The Positive Dominance Algorithm

Richard Evan Schwartz *

August 10, 2014

1 Introduction

In this note I will explain a computational algorithm which I call the *positive dominance algorithm*. The input is a polynomial $F \in \boldsymbol{R}[X_1, ..., X_n]$ and a polytope $P \subset \boldsymbol{R}^n$. One version of the algorithm tries to verify that $F > 0$ on P. This version halts if and only if the the assertion is true.

Another version of the algorithm tries to verify that $F \geq 0$ on P. This case is more interesting, because it allows for sharper applications. In this case, we have $F \geq 0$ on P if the algorithm halts, but the converse is not necessarily true. The algorithm will probably fail if $F = 0$ on some interior points of the polytope.

The algorithms require that P has some triangulation into simplices, so we will restrict our attention to the case when P is a simplex. In case F is a rational polynomial and P is a rational simplex, the algorithm can be implemented with exact rational arithmetic. Another useful case happens when both P and F are defined over the same number field. Here, again, the algorithm can be implemented using exact arithmetic.

I have used the algorithm in two diverse situations. In one situation [**S1**], I proved some inequalities converning the Cayley-Menger determinant on the space of tetrahedra, and in another situation [**S2**] I used it to analyze the Julia set of a high degree rational map of 2 variables. I'm sure it has many other applications. After presenting the algorithm, I'll discuss the geometric applications I have in mind.

* Supported by N.S.F. Research Grant DMS-1204471

2 Single Variable Case

2.1 Positive Dominance

I will concentrate on the ($\geq$) case first, and explain the changes needed for the ($>$) case at the end. As a warmup, we consider the situation for polynomials in a single variable. Let

$$F(x) = a_0 + a_1 x + ... + a_n x^n \tag{1}$$

be a polynomial with real coefficients. Here we describe a method for showing that $F \geq 0$ on $[0,1]$,

Define

$$A_k = a_0 + \cdots + a_k. \tag{2}$$

We call F *positive dominant* (or *PD* for short) if $A_k \geq 0$ for all k.

Lemma 2.1 *Suppose F is positive dominant. Then $F \geq 0$ on $[0,1]$. Moreover, if F is positive dominant and nontrivial, then $F > 0$ on $(0,1)$.*

Proof: The proof goes by induction on the degree of F. The case $\deg(F) = 0$ follows from the fact that $a_0 = A_0 \geq 0$. Let $x \in [0,1]$. We have

$$F(x) = a_0 + a_1 x + x_2 x^2 + \cdots + a_n x^n \geq$$

$$a_0 x + a_1 x + a_2 x^2 + \cdots + a_n x^n =$$

$$x(A_1 + a_2 x + a_3 x^2 + \cdots a_n x^{n-1}) = xG(x) \geq 0$$

Here $G(x)$ is positive dominant and has degree $n-1$.

Examining the proof, we see that the only way we could have $F(x) = 0$ for some $x \in (0,1)$ is if $G(x) = 0$ as well. So, the statement that $F > 0$ on $(0,1)$ also follows from induction. ♠

Remark: If we had the stronger hypothesis that $A_k > 0$ for all k, then the proof in Lemma 2.1 could be modified to show that $F > 0$ on $[0,1]$.

2.2 The Algorithm

Given a polynomial F, we define

$$F_0(x) = F(x/2), \qquad F_1(x) = F(1 - x/2). \tag{3}$$

Lemma 2.2 *$F \geq 0$ on $[0,1]$ iff $F_0 \geq 0$ on $[0,1]$ and $F_1 \geq 0$ on $[0,1]$.*

Proof: Let $A_0(x) = x/2$ and $A_1(x) = 1 - x/2$. Then A_0 is a bijection from $[0,1]$ to $[0,1/2]$ and A_1 is a bijection from $[0,1]$ to $[1/2,1]$. So, $F \geq 0$ on $[0,1/2]$ iff $F_0 \geq 0$ on $[0,1]$ and $F \geq 0$ on $[1/2,1]$ iff $F_1 \geq 0$ on $[0,1]$. ♠

The polynomial pair $\{F_0, F_1\}$ is defined to be the *subdivision* of F. The basic idea behind our technique is that the subdivision process tends to increase the changes that a polynomial is positive dominant, because it tends to put more weight on the earlier terms in the polynomial. With this in mind, we define the basic algorithm.

