Math. Struct. in Comp. Science (2011), *vol.* 21, *pp.* 731–761. © Cambridge University Press 2011
doi:10.1017/S0960129511000090

A formal study of Bernstein coefficients and polynomials[†]

YVES BERTOT[‡], FRÉDÉRIQUE GUILHOT[‡] and
ASSIA MAHBOUBI[§]

[‡]*INRIA Sophia Antipolis - Méditerranée, 2004, route des Lucioles - BP 93, 06902 Sophia Antipolis Cedex, France*
Email: {Yves.Bertot;Frederique}.Guilhot@inria.fr
[§]*Inria Saclay – Île-de-France, Laboratoire d'Informatique de l'École Polytechnique (LIX), Parc Orsay Université, 4, rue Jacques Monod, 91893 Orsay cedex, France and École polytechnique Laboratoire d'informatique (LIX), 91128 Palaiseau Cedex, France*
Email: Assia.Mahboubi@inria.fr

Received 11 July 2010; revised 15 December 2010

Bernstein coefficients provide a discrete approximation of the behaviour of a polynomial inside an interval. This can be used, for example, to isolate the real roots of polynomials. We prove formally a criterion for the existence of a single root in an interval and the correctness of the de Casteljau algorithm for computing Bernstein coefficients efficiently.

1. Introduction

Bernstein coefficients provide a discrete approximation of polynomials inside a bounded interval. As such, they are useful tools for solving problems like locating the roots of polynomials, isolating these roots or solving systems of inequations with polynomial members. In computer aided design, they are also used intensively as they give an efficient tool for drawing curves that are controlled by points that users can grab and drag, with instantaneous and intuitive feedback on the curve's shape. Bernstein coefficients are closely related to Bézier curves and have a very simple geometric interpretation, which we illustrate in Section 4.

Bernstein coefficients are defined for a given polynomial, a given degree and a given interval. If the degree is n, the coefficients form a sequence of size $n+1$. In this paper, we are interested in three important properties of these coefficients:

(1) If the coefficients taken in order exhibit exactly one sign change, the polynomial is guaranteed to have exactly one root inside the interval.
(2) If all coefficients have the same sign, the polynomial is guaranteed to have no root inside the interval.
(3) There is an easy method for computing new Bernstein coefficients when splitting the interval in two.

We describe a formal proof of these three properties, but concentrate on (1) and (3).

[†] This work has been partially funded by the FORMATH project, nr. 243847, of the FET program within the 7th Framework program of the European Commission, and by the Galapagos project of the French ANR.

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press

The main plan of the proof of the first property is to describe a sequence of pairs (I_0, p_0) to (I_3, p_3), each pair containing an interval and a polynomial, such that every root of polynomial p_i inside I_i is in bijective correspondence with a root of polynomial p_{i+1} in the interval I_{i+1}. If we study the roots of the polynomial p on the interval (l, r), then I_0 and p_0 are (l, r) and p, respectively. The last interval, I_3, is $(0, +\infty)$ and the last polynomial, p_3, is $c_0 + c_1X + \cdots + c_nX^n$, where the coefficients c_i have the same sign as the Bernstein coefficients. Going from p_i to p_{i+1}, we apply a given transformation. The first transformation is a change of variable such that I_1 is $(0, 1)$ and $p_1(x) = p_0(rx + l(1-x))$. The second transformation is such that I_2 is $(1, +\infty)$ and $p_2(x) = 0$ exactly when $p_1(1/x) = 0$, provided $x \neq 0$. The third transformation is a translation such that $I_3 = (0, +\infty)$ and $p_3(x) = p_2(1 + x)$. We will show that the condition on Bernstein coefficients simply boils down to Descartes' law of sign (Descartes 1969; Basu *et al.* 2006) for polynomial p_3 in the case where there is exactly one sign change in this polynomial's coefficients. This path from one polynomial to another is described in Section 5.

Descartes' law of signs provides a sufficient criterion for the existence of exactly one root for a polynomial between 0 and $+\infty$. In its most general form, this law expresses a relation between the number of roots of a polynomial between 0 and $+\infty$ and the number of sign changes in the coefficients of this polynomial. The number of sign changes is larger than the number of roots, and the difference between the two numbers is even. Thus, if the number of sign changes is 1, there is exactly one root between 0 and $+\infty$.

For our development, we only prove the corollary of Descartes' law of signs for the case where there is only one sign change. This proof is carried out in Section 3. Expressing Descartes' law on the coefficients of polynomial p_3 directly yields a law expressed in terms of sign changes for Bernstein's coefficients of p with respect to the interval (l, r).

Another part of our work is to describe dichotomy. Knowing Bernstein coefficients for a polynomial and a given interval, it is easy to obtain the Bernstein coefficients for the two half intervals, using an algorithm due to de Casteljau (de Castleljau 1985). In the process, we increase the precision of the approximation given by the Bernstein coefficients. De Casteljau's algorithm is a simple combinatorial algorithm based on taking arithmetic means of successive coefficients. To justify this combinatorial process, we show in Section 4 that Bernstein coefficients actually are the coefficients of the polynomial in a different basis from the usual monomials, which is called the *Bernstein basis*.

In the following, we will assume that we are working with polynomials whose roots are all simple – such polynomials are called *separable polynomials*. Starting from an arbitrary polynomial, it is easy to produce a separable polynomial with the same roots by computing the greatest common divisor of this polynomial and its derivative.

All these results deal with polynomial functions over real numbers. In order to make this formal study as generic as possible, we abstract from the choice of the subset of real numbers of interest for the user and only rely on an abstract structure of an archimedean field. However, we require this field to be equipped with a decidable comparison: the process we describe can, in turn, be used effectively in a decision procedure.

If the field we work in is only guaranteed to be ordered and archimedean, the existence of roots takes a different meaning: if a polynomial has a single simple real root in an interval, this root may not belong to the field of its coefficients. However, we can use a corresponding property, which can be expressed in the field of coefficients: there exists a

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press

sub-interval inside which the absolute value of the slope is bounded below, and such that the values of the polynomial at the sub-interval bounds have opposite signs. In a similar vein, the intermediate value theorem does not hold within the language we have chosen, but a corresponding statement, expressed as a bounded-value property, does. Our proof development relies on this approach. We describe the formal aspects of this approach to describing roots in Section 2.

The formal work described in this paper has been performed using the Coq[†] system (Bertot and Castéran 2004), with the SSReflect extension (Gonthier and Mahboubi 2008). We think some characteristics of the proof system played a key role in making this development possible. We describe these key aspects in Section 7. The sources for this development are available at `http://hal.inria.fr/inria-00503017/`

2. Formalisation viewpoints

2.1. *A constructive and abstract approach*

Our aim is to provide a constructive and effective formalisation of the theory of Bernstein polynomials, and to be as independent as possible of the implementation choices of rational and real numbers. To this end, we eliminate real points from the formalisation. We consider polynomials with coefficients in an abstract ordered and archimedean field. The axioms of this structure are the axioms of a field with decidable equality and a decidable order relation that is compatible with the field operations. The decidability of the equality and order relations are crucial for the algorithms we describe to be truely effective. Such a field necessarily contains a copy of rational numbers, which are the carrier of the computations performed in our proofs. We do not provide nor rely on a general theory of continuous functions, but rather prove that such polynomials are necessarily continuous (and even uniformly continuous on bounded intervals) using ϵ-δ statements quantified on elements of the coefficient field.

Since such an archimedean field has no reason to be algebraically closed, the roots of the polynomials are not necessarily elements of this type. We hence need to adjust the definition of a *root of a polynomial*. In particular, we provide a weak version of the intermediate value theorem for polynomials, which results in a sufficient criterion for the existence of a single real root in an interval, expressed in the language of decidable, ordered archimedean fields. This field of coefficients cannot be directly instantiated by constructive reals since their comparison is not decidable, but this was not our purpose since our weak intermediate value theorem is only there to support the further study of Bernstein polynomials, which have rational coefficients. If coefficients are indeed instantiated by any implementation of rational numbers, for instance the one embedded in some implementation of constructive numbers, then the conclusion of the intermediate value theorem is sufficient to implement the modulus of convergence of a Cauchy sequence, which will be the constructive root of the polynomial. Of course, this work is readily usable with the 'classical real numbers' of the standard Coq library, but it also complies with the constructive libraries also implemented in the Coq system like (Cruz-Filipe *et al.* 2004; O'Connor 2007; O'Connor 2008).

[†] The Coq System can be downloaded from `http://coq.inria.fr/`.

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press

2.2. *Small scale reflection libraries*

This work is based on the SSReflect libraries[†] developed with the SSReflect extension (Gonthier and Mahboubi 2008) of the Coq system. These libraries cover basic theories (sequences, natural numbers, finite types, finite sets), infrastructure for notations and theory sharing (containers, iterated operators, algebraic hierarchies) and elementary algebra and arithmetic. They were developed along a path leading to a complete formal proof of a historic result in finite group theory (namely the Odd Order Theorem), to demonstrate that a well-understood art of formalisation leads to modular, reusable formal libraries. This work addresses theories that are not on this path, and thus tests the reusability of the distributed libraries and demonstrates that they can indeed be reused and extended.

One of the main characteristics of the methodology deployed in these libraries is small scale reflection (Gonthier and Mahboubi 2010), a proof methodology based on the pervasive use of computations with symbolic symbols. As a consequence, predicates are formalised as often as possible as boolean functions and a set of specialised tactics allows the user to reason on how these functions compute. For instance, it provides powerful automation for propositional reasoning: in this framework, reasoning in the intutionistically classical fragment of the Coq logic is almost as convenient as it would be in a fully classical system. Boolean representations play a role that goes beyond automation. They also affect the manipulation of dependent types, especially dependent pairs (also called Σ-types). The SSReflect library for polynomials illustrates this fact, as described in Section 2.3.

