Algorithms for Weighted Sum of Squares Decomposition of Non-negative Univariate Polynomials

Victor Magron

CNRS Verimag; 700 av Centrale 38401 Saint-Martin d'Hères, France

Mohab Safey El Din

Sorbonne Université, CNRS, Inria, Laboratoire d'Informatique de Paris 6, Équipe PolSys, France

Markus Schweighofer

Fachbereich Mathematik und Statistik, Universität Konstanz, 78457 Konstanz, Germany

---

Abstract

It is well-known that every non-negative univariate real polynomial can be written as the sum of two polynomial squares with real coefficients. When one allows a (non-negatively) *weighted* sum of *finitely many* squares instead of a sum of two squares, then one can choose all coefficients in the representation to lie in the field generated by the coefficients of the polynomial. In particular, this allows for an effective treatment of polynomials with rational coefficients.

In this article, we describe, analyze and compare, from both the theoretical and practical points of view, two algorithms computing such a weighted sum of squares decomposition for univariate polynomials with rational coefficients.

The first algorithm, due to the third author, relies on real root isolation, quadratic approximations of positive polynomials and square-free decomposition, but its complexity was not analyzed. We provide bit complexity estimates, both on the runtime and the output size of this algorithm. They are exponential in the degree of the input univariate polynomial and linear in the maximum bitsize of its complexity. This analysis is obtained using quantifier elimination and root isolation bounds.

The second algorithm, due to Chevillard, Harrison, Joldes and Lauter, relies on complex root isolation and square-free decomposition, and was introduced for certifying positiveness of polynomials in the context of computer arithmetic. Again, its complexity was not analyzed. We provide bit complexity estimates, both on the runtime and the output size of this algorithm, which are polynomial in the degree of the input polynomial and linear in the maximum bitsize of its complexity. This analysis is obtained using Vieta's formula and root isolation bounds.

Finally, we report on our implementations of both algorithms and compare them in practice on several application benchmarks. While the second algorithm is, as expected from the complexity result, more efficient on most of examples, we exhibit families of non-negative polynomials for which the first algorithm is better.

Keywords: Non-negative univariate polynomials, Nichtnegativstellensätze, sum of squares decomposition, root isolation, real algebraic geometry.

---

1. Introduction

Given a subfield K of $\mathbb{R}$ and a non-negative univariate polynomial $f \in K[X]$, we consider the problem of proving the existence of, and computing, weighted sum of squares decompositions of f with coefficients also lying in K, i.e., $a_1, \ldots, a_r \in K^{\geq 0}$ and $g_1, \ldots, g_r \in K[X]$ such that $f = \sum_{i=1}^{r} a_i g_i^2$.

Beyond the theoretical interest of this question, finding certificates of non-negative polynomials is mandatory in many application fields. Among them, one can mention the stability proofs of critical control systems often relying on Lyapunov functions ([33]), the certified evaluation of mathematical functions in the context of computer arithmetics (see for instance [5]), the formal verification of real inequalities ([12]) within proof assistants such as Coq ([7]) or Hol-light ([13]); in these situations the univariate case is already an important one. In particular, formal proofs of polynomial non-negativity can be handled with weighted sum of squares certificates. These certificates are obtained with tools available outside of the proof assistants and eventually verified inside. Because of the limited computing power available inside such proof assistants, it is crucial to devise algorithms that produce certificates, whose checking is computationally reasonably simple. In particular, we would like to ensure that such algorithms output weighted sum of squares certificates of moderate bitsize and ultimately with a computational complexity being polynomial with respect to the input.

Related Works. Decomposing non-negative univariate polynomials into weighted sums of squares has a long story; very early quantitative aspects like the number of needed squares have been studied. For the case $K = \mathbb{Q}$, Landau showed in [21] that for every non-negative polynomial in $\mathbb{Q}[X]$, there exists a decomposition involving a weighted sum of (at most) eight polynomial squares in $\mathbb{Q}[X]$. In [31], Pourchet improves this result by showing the existence of a decomposition involving only a weighted sum of (at most) five squares. This is done using approximation and valuation theory; extracting an algorithm from these tools is not the subject of study of this paper.

More recently, the use of semidefinite programming for computing weighted sum of squares certificates of non-negativity for polynomials has become very popular since [22, 29]. Given a polynomial f of degree n, this method consists in finding a real symmetric matrix G with non-negative eigenvalues (a *positive semidefinite* matrix) such that $f(x) = v(x)^T G v(x)$, where v is the vector of monomials of degree less than $n/2$. Hence, this leads to the problem of solving a so-called Linear Matrix Inequality, and one can rely on semidefinite programming (SDP) to find the coefficients of G. This task can be delegated to an SDP solver (e.g., SeDuMi, SDPA, SDPT3). An important technical issue arises from the fact that such SDP solvers are most of the time implemented with floating-point double precision. More accurate solvers are available (e.g., SDPA-GMP [28]). However, these solvers always compute numerical approximations of the algebraic solution to the semidefinite program under consideration. Hence, they are not sufficient to provide algebraic certificates of posivity with rational coefficients. Hence, a process is needed to replace the computed numerical approximations of a sum of squares certificate by an exact, weighted sum of squares certificate with all weights and coefficients rational. This issue

Email addresses: victor.magron@univ-grenoble-alpes.fr (Victor Magron), Mohab.Safey@lip6.fr (Mohab Safey El Din), markus.schweighofer@uni-konstanz.de (Markus Schweighofer)

URL: http://www-verimag.imag.fr/~magron/ (Victor Magron), http://www-polsys.lip6.fr/~safey/ (Mohab Safey El Din), http://www.math.uni-konstanz.de/~schweigh/ (Markus Schweighofer)

was tackled in [30, 19]. The certification scheme described in [24] allows one to obtain lower bounds of non-negative polynomials over compact sets. However, despite their efficiency, there is no guarantee that these methods will output a rational solution to a Linear Matrix Inequality when it exists (and especially when it is far from the computed numerical solution).

A more systematic treatment of this problem has been brought by the symbolic computation community. Linear Matrix Inequalities can be solved as a decision problem over the reals with polynomial constraints using the Cylindrical Algebraic Decomposition algorithm [6] or more efficient critical point methods (see e.g., [1] for complexity estimates, and see [17, 10] for practical algorithms). But using such general algorithms is overkill, and, dedicated algorithms have been designed for computing exact algebraic solutions to Linear Matrix Inequalities [14, 15]. Computing rational solutions can also be considered, thanks to convexity properties [34]. In particular, the algorithm in [11] can be used to compute weighted sum of squares certificates with rational coefficients for a non-negative univariate polynomial of degree n with coefficients of bitsize bounded by τ using at most $\tau^{O(1)}2^{O(n^3)}$ boolean operations at (see [11, Theorem 1.1]). In [3], the authors derive positivity certificates of polynomials positive over $[-1, 1]$ in the Bernstein basis. This certificate allows one in turn to produce a Positivstellensatz identity of total bitsize bounded by $O(n^4 \log n + n^4\tau)$, thus polynomial in n and τ (see [3, Theorem 8]). To the best of our knowledge, there is no available implementation of this method.