The Positive Dominant Algorithm: Here is a recursive algorithm which tries to show that a polynomial F is non-negative on $[0,1]$.

1. Start with a list LIST of polynomials. Initially LIST=$\{F\}$.

2. Let G be the last polynomial on LIST. We delete G from LIST and then test whether G is PD.

3. Suppose G is PD. We go back to Step 2 if LIST is nonempty, and otherwise halt.

4. Suppose G is not PD. We append to LIST the two polynomials G_0 and G_1 obtained by subdividing G, then go back to Step 2.

Remark: In our definition of the subdivision, we might have used the formula $F_1'(x) = F(1/2+x/2)$ instead. This does not work as well. For instance, if $F(x) = (x-1)^2$ then $F_1'(x) = F(x)/4$. So, with this input, the algorithm would run forever.

2.3 The Power of the Algorithm

Now we discuss what it means when the algorithm halts. We always assume that our polynomials are nontrivial.

Lemma 2.3 *If the positive dominance algorithm halts for F, then $F \geq 0$ on $[0,1]$ and all the roots of F in $[0,1]$ are dyadic rationals.*

Proof: Suppose first that the algorithm halts. Then we have a partition of $[0,1]$ into intervals $I_1, ..., I_n$, together with affine maps $A_k : [0,1] \to I_k$ so that $F \circ A_k$ is positive dominant for all k. But then $F > 0$ on the interior of A_k for all k. Also, by construction, the endpoints of A_k are dyadic rational. Hence, the only roots of F in $[0,1]$ lie in dyadic rationals. ♠

The converse is also true. We will build up to the converse result in stages.

Lemma 2.4 *Suppose that $F > 0$ on $[0,1]$ then there is some $\epsilon > 0$ with the following property: If $I \subset [0,1]$ is any interval whose length is less than ϵ, and A is one of the two affine isomorphisms from $[0,1]$ to I, then $F \circ A$ is positive dominant.*

Proof: Since $F > 0$ on $[0,1]$, we have $F = a_0 + a_1 x + ...$ where $a_0 > 0$. There is some $\epsilon > 0$ so that

$$a_0 > \epsilon(|a_1 + ... + |a_n|). \tag{4}$$

The affine map A has the form

$$A(x) = c_1 x + c_2,$$

where $|c_1| < |I|$, the length of I. But then, if $|I| < \epsilon$, we have $F \circ A = b_0 + b_1 x + ...$ where

$$b_0 > |b_1| + ... + ||b_n|.$$

Such a polynomial is positive dominant. ♠

Corollary 2.5 *If $F > 0$ on $[0,1]$ then the positive dominant algorithm halts for F.*

Proof: We can think of our algorithm as taking place on the level of intervals. For each function G on LIST, there is some interval I and some affine map $A : [0,1] \to I$ so that $G = F \circ A$. There is also the constant $\epsilon > 0$ from the preceding result. By the preceding result, LIST will pass any polynomial whose associated interval has length less than ϵ. Hence LIST will only contain finitely many polynomials. Hence the algorithm halts. ♠

Lemma 2.6 *If $F > 0$ on $(0, 1]$ then the algorithm halts for F.*

Proof: There is some k so that $F(x) = x^k G(x)$ where $G > 0$ in $[0, 1]$. The algorithm runs the same for F and for G. Since the algorithm halts for F it also halts for G. ♠

Lemma 2.7 *If $F > 0$ on $(0, 1)$ then the algorithm halts for F.*

Proof: Let $\{F_0, F_1\}$ be the subdivision of F. By construction, $F_j > 0$ on $(0, 1]$. Hence, the algorithm halts for F_0 and F_1. Hence, the algorithm halts for both functions together. ♠

Finally, we come to the main result.