2.3. *Representation of polynomials*

In the SSReflect libraries, polynomials are represented by a big endian list of coefficients in the monomial basis. A polynomial function can easily be defined from such a list through the Horner evaluation scheme. However, a bare list of coefficients would not provide a canonical representation since the same polynomial can be represented by an infinite number of coefficient lists, only differing by the number of tail zeroes. The actual definition is as follows:

```
Record polynomial R :=
  Polynomial {polyseq :> seq R; _ : last 1 polyseq != 0}.
```

A polynomial is thus represented by a dependent pair consisting of a list `polyseq` of type `(seq R)` and a proof that the last element of this list is non-zero (this proof is not given a name, hence the use of the character `_`). The type `(polynomial R)` is used with the notation `{poly R}`. The parameter `R` is the type of the coefficients. It should in fact be equipped with a ring structure, which provides the zero element used in the test of the proof component.

The constant zero polynomial is represented by the empty list. A constant non-zero polynomial is represented by a list with a single element. The monomial X is represented

[†] The Ssreflect libraries can be downloaded from `http://www.msr-inria.inria.fr/Projects/math-components/index_html`.

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press

by the list [0, 1], where 1 and 0 are the one and zero constants of the underlying coefficient ring. More generally, a non-zero polynomial of degree $n+1$ is represented by a list with $n+1$ elements. The `:>` symbol indicates that the `polyseq` constructor is declared as a coercion: a polynomial can at any time be seen as a list of coefficients, forgetting about the proof that it is in normal form.

In this representation, comparing two polynomials seems to amount to comparing component-wise two dependent pairs: the first components of the two pairs being the actual polynomials, and the second components of the pairs being the proofs that they are in normal form. The calculus of inductive constructions underlying the Coq system is not proof irrelevant: two proofs of the same statements are not equal in general. Hence a naive design of these polynomials as dependent pairs would lead to an uncomfortable situation where two equal lists of coefficients would not necessarily give two equal polynomials because they would be associated with two different proofs of normalisation. Yet if proof irrelevance does not hold in general, two proofs of the same boolean equality are nonetheless provably equal in the calculus of inductive constructions without axioms. This is known as *boolean proof irrelevance*.

If the type `R` of coefficients is equipped with a boolean equality test, the second component of polynomials becomes a proof that a certain boolean is equal to the boolean `true`, and hence a proof of a boolean equality. Comparing two polynomials now boils down to comparing only their first component, and the equality of the proof components is given by boolean proof irrelevance. This kind of proof irrelevant dependent pairs is widely used in SSReflect libraries, which therefore provide a generic infrastructure to manipulate comparisons of their inhabitants. Moreover, again thanks to the infrastructure of SSReflect libraries, polynomials canonically inherit from the boolean comparison available on coefficients, lifted to the element-wise comparison of lists.

2.4. *Criteria for the existence of a unique root*

We concentrate on a sufficient criterion for the existence of a root inside an interval. Our criterion is based on slopes.

Ensuring that the slope is positive or negative in some interval helps make sure that there is no more than one root in this interval. In our setting, where the polynomials we consider only have simple roots, we have the stronger property that the slope is separated from 0 by a given ratio. In the case of positive slopes, we write the slope requirement for a polynomial p inside a given interval I as follows.

Definition 2.1 (Slope requirement – positive case). A polynomial p satisfies the positive slope requirement if

$$\exists k, (0 < k \quad \wedge \quad \forall x\ y, [x \in I \wedge y \in I \wedge x < y \Rightarrow k(y-x) < p(y) - p(x)]).$$

We also define the analogous requirement for negative slopes.

Depending on the kind of interval we consider, we will use two different sufficient conditions to express the existence of a single root in the interval:

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press

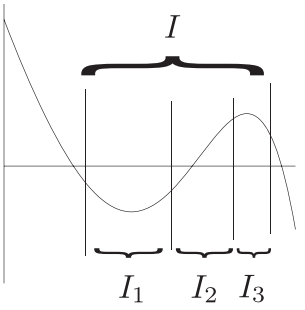


Fig. 1. A sufficient criterion for the existence of a single root in a bounded interval

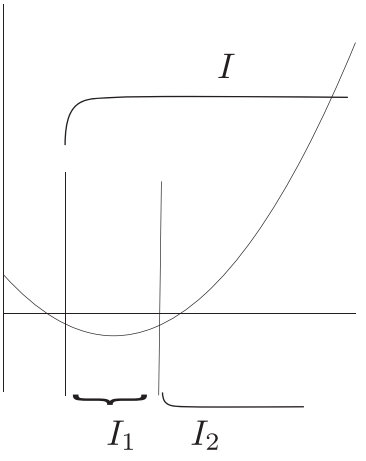


Fig. 2. A sufficient criterion for the existence of a single root in an unbounded interval

(1) If the interval is bounded, we express the fact that the interval can be decomposed into three parts, the first part I_1 where the polynomial's value is always negative (respectively, positive), the second part I_2 where the polynomial's value goes from negative to positive (respectively, from positive to negative) with a requirement on the slope, and the third part I_3 where the polynomial's value is always positive (respectively, negative). This is illustrated in Figure 1).
(2) If the interval is unbounded, we express the fact that the interval can be decomposed into two parts, the first part I_1 where the polynomial's value is always negative (respectively, positive), and the second part I_2 where there is a requirement on the slope with a positive (respectively, negative) slope. This is illustrated in Figure 2.

2.5. *Finding locations where a polynomial's value is arbitrarily small*

In classical mathematics dealing with real closed fields, once we know that the polynomial takes values of opposite sign at the bounds of an interval, we know that there is a root for this polynomial in this interval, thanks to the *intermediate value theorem.* For this work, we establish a simplified constructive, real point free, replacement for the intermediate value

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press

theorem specialised for polynomials. In our setting with a field that is only guaranteed to be ordered and archimedean, we cannot be sure to produce a value on which the polynomial of interest evaluates to zero, but we are able to produce an input for which the polynomial's absolute value is arbitrarily close to zero. The statement we prove has the following form.

Theorem 2.1 (Weak intermediate value theorem for polynomials).

$$\begin{aligned}\forall p\ x\ y\ \varepsilon,\quad & x < y \quad \wedge \quad 0 < \varepsilon \quad \wedge \quad p(x) < 0 \leqslant p(y) \\ & \Rightarrow [\exists x'\ y', -\varepsilon \leqslant p(x') < 0 \leqslant p(y') \leqslant \varepsilon \quad \wedge \quad x \leqslant x' < y' \leqslant y]\end{aligned}$$

Proof. We again rely on reasoning about slopes. Without loss of generality, we can assume that the two values x and y are positive. Assuming that the polynomial p has the shape $a + X \times p_1$, we construct another polynomial p_2 whose coefficients are the absolute values of the coefficients of p_1. This polynomial p_2 is increasing, so its maximum value in $[x, y]$ is reached at y. We prove that the slope of the polynomial between any two points inside $[x, y]$ is smaller than $k = p_2(y)$. Thus, we establish that the slope of any polynomial is bounded in absolute value on any bounded interval. In particular, for any $z, t \in [x, y]$, we have

$$|p(z) - p(t)| \leqslant |k \times (z - t)|.$$

For a given ε, because we work in an archimedean field, we can choose an n such that

$$\frac{k(y - x)}{n} < \varepsilon.$$

We then consider the $n + 1$ values

$$a_i = x + \frac{i \times (y - x)}{n}$$

for $i = 0 \ldots n$ and solve a discrete problem over the values a_i. We simply need to find the largest prefix $a_0, \ldots, a_{j-1}$ such that all values $p(a_k)$ in this prefix are negative. We can set $x' = a_{j-1}$, because the next value $p(a_j)$ is necessarily non-negative and $p(a_j) - p(x') < \varepsilon$, thus $-\varepsilon < p(x') < 0$. In a similar way, we can set $y' = a_j$ because $0 \leqslant p(y') < \varepsilon$. □

Our algorithm is illustrated in Figure 3. The point selected by our algorithm is a_8, even though there are more roots in the vicinity of a_1 and a_2 but neither a_1 nor a_2 is a point where the polynomial takes a positive value.

This theorem gives a similar result to Troelstra and van Dalen (1988, Theorem 6.1.4), but the proof in that book relies on a more complete description of abstract topology than we performed for our development, where opens and their properties with respect to continuous functions are not mentioned. Troelstra and van Dalen (1988, Theorem 6.1.5) enables the construction of a Cauchy sequence to a root of the continuous function to be considered. A variant of our theorem would also make this possible, but it was not needed for our purposes.

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press

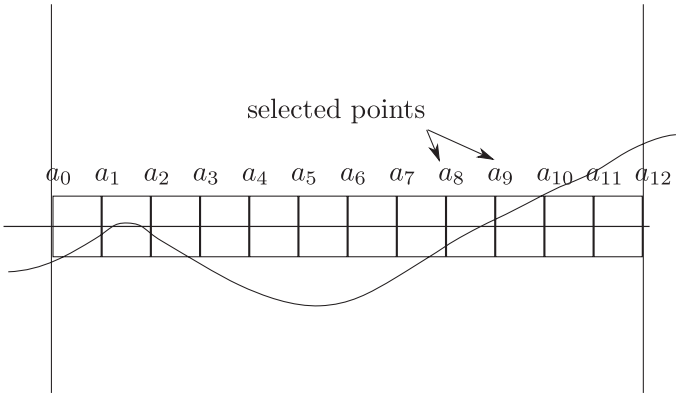


Fig. 3. Bounding a polynomial's value

3. A simple form of Descartes' law of signs

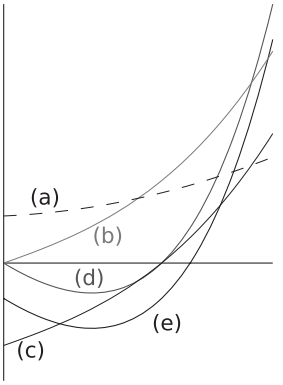
One of the main results studied in this paper is that having only one sign change in the sequence of Bernstein coefficients for the square-free polynomial p and the interval (l, r) ensures that there is only one root of p inside (l, r). The proof of this result relies on a similar property for the standard coefficients of another polynomial q: if there is only one sign change in the coefficients of q, then q has only one root inside the interval $(0, +\infty)$. In this section, we discuss how this property can be proved formally.