For the case where K is an arbitrary subfield of $\mathbb{R}$, Schweighofer gives in [35] a new proof of the existence of a decomposition involving a sum of (at most) n polynomial squares in $K[X]$. This existence proof comes together with a recursive algorithm to compute such decompositions. At each recursive step, the algorithm performs real root isolation and quadratic approximations of positive polynomials. Later on, a second algorithm is derived in [5, Section 5.2], where the authors show the existence of a decomposition involving a sum of (at most) $n + 3$ polynomial squares in $K[X]$. Note that this second algorithm was presented earlier in [18, Section 7] (albeit with less detail and without a pointer to the code). This algorithm is based on approximating complex roots of perturbed positive polynomials.

Neither of these latter algorithms were analyzed, despite the fact that they were implemented and used. An outcome of this paper is a bit complexity analysis for both of them, showing that they have better complexities than the algorithm in [11], the second algorithm being polynomial in n and τ.

Notation for complexity estimates. For complexity estimates, we use the bit complexity model. For an integer $b \in \mathbb{Z}\backslash\{0\}$, we denote by $\tau(b) := \lfloor \log_2(|b|) \rfloor + 1$ the bitsize of b, with the convention $\tau(0) := 1$. We write a given polynomial $f \in \mathbb{Z}[X]$ of degree $n \in \mathbb{N}$ as $f = \sum_{i=0}^n b_i X^i$, with $b_0, \dots, b_n \in \mathbb{Z}$. In this case, we define $\|f\|_\infty := \max_{0 \le i \le n} |b_i|$ and, using a slight abuse of notation, we denote $\tau(\|f\|_\infty)$ by $\tau(f)$. Observe that when f has degree n, the bitsize necessary to encode f is bounded by $n\tau(f)$ (when storing the coefficients of f). The derivative of f is $f' = \sum_{i=1}^n i b_i X^{i-1}$. For a rational number $q = \frac{b}{c}$, with $b \in \mathbb{Z}, c \in \mathbb{Z}\backslash\{0\}$ and $\gcd(b, c) = 1$, we denote $\max\{\tau(b), \tau(c)\}$ by $\tau(q)$. For two mappings $g, h : \mathbb{N}^l \to \mathbb{R}^{>0}$, the expression "$g(v) = O(h(v))$" means that there exists an integer $b \in \mathbb{N}$ and $N \in \mathbb{N}$ such that when all coordinates of v are greater than or equalled to N, $g(v) \le bh(v)$. The expression "$g(v) = \widetilde{O}(h(v))$" means that there exists an integer $c \in \mathbb{N}$ such that for all $v \in \mathbb{N}^l$, $g(v) = O(h(v)(\log(h(v))^c)$.

Contributions. We present and analyze two algorithms, denoted by `univsos1` and `univsos2`, allowing one to decompose a non-negative univariate polynomial f of degree n with coefficients lying in any subfield K of $\mathbb{R}$ into a sum of squares. To the best of our knowledge, there was no prior complexity estimate for the output of such certification algorithms based on sums of squares in the univariate case. We summarize our contributions as follows:

- We describe in Section 3 the first algorithm, called `univsos1` in the sequel. It was originally given in [35, Chapter 2]; Section 3 can be seen as a partial English translation of this German text, since some proofs have been significantly simplified. In the same section, we analyze its bit complexity. When the input is a polynomial of degree n with integer coefficients of maximum bitsize τ, we prove that Algorithm `univsos1` uses $\widetilde{O}((\frac{n}{2})^{\frac{3n}{2}}\tau)$ boolean operations and returns polynomials of bitsize bounded by $O((\frac{n}{2})^{\frac{3n}{2}}\tau)$. This is not restrictive: when $f \in \mathbb{Q}[X]$, one can multiply it by the least common multiple of the denominators of its coefficients and apply our statement for polynomials in $\mathbb{Z}[X]$.

- We describe in Section 4 the second algorithm, `univsos2`, initially given in [5, Section 5.2]. We also analyze its bit complexity. When the input is a univariate polynomial of degree n with integer coefficients of maximum bitsize τ, we prove that Algorithm `univsos2` returns a decomposition as the sum of $n + 3$ squares of polynomials with coefficients of bitsize bounded by $O(n^3 + n^2\tau)$ using $\widetilde{O}(n^4 + n^3\tau)$ boolean operations. Thus, Algorithm `univsos2` outputs decompositions with total bitsize bounded by $O(n^4 + n^3\tau)$, yielding (slightly) better complexity that the algorithm in [3].

- Both algorithms are implemented within the `univsos` tool. The first release of `univsos` is a Maple library, is freely available[1] and is integrated in the RAGlib (Real Algebraic Library) Maple package[2]. The scalability of the library is evaluated in Section 5 on several non-negative polynomials issued from the existing literature. Despite the significant difference of theoretical complexity between the two algorithms, numerical benchmarks indicate that both may yield competitive performance w.r.t. specific subclasses of problems.

Acknowledgment. We are very grateful to the referees who helped to improve this article significantly. The figure illustrating the proof of Proposition 11 is due to one of the referees.

2. Preliminaries

We first recall the proof of the following classical result for non-negative real-valued univariate polynomials (see e.g., [32, Section 8.1]).

Theorem 1. *Let $f \in \mathbb{R}[X]$ be a non-negative univariate polynomial, i.e., $f(x) \geq 0$ for all $x \in \mathbb{R}$. Then f can be written as the sum of two polynomial squares in $\mathbb{R}[X]$.*

Proof. Without loss of generality, one can assume that f is monic, i.e., the leading coefficient (nonzero coefficient of highest degree) of f is 1. Then we decompose f as follows in $\mathbb{C}[x]$:

$$f = \prod_j (X - a_j)^{r_j} \prod_k ((X - (b_k + ic_k))(X - (b_k - ic_k)))^{s_k} ,$$

[1] `https://github.com/magronv/univsos`

[2] `http://www-polsys.lip6.fr/~safey/RAGLib/`

with $a_j, b_k, c_k \in \mathbb{R}$, $r_j, s_k \in \mathbb{N}^{>0}$, a_j standing for the real roots of f and $b_k \pm ic_k$ standing for the complex conjugate roots of f. Since f is non-negative, all real roots must have even multiplicity r_j, yielding the existence of polynomials $g, q, r \in \mathbb{R}[X]$ satisfying the following:

$$g^2 = \prod_j (X - a_j)^{r_j}, \quad q + ir = \prod_k (X - (b_k + ic_k))^{s_k}, \quad q - ir = \prod_k (X - (b_k - ic_k))^{s_k}.$$

Then one has $f = g^2(q + ir)(q - ir) = g^2(q^2 + r^2) = (gq)^2 + (gr)^2$, which proves the claim. □

Let K be a field and $g \in K[X]$. One says that g is a *square-free* polynomial when there is no prime element $p \in K[X]$ such that p^2 divides g. Now let $f \in K[X] \setminus \{0\}$. A decomposition of f of the form $f = a g_1^1 g_2^2 \dots g_n^n$ with $a \in K$ and normalized pairwise coprime square-free polynomials $g_1, g_2, \dots, g_n$ is called a *square-free decomposition* of f in $K[X]$.