Lemma 2.8 *The positive dominance algorithm halts on F if and only if $F \geq 0$ on $[0, 1]$ and if the roots of F are dyadic rationals.*

Proof: We have already proved one direction. So, suppose that $F \geq 0$ on $[0, 1]$ and the roots of F are all dyadic rationals. There exists a partition $\mathcal{J}$ of I into dyadic intervals such that $F > 0$ on the interior of each interval. The algorithm does a depth-first-search through the tree of dyadic intervals. If it runs forever, it eventually produces an infinite list of intervals all contained within a single interval of $\mathcal{J}$. But this contradicts the preceding results: The algorithm will halt on a polynomial associated to a sub-interval of an interval of $\mathcal{J}$. ♠

2.4 Variants

Positivity: There is a variant of the algorithm in which we test for strict positivity. In this case, we test whether all the sums in Equation 2 are positive. The results above show that the "positive version" of the algorithm converges if and only if $F > 0$ on $[0, 1]$.

Alternate Subdivision Scheme: The polynomial $F(x) = (x - 1/3)^2$ is the simplest polynomial which defeats the algorithm. This polynomial is non-negative on $[0, 1]$ but the algorithm runs forever. The problem is that the root of F is not a dyadic rational. One could imagine re-running the algorithm based on a different subdivision rule, so as to pick additional roots. For instance, if one uses *Farey subdivision* rather than dyadic subdivision, the algorithm halts if and only if $F \geq 0$ on $[0, 1]$ and all roots are rational.

Detecting Negativity: Here is a variant of the algorithm which will also halt (declaring failure) if $F(x) < 0$ for some $x \in [0, 1]$. The variant is the same as the original, except that:

- At Step 2, we also test whether $G(0) < 0$, and halt if so.
- At Step 4 we prepend G_0 and G_1 to LIST rather than append them.

This version does a breadth first search (rather than a depth first search) through the tree of dyadic intervals. So, it will find points of negativity eventually. Thus, if $F(x) < 0$ for some $x \in [0, 1]$, the above algorithm will halt (with failure). The results about positivity are the same: The algorithm will halt with success if and only if $F \geq 0$ on $[0, 1]$ and all the roots are dyadic rationals.

3 The General Case

3.1 Positive Dominance

Now we go to the higher dimensional case. We consider real polynomials in the variables $x_1, ..., x_k$. Given a multi-index $I = (i_1, ..., i_k) \in (\boldsymbol{N} \cup \{0\})^k$ we let

$$x^I = x_1^{i_1} ... x_k^{i_k}. \tag{5}$$

Any polynomial $F \in \boldsymbol{R}[x_1, ..., x_k]$ can be written succinctly as

$$F = \sum A_I X^I, \qquad A_I \in \boldsymbol{R}. \tag{6}$$

If $I' = (i'_1, ..., i'_k)$ we write $I' \le I$ if $i'_j \le i_j$ for all $j = 1, ..., k$. We call F *positive dominant* if

$$\sum_{I' \le I} A_{I'} \ge 0 \qquad \forall I, \tag{7}$$

Lemma 3.1 *If F is positive dominant then $F \ge 0$ on $[0,1]^k$. If F is nontrivial and positive dominant, then $F > 0$ on $(0,1)^k$.*

Proof: The 1 variable case is Lemma 2.1. In general, we write

$$F = f_0 + f_1 x_k + ... + f_m x_k^m, \qquad f_j \in \boldsymbol{R}[x_1, ..., x_{k-1}]. \tag{8}$$

Let $F_j = f_0 + ... + f_j$. Since F is positive dominant, we get that F_j is positive dominant for all j. By induction on k, we get $F_j \ge 0$ on $[0,1]^{k-1}$. But now, if we hold $x_1, ..., x_{k-1}$ fixed and let $t = x_k$ vary, the polynomial $g(t) = F(x_1, ..., x_{k-1}, t)$ is positive dominant.. Hence, by Lemma 2.1, we get $g \ge 0$ on $[0,1]$. Hence $F \ge 0$ on $[0,1]^k$.