3.1. *A geometrical explanation of the proof*

We begin by describing a simple graphical argument based on curves for polynomial functions between 0 and $+\infty$, as shown in Figure 4. To describe our proof, we rely on an inductive view of polynomials where new polynomials are built from existing ones by multiplying them by the polynomial X and adding a constant; this operation is known as 'Horner's scheme'. Polynomials with one sign change and a positive leading coefficient are obtained by starting with a positive constant, applying Horner's scheme a certain number of times with non-negative constants, then applying it with a negative constant, and then applying it again a certain number of times with non-positive constants.

Polynomials with only non-negative coefficients have curves that look like the curves 4 (a) or 4 (b) depending on whether the first coefficient is 0, adding a positive coefficient to a polynomial of the form 4 (a) or 4 (b) yields a polynomial of the form 4 (a), multiplying a polynomial of the form 4 (a) or 4 (b) by the polynomial X yields a new polynomial of the form 4 (b). Thus, Horner's scheme with non-negative constants keeps polynomials in the (a)–(b) form. Then, when applying Horner's scheme with a negative coefficient (thus introducing a sign change), the multiplication by X first builds a polynomial of the 4 (b) form, and adding a negative constant, one obtains a curve whose shape is given by 4 (c). From then on, multiplying a polynomial of the form 4 (c), 4 (d) or 4 (e) by X yields a polynomial of the form 4 (d); adding a negative constant to a polynomial of the form 4 (d) or 4 (e) yields a polynomial of the form 4 (e). Polynomials of the form 4 (d) or 4 (e) share

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press



(a) Non-negative coefficients and a non-zero constant coefficient.
(b) Non-negative coefficients and a zero constant coefficient.
(c) A negative constant coefficients and all others non-negative.
(d) One sign change and a zero constant coeffcient.
(e) One sign change and a negarive constant coeffcient.

Fig. 4. Classes of polynomials with or without sign change

the following characteristic: there exists a positive value x such that the polynomial has a negative value between 0 and x, and the slope of the curve is strictly positive above x. Because of the slope condition, we can also find a point where the polynomial is positive.

We will now give a more precise proof, outlining the concepts that are used in the formal proof.

3.2. *Lemmas for polynomials with non-negative coefficients*

Polynomials are simply encoded by their lists of coefficients, evaluating a polynomial at a given point is done recursively following Horner's scheme, and recognising polynomials with only non-negative coefficients is also done using a simple recursive function, written in the following form:

```
Fixpoint all_pos_or_zero (l : seq Q) : bool :=
if l is a::tl then (0 <= a) && all_pos_or_zero tl else true.
```

The type `Q` refers to the ordered field parameterising the development, `(0 <= a)` is a boolean value and the type constructor `seq` is a type for lists of elements in a type equipped with a boolean equality test.

We should note that polynomials satisfying the boolean predicate `all_pos_or_zero` may contain no positive coefficients: for this reason, we cannot guarantee that they are increasing or strictly positive anywhere between 0 and $+\infty$.

It is easy to prove by induction on lists that if they contain only non-negative coefficients, then the corresponding polynomial always has a non-negative value for inputs in $(0, +\infty)$ and from this, we can also prove by induction that any polynomial with only non-negative coefficients is constant or increasing.

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press

We then prove that for every polynomial p with non-negative coefficients, the product $x \times p(x)$ can be made arbitrarily close to 0 while x stays in $(0, +\infty)$.

3.3. *Two lemmas on slopes*

A first lemma on slopes concerns the existence of points where a polynomial p takes a value above an arbitrary bound a. This is guaranteed if the slope is bounded below by a positive ratio k, as it then suffices to take an input that is large enough. As the proof is constructive, we need to be more precise: assuming the slope is larger than k for any y larger than x_1, it suffices to take any input larger than

$$x_1 + \frac{a - p(x_1)}{k}.$$

This result is stored in our development under the name `above_slope`.

A second lemma on slopes concerns the slope of a product of the form $x \times p(x)$. This lemma reproduces the known formulas for the derivative of products of derivable functions, but is expressed solely in terms of lower bounds of slopes. If a function f has a slope larger than or equal to a non-negative ratio k_f when x is larger than a certain bound a, then the slope of the product $x \times f(x)$ is larger than $ak_f + f(x)$.

This statement requires f to have a positive slope, but it leaves open whether $f(x)$ is positive or not. In particular, the values a and k_r can be fixed for a large interval: we intend a to be the lower bound of interval I_2 as used in the criterion for existence of a unique root in an unbounded interval (see Figure 2).

3.4. *Polynomials with exactly one sign change*

We can now address the case of polynomials with exactly one sign change. We want to show that these polynomials have exactly one root. We exhibit the two intervals described in the criterion for unbounded intervals (see Figure 2): the positive value x_1 and the positive ratio k such that the polynomial is negative in the interval $(0, x_1)$ and the slope between any two values above x_1 is larger than k.

To detect polynomials with exactly one sign change, we use two recursive functions. The first, called `alternate_1`, recognises polynomials with at least one positive coefficient, preceded by any number of non-positive coefficients (possibly 0), and followed by only non-negative coefficients, as checked by `all_pos_or_zero`. This function is defined as follows:

```
Fixpoint alternate_1 (l:seq Q) : bool :=
  if l is a::tl then
     if 0 < a then all_pos_or_zero tl else alternate_1 tl
  else false.
```

The second function, called `alternate`, checks for the presence of at least one negative coefficient and then calls `alternate_1`.

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press

As we have two recursive functions, `alternate_1` and `alternate`, we actually need to perform two proofs by induction. Each proof by induction shows that some invariant is satisfied.

The invariant for `alternate_1` must be satisfied by a polynomial p that may or may not contain a negative coefficient, so this invariant cannot guarantee the existence of places where the polynomial takes a negative value. Instead, this invariant guarantees for any positive ε the existence of a positive `x` and a k such that:

(1) For any y between 0 and x, $p(y) \leqslant p(x)$.
(2) The slope between two points larger than x is guaranteed to be larger than k.
(3) The number $x \times p(x)$ is between 0 and ε.

The invariant for `alternate` is exactly the criterion we use to describe the existence of exactly one root in an unbounded interval as in Section 2.4. This proof by induction is done by induction on the list. The empty list does not satisfy the predicate `alternate`, so this case is easily taken care of. The other case is when the polynomial is described by a list of the form $a::l$, so that l represents another polynomial p_l and $p(x) = a + x \times p_l(x)$. Here another case distinction must be studied, depending on whether a is zero or negative.

If a is negative, we cannot use an induction hypothesis, because in this case l is only guaranteed to satisfy the predicate `alternate_1`. On the other hand, the invariant for `alternate_1` guarantees the existence of an x such that $x \times p_l(x)$ is positive and smaller than $-a$, this x is the correct witness and the slope is $p_l(x)$. Since $p_l(y) \leqslant p_l(x)$ when $y < x$, it is easy to prove that $a + y \times p_l(y) = p(y)$ is negative when $0 < y \leqslant x$. To reason about the slope, we use our lemma about the slope of $x \times p_l(x)$, using 0 as a lower bound for the slope of p_l (we only know that it is increasing).

If `a` is zero, we have by the induction hypothesis that there exist x and k such that p_l is negative on the left of x and has a slope larger than k on the right of x. However, this does not guarantee that x is the correct witness for p because the slope of $x \times p_l(x)$ is only larger than $p_l(x) + x \times k$, and $p_l(x)$ is negative. The solution is to note that p_l necessary takes a positive value at some point v_1 on the right of x, and then to use our *constructive intermediate value theorem* from Section 2.5 to build a new value x_1 such that

$$-\frac{kv_1}{2} \leqslant p_l(x_1) \leqslant 0$$

and $x_1 < v_1$. Now p_l is still guaranteed to be negative between x and x_1, because of the slope condition, and the slope on the right of x_1 is guaranteed to be larger than

$$\frac{kv_1}{2},$$

which is positive.

4. Bernstein polynomials and Bernstein coefficients

Bézier curves (Bézier 1986) are parametric curves that are widely used to construct smooth plane curves whose shapes are governed by a finite number of *control points*. A Bézier curve controlled by $n + 1$ points is a polynomial expression of degree n in its parameter t. For instance, given two points P_0 and P_1, the corresponding Bézier curve

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press

is the segment $B_{(P_0,P1)}(t) = tP_0 + (1-t)P_1$. For three control points P_0, P_1, P_2, the Bézier curve is $B_{(P_0,P_1,P_2)}(t) = (1-t)^2P_0 + 2(1-t)tP_1 + t^2P_2$. We already see in this case that a Bézier curve does not meet all its control points. In fact, it is only guaranteed to pass through the first and last control points. In the case of the quadratic Bézier curve, the middle control point P_1 is the intersection of the tangents to the curve at P_0 and P_2. The general formula for the Bézier curve with $n+1$ control points is

$$B_{(P_0,\ldots,P_n)}(t) = \sum_{k=0}^{n} \binom{n}{k} (1-t)^{n-k} t^k P_i,$$

which satisfies the recurrence relation

$$B_{(P_0,\ldots,P_n)}(t) = (1-t)B_{(P_0,\ldots,P_{n-1})}(t) + tB_{(P_1,\ldots,P_n)}(t).$$

Bézier curves are named after the engineer Paul Bézier who was working in the design division of a car manufacturing company. These curves have very interesting properties for interpolation but also for computer graphics: a Bézier curve is contained in the convex hull of its control points and uniform transformations on the control points, such as translation or rotation have the same effect on the curve. Points control the shape of the curve since the kth derivative of the curve at its extreme points is governed by the $k+1$ nearest control points.