We recall the following useful classical bounds.

Lemma 2. *[2, Corollary 10.12] If $p \in \mathbb{Z}[X]$ and $q \in \mathbb{Z}[X]$ divides p in $\mathbb{Z}[X]$, then one has $\tau(q) \leq \deg q + \tau(p) + \tau(1 + \deg p)$.*

Yun's algorithm [39] (also described in [8, Algorithm 14.21]) allows one to compute a square-free decomposition of polynomials with coefficients in a field of characteristic 0.

Lemma 3. *[8, § 14.23 & Table 8.7] Let $f \in \mathbb{Z}[X]$ of degree at most n and with coefficient bitsize bounded from above by τ. Then the square-free decomposition of f using Yun's Algorithm [39] can be computed using an expected number of $\widetilde{O}(n^2\tau)$ boolean operations.*

Lemma 4. *[27, § 6.3.1] & [20, Lemma 9.26] Let K be a field of characteristic 0 and L a field extension of K. The square-free decomposition in $L[X]$ of any polynomial $f \in K[X] \setminus \{0\}$ is the same as the square-free decomposition of f in $K[X]$. Any polynomial $f \in K[X] \setminus \{0\}$ which is a square-free polynomial in $K[X]$ is also square-free in $L[X]$.*

The following lemma allows one to obtain upper bounds on the magnitudes of the roots of a univariate polynomial.

Lemma 5. ***(Cauchy Bound** [4]) Let K be an ordered field. Let $a_0, \dots, a_n \in K$ with $a_n \neq 0$. Let $x \in K$ such that $\sum_{i=0}^n a_i x^i = 0$. Then, one has:*

$$|x| \leq \max \left\{ 1, \frac{|a_0|}{|a_n|} + \cdots + \frac{|a_{n-1}|}{|a_n|} \right\}.$$

For a polynomial with integer coefficients, one has the following:

Lemma 6. *[27, Theorem 4.2 (ii)] Let $f \in \mathbb{Z}[X]$ of degree n, with coefficient bitsize bounded from above by τ. If $f(x) = 0$ and $x \neq 0$, then $\frac{1}{2^\tau + 1} \leq |x| \leq 2^\tau + 1$.*

The real (resp. complex) roots of a polynomial can be approximated using root isolation techniques. To compute the real roots one can use algorithms based on Uspensky's method relying on Descartes' rule of signs, see e.g., [2, Chap. 10] for a general description of real root isolation algorithms.

Lemma 7. *[25, Theorem 5] Let $f \in \mathbb{Z}[X]$ with degree at most n with coefficient bitsize bounded from above by τ. Isolating intervals (resp. disks) of radius less than $2^{-\kappa}$ for all distinct real (resp. complex) roots of f can be computed in $\widetilde{O}(n^3 + n^2\tau + n\kappa)$ boolean operations, where κ is an arbitrary positive integer.*

Vieta's formulas provide relations between the coefficients of a polynomial and signed sums and products of the complex roots of this polynomial:

Lemma 8. ***(Vieta's formulas [9])*** *Let K be an ordered field. Given a polynomial $f = \sum_{i=0^n} a_i X^i \in K[X]$ with $a_n \neq 0$ with (not necessarily distinct) complex roots $z_1, \ldots, z_n$, one has for all $j = 1, \ldots, n$:*

$$\sum_{1 \leq i_1 < \cdots < i_j \leq n} z_{i_1} \ldots z_{i_j} = (-1)^j \frac{a_{n-j}}{a_n} .$$

3. Nichtnegativstellensätze with quadratic approximations

3.1. A proof of the existence of weighted SOS decompositions in K[X]

Lemma 9. *Let K be an ordered field. Let $g = aX^2 + bX + c \in K[X]$ with $a, b, c \in K$ and $a \neq 0$. Then g can be rewritten as $g = a\left(X + \frac{b}{2a}\right)^2 + \left(c - \frac{b^2}{4a}\right)$. Moreover, when g is non-negative over K, one has $a > 0$ and $c - \frac{b^2}{4a} \geq 0$.*

Proof. The decomposition of g is straightforward. Assume that g is non-negative over K. Observe that $c - \frac{b^2}{4a} = g\left(-\frac{b}{2a}\right)$; hence, since we assume that g is non-negative over K, we deduce that $c - \frac{b^2}{4a} \geq 0$.

It remains to prove that $a > 0$, which we do by contradiction, assuming that $a < 0$. This implies that for all $x \in K$, one has $\left(x + \frac{b}{2a}\right)^2 \leq -\frac{1}{a}\left(c - \frac{b^2}{4a}\right)$. Taking $y = x + \frac{b}{2a}$, there exists $C \in K$ such that $y^2 \leq C$, for each $y \in K$. This implies in particular for $y = 2$ that $4 \leq C$ and for $y = C$ that $C^2 \leq C$, thus $C \leq 1$. Finally, one obtains $4 \leq C \leq 1$, yielding a contradiction. □

Let $f \in K[X]$ be a square-free polynomial that is non-negative over $\mathbb{R}$. Then f is positive over $\mathbb{R}$; otherwise f would have at least one real root, implying that f would be neither a square-free polynomial in $\mathbb{R}[X]$ nor a square-free polynomial in $K[X]$, according to Lemma 4. We want to find a polynomial $g \in K[X]$ that fulfills the following conditions:

(i) $\deg g \leq 2$,
(ii) g is non-negative over $\mathbb{R}$,
(iii) $f - g$ is non-negative over $\mathbb{R}$,
(iv) $f - g$ has a root $t \in K$.

Assume that Property (i) holds. Then the existence of a weighted sum of squares decomposition in $K[X]$ for g is ensured from Property (ii). Property (iii) implies that $h = f - g$ has only non-negative values over $\mathbb{R}$. The aim of Property (iv) is to ensure the existence of a root $t \in K$ of h, which is stronger than the existence of a real root. Note that the case where the degree of $h = f - g$ is less than the degree of f occurs only when $\deg f = 2$. In this latter case, we can rely on Lemma 9 to prove the existence of a weighted sum of squares decomposition.

Now, we investigate the properties of a polynomial $g \in K[X]$ that fulfills conditions (i)-(iv). Using Property (i) and Taylor Decomposition, we obtain $g(X) = g(t) + g'(t)(X - t) + c(X - t)^2$. By Property (iv), one has $g(t) = f(t)$. In addition, Property (iii) yields $f(x) - g(x) \geq 0 = f(t) - g(t)$, for all $x \in K$, which implies that $(f - g)'(t) = 0$ and $g'(t) = f'(t)$. By Property (ii), the quadratic polynomial $g(X + t) = f(t) + f'(t)X + cX^2$ has at most one real root. This implies that the discriminant of $g(X + t)$, namely $f'(t)^2 - 4cf(t)$, cannot be positive; thus one has $c \geq \frac{f'(t)^2}{4f(t)}$ (since $f(t) > 0$).