If F is nontrivial, then there is some point $p \in (0,1)^k$ where $f(p) > 0$. But then, in Equation 8, at least one f_j is a nontrivial polynomial. But then, by induction on k, the function f_j is positive on $(0,1)^{k-1}$. But then the polynomial $g(t)$ nontrivial and positive dominant for each choice of $x_1, ..., x_{k-1} \in (0,1)^{k-1}$. But then $g(t) > 0$ on $(0,1)$. This proves what we want. ♠

3.2 The Algorithm

We can perform the same kind of divide-and-conquer algorithm as in the 1-dimensional case. We always take our domain to be $[0,1]^k$. Let P be a polynomial. We define the kth subdivision of F to be the pair of polynomials $\{F_0, F_1\}$, where

$$F_{k,0}(x_1, ..., x_n) = F(x_1, ..., (x_k/2), ..., x_n),$$
$$F_{k,1}(x_1, ..., x_n) = F(x_1, ..., (1 - x_k/2), ..., x_n). \tag{9}$$

The same argument as in the previous section shows that $F \geq 0$ on $[0,1]^n$ if and only if $F_0 \geq 0$ on $[0,1]^n$ and $F_1 \geq 0$ on $[0,1]^n$.

Now we will build up to the general version of our subdivision algorithm. Say that a *marker* is a non-negative integer vector in $\boldsymbol{R}^k$. Say that the *youngest entry* in the the marker is the first minimum entry going from left to right. The *successor* of a marker is the marker obtained by adding one to the youngest entry. For instance, the successor of $(2,2,1,1,1)$ is $(2,2,2,1,1)$. Let μ_+ denote the successor of μ.

We say that a *marked polynomial* is a pair (F, μ), where F is a polynomial and μ is a marker. Let k be the position of the youngest entry of μ. We define the *subdivision* of (F, μ) to be the pair

$$\{(F_{k,1}, \mu_+, (F_{k,2}, \mu_+)\}. \tag{10}$$

Geometrically, we are cutting the domain in half along the longest side, and using a particular rule to break ties when they occur.

Positive Dominance Algorithm:

1. Start with a list LIST of marked polynomials. Initially, LIST consists only of the marked polynomial $(F, (0, ..., 0))$.

2. Let (G, μ) be the last element of LIST. We delete (G, μ) from LIST and test whether G is positive dominant.

3. Suppose G is positive dominant. we go back to Step 2 if LIST is not empty. Otherwise, we halt.

4. Suppose G is not positive dominant. we append to LIST the two marked polynomials in the subdivision of (G, μ) and then go to Step 2.

We call F *Recursively Positive Dominant* or (RPD) if the positive dominance algorithm halts for F. If F is RPD then $F \geq 0$ on $[0,1]^k$.

3.3 The Limitations of the Algorithm

The converse results are not as strong as in the one dimensional case. One result which goes through easily is:

Lemma 3.2 *If $F > 0$ on $[0,1]^n$ then the positive dominant algorithm halts on F.*

Proof: Essentially the same as in the 1-dimensional case. ♠

When F is only non-positive on $[0,1]^n$, the situation is much trickier. The polynomial

$$F(x,y) = (x-y)^2 \tag{11}$$

defeats the algorithm because $F_0 = 1/4F$, and then $F_{00} = 1/16F$, etc. One might expect problems with this function because it vanishes along the diagonal in $\boldsymbol{R}^2$.

Here is a related example. The function

$$F(x,y) = (x-y)^2 + x^4 + y^4 \tag{12}$$

is positive on $\boldsymbol{R}^2 - (0,0)$ but defeats the algorithm.

Here is another general source of failure in the non-negative case. The algorithm has no chance of working if $F \geq 0$ on $[0,1]^n$ and the variety $\{F = 0\}$ intersects $[0,1]^n$ in a non-linear set. The problem is that one cannot completely eliminate polynomials which vanish at points in $(0,1)^n$. Such polynomials cannot be positive dominant, by Lemma 3.1.

A simple example of the problematic kind of polynomial we are taking about is:

$$F(x,y) = (x^2 + y^2 - 1)^2. \tag{13}$$

This function vanishes on the unit circle and otherwise is positive.

Some experience shows that the algorithm has a pretty good chance of working (the example in Equation 12 notwithstanding) provided that it vanishes only along linear subsets contained in the boundary of $[0,1]^n$. The example in Equation 12 vanishes "up to higher order" on the diagonal $x = y$, which intersects the interior of $[0,1]^2$, and this is what causes the problem. However, I don't know a general criterion which describes exactly when the algorithm halts.