Computer graphics algorithms usually use piecewise polynomial paths (called splines) of low degree. Bézier curves are often used to build these splines, resulting in the so-called Bézier splines. Most modern vector graphics standards, for instance SVG, feature support for Bézier splines. TrueType fonts use quadratic Bézier splines, whereas PostScript and MetaFont (Knuth 1986) use cubic Bézier splines.

Bernstein polynomials are defined as the weight assigned to each control point: the kth Bernstein polynomial $P_b(n,k)$ is defined by

$$B_{(P_0,\ldots,P_n)}(t) = \sum_{k=0}^{n} P_b(n,k)(t)P_k,$$

hence

$$P_b(n,k)(t) = \binom{n}{k} (1-t)^{n-k} t^k.$$

For arbitrary numbers l and r, we can also consider the following polynomials, called the *Bernstein polynomials for degree n and the interval* (l,r) for $0 \leqslant k \leqslant n$:

$$P_b(n,l,r,k) = \binom{n}{k} \frac{(x-l)^k (r-x)^{n-k}}{(r-l)^n}.$$

These polynomials also constitute a basis of the vector space of polynomials of degree n, and we will usually call it the *Bernstein basis*, leaving the degree and the values l and r unspecified. Hence, every polynomial p of degree n has a sequence of coefficients $(b_i)_{0 \leqslant i \leqslant n}$, such that

$$p(x) = \sum_{i=0}^{n} b_i P_b(n,l,r,i)(x).$$

The coefficients b_i are the *Bernstein coefficients* of p.

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press

When $l < r$, the Bernstein polynomials are positive on (l,r), and each polynomial of index k reaches its maximum at the point

$$d_k = l + \frac{(r-l)k}{n},$$

so each coefficient b_k somehow has a dominant influence on the value of the polynomial around d_k. Moreover, the coefficients $\binom{n}{k}$ included in the definition of $P_b(n,l,r,k)$ are chosen in such a way that the kth coefficient b_k of p would tend to have a value close to that of p in d_k. For instance, if p_1 is the constant polynomial with value 1, then all its Bernstein coefficients are equal to 1. Similarly, if p_2 is the identity polynomial and n is larger than 0, the Bernstein coefficients for p_2 are

$$l + \frac{(r-l)k}{n},$$

as can be verified through the computation

$$\begin{aligned}
&\sum_{i=0}^{n} \left(l + \frac{(r-l)i}{n}\right) \binom{n}{i} \frac{(x-l)^i (r-x)^{n-i}}{(r-l)^n} \\
&\qquad = \sum_{i=0}^{n} l \binom{n}{i} \frac{(x-l)^i (r-x)^{n-i}}{(r-l)^n} + \sum_{i=0}^{n} \frac{i}{n} \binom{n}{i} \frac{(x-l)^i (r-x)^{n-i}}{(r-l)^{n-1}} \\
&\qquad = l \frac{((x-l)+(r-x))^n}{(r-l)^n} + \sum_{i=1}^{n} \binom{n-1}{i-1} \frac{(x-l)^i (r-x)^{n-i}}{(r-l)^{n-1}} \\
&\qquad = l + (x-l) \sum_{i=0}^{n-1} \binom{n-1}{i} \frac{(x-l)^i (r-x)^{(n-1)-i}}{(r-l)^{n-1}} \\
&\qquad = l + (x-l) \frac{((x-l)+(r-x))^{n-1}}{(r-l)^{n-1}} \\
&\qquad = x,
\end{aligned}$$

where:

— In the first equation, we distribute inside the first sum, and in the second term, we simplify the denominator with the numerator $(r-l)$.

— In the second equation, we recognise that the first term contains a binomial formula corresponding to $((x-l)+(r-x))^n$, and in the second term, we recognise that the first element of the sum can be removed because it is 0, also we recognise that

$$\frac{i}{n}\binom{n}{i}$$

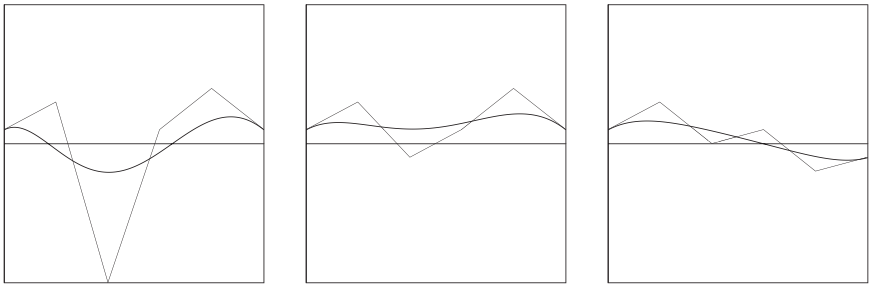
is

$$\binom{n-1}{i-1}$$

when $i \neq 0$.

— In the third equation, we use $(x-l)+(r-x) = r-l$ for the first term and factor out $(x-l)$ from the remaining indexed sum and re-index that sum.

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press



(a) 1, 3, -10, 1, 4, 1 (b) 1, 3, -1, 1, 4, 1 (c) 3, 0, 1, 10, -2, -1

Fig. 5. Bernstein control points and corresponding polynomial curves

— We then recognise another binomial formula and can conclude.

The Bernstein coefficients are related to a broken line (made up of contiguous straightline segments), which gives a rough approximation of the polynomial's function graph. More precisely, given the bounds (l, r) of the interval, and the Bernstein coefficients $(b_0, \ldots, b_n)$ of polynomial p, the $n+1$ points with coordinates

$$\left(l + i\frac{r-l}{n}, b_i\right)$$

are the control points for the Bézier curve that coincides with the polynomial's graph. This is illustrated in Figure 5.

In Figure 5 (a) the illustration shows that a peak in the disposition of the control points corresponds to an inflexion in the polynomial's curve (the Bernstein coefficients are 1, 3, −10, 1, 4, 1 and −10 corresponds to a downward peak). In this case, the peak provokes two sign changes, which are reproduced in the shape of the curve and correspond to the existence of two roots inside the interval. In Figure 5 (b), the coefficients still exhibit a downward peak with a negative coefficient, but the polynomial's curve stays above the x-axis and the two sign changes in the Bernstein coefficients do not correspond to any real root for the polynomial (this is a false alert). In Figure 5 (c), there is one sign change, so that the first and last coefficients have opposite signs. In fact, the first and last Bernstein coefficients are equal to the values of the polynomial at the bounds of the interval, so this imposes the existence of at least one root. But because there is exactly one sign change in the coefficients, we can be sure that any other bend in the curve stays away from the x-axis.

5. From Bernstein to Descartes

In this section, we clarify the polynomial transformations that link the problem of finding the roots of a polynomial inside an arbitrary bounded interval (l, r) successively with the problem of finding the roots of another polynomial inside the interval $(0, 1)$ and with the problem of finding the roots of yet another polynomial inside the interval $(0, +\infty)$. These transformations make it possible to compute another collection of coefficients, which happen to be very simply related to the Bernstein coefficients.

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press

5.1. *A criterion for the interval* $(1, +\infty)$

Descartes' weak law of signs gives us a sufficient condition to determine when the unbounded interval $(0, +\infty)$ contains exactly one root for a polynomial. Through a change of variable, we obtain a similar criterion for the interval $(1, +\infty)$.

In the following, we will write θ_v for the transformation that maps any polynomial p to the polynomial $y \mapsto p(y + v)$. If

$$p = \sum_{i=0}^{n} a_i x^i,$$

we have the formula

$$p(y+v) = \sum_{i=0}^{n} a_i (y+v)^i = \sum_{k=0}^{n} \left(\sum_{i=k}^{n} a_i \binom{i}{k} v^{i-k} \right) y^k.$$

The polynomial p has exactly one root in the interval $(v, +\infty)$ if and only if the polynomial $\theta_v(p)$ has exactly one root in the interval $(0, +\infty)$. We have proved this lemma.

Thus, if we apply Descartes's law of signs to the coefficients

$$\sum_{i=k}^{n} a_i \binom{i}{k},$$

we can obtain a sufficient criterion for the existence of exactly one root of the polynomial

$$p = \sum_{i=0}^{n} a_i x^i$$

in the interval $(1, +\infty)$.

5.2. *A criterion for the interval* $(0, 1)$

Descartes' law of signs works for unbounded intervals. In this section, we see how to cover bounded intervals as well. The trick here relies on reversing the polynomial's list of coefficients. Obviously, the number of sign changes in a list of coefficients is the same as the number of sign changes in the reversed list.

However, the roots of a polynomial on the interval $(1, +\infty)$ are in one-to-one correspondence with the roots of the reversed polynomial in $(0, 1)$. This is due to the following equation:

$$\sum_{i=0}^{n} a_i x^i = x^n \times \sum_{i=0}^{n} a_i x^{i-n}$$

We can now perform another change of variable, here $y = 1/x$ and a change of index $j = n - i$ in the sum.

$$\sum_{i=0}^{n} a_i x^i = \left(\frac{1}{y}\right)^n \sum_{j=0}^{n} a_{n-j} y^j$$

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press

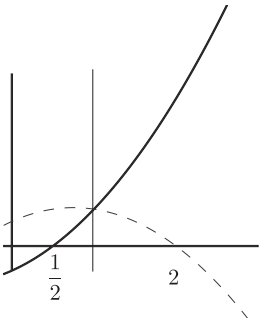


Fig. 6. Curves of $x^2 + \frac{3}{2}x - 1$ (solid line) and its reverse $-x^2 + \frac{3}{2}x + 1$ (dashed)

The polynomial

$$\sum_{j=0}^{n} a_{n-j} y^j$$

is exactly the reversed polynomial in y, and the expression

$$\left(\frac{1}{y}\right)^n$$

never becomes 0 for $y \in (0, 1)$. Thus, x is a root of the polynomial between 1 and $+\infty$ if and only if $y = x^{-1}$ is a root of the reversed polynomial between 0 and 1.