Finally, given a polynomial g satisfying (i)-(iii) and (iv), one necessarily has $g = f_{t,c}$ with $\frac{f'(t)^2}{4f(t)} \leq c \in K$, and $f_{t,c} = f(t) + f'(t)(X-t) + c(X-t)^2$.

In this case, one also has that the polynomial $g = f_{t,c'}$, with $c' = \frac{f'(t)^2}{4f(t)}$, fulfills (i)-(iii) and (iv). Indeed, (i) and (iv) trivially hold. Let us prove that (ii) holds: when $\deg f_{t,c'} = 0$, $g = f(t) \geq 0$, and when $\deg f_{t,c'} = 2$, g has a single root $t - \frac{f'(t)}{2c'}$, and the minimum of g is $g\left(t - \frac{f'(t)}{2c'}\right) = 0$. The inequalities $f_{t,c'} \leq f_{t,c} \leq f$ over $\mathbb{R}$ yield (iii).

Therefore, given $f \in K[X]$ with f positive over $\mathbb{R}$, we are looking for $t \in K$ such that the inequality $f \geq f_t$ holds over $\mathbb{R}$, with

$$f_t := f(t) + f'(t)(X-t) + \frac{f'(t)^2}{4f(t)}(X-t)^2 \in K[X]\,.$$

The main problem is to ensure that t lies in K. If we choose t to be a global minimizer of f, then f_t would be the constant polynomial $\min\{f(x) \mid x \in \mathbb{R}\}$. The idea is then to find t in the neighborhood of a global minimizer of f. The following lemma shows that the inequality $f_t \leq f$ can always be satisfied for t in some neighborhood of a local minimizer of f.

Lemma 10. *Let $f \in \mathbb{R}[X]$ and assume that f is positive over $\mathbb{R}$. Let a be a local minimizer of f. For all $t \in \mathbb{R}$ with $f(t) \neq 0$, let us define the polynomial f_t:*

$$f_t := f(t) + f'(t)(X-t) + \frac{f'(t)^2}{4f(t)}(X-t)^2 \in \mathbb{R}[X]\,.$$

Then there exists a neighborhood $U \subset \mathbb{R}$ of a such that the inequality $f_t(x) \leq f(x)$ holds for all $(x,t) \in U \times U$.

Proof. Set $n := \deg f$. After adequate change of variable and scaling, it is easy to see that we can suppose without loss of generality that a is the origin and that $f(0) = 1$. Because of the Taylor formula

$$f = \sum_{k=0}^{n} \frac{f^{(k)}(t)}{k!}(X-t)^k,$$

we have

$$f - f_t = \sum_{k=2}^{n} \frac{f^{(k)}(t)}{k!}(X-t)^k - \frac{f'(t)^2}{4f(t)}(X-t)^2 = (X-t)^2\left(\sum_{k=2}^{n} \frac{f^{(k)}(t)}{k!}(X-t)^{k-2} - \frac{f'(t)^2}{4f(t)}\right)$$

for all $t \in \mathbb{R}$ with $f(t) \neq 0$. Let h be the bivariate polynomial defined as follows:

$$h := f(T)\left(\sum_{k=2}^{n} \frac{f^{(k)}(T)}{k!}(X-T)^{k-2}\right) - \frac{1}{4}f'(T)^2 \in \mathbb{R}[T,X].$$

Let us prove that $(0,0)$ is a local minimizer of h.

Since $f(0) = 1$, there exists $c \neq 0$, $\alpha \in \mathbb{N}$ and $g \in \mathbb{R}[X]$ such that $f - 1 = cX^{\alpha} + X^{\alpha+1}g$. Therefore, $\lim_{x\to 0} \frac{f(x)-1}{cx^{\alpha}} = 1$. Since $f - 1$ is non-negative over $\mathbb{R}$, one concludes that $c > 0$ and α is even. Let us consider the lowest homogeneous part H of h, that is the sum of all monomials of lowest degree involved in h. The lowest homogeneous part of $f'(T)^2$ is $c^2\alpha^2T^{2\alpha-2}$ with degree

$2\alpha-2$, while the lowest homogeneous part of $\sum_{k=2}^{n}\frac{f^{(k)}(T)}{k!}(X-T)^{k-2}$ is $c\sum_{k=2}^{n}\binom{\alpha}{k}T^{\alpha-k}(X-T)^{k-2}$ with degree $\alpha-2$ (where $\binom{\alpha}{k}=0$ for $k>\alpha$). Then

$$H = c\sum_{k=2}^{n}\binom{\alpha}{k}T^{\alpha-k}(X-T)^{k-2}\,,$$

and thus

$$(X-T)^2H = c((T+(X-T))^{\alpha}-T^{\alpha}-\alpha T^{\alpha-1}(X-T)) = c(X^{\alpha}-\alpha T^{\alpha-1}X+(\alpha-1)T^{\alpha}).$$

We shall prove that H is positive except at $(0,0)$. Then it will be clear that $\lim_{\|(x,t)\|\to 0}\frac{h(x,t)}{H(x,t)}$ exists. And since this limit obviously equals 1, we will conclude that $(0,0)$ is a local minimizer of h.

Let us consider $(x,t)\in\mathbb{R}^2\setminus\{(0,0)\}$ and show that $H(x,t)>0$. If $t=x$, we have $H(x,t)=H(x,x)=\binom{\alpha}{2}x^{\alpha-2}>0$. If $t\neq x$, then it is enough to show that $(x-t)^2H(x,t)=c(x^{\alpha}-\alpha t^{\alpha-1}x+(\alpha-1)t^{\alpha})>0$. This is clear if $t=0$, since $c>0$ and α is even. Now suppose that $t\neq 0$ and define $\xi:=\frac{x}{t}\neq 1$. Then one has $t^{-\alpha}(x-t)^2H(x,t)=c(\xi^{\alpha}-\alpha\xi+\alpha-1)>0$, since the univariate polynomial $r:=X^{\alpha}-\alpha X+\alpha-1$ is positive except at 1, as $r'=\alpha X^{\alpha-1}-\alpha$. The positivity of H implies that $(0,0)$ is a local minimizer of h.

Let us define $q(X,T):=(X-T)^2h$. Combining the fact that $(0,0)$ is a local minimizer of the two polynomials h, $(X-T)^2$, and the fact that $h(0,0)=\frac{1}{2}f(0)f''(0)-\frac{1}{4}f'(0)^2=0$, we conclude that $(0,0)$ is also a local minimizer of q. Since $f(x)-f_t(x)=q(x,t)/f(t)$, this yields the existence of a neighborhood $O\subset\mathbb{R}^2$ of $(0,0)$ such that the inequality $f-f_t\geq 0$ holds for all $(x,t)\in O$. Since there exists some neighborhood $U\subset\mathbb{R}$ of 0 such that the rectangle $U\times U$ is included in O, this proves the initial claim.

□

Lemma 10 states the existence of a neighborhood U of a local minimizer of f such that the inequality $f_t(x)\leq f(x)$ holds for all $(x,t)\in U\times U$. Now, we show that with such a neighborhood U of the smallest global minimizer a of f, there exists $\varepsilon>0$ such that the inequality $f_t(x)\leq f(x)$ holds for all $t\in(a-\varepsilon,a)$, and for all $x\in\mathbb{R}$.