3.4 Variants

Detecting Negativity: One can run the same variants as discussed in the one dimensional case, in order to detect negativity.

Multiple Functions: One can also run a version of the algorithm which works with a finite number of functions at the same time.

1. Let LIST denote a list of k-tuples of polynomial pairs. Initially LIST has the one member $(\Pi_1, ..., \Pi_k)$. Here $\Pi_j = (F_j, (0, ..., 0))$.
2. Let α be the last member of LIST. Delete α from LIST and test whether at least one member of α is positive dominant.
3. If some member of α is positive dominant and LIST is nonempty, return to step 2. Otherwise halt.
4. If no member of α is positive dominant, append to LIST the two k-tuples α_0 and α_1. Here α_0 is obtained by replacing each member of α by its 0th subdivision, and likewise α_1. Then go to Step 2.

If this algorithm halts, it produces a partition of $[0, 1]^n$ into sets $S_1, ..., S_k$ with the property that the restriction $f_j|S_j$ is non-negative for each $j = 1, .., k$. I will give a geometric application of this version below.

4 The Cube Transform

4.1 Main Definition

So far we have discussed how we show $F \geq 0$ on $[0,1]^k$. But now we want to explain how we deal with simplices.

Let $S_n \subset \boldsymbol{R}^n$ denote the simplex

$$\{(x_1, ..., x_n)|\ 1 \geq x_1 \geq \cdots \geq x_n \geq 0\}. \tag{14}$$

This simplex is the convex hull of the vectors

$$(0, ..., 0), \quad (1, 0, ..., 0), \quad (1, 1, 0, ..., 0), \cdots, \quad (1, ..., 1).$$

There is a surjective polynomial map $U : [0,1]^n \to S_n$ given by

$$U(x_1, ..., x_n) = \Big((x_1), (x_1x_2), (x_1x_2x_3), ..., (x_1...x_n)\Big). \tag{15}$$

Let $\Delta_n \subset \boldsymbol{R}^{n+1}$ denote the regular n-simplex in $\boldsymbol{R}^{n+1}$ consisting of the convex hull of the standard basis vectors. That is, Δ_n consists of points $(x_1, ..., x_{n+1})$ such that $x_j \geq 0$ for all j and $\sum x_j = 1$. There is an affine isomorphism from S_n to Δ_n:

$$V(x_1, ..., x_n) = (1 - x_1, x_1 - x_2, x_2 - x_3, ..., x_{n-1} - x_n, x_n). \tag{16}$$

The easiest way to see that this works is to check it on the vertices of S_n.

Finally, let Σ denote an n-simplex in some Euclidean space. We form a matrix W_Σ whose columns are the vertices of Σ.

We define the *cube transform* of the pair (F, Σ) to be the composition

$$F_\Sigma = F \circ W_\Sigma \circ V \circ U. \tag{17}$$

By construction, $F \geq 0$ on Σ provided that $F_\Sigma \geq 0$ on $[0,1]^n$. So, if we want to decide if some polynomial F is positive on some simplex Σ, we use the positive dominance algorithm to show that $F_\Sigma \geq 0$ on $[0,1]^n$.

Integrality: If Σ is a simplex with integer vertices, then W_Σ is represented by an integer matrix. Thus, if (F, Σ) are both defined over $\boldsymbol{Z}$, so is F_Σ. Similarly, if F and Σ are defined over a number ring (respectively field), then F_Σ is defined over the same number ring (respectively field).

4.2 Alternate Subdivision Schemes

One can combine the cube transform and the notion of positive dominance in a different way. Let F be a polynomial and let T be a triangle. We say that the pair (F,T) is positive dominant if F_T is positive dominant. Here F_T is the cube transform of F with respect to T. That is, the positive dominance algorithm halts immediately for the polynomial F_T. To show that $F \geq 0$ on T we could apply the following algorithm.