We will write ρ for the function that computes the reverse of a polynomial. Here we need to be precise: the coefficients of a polynomial of degree n actually are the coefficients of a vector in an $n+1$ dimensional space, whose basis is made of the monomials X^i where $i \in \{0, \ldots, n\}$. Seen as an operation on this vector space, ρ is an involutive automorphism. But polynomials of degree less than n are also elements of this vector space, and the reverse operation must be understood as reversing the list of coefficients of length $n + 1$ obtained by padding the polynomials description with enough 0 coefficients.

To illustrate the correspondence between a polynomial and its reverse, consider the polynomial

$$p(x) = x^2 + \tfrac{3}{2}x - 1.$$

The reversed polynomial is

$$q(x) = -x^2 + \tfrac{3}{2}x + 1$$

and after the variable change we obtain the polynomial $-x^2 - 2x + 1$ which exhibits only one sign change. This predicts that the polynomial has exactly one root between 0 and 1, and, indeed, the two roots of the initial polynomial are -2 and $1/2$. This is illustrated in Figure 6, where the solid line is the curve for the polynomial p, and the dashed line is the curve for the polynomial q, which has a single root between 1 and $+\infty$.

As a conclusion, we can establish a correspondence between unique roots in $(1, +\infty)$ of a polynomial p and unique roots in $(0, 1)$ of the polynomial $\rho(p)$. When working in a field that is only guaranteed to be ordered and archimedean, this correspondence works by linking the criterion for unbounded intervals with the criterion for bounded intervals.

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press

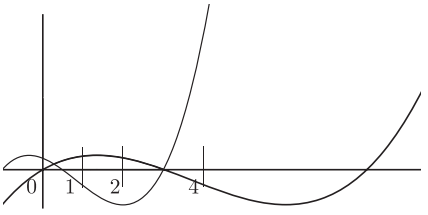


Fig. 7. Curves of a polynomial inside $(2,4)$ and the corresponding transformed polynomial inside $(0,1)$

The proof of this correspondence involves the computation of the slope for $x^n p(1/x)$ from the slope of p, which makes it trickier than other proofs of this section. This is again a place where our constructive intermediate value theorem plays a role.

5.3. *Handling arbitrary bounded intervals*

The next step is to relate the roots of any polynomial inside an arbitrary interval (l,r) with the roots of another polynomial inside the interval $(0,1)$. This is done with another change of variable, this time $x = (r-l)y + l$. In other words, the polynomial function that maps any x to $p(x)$ has a root between l and r if and only if the polynomial function that maps any y to $p((r-l)y+l)$ has a root between 0 and 1.

Here again, we can define a generic transformation on polynomials, written χ_k, that corresponds to expanding with a given ratio k. For an arbitrary polynomial

$$p = \sum_{i=0}^{n} a_i X^i,$$

the polynomial $\chi_k(p)$ is defined by

$$\chi_k(p) = \sum_{i=0}^{n} a_i (kX^i) = \sum_{i=0}^{n} a_i k^i X^i.$$

Thus, the change of variable to study the roots of polynomial p is actually represented by $\chi_{r-l} \circ \theta_l$.

The geometric effect of the polynomial transformation is illustrated in Figure 7, where the shape of the curve for the polynomial

$$\frac{x^3}{8} - \frac{x^2}{8} + 3x$$

inside the interval $(2,4)$ is reproduced by the shape of the curve for the polynomial

$$x^3 - \tfrac{5}{2}x^2 - 2x + \tfrac{3}{2}$$

inside the interval $(0,1)$.

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press

5.4. *Recapitulating operations*

In our formal development, we defined the three operations for translating (θ), expanding (χ), and reversing the list of coefficients (ρ). We can then compute a sequence of coefficients by applying the transformation

$$\tau = \theta_1 \circ \rho \circ \chi_{r-l} \circ \theta_l.$$

When the coefficients we obtain have exactly one sign change, we know that the polynomial has exactly one root inside the interval (l, r).

By construction, each of the operations θ, ρ, χ is actually a linear application of the vector space of polynomials of degree less than n into itself. The inverse of θ_a is θ_{-a}, and this is easily proved, so θ_a is always bijective. When k is non-zero, the inverse of χ_k is $\chi_{\frac{1}{k}}$, so this linear application is also bijective. The inverse of ρ is itself. As a result, the whole transformation is also inversible and its inverse is

$$\tau^{-1} = \theta_{-l} \circ \chi_{\frac{1}{r-l}} \circ \rho \circ \theta_{-1}.$$

We proved that the images of the monomials X^i by this inverse transformation are multiples of the Bernstein polynomials, in reverse order. Let us first observe the effect of $\rho \circ \theta_{-1}$:

$$\begin{aligned}
\tau^{-1}(x^i) &= \theta_{-l} \circ \chi_{\frac{1}{r-l}} \circ \rho \left(\sum_{j=0}^{i} \binom{i}{j} (-1)^j x^{i-j} \right) \\
&= \theta_{-l} \circ \chi_{\frac{1}{r-l}} \left(\sum_{j=0}^{i} \binom{i}{j} (-1)^j x^{n-i+j} \right) \\
&= \theta_{-l} \circ \chi_{\frac{1}{r-l}} \left(x^{n-i} \sum_{j=0}^{i} \binom{i}{j} (-x)^j \right) \\
&= \theta_{-l} \circ \chi_{\frac{1}{r-l}} \left(x^{n-i}(1-x)^i \right).
\end{aligned}$$

Then the effect of $\theta_{-l} \circ \chi_{\frac{1}{r-l}}$:

$$\begin{aligned}
\tau^{-1}(X^i) &= \theta_{-l} \left(\frac{x^{n-i}}{(r-l)^{n-i}} \left(1 - \frac{x}{r-l}\right)^i \right) \\
&= \theta_{-l} \left(\frac{x^{n-i}}{(r-l)^{n-i}} \frac{(r-l-x)^i}{(r-l)^i} \right) \\
&= \theta_{-l} \left(\frac{x^{n-i}(r-l-x)^i}{(r-l)^n} \right) \\
&= \frac{(x-l)^{n-i}(r-l-(x-l))^i}{(r-l)^n} \\
&= \frac{1}{\binom{n}{i}} (P_b(n, l, r, n-i)).
\end{aligned}$$

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press

If the transformation $\tau(p)$ leads to a sequence of coefficients c_i, this means

$$\tau(p) = \sum_{i=0}^{n} c_i x^i.$$

Now, using the fact that both τ and τ^{-1} are linear, we can see that the polynomial p is

$$\begin{aligned} p &= \tau^{-1}\left(\sum_{i=0}^{n} c_i X^i\right) \\ &= \sum_{i=0}^{n} c_i \tau^{-1}(x^i) \\ &= \sum_{i=0}^{n} c_i \frac{1}{\binom{n}{i}} P_b(n, l, r, n-i). \end{aligned}$$

Thus, the Bernstein coefficients are obtained as follows:

$$b_i = \binom{n}{n-i}^{-1} c_{n-i} = \binom{n}{i}^{-1} c_{n-i}.$$

Since the number of sign changes does not depend on the order in which the list is observed, we obtain the proof that one sign change in the sequence of Bernstein coefficients implies the existence of a root in the interval (l, r).

6. Dichotomy

Bernstein coefficients give precise information when they exhibit either zero or one sign change. In the first case, we have a guarantee that there are no roots of the considered polynomial in the considered interval. In the second case, we have a guarantee that there is exactly one root.

When Bernstein coefficients exhibit more than one sign change, no conclusion can be drawn about the existence and unicity of roots in the interval. For instance, in Figure 5 (b), the Bernstein coefficients exhibit two sign changes, but there is no root inside the interval. When facing this kind of inconclusive information, the solution is to refine the approximation given by the control line.

6.1. *Geometric intuition for dichotomy*

When cutting an interval into two halves, the number of control points is approximately doubled, because each of the new half-intervals receives a new sequence of n Bernstein coefficients. As a result, the control points are closer to each other and to the polynomial's curve and they give a more accurate account of the curve's position with respect to the real x-axis. This is illustrated in Figure 8, where the initial Bernstein coefficients exhibit two sign changes, which are needed to account for the bend in the first half of the interval (a positive local minimal, but expressed by a negative Bernstein coefficient). In the halved interval two more points are added in the vicinity of the bend, and none of the control points has to be negative anymore.

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press

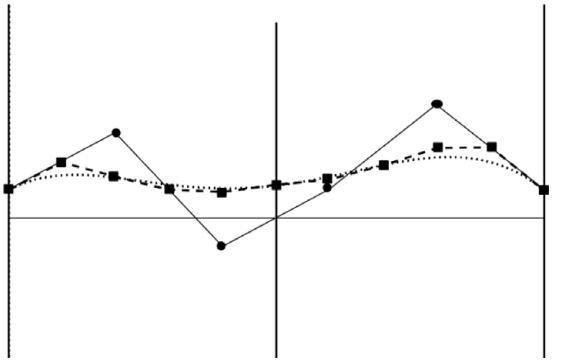


Fig. 8. Bernstein control points for halved intervals

In Figure 8, the dotted line represents the polynomial's curve, the solid line links the control points for the complete interval, marked by round bullets (the Bernstein coefficients are 1, 3, −1, 1, 4, 1 for this interval). The dashed line links the control points for the two half intervals, marked by square boxes (the Bernstein coefficients are 1, 2, 1.5, 1, 0.9375, 1.15625 for the first half interval, and 1.15625, 1.375, 1.875, 2.5, 2.5, 1 for the second half interval). This figure shows that the control line really does get closer to the polynomial's curve, and provides a much better approximation of the polynomial.

The formula given in Section 4 is useful for computing an initial series of Bernstein coefficients, and the correctness of the conditions for existence of roots based on these coefficients can be justified using the transformation described in Section 5.4.