Proposition 11. *Let $f\in\mathbb{R}[X]$ with $\deg f>0$. Assume that f is positive over $\mathbb{R}$. Let a be the smallest global minimizer of f. Then there exists a positive $\epsilon\in\mathbb{R}$ such that for all $t\in\mathbb{R}$ with $a-\epsilon<t<a$, the quadratic polynomial f_t, defined by*

$$\begin{aligned} f_t &:= f(t)+f'(t)(X-t)+\frac{f'(t)^2}{4f(t)}(X-t)^2 \\ &= \frac{f'(t)^2}{4f(t)}\left[\frac{2f(t)}{f'(t)}+(X-t)\right]^2\in\mathbb{R}[X]\,, \end{aligned} \tag{1}$$

satisfies $f_t\leq f$ over $\mathbb{R}$.

Proof. First, we handle the case when $\deg f=2$. Using Taylor Decomposition of f at t, one obtains $f=f(t)+f'(t)(X-t)+\frac{f''(t)}{2}(X-t)^2$. Since f has no real root, the discriminant of f is negative, namely $f'(t)^2-4f(t)\frac{f''(t)}{2}<0$. This implies that $\frac{f'(t)^2}{4f(t)}<\frac{f''(t)^2}{2}$, ensuring that the inequality $f_t\leq f$ holds over $\mathbb{R}$, for all $t\in\mathbb{R}$.

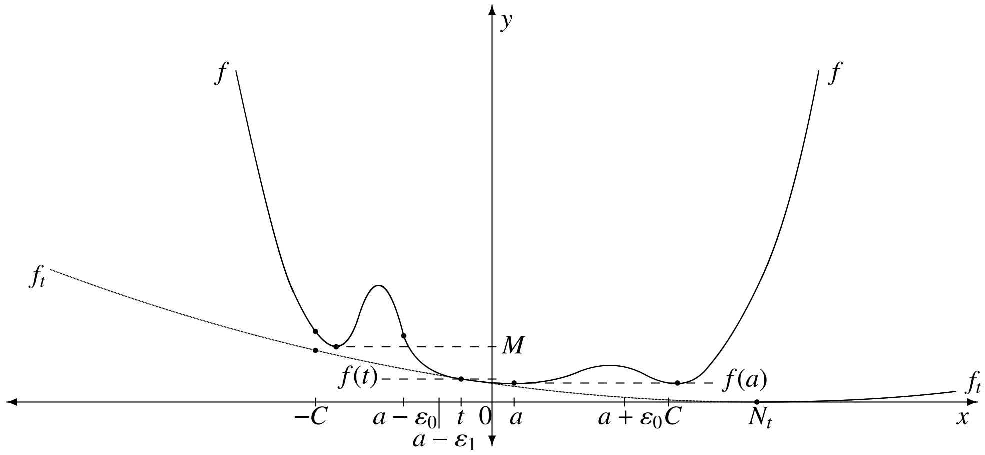
In the sequel, we assume that $\deg f > 2$. We can find a neighborhood U as in Lemma 10 and without loss of generality, let us suppose that $U = [a - \epsilon_0, a + \epsilon_0]$ for some positive ϵ_0 small enough so that f' has no real root in $[a - \epsilon_0, a)$. Then the inequality $f_t(x) \leq f(x)$ holds for all $x, t \in U$. Next, we write $f - f_t = \sum_{i=0}^{n} a_{it} x^i$, with $a_{it} \in \mathbb{R}$ and $n = \deg f > 2$ and define the following function:

$$U \to \mathbb{R} : t \mapsto C_t := \max \left\{ 1, \frac{|a_{0t}|}{|a_{nt}|} + \cdots + \frac{|a_{(n-1)t}|}{|a_{nt}|} \right\} .$$

Note that the Cauchy bound (Lemma 5) implies that for all $t \in U$, all real roots of $f - f_t$ lie in $[-C_t, C_t]$. In addition, the closed interval domain U is compact, implying that the range values of the function $U \to \mathbb{R} : t \mapsto C_t$ are bounded. Let $C \in \mathbb{R}$ with $C \geq C_t$ for all $t \in U$. Then, for all $t \in U$, all real roots of $f - f_t$ lie in the interval $[-C, C]$ and we can assume without loss of generality that $-C < a - \epsilon_0 < a < a + \epsilon_0 < C$. Let us define $M := \min\{f(x) \mid x \in [-C, a - \epsilon_0]\}$.

Since a is the smallest global minimizer of f, $f(a) < M$. For all $t \in [a - \epsilon_0, a)$, the quadratic polynomial f_t has one real root $N_t := \frac{-2f(t)}{f'(t)} + t$. When $t \in [a - \epsilon_0, a)$ converges to a, $f'(t) < 0$ converges towards 0 and $-2f(t)$ converges towards $-2f(a) < 0$. Thus, the corresponding limit of N_t is $+\infty$. In addition, $f_t(-C)$ tends to $f_a(-C) = f(a) < M$. Therefore, there exists some $\epsilon \in (0, \epsilon_0]$ such that for all $t \in (a - \epsilon, a)$, one has $N_t \in [C, \infty)$ and $f_t(-C) < M$. For all $t \in (a - \epsilon, a)$, we partition $\mathbb{R}$ into five intervals and prove that the inequality $f_t \leq f$ holds on each interval (see the picture below):

- The inequality $f_t \leq f$ holds over $(-\infty, -C]$: this comes from the fact that $f_t(-C) < M \leq f(-C)$ and the fact $f - f_t$ has no real root in $(-\infty, -C]$.
- The inequality $f_t \leq f$ holds over $(-C, a - \epsilon_0]$: f_t is monotonically decreasing over $(-\infty, N_t]$. Since one has $-C < a - \epsilon_0 < C \leq N_t$, f_t is monotonically decreasing over $(-C, a - \epsilon_0]$. This implies that for all $x \in (-C, a - \epsilon_0]$, one has $f_t(x) \leq f_t(-C) < M \leq f(x)$.
- The inequality $f_t \leq f$ holds over $[a - \epsilon_0, a)$: it follows from the fact that $[a - \epsilon_0, a) \subseteq U$.
- The inequality $f_t \leq f$ holds over $[a, C)$: f_t is monotonically decreasing over $(-\infty, N_t]$. Since one has $a < C \leq N_t$, f_t is monotonically decreasing over $[a, C)$. Since a is a global minimizer of f and $a \in U$, one has $f_t(x) \leq f_t(a) \leq f(a) \leq f(x)$ for all $x \in [a, C)$.
- The inequality $f_t \leq f$ holds over $[C, \infty]$: this claim is implied by the facts that $f_t(N_t) = 0 < f(N_t)$ and $N_t \in [C, \infty]$, together with the fact that $f - f_t$ has no real root in $[C, \infty]$.



