  shows all-roots-real (fcompose p [:a,b:] [:1,1:]) (proof)
  If all roots are real, we can use the Budan-Fourier theorem to EXACTLY count the number of real roots.

corollary budan-fourier-real:
  assumes p≠0
  assumes all-roots-real p
  shows proots-count p {x. x ≤a} = changes-le-der a p
  a<b ⇒ proots-count p {x. a<x ∧ x ≤b} = changes-itv-der a b p
  proots-count p {x. b<x} = changes-gt-der b p (proof)

  Similarly, Descartes’ rule of sign counts exactly when all roots are real.

corollary descartes-sign-real:
  fixes p::real poly and a b::real
  assumes p≠0
  assumes all-roots-real p
  shows proots-count p {x. 0 < x} = changes (coeffs p) (proof)

end

3 Extension of Sturm’s theorem for multiple roots

theory Sturm-Multiple-Roots
  imports
    BF-Misc
begin

  The classic Sturm’s theorem is used to count real roots WITHOUT multiplicity of a polynomial within an interval. Surprisingly, we can also extend Sturm’s theorem to count real roots WITH multiplicity by modifying the signed remainder sequence, which seems to be overlooked by many textbooks.

3.1 More results for smods

lemma last-smods-gcd:
fixes p q :: real poly
defines pp ≡ last (smods p q)
assumes p≠0
shows pp = smult (lead-coeff pp) (gcd p q)
⟨proof⟩

lemma last-smods-nzero:
assumes p≠0
shows last (smods p q) ≠ 0
⟨proof⟩

3.2 Alternative signed remainder sequences

function smods-ext:: real poly ⇒ real poly ⇒ real poly list where
smods-ext p q = (if p=0 then [] else
(if p mod q ≠ 0
then Cons p (smods-ext q (−(p mod q)))
else Cons p (smods-ext q (pderiv q)))
)
⟨proof⟩
termination
⟨proof⟩

lemma smods-ext-prefix:
fixes p q :: real poly
defines pp ≡ last (smods p q)
assumes p≠0 q≠0
shows smods-ext p q = smods p q @ tl (smods-ext pp (pderiv pp))
⟨proof⟩

lemma no-0-in-smods-ext: 0∉set (smods-ext p q)
⟨proof⟩

3.3 Sign variations on the alternative signed remainder sequences

definition changes-itv-smods-ext:: real ⇒ real ⇒ real poly ⇒ real poly ⇒ int where
changes-itv-smods-ext a b p q= (let ps= smods-ext p q in changes-poly-at ps a
− changes-poly-at ps b)
definition changes-gt-smods-ext:: real ⇒ real poly ⇒ real poly ⇒ int where
changes-gt-smods-ext a p q= (let ps= smods-ext p q in changes-poly-pos-inf ps)
definition changes-le-smods-ext:: real ⇒ real poly ⇒ real poly ⇒ int where
changes-le-smods-ext b p q= (let ps= smods-ext p q in changes-poly-neg-inf ps)
-- changes-poly-at ps b)

**definition** changes-R-smods-ext:: \text{real poly} \Rightarrow \text{real poly} \Rightarrow \text{int} where
changes-R-smods-ext p q = (let ps = smods-ext p q in changes-poly-neg-inf ps
-- changes-poly-pos-inf ps)

3.4 Extension of Sturm’s theorem for multiple roots

**theorem** sturm-ext-interval:
assumes \(a < b\) poly p \(a \neq 0\) poly p \(b \neq 0\)
shows proots-count p \(\{x. a < x \land x < b\}\) = changes-itv-smods-ext a b p (pderiv p)
⟨proof⟩

**theorem** sturm-ext-above:
assumes poly p \(a \neq 0\)
shows proots-count p \(\{x. a < x\}\) = changes-gt-smods-ext a p (pderiv p)
⟨proof⟩

**theorem** sturm-ext-below:
assumes poly p \(b \neq 0\)
shows proots-count p \(\{x. x < b\}\) = changes-le-smods-ext b p (pderiv p)
⟨proof⟩

**theorem** sturm-ext-R:
assumes \(p \neq 0\)
shows proots-count p UNIV = changes-R-smods-ext p (pderiv p)
⟨proof⟩

end

4 Descartes Roots Test

**theory** Descartes-Roots-Test imports Budan-Fourier
begin

The Descartes roots test is a consequence of Descartes’ rule of signs: through counting sign variations on coefficients of a base-transformed (i.e. Taylor shifted) polynomial, it can over-approximate the number of real roots (counting multiplicity) within an interval. Its ability is similar to the Budan-Fourier theorem, but is far more efficient in practice. Therefore, this test is widely used in modern root isolation procedures.

lemma bij-betw-pos-interval:
  fixes a b :: real
  assumes a < b
  shows bij-betw (λx. (a + b * x) / (1 + x)) {x. x > 0} {x. a < x ∧ x < b}
⟨proof⟩

lemma proots-sphere-pos-interval:
  fixes a b :: real
  defines q1 ≡ [a..b] and q2 ≡ [1..1]
  assumes p ≠ 0 a < b
  shows proots-count p {x. a < x ∧ x < b} = proots-count (fcompose p q1 q2) {x. 0 < x}
⟨proof⟩

definition descartes-roots-test::real ⇒ real ⇒ real poly ⇒ nat where
descartes-roots-test a b p = nat (changes (coeffs (fcompose p [a..b] [1..1])))

theorem descartes-roots-test:
  fixes p :: real poly
  assumes p ≠ 0 a < b
  shows proots-count p {x. a < x ∧ x < b} ≤ descartes-roots-test a b p ∧
  even (descartes-roots-test a b p − proots-count p {x. a < x ∧ x < b})
⟨proof⟩

The roots test descartes-roots-test is exact if its result is 0 or 1.

corollary descartes-roots-test-zero:
  fixes p :: real poly
  assumes p ≠ 0 a < b descartes-roots-test a b p = 0
  shows ∀ x. a < x ∧ x < b → poly p x ≠ 0
⟨proof⟩

corollary descartes-roots-test-one:
  fixes p :: real poly
  assumes p ≠ 0 a < b descartes-roots-test a b p = 1
  shows proots-count p {x. a < x ∧ x < b} = 1
⟨proof⟩

Similar to the Budan-Fourier theorem, the Descartes roots test result is
exact when all roots are real.

corollary descartes-roots-test-real:
  fixes p :: real poly
  assumes p ≠ 0 a < b
  assumes all-roots-real p
  shows proots-count p {x. a < x ∧ x < b} = descartes-roots-test a b p
⟨proof⟩

end
5 Acknowledgements

The work was 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.

References