It may seem that computing Bernstein coefficients is a costly process. Around 1950, while studying Bézier curves, De Casteljau noticed that the coefficients for the sub-intervals were easy to compute from the coefficients for the big interval through a simple recursive process, exploiting the recurrence relation already given in Section 4. This suggests another recursive algorithm, starting from a large interval that is guaranteed to contain all the roots of a polynomial (see Section 6.2), and splitting this interval into smaller pieces until all roots have been isolated (see Section 6.3).

6.2. *Initialisation*

Given an arbitrary non-constant polynomial p of degree n defined by

$$p = \sum_{i=0}^{n} a_i X^i,$$

it is actually possible to bound the absolute values of its roots by a simple constant, called the Cauchy bound (Basu *et al.* 2006), defined from the coefficients $a_i (i = 0 \ldots n)$.

Theorem 6.1 (Cauchy bounds).

$$\forall x \in \mathbb{R}, p(x) = 0 \Rightarrow |x| \leqslant C(p)$$

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press

with

$$C(p) = \sum_{i=0}^{n} \frac{|a_i|}{|a_n|}.$$

Proof. Let x be a root of p.

If $|x| \leqslant 1$, since $1 \leqslant C(p)$, the inequality holds trivially.

For $|x| > 1$, since x is a root and $a_n \neq 0$, we have

$$x^n = -\frac{1}{a_n} \sum_{i=0}^{n-1} a_i x^i.$$

Hence,

$$|x|^n \leqslant \frac{1}{|a_n|} \sum_{i=0}^{n-1} |a_i||x|^i \leqslant \frac{1}{|a_n|} \sum_{i=0}^{n} |a_i||x|^i$$

by the triangle inequality. So

$$|x| \leqslant \frac{1}{|a_n|} \sum_{i=0}^{n-1} |a_i||x|^{i-(n-1)} \leqslant \frac{1}{|a_n|} \sum_{i=0}^{n-1} |a_i|$$

since $|x| > 1$ implies that for all $i = 0 \ldots n-1$, we have $|x|^{i-(n-1)} \leqslant 1$. Finally, since

$$\frac{1}{|a_n|} \sum_{i=0}^{n-1} |a_i| \leqslant \frac{1}{|a_n|} \sum_{i=0}^{n} |a_i|,$$

we have $|x| \leqslant C(p)$. □

This means that when studying the roots of a polynomial p, we can restrict the infinite real line to a bounded interval $(-C(p), C(p))$. This justifies us starting a real root isolation process by providing the initial interval of interest. On this first interval, we compute Bernstein coefficients from the transformations presented in the previous section. Then, if there is more than one sign change, we continue by invoking the splitting de Casteljau algorithm explained in the next subsection.

6.3. *Splitting algorithm*

Given three pairwise distinct rational numbers l, r, m, there exists an efficient algorithm to deduce the two respective lists of Bernstein coefficients of a polynomial p on intervals (l, m) and (m, r) from the list of Bernstein coefficients of p on interval (l, r).

Let $\mathtt{b}$ be the sequence of Bernstein coefficients of a polynomial p of degree n for an interval (l, r). Let m be a number distinct from l and r. We put

$$\alpha = \frac{m-l}{r-l}$$
$$\beta = \frac{r-m}{r-l}.$$

The `de_casteljau` algorithm is defined recursively by:

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press

```
Variables (alpha beta : Q).
Fixpoint de_casteljau (b : nat -> Q) (n : nat) :=
  match n with
  | O => b
  | i.+1 => fun j =>
    (alpha * de_casteljau b i j + beta * de_casteljau b i j.+1)
end.
```

where the initial sequence of coefficients `b` is represented by an infinite sequence of rational numbers, for which only the first n elements are relevant. The following function gives the Bernstein coefficients of p on the finite interval (l, m):

```
Definition dicho' alpha beta c i :=
  de_casteljau alpha beta c i O.
```

The following function gives the Bernstein coefficients of p on the finite interval (m, r):

```
Definition dicho alpha beta p c i :=
   de_casteljau alpha beta c (p - i) i.
```

Looking at the function `de_casteljau` more closely, we see that the algorithm actually proceeds by creating a succession of lines where the element at rank j in a given line is obtained by computing a weighted sum of the two elements at rank j and $j+1$ on the previous line.

This process can be illustrated geometrically by a succession of broken lines. For the first line, we take the control line of the initial interval. Then, for each of the segments that compose this control line, we cut this segment in the same proportion as the proportion in which the interval is split between (l, m) and (m, r). This gives us a new collection of points. We started with $n+1$ control points and thus had n segments, we now have n new points, defining $n-1$ new segments. We repeat this process with the new segments, until we reach a situation where there is only one segment and we again split this segment into two parts in the proportion of (l, m) to (m, r). The last point is guaranteed to lie on the polynomial's curve.

Although we actually only use de Casteljau's algorithm when m is the midpoint of the initial interval, it works for any relative positions of l, m, and r, as long as they are pairwise distinct.

The different points computed by the de Casteljau algorithm are represented on Figure 9. The innermost points are the control points in the two new bases, which are computed from the original control points $\{B0, \ldots, B5\}$. The middle innermost control point, which lies on the curve, belongs to the two new lists of control points. Points $\{C0, \ldots, C5\}$ are the control points in the left half, $\{D0, \ldots, D5\}$ are the control points on the right half. De Casteljau's algorithm is used extensively in computer graphics for rasterising Bézier curves.

The aim of this section is to prove that this algorithm is correct, that is, that the `dicho` and `dicho'` functions do indeed compute the expected Bernstein coefficients. The correctness theorems as stated in Coq are:

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press

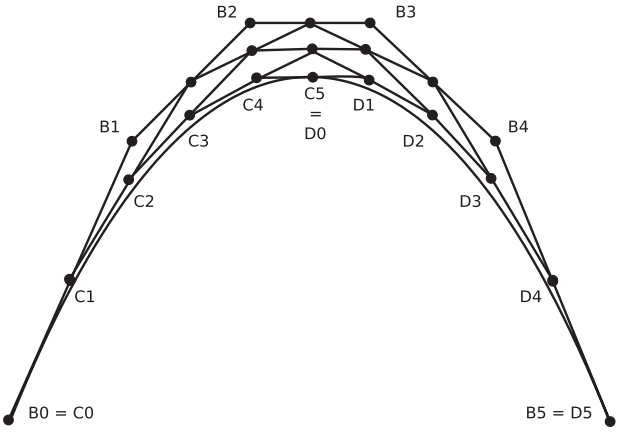


Fig. 9. Intermediate points computed by de Casteljau's algorithm on $[B0, B5]$

```
Lemma dicho'_correct : forall (l r m : Q)(q : {poly Q})(p : nat)
(c : nat -> Q)
(alpha := (r - m) * (r - l)^-1) (beta := (m - l) * (r - l)^-1),
  m != l ->
  q = \sum_(i < p.+1)(c i) * bernp l r p i ->
  q = \sum_(j < p.+1)(dicho' alpha beta c j) * bernp l m p j.

Lemma dicho_correct : forall (l r m : Q)(q : {poly Q})(p : nat)
(c : nat -> Q)
(alpha := (r - m) * (r - l)^-1) (beta := ((m - l) * (r - l)^-1)),
  m != r ->
  q = \sum_(i < p.+1)(c i) * bernp l r p i ->
  q = \sum_(j < p.+1)(dicho alpha beta p c j) * bernp m r p j.
```

where `(bernp l r p i)` is the ith polynomial in the Bernstein basis of degree p with parameters l and r.

The properties of the computations performed by the de Casteljau algorithm are summarised in Figure 10. Starting from the input list $b = (b_0^{(0)} \dots b_p^{(0)})$ of coefficients in the basis with parameters l and r on the upper side of the triangle, it computes the full triangle, so that in the end the two expected output lists can be read on the two other sides of the triangle. The list $b' = b_0^{(0)} \dots b_0^{(j)} \dots b_0^{(p)}$ is the list of coefficients in the basis with parameters l and m output by `dicho'`. The list $b'' = b_0^{(p)} \dots b_j^{(p-j)} \dots b_p^0$ is the list of coefficients in the basis with parameters m and r output by `dicho`. The small triangle area in Figure 10 shows which values the computation of an arbitrary given point in the triangle relies on. This structure is imposed by the fixpoint equation of the recursive definition of the de Casteljau algorithm:

```
de_casteljau alpha beta b n.+1 i =
(de_casteljau alpha beta b n i) + (de_casteljau alpha beta b n i.+1)
```

This looks very similar to the recursive relation governing the Pascal triangle.

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press

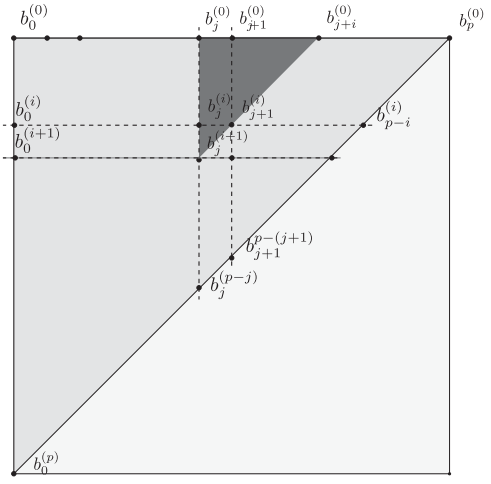


Fig. 10. Properties of de Casteljau computations

First note that the shape of Bernstein polynomials implies that

```
Lemma bern_swap :
 forall n i l r,
 (i <= n) -> r != l -> bernp r l n i = bernp l r n (n - i).
```

This remark implies that if b is the list of coefficients of the polynomial p in the Bernstein basis of degree n with parameters l and r, then the reverse of b is the list of coefficients of the same polynomial p in the Bernstein basis of degree n with parameters r and l:

```
Lemma bern_rev_coef : forall (n : nat)(l r : Q)(b : nat -> Q),
  \sum_(i < n.+1)(b i) * (bernp l r n i) =
  \sum_(i < n.+1)(b (n - i)) * (bernp r l n i).
```

This shows that the correctness of the `dicho'` function is enough to get a certified computation of both Bernstein coefficient lists. If b is the initial list of Bernstein coefficients with parameters l and r, then reversing b gives the coefficients with parameters r and l. Applying `dicho'` to the reverse of b using r, l, and m computes the coefficients with parameters r and m, hence reversing this output gives the result expected for `dicho` on b using l, r, and m. Using a similar symmetry in the `de_casteljau` algorithm, we in fact reduce the proof of the `dicho_correct` specification to the proof of `dicho'_correct`.