□

Proposition 12. *Let K be a subfield of $\mathbb{R}$ and $f \in K[X]$ with $\deg f = n \geq 1$. Then f is non-negative on $\mathbb{R}$ if and only if f is a weighted sum of n polynomial squares in $K[X]$, i.e., there exist $a_1, \ldots, a_n \in K^{\geq 0}$ and $g_1, \ldots, g_n \in K[X]$ such that $f = \sum_{i=1}^n a_i g_i^2$. (In fact, for $n \geq 4$, $n-1$ squares suffice.)*

Proof. The *if* part is straightforward. For the other direction, assume that f is non-negative on $\mathbb{R}$, whence n is even. The proof is by induction over n. The base case $n = 2$ follows from Lemma 9. So assume $n \geq 4$, and that the result has been established for polynomials of degree $< n$.

When f is not a square-free polynomial, we show that f is a weighted sum of $n-2$ polynomial squares. We can write $f = gh^2$, for some polynomials $g, h \in K[X]$ with $\deg g \leq n-2$. This gives $g(x) = \frac{f(x)}{h(x)^2} \geq 0$ for all $x \in \mathbb{R}$ such that $h(x) \neq 0$. Since h has a finite number of real roots, g is non-negative on $\mathbb{R}$. Using the induction hypothesis, g is a weighted sum of $n-2$ polynomial squares. Therefore, f is also a weighted sum of $n-2$ polynomial squares.

Next assume f is square-free in $K[X]$ (and hence also square-free in $\mathbb{R}[X]$ by Lemma 4). Thus, f is positive on $\mathbb{R}$. Using Proposition 11, there exists some $t \in K$ (K is dense in $\mathbb{R}$) and a quadratic polynomial $f_t \in K[X]$ such that the inequalities $0 \leq f_t(x) \leq f(x)$ hold for all $x \in \mathbb{R}$ and $f_t(t) = f(t)$. The polynomial $f - f_t$ has degree n, and takes only non-negative values. In addition, $(f - f_t)(t) = 0$, whence $f - f_t$ is not a square-free polynomial. Hence, we are in the above case, implying that $f - f_t$ is a weighted sum of $n-2$ polynomial squares. From (1) in Lemma 9, f_t is a weighted perfect square in $K[X]$, implying that f is a weighted sum of $n-1$ polynomial squares, as required. □

3.2. Algorithm `univsos1`

The smallest global minimizer a of f is a real root of $f' \in K[X]$. Therefore, by using root isolation techniques [2, Chap. 10], one can isolate all the real roots of f' in non-overlapping intervals with endpoints in K. Such techniques rely on applying successive bisections, so that one can arbitrarily reduce the width of every interval and sort them w.r.t. their left endpoints. Eventually, we apply this procedure to find a sequence of elements in K converging from below to the smallest global minimizer of f in order to find a suitable t. We denote by `parab`(f) the

Input: non-negative polynomial $f \in K[X]$ of degree $n \geq 2$, with K a subfield of $\mathbb{R}$
Output: pair of lists of polynomials ($\texttt{h\_list}, \texttt{q\_list}$) with coefficients in K

1: $\texttt{h\_list} := [\,], \texttt{q\_list} := [\,]$.
2: **while** $\deg f > 2$ **do**
3: $\quad (g, h) := \texttt{sqrfree}(f)$ $\qquad \triangleright\ f = gh^2$
4: $\quad$ **if** $\deg h > 0$ **then** $\texttt{h\_list} := \texttt{h\_list} \cup \{h\}, \texttt{q\_list} := \texttt{q\_list} \cup \{0\}, f := g$
5: $\quad$ **else**
6: $\quad\quad f_t := \texttt{parab}(f)$
7: $\quad\quad (g, h) := \texttt{sqrfree}(f - f_t)$
8: $\quad\quad \texttt{h\_list} := \texttt{h\_list} \cup \{h\}, \texttt{q\_list} := \texttt{q\_list} \cup \{f_t\}, f := g$
9: $\quad$ **end**
10: **done**
11: $\texttt{h\_list} := \texttt{h\_list} \cup \{0\}, \texttt{q\_list} := \texttt{q\_list} \cup \{f\}$
12: **return** $\texttt{h\_list}, \texttt{q\_list}$

Figure 1: `univsos1`: algorithm to compute SOS decompositions of non-negative univariate polynomials.

corresponding procedure performing root isolation and returning the polynomial $f_t := \frac{f'(t)^2}{4f(t)}(X - t)^2 + f'(t)(X - t) + f(t)$ such that $t \in K$ and $f \geq f_t$ over $\mathbb{R}$.

Algorithm `univsos1`, depicted in Figure 1, takes as input a polynomial $f \in K[X]$ of even degree $n \geq 2$. The steps performed by this algorithm correspond to what is described in the proof of Proposition 12 and rely on two auxiliary procedures. The first one is the procedure `parab` (see Step 6). The second one is denoted by `sqrfree` and performs square-free decomposition: for a given polynomial $f \in K[X]$, $\texttt{sqrfree}(f)$ returns two polynomials g and h in $K[X]$ such that $f = gh^2$ and g is square-free. When f is square-free, the procedure returns $g = f$ and $h = 1$ (in this case $\deg h = 0$). As in the proof of Proposition 12, this square-free decomposition procedure is performed either on the input polynomial f (Step 3) or on the non-negative polynomial $(f - f_t)$ (Step 7). The output of Algorithm `univsos1` is a pair of lists of polynomials in $K[X]$, allowing one to retrieve an SOS decomposition of f. By Proposition 12 the length of all output lists, denoted by r, is bounded by $n/2$. If we write $h_r, \ldots, h_1$ for the polynomials belonging to `h_list`, and $q_r, \ldots, q_1$ the positive definite quadratic polynomials belonging to `q_list`, one obtains the following Horner-like decomposition: $f = h_r^2(h_{r-1}^2(h_{r-2}^2(\ldots) + q_{r-2}) + q_{r-1}) + q_r$. Since each positive definite quadratic polynomial q_i is a weighted SOS polynomial, this yields a weighted SOS decomposition for f.

Example 13. *Let us consider the polynomial* $f := \frac{1}{16}X^6 + X^4 - \frac{1}{9}X^3 - \frac{11}{10}X^2 + \frac{2}{15}X + 2 \in \mathbb{Q}[X]$. *We describe the different steps performed by Algorithm* `univsos1`*:*

- *The polynomial f is square-free, and the algorithm starts by providing the value $t = -1$ as an approximation of the smallest minimizer of f. With $f(t) = \frac{1397}{720}$ and $f'(t) = \frac{-19}{8}$, one obtains $f_{-1} = \frac{720}{1397}(-\frac{19}{16}X + \frac{271}{360})^2$.*
- *Next, after obtaining the square-free decomposition $f(X) - f_{-1} = (X + 1)^2 g$, the same procedure is applied on g. One obtains the value $t = 1$ as an approximation of the smallest minimizer of g and $g_1 = \frac{502920}{237293}(-\frac{1}{18}X + \frac{88411}{167640})^2$.*
- *Eventually, one obtains the square-free decomposition $g(X) - g_1 = (X - 1)^2 h$ with $h = \frac{1}{16}(X - \frac{19108973}{17085096})$.*