1. Let LIST denote a list of pairs (G,U) where G is a polynomial and U is a triangle. Initially, LIST=$\{(F,T)\}$.

2. Let $\{G,U\}$ be the last member of LIST. We delete (G,U) from LIST and test whether this pair if positive dominant.

3. Suppose (G,U) is positive dominant. If LIST is not empty we return to Step 2. Otherwise halt.

4. If (G,U) is not positive dominant, replace (G,U) by the 4 pairs

$$(G,U_1),...,(G,U_4),$$

   where $\{U_j\}$ is some subdivision of U. Then return to Step 2.

If the algorithm halts, it means that $F \geq 0$ on T. During the algorithm, we are subdividing the triangle repeatedly but keeping the polynomial the same. Of course, when we take the cube transform of F with respect to different triangles we get different polynomials.

I took this alternate approach in [**S2**]. I used the obvious subdivision scheme, where each triangle is broken into 4 congruent triangles all having have the size, and the same shape, as the original. In higher dimensions, one would need a convenient subdivision scheme for a simplex in order to take this alternate approach. In the variant where we test for strict positivity, we again have the result that the algorithm halts if and only $F > 0$ on T.

In the case of non-negativity, I'm not sure whether the approach outlined in this section is better or worse than the the original approach. Probably the approach here works better for some functions and worse for others.

5 Rational Maps and Polytopes

Now I will give some geometric applications of the positive dominance algorithm. Really, these are applications of any algorithm which can certify that a polynomial is non-negative or positive on a polyhedron.

5.1 The Confinement Test

Let $\Phi : \boldsymbol{R}^n \to \boldsymbol{R}^n$ be a rational map, let $X \subset \boldsymbol{R}^n$ be a polytope, and let $Y \subset \boldsymbol{R}^n$ be a convex polytope. In this section we will explain how to verify that $\Phi(X) \subset Y$.

The building block for the general method is the situation when Y is a half-space. That is, there is some linear functional $L : \boldsymbol{R}^n \to \boldsymbol{R}$ such that $L(p) \geq 0$ if and only if $p \in Y$. The function $L \circ \Phi$ is a rational function on $\boldsymbol{R}^n$. Let's write

$$L \circ \Phi = F/G, \tag{18}$$

where F and G are polynomials. We normalize so that $G > 0$ at some point in X. If $P \geq 0$ on X and $G > 0$ on X, then $\Phi(X) \subset Y$.

Any convex polytope is the intersection of a finite union of half-spaces. So, we may apply the above test finitely many times in order to show that $\Phi(X) \subset Y$. If Y is an open polytope we can perform the same tests, replacing $F \geq 0$ with $F > 0$.

5.2 The Exclusion Test

There is a variant of the test above, in which we try to show that $\Phi(X)$ is disjoint from Y. Consider the case When X is a simplex. Let $L_1, .., L_k$ denote linear functionals defining the faces of Y. We write

$$L_j \circ \Phi = F_j/G_j. \tag{19}$$

We normalize so that $G_j > 0$ on some point of X, for each $j = 1, ..., k$. We first try to show that $G_j > 0$ on X for all j. Next, we perform a the variant of the algorithm discussed in §3.4.

If the algorithm halts, it means that we have a partition of X into k piecesm say $X = X_1 \cup ... \cup X_k$ so that $L_j \circ \Phi < 0$ on X_j. But then the jth side of Y separates $\Phi(X_j)$ from Y. Hence $\Phi(X)$ is disjoint from Y.

The Exclusion Test is more versatile than the Confinement Test. Suppose that we want to verify that $\Phi(X) \subset Y$ where Y is some complicated set, say a smooth compact manifold with boundary. We could cover ∂Y by some finite union of of convex polytopes, and then use the Exclusion Test repeatedly to show that $\Phi(X)$ is disjoint from the covering. This would prove that $\Phi(X)$ is disjoint from ∂Y. We then would check that $\Phi(p) \in Y$ for some $p \in X$. We could then conclude that $\Phi(X) \subset Y$. Similarly, if we had $\Phi(p) \notin Y$ for some $p \in X$ then we could conclude that $\Phi(X)$ is disjoint from Y.

6 References

[**S1**] R. Schwartz, *Lengthening a Tetrahedron*, preprint 2014
[**S2**] R. Schwartz, *The Projective Heat Map on Pentagons*, preprint 2014