By linearity, we can also reduce the proof of the `dicho'_correct` specification to the case where the input polynomial p is in fact itself a Bernstein polynomial. This means that the input coefficient list b only contains zeros except at one position where the coefficient is one.

We now first compute the expected output of the `dicho'` function on such a list. In other words, for any distinct rational numbers l, r, m and any $n \in \mathbb{N}$, given $i \leqslant n$, we want

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press

to compute the coefficients of

$$P_b(n,l,r,i) = \binom{n}{i}\frac{(X-l)^i(X-r)^{n-i}}{(r-l)^n}$$

in the basis $(P_b(n,l,m,i))_{i=0,\ldots,n}$. We put

$$\alpha = \frac{r-m}{r-l}$$
$$\beta = \frac{m-l}{r-l}.$$

In the polynomials of the new basis, formal denominators are of the form $(m-l)$. Then, noting that

$$\frac{X-l}{r-l} = \beta\frac{X-l}{m-l}$$
$$\frac{r-X}{r-l} = \alpha\frac{X-l}{m-l} + \frac{m-X}{m-l}$$

and using the binomial identity,

$$\binom{n}{i}\binom{n-i}{j-i} = \binom{j}{i}\binom{n}{j},$$

we get

$$P_b(n,l,r,i) = \sum_{j=i}^{n}\binom{j}{i}\alpha^{j-i}\beta^i P_b(n,l,m,j). \qquad (*)$$

Now to prove the `dicho'_correct` lemma, it is sufficient to prove that the values output by the `dicho'` function coincide with the ones of (*), which boils down to an induction on *i*.

7. Formalisation issues

7.1. *Sequences, iterations and polynomials*

One of the notable features of the SSReflect libraries is that they devote a substantial effort to infrastructure. For instance, the library for sequences contains as many operators as one would expect to find in the standard library for a functional language, including, factories, access, filtering, surgery, and so on, plus a comprehensive set of specifications for these operators. For instance, to construct the subdivision used in the proof of the weak intermediate value theorem proposed in Section 2.5, it is enough to use the following tactic line in the proof:

```
pose sl := map (fun k => x + (y - x) * (k%:R / (n.+1%:R))) (iota 0 n.+2).
```

The `(iota 0 n.+2)` operator constructs the sequence of integers between `0` and `n.+2`. Then the `map` generic list operator maps this sequence to the desired subdivision of the

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press

interval $[x, y]$. The infix `+`, `*` and `\` operators are the generic notations for addition, product and division in a field, respectively, and here used for the ordered archimedean field that is a parameter to the development. The `%:R` annotation is a postfix notation for the embedding of integers in any ring structure (see Section 7.2). Now the index of the first point in this sequence where the polynomial has a negative value is found by the definition

```
pose a'_index := find (fun x => p.[x] >= 0) sl.
```

where again the `find` operator is a generic construction. The theory available for `find` is sufficient to prove all the results needed in the proof.

In the SSReflect distribution, the sequence library is extended with a library for iterated operations (Bertot *et al.* 2008), which provides a comprehensive infrastructure for working with the finite iteration of operators equipped with some known properties like associativity, commutativity or distributivity. Although the definition of iteration is naturally based on a sequence, the library includes a variety of indexing facilities. This library is crucial in the proof of the de Casteljau algorithm presented in Section 6.3, both for the initialisation with Cauchy bounds and for proving the correctness of the computation, that is, that the output coefficients do indeed provide a correct decomposition of the initial polynomial on the Bernstein basis. The `\sum` operator implicitly carries the properties that the iterated operation is commutative and associative, and that there is a product law that distributes over the sum. The `\sum_(i < n)` notation indicates that the iteration is performed on the sequence of integers from `0` to n.-1 (meaning the sum might be empty). Manipulations of these expressions are treated in the generic theory of iterated operators.

Sequences also provide the core of the definition of polynomials. However, the definition given in Section 2.3 provides few facilities for defining a polynomial extensionally, by simply providing the values of its coefficients. This is made possible by the following construction, and its associated notation:

```
Notation "\poly_ ( i < n ) E" := (Poly (mkseq (fun i : nat => E) n)).
```

The expression `(mkseq (fun i : nat => E)n)` builds a sequence containing the `n` first values of the function `E` over natural numbers thanks to the generic `mkseq` operator. The variable `i` is bound in the body expression `E`. The `Poly` function of type

```
forall T, seq T -> {poly T}
```

normalises the sequence it takes as its argument into a polynomial, such that the tail zeroes are erased and a proof of normal form is added to the resulting sequence. Hence, the size of the (sequence contained in the) polynomial `\poly_(i < n)E` is only smaller than or equal to `n`, but is only equal to `n` if `E n.-1` is known to be non-zero. This construction is used to define the expansion and translation introduced in Section 5, for instance,

```
Definition expand (p : {poly Q})(k : Q) :=
  \poly_(i < size p)(p`_i * k ^+i).
```

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press

where `Q` is the ordered archimedean field parameterising the development and `^+` is the exponentiation operation (see Section 7.2). These definitions are accompanied by correctness lemmas in term of evaluation, such as

```
Lemma eval_expand : forall p k x, (expand p k).[x] = p.[k * x].
```

where `p.[x]` denotes the evaluation of the polynomial `p` at the point `x`.

The list of coefficients defining a polynomial is, of course, the coefficients of the polynomial in the (countable) infinite basis of monomials. In this development, we also need to consider polynomials as vectors in the vector space of polynomials of degree less than a fixed value, again with coordinates in the basis of monomials or in Bernstein bases. More precisely, we only use the fact that monomials and Bernstein polynomials are two families of generators and do not use their linear independence. Unfortunately, the linear algebra part of the SSReflect archive is not yet sufficiently well integrated to get this easily from an existing infrastructure, so we have only defined these linear algebra operations at a low level: the list of coefficients defining a polynomial is padded with zeroes when the list of coefficients in a monomial basis of a certain degree is needed, and, conversely, the list of coefficients in a monomial basis is only normalised when we need to go back to the actual polynomial. It would certainly be more elegant to use a generic library than to rely on this *ad hoc* solution.

7.2. *Algebraic structures*

This formalisation is structured along the algebraic hierarchy proposed by the SSReflect libraries (Gonthier *et al.* 2007). This hierarchy builds a graph of algebraic structures based on types with decidable equalities, and implements inheritance and sharing of theories and notations through the canonical structures type inference mechanism of the Coq system. Moreover, we base our proofs on an extension of this hierarchy for ordered structures (Cohen 2010). This extension allowed us to consider ordered unit rings and fields, which are unit rings and fields equipped with a decidable (total) boolean order relation that is compatible with addition and product. In the proofs we describe here, the main algebraic structures involved are the field of coefficients and the commutative ring structure that polynomials inherit. This inheritance is completely automated by the infrastructure already present in the SSReflect libraries. However, natural numbers are not equipped with any such algebraic structure in the SSReflect library, and their theory and notations are defined separately. However, we can still use the same symbols for their algebraic operations and ordering thanks to the scoping mechanism of the Coq system, even if this sometimes requires an explicit scoping annotation in theorem statements.

Natural numbers are used to define the (discrete) exponentiation as iterated product in ring structures: if `x` belongs to a type equipped with a ring structure and `(n : nat)` is a natural number, `x^+n` denotes `x` to the *n*th power, that is, `x * (x * .. (x * x)...)`. Since, at the time of writing, the SSReflect libraries did not feature a formalisation of integers, exponentiation in a field structure can become artificially technical: `x^-n` only denotes the inverse of `x^+n`, with a (positive) natural number exponent. However, this defect was not a severe limitation for the current work. An other interesting operation

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press

is the iteration of the addition operation in a ring structure: `x *+n` denotes `n` times `x`, that is, `x + (x + .. (x + x)...)`. Thus, in a commutative ring structure, the expression developing the power of a sum is

```
Lemma exprn_addl : forall x y n,
  (x + y) ^+ n = \sum_(i < n.+1) (x ^+ (n - i) * y ^+ i) *+ 'C(n, i).
```

as available in the SSReflect library. This allows us to combine smoothly the binomial identities also available in SSReflect with operations in a commutative ring, as required in the correctness proofs of the de Casteljau algorithm presented in Section 6.3. When the iterated element `x` is in fact `1`, the unit element of the ring, this operation defines a generic image of the semi-ring of natural numbers in the ring. Note that this operation need not be injective: this will only be the case if the characterictic is zero. This embedding is so widely used that a new notation was defined: `n%:R` denotes `1 *+ n` in any ring structure. Note that this expression needs a context or an explicit cast to determine the ring in which the natural number should be injected, that is, the ring for which `1` is the unit. In particular, this injection appears in the hypothesis that the field is archimedean, which is necessary for the weak intermediate value theorem. This hypothesis is stated as follows:

```
Variable Q : oFieldType.
Hypothesis Q_arch : forall x:Q, 0 <= x -> {n : nat | x <= n%:R}.
```

where the `Variable Q` declares an ordered field structure parameter, and the `Q_arch` hypothesis gives access to the explicit value of an integer larger than a given element of the ordered field. Again, the SSReflect distribution lacks a library for integers and rational numbers, so the manipulation of the embedding of rational numbers in this archimedean field is sometimes unnecessarily tedious, though again this was not really an issue in this development.