Overall, Algorithm $\texttt{univsos1}$ *provides the lists* $\texttt{h\_list} = [1, X+1, 1, X-1, 0]$ *and* $\texttt{q\_list} = [\frac{720}{1397}(-\frac{19}{16}X + \frac{271}{360})^2, 0, \frac{502920}{237293}(-\frac{1}{18}X + \frac{88411}{167640})^2, 0, \frac{1}{16}(X - \frac{19108973}{17085096})]$*, yielding the following weighted SOS decomposition:*

$$f = (X+1)^2\Big[(X-1)^2\Big(\frac{1}{16}\Big(X-\frac{19108973}{17085096}\Big)^2\Big)+\frac{502920}{237293}\Big(-\frac{1}{18}X+\frac{88411}{167640}\Big)^2\Big]+\frac{720}{1397}\Big(-\frac{19}{16}X+\frac{271}{360}\Big)^2.$$

In the sequel, we analyze the complexity of Algorithm univsos1 in the particular case $K = \mathbb{Q}$. We provide bounds on the bitsize of related SOS decompositions as well as bounds on the arithmetic cost required for computation and verification.

3.3. Bitsize of the output

Lemma 14. *Let $f \in \mathbb{Z}[X]$ be a positive polynomial over $\mathbb{R}$, with $\deg f = n$ and τ an upper bound on the bitsize of the coefficients of f. When applying Algorithm* $\texttt{univsos1}$ *to f, the subprocedure* $\texttt{parab}$ *outputs a polynomial f_t such that $\tau(t) = O(n^2\tau)$.*

Proof. Let us consider the set $S \subseteq \mathbb{Q}$ defined by:

$$\begin{aligned} S &:= \{t \in \mathbb{Q} \mid \forall x \in \mathbb{R}\,, f_t(x) \le f(x)\} \\ &= \{t \in \mathbb{Q} \mid \forall x \in \mathbb{R}\,, 4f(t)^2 + 4f(t)f'(t)(x-t) + f'(t)^2(x-t)^2 \le 4f(t)f(x)\}\,. \end{aligned}$$

The polynomial involved in S has degree $2n$, with maximum bitsize of the coefficients bounded from above by 2τ. Observe that the set S can be described by solving a quantifier elimination problem involving a single quantified variable, and a single free variable. In addition, solving such a problem can be done with the Cylindrical Algebraic Decomposition algorithm (see e.g., [2, Chap. 11]) which, here, reduces to performing subresultant computations with bivariate polynomials as input. Using the complexity analysis of the Cylindrical Algebraic Decomposition algorithm as done for [2, Algorithm 11.1], one shows that the set S can be described by polynomials with maximum bitsize coefficients bounded from above by $O(n^2\tau)$. Since t is a rational root of one of these polynomials, the rational zero theorem [37] implies that $\tau(t) = O(n^2\tau)$. □

Lemma 15. *Let $f \in \mathbb{Z}[X]$ be a positive polynomial over $\mathbb{R}$, with $\deg f = n$ and τ an upper bound on the bitsize of the coefficients of f. Let t and f_t be as in Lemma 14. Let us write $t = \frac{t_1}{t_2}$, with $t_1 \in \mathbb{Z}, t_2 \in \mathbb{Z}\backslash\{0\}$, t_1 and t_2 being coprime. Let $\hat{f}(X) := t_2^{2n} f(t) f(X)$ and $\hat{f}_t(X) := t_2^{2n} f(t) f_t(X)$. The polynomial f_t has coefficients of bitsize bounded by $O(n^3\tau)$. Moreover, there exists $g \in \mathbb{Z}[X]$ such that $\hat{f} - \hat{f}_t = (X-t)^2 g$ and $\tau(g) = O(n^3\tau)$.*

Proof. One can write $f_t = M_2(t)X^2 + M_1(t)X + M_0(t)$ with

$$\begin{aligned} M_2(t) &:= \frac{f'(t)^2}{4f(t)}\,, \\ M_1(t) &:= \frac{2f'(t)(2f(t) - tf'(t))}{4f(t)}\,, \\ M_0(t) &:= \frac{(2f(t) - tf'(t))^2}{4f(t)}\,, \end{aligned}$$

and $\|f_t\|_\infty = \max\{M_2(t), |M_1(t)|, M_0(t)\}$. One has $0 \le M_0(t) = f_t(0) \le f(0) \le \|f\|_\infty$.

In addition, $0 \leq M_0(t) + M_1(t) + M_2(t) = f_t(1) \leq f(1) \leq (n+1)\|f\|_\infty$ and $0 \leq M_0(t) - M_1(t) + M_2(t) = f_t(-1) \leq f(-1) \leq (n+1)\|f\|_\infty$. Thus, one has $M_0(t) + |M_1(t)| + M_2(t) \leq (n+1)\|f\|_\infty$, which implies that $\|f_t\|_\infty \leq (n+1)\|f\|_\infty$.

By writing $f(X) = \sum_{i=0}^n a_i X^i$, one has $t_2^{2n} f(t) = \sum_{i=0}^n a_i t_1^i t_2^{2n-i} \leq \|f\|_\infty \sum_{i=0}^n |t_1|^i |t_2|^{2n-i}$. This implies that $\tau(\hat{f}) \leq \tau + \tau(n+1) + \tau(t^{2n})$. By Lemma 14, one has $\tau(\hat{f}) = O(n^3\tau)$.

The polynomials $\hat{f}(X), \hat{f}_t(X)$ are polynomials in $\mathbb{Z}[X]$, and since $\|\hat{f}_t\|_\infty \leq (n+1)\|\hat{f}\|_\infty$, the triangle inequality $\|\hat{f} - \hat{f}_t\|_\infty \leq \|\hat{f}\|_\infty + \|\hat{f}_t\|_\infty \leq (n+2)\|\hat{f}\|_\infty$ implies that $\tau(\hat{f} - \hat{f}_t) \leq \tau(n+2) + \tau(\hat{f}) = O(n^3\tau)$. In addition, the polynomial f_t has coefficients of bitsize bounded by $\tau(\hat{f}_t) + \tau(t_2^{2n} f(t)) = O(n^3\tau)$.

As in the proof of Proposition 12, one has $(\hat{f} - \hat{f}_t)(t) = 0$, which allows one to write the square-free decomposition of the polynomial $\hat{f} - \hat{f}_t \in \mathbb{Z}[X]$ as $\hat{f} - \hat{f}_t = (X - t)^2 g$, with $g \in \mathbb{Z}[X]$. By Lemma 2, one has $\tau(g) \leq n - 2 + \tau(\hat{f} - \hat{f}_t) + \log_2(n+1) \leq n - 2 + 2\log_2(n+2) + \tau(\hat{f}) = O(n^3\tau)$, which concludes the proof. $\square$

Theorem 16. *Let $f \in \mathbb{Z}[X]$ be a positive polynomial over $\mathbb{R}$, with $\deg f = n = 2k$ and τ an upper bound on the bitsize of the coefficients of f. Then the maximum bitsize of the coefficients involved in the SOS decomposition of f obtained with Algorithm `univsos1` is bounded from above by $O((k!)^3\tau) = O\left(\left(\frac{n}{2}\right)^{\frac{3n}{2}}\tau\right)$.*

Proof. With $k = n/2$ and starting from the polynomial f, Algorithm `univsos1` generates, in the worst-case scenario, two sequences of polynomials $f_k, \ldots, f_1 \in \mathbb{Z}[X], q_k, \ldots, q_2 \in \mathbb{Z}[X]$, as well as rational numbers $t_k, \ldots, t_2 \in \mathbb{Q}$ such that $f_k = f$, $t_i = \frac{t_{i1}}{t_{i2}}$, with $t_{i1} \in \mathbb{Z}$, $t_{i2} \in \mathbb{Z}\backslash\{0\}$ and

$$t_{i2}^{4i} f_i(t_i) f_i - q_i = (X - t_i)^2 f_{i-1}\,, \quad i = 2, \ldots, k\,. \tag{2}$$

From Lemma 15, for all $i = 2, \ldots, k$, one has $\tau(f_{i-1}) = O(i^3\tau(f_i))$. This yields $\tau(f_1) = O((k!)^3\tau(f))$.

Using Stirling's formula, we obtain $k! \leq 2\sqrt{2\pi k}(\frac{k}{e})^k$ and $(k!)^3 \leq 16\sqrt{2}\pi^{\frac{3}{2}} k^{\frac{3}{2}} (\frac{k}{e})^{3k}$, where e denotes the Euler number. Since $k \leq e^k$ for each integer $k \geq 1$ and $\frac{3}{2} < 3$, one has $(k!)^3 \in O(k^{3k})$, yielding $\tau(f_i) = O((\frac{n}{2})^{\frac{3n}{2}}\tau)$, for all $i = 1, \ldots, k$. Similarly, we obtain $\tau(q_i) = O((\frac{n}{2})^{\frac{3n}{2}}\tau)$, for all $i = 1, \ldots, k$. Finally, using Lemma 14, one has $\tau(t_i) = O(i^2\tau(f_i))$, yielding the desired result. $\square$

3.4. Bit complexity analysis

Theorem 17. *Let $f \in \mathbb{Z}[X]$ be a positive polynomial over $\mathbb{R}$, with $\deg f = n = 2k$ and τ an upper bound on the bitsize of the coefficients of f. Then, on input f, Algorithm `univsos1` runs in*

$$\tilde{O}(k^3 \cdot (k!)^3\tau) = \tilde{O}\left(\left(\frac{n}{2}\right)^{\frac{3n}{2}}\tau\right)$$

boolean operations.

Proof. For $i = 2, \ldots, k$ we obtain each polynomial f_{i-1} as in the proof of Theorem 16 by computing the square-free decomposition of the polynomial $t_{i2}^{4i} f_i(t_i) f_i - q_i$. As in the proof of Theorem 16, one has $\tau(f_{i-1}) = O(i^3\tau(f_i))$. Hence, this follows by Lemma 3 that the polynomial f_{i-1} can be computed using an expected number of $\tilde{O}(i^2 \cdot i^3\tau(f_i))$ boolean operations. The number of boolean operations to compute all polynomials $f_1, \ldots, f_{k-1}$ is thus bounded by $\tilde{O}(k^2 \cdot k^3\tau + (k-1)^2(k-1)^3k^3\tau + \cdots + 2^2(k!)^3\tau) = \tilde{O}\left(\sum_{i=2}^k \left(i^2 \prod_{j=i}^k j^3\right)\tau\right)$.

For each $i = 2, \dots, k$, the bitsize of t_i is bounded from above by $O(i^2 \tau(f_i))$. Therefore, t_i can be computed by approximating the roots of f_i' with isolating intervals of radius less than $2^{-i^2 \tau(f_i)}$. By Lemma 7, the corresponding computation cost is $\tilde{O}(i^3 \tau(f_i))$ boolean operations. The number of boolean operations to compute all rational numbers $t_2, \dots, t_k$ is bounded from above by $\tilde{O}(k^3 \cdot k^3 \tau + (k-1)^3 (k-1)^3 k^3 \tau(f) + \cdots + 2^3 (k!)^3 \tau) = \tilde{O}\left(\sum_{i=2}^k \left(i^3 \prod_{j=i}^k j^3\right) \tau\right)$.

In addition, one has

$$\sum_{i=2}^{k} \left(i^3 \prod_{j=i}^{k} j^3\right) = (k!)^3 \sum_{i=2}^{k} \frac{1}{((i-1)!)^3} < (k!)^3 \sum_{i=1}^{\infty} \frac{1}{i!} = (k!)^3 (e^1 - 1) < 2(k!)^3 .$$

Using Stirling's formula, we obtain $2(k!)^3 \leq 32\sqrt{2}\pi^{\frac{3}{2}} k^{\frac{3}{2}} \left(\frac{k}{e}\right)^{3k}$. As in the proof of Theorem 16, we obtain the announced complexity. □

For a given polynomial f of degree $2k$, one can check the correctness of the SOS decomposition obtained with Algorithm `univsos1` by evaluating this SOS polynomial at $2k+1$ distinct points and compare the results with the ones obtained while evaluating f at the same points.

Theorem 18. *Let $f \in \mathbb{Z}[X]$ be a positive polynomial over $\mathbb{R}$, with $\deg f = n = 2k$ and τ an upper bound on the bitsize of the coefficients of f. Then one can check the correctness of the SOS decomposition of f obtained with Algorithm `univsos1` within*

$$\tilde{O}(k \cdot (k!)^3 \tau) = \tilde{O}\left(\left(\frac{n}{2}\right)^{\frac{3n}{2}} \tau\right)$$

boolean operations.

Proof. From [8, Corollary 8.27], the cost of multiplying two polynomials in $\mathbb{Z}[X]$ of degree less than $n = 2k$ with coefficients of bitsize less than B is bounded by $\tilde{O}(k \cdot B)$. By Theorem 16, the maximal bitsize of the coefficients of the SOS decomposition of f obtained with Algorithm `univsos1` is bounded from above by $B = O((k!)^3 \tau)$. Let us consider $2k+1$ distinct integers (e.g., all integers between 0 and n), with maximal bitsize bounded from above by $\log_2 n$. Therefore, from [8, Corollary 10.8], the cost of the evaluation of this decomposition at the $2k+1$ points can be performed using at most $\tilde{O}(k \cdot (k!)^3 \tau)$ boolean operations, the desired result. □

Remark 19. *Let $f_k = f \in \mathbb{Z}[X]$. Under the strong assumption that each polynomial $f_k, \dots, f_1$ involved in Algorithm `univsos1` has at least one integer global minimizer, Algorithm `univsos1` has polynomial complexity. Indeed, in this case, $q_i = f_i(t_i)$, $\tau(t_i) = O(\tau(f_i))$ and $\tau(f_{i-1}) = O(2(i-1) + \tau(f_i))$, for all $i = 2, \dots, k$. Hence