7.3. *Automation issues*

Sadly, a significant proportion of the scripts is devoted to too many atomic rewrite steps for normalising ring expressions or proving trivial consequences of the properties of the order, such as transitivity or compatibility with field operations. The SSReflect libraries still lack the standard automated proof producing decision procedures available in the Coq system, such as ring normalisation or linear arithmetic decision – the SSReflect structures are still not connected to these mechanisms. The Coq tactics for linear arithmetic hardcode the representation of integers and coefficients, but should rely on a more abstract structure such as an ordered field. In the case of the automation of ring identities, it would significantly help the user if normalisation could handle simultaneously the various ring and semi-ring structures that can occur in an expression, like the rings of polynomial coefficients, polynomials themselves and, possibly, the semi-ring of natural numbers. In the case of the automation of ordered arithmetic, proof steps often involve non-linear expressions, for which it is quite difficult to get a truly generic and efficient proof producing decision procedure. This is in fact one of the long term objectives of this work, namely, to certify a complete decision procedure for the full first-order theory of real closed fields.

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press

However, some incomplete lightweight tools could probably be crafted to relieve the user of tedious steps when possible.

7.4. *Current state of the formalisation*

In this section, we recapitulate the main results described in this paper that have a formal proof in our development:

— The absolute values of the real roots of a polynomial are bounded by the Cauchy bound, which is expressed only using the absolute values of the coefficients of the polynomial.
— If a polynomial function p has a negative value in x and a positive value in y, with $x < y$, then for any ε, one can exhibit x' and y' such that $-\varepsilon < p(x') < p(y') < \varepsilon$.
— If a polynomial only has one sign change in its coefficients for the standard monomial basis, then this polynomial has exactly one root between 0 (excluded) and $+\infty$.
— If a polynomial only has one sign change in its Bernstein coefficients for a given interval (l, r), then this polynomial has exactly one root between l and r (excluded).
— The transformation that maps any polynomial to its Bernstein coefficients maps Bernstein polynomials to plain monomials.
— De Casteljau's algorithm correctly computes the Bernstein coefficients for the intervals (l, m) and (m, r) from the Bernstein coefficients for the interval (l, r).

This work is part of a more ambitious plan, aiming at providing an efficient procedure to isolate the roots of any polynomial. We still need to develop the connections between the various results that will constitute this procedure and its proof of correctness. To certify an algorithmically naive version of such a procedure, we still need to describe the procedure to reduce any polynomial to a separable polynomial (a polynomial where all roots have multiplicity 1). The easy approach would be to divide by the greatest common divisor between the polynomial and its derivative. The reduction to separable polynomials should not require too much effort considering the libraries already available in the SSReflect package. We also need to describe the termination of a procedure based on successive dichotomy. However, such a study of termination might require more substantial work.

Another issue will be to connect the correctness proof of such a naive implementation with more realistic programs, such as an implementation of de Casteljau whose complexity would be linear in the degree of the input, as implemented in Mahboubi (2007), or even more optimised codes like the ones of Mourrain *et al.* (2005).

8. Conclusion

Real root isolation methods by sign-change-based methods is a classical topic, which has been extensively studied (see Rouillier and Zimmermann (2003) for a review of the related literature) after the pioneering work of Uspensky (Uspensky 1948). Bernstein polynomials are used to provide efficient implementations of these methods (Mourrain *et al.* 2005; Rouillier and Zimmermann 2003). To our knowledge, the current work is the first mechanised proof of de Casteljau's algorithm, and of the building blocks of a real root isolation procedure. The closest work to ours is probably the study of global optimisation methods in Coq lead by Roland Zumkeller (Zumkeller 2008), where Bernstein polynomial

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press

bases were also used to approximate continuous functions on a closed domain. This work results in an implementation in the Coq system of a tool to bound optima of multivariate continuous real functions. However, we could not find any mention of a formalisation of the correctness proofs of this tool.

Other work concerned with roots of continuous functions often relies on Newton's method. This method has also been described formally using Coq and SSReflect, although in a less constructive setting (Paşca 2011). However, the applicability of the work is quite different: the work in Paşca (2011) applies to a wide variety of functions of class C2 and it guarantees the unicity of a root in an interval using strong constraints on the first and second derivative of the function inside the interval. It does not provide any tool for the case when the chosen interval does not satisfy the conditions, while our work provides a study of a dichotomy approach, but is concerned solely with polynomials.

This work on Bernstein polynomials combines techniques coming from analysis, algebra and geometry. For instance, the properties of reversing the list of coefficients of a polynomial are studied by looking at the polynomial as a function from rational numbers to rational numbers. Similarly, the proof of Descartes' law of signs works by looking at functions and bounds on their values in various intervals. On the other hand, the definition of Bernstein coefficients relies on concepts that come from linear algebra: vector spaces, bases and morphisms. Finally, de Casteljau's algorithm relies on geometry with midpoints or segments. It is particularly exciting that we can now study mathematical algorithms formally that use all these aspects of mathematics.

This development is not made just for the beauty of it. The initial goal is to provide one of the basic building blocks required for cylindrical algebraic decomposition (Basu *et al.* 2006; Mahboubi 2007). In the short term, we want to develop this into a full algorithm for isolating the roots of an arbitrary polynomial. This involves proving the technique to reduce the multiplicity of roots that we already described in the introduction, initialising the search for roots with an interval large enough to contain all the roots, programming the recursive dichotomy process and proving that this process always terminates.

For the proof of termination, we already know a mathematical argument, described in Basu *et al.* (2006) under the name 'theorem of three circles'. However, this theorem uses arguments based on complex numbers and we wish to find a more elementary proof, as we still want to express our result using mainly rational numbers. Our proof of the law of signs already is more elementary than those found in the literature.

In the long run, a good knowledge of Bernstein polynomials and coefficients opens the door to a wide variety of tools that are commonplace in computer aided design and robotics. Bézier curves, which are often used in drawing tools, share a lot of properties with Bernstein control points. Splines and Bézier curves are also used in the field of robotics to describe the trajectory of moving vehicles. Thus, this work may eventually be useful for the formal verification of critical software in robotics.

Acknowledgments

The authors wish to thank the anonymous referees who have suggested numerous improvements to both the formalisation and its description.

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press

References

Basu, S., Pollack, R. and Roy, M.-F. (2006) *Algorithms in Real ALgebraic Geometry*, second edition, Algorithms and Computations in Mathematics **10**, Springer-Verlag.

Bertot, Y. and Castéran, P. (2004) *Interactive Theorem Proving and Program Development, Coq'Art: the Calculus of Inductive Constructions*, Springer-Verlag.

Bertot, Y., Gonthier, G., Ould Biha, S. and Paşca, I. (2008) Canonical Big Operators. In: Mohamed, O. A., Muñoz, C. and Tahar, S. (eds.) Proceedings of the 21st International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2008). *Springer-Verlag Lecture Notes in Computer Science* **5170** 12–16. (Available at `http://hal.inria.fr/inria-00331193/`.)

Bézier, P. (1986) *Courbes et Surfaces*, Hermès.

Cohen, C. (2010) Formalizing real analysis for polynomials. Technical report, Inria. (Available at `http://hal.inria.fr/inria-00545778/en/`.)

Cruz-Filipe, L., Geuvers, H. and Wiedijk, F. (2004) C-CoRN, the Constructive Coq Repository at Nijmegen. In: Asperti, A., Bancerek, G. and Trybulec, A. (eds.) Proceedings Mathematical Knowledge Management, Third International Conference, MKM 2004. *Springer-Verlag Lecture Notes in Computer Science* **3119** 88–103.

de Castleljau, P. (1985) *Formes à pôles*, Hermès.

Descartes, R. (1969) *Géométrie (1636) A source book in Mathematics*, Harvard University Press.

Gonthier, G. and Mahboubi, A. (2008) A Small Scale Reflection Extension for the Coq system. Research Report RR-6455, INRIA. (Available at `http://hal.inria.fr/inria-00258384/en/`.)

Gonthier, G. and Mahboubi, A. (2010) An introduction to small scale reflection in Coq. Technical report, INRIA.

Gonthier, G., Mahboubi, A., Rideau, L., Tassi, E. and Théry, L. (2007) A modular formalisation of finite group theory. In: Schneider, K. and Brandt, J. (eds.) Proceedings Theorem Proving in Higher Order Logics, 20th International Conference, TPHOLs 2007. *Springer-Verlag Lecture Notes in Computer Science* **4732** 86–101.

Knuth, D. (1986) *Metafont: the Program*, Addison Wesley.

Mahboubi, A. (2007) Implementing the cylindrical algebraic decomposition within the Coq system. *Mathematical Structures in Computer Science* **17** (1) 99–127.

Mourrain, B., Rouillier, F. and Roy, M.-F. (2005) *Bernstein's basis and real root isolation*, Mathematical Sciences Research Institute Publications.

O'Connor, R. (2007) A monadic, functional implementation of real numbers. *Mathematical Structures in Computer Science* **17** (1) 129–159.

O'Connor, R. (2008) Certified exact transcendental real number computation in Coq. In: Mohamed, O. A., Muñoz, C. and Tahar, S. (eds.) Proceedings of the 21st International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2008). *Springer-Verlag Lecture Notes in Computer Science* **5170** 246–261.

Paşca, I. (2011) Formal proofs for theoretical properties of Newton's method. *Mathematical Structures in Computer Science* **21** (4) 683–714.

Rouillier, F. and Zimmermann, P. (2003) Efficient isolation of polynomial real roots. *Journal of Computational and Applied Mathematics* **162** (1) 33–50.

Troelstra, A. S. and van Dalen, D. (1988) *Constructivism in Mathematics, an introduction* **1**, North-Holland.

Uspensky, J. V. (1948) *Theory of Equations*, MacGraw-Hill.

Zumkeller, R. (2008) *Global Optimization in Type Theory*, Ph.D. thesis, École Polytechnique.

https://doi.org/10.1017/S0960129511000090 Published online by Cambridge University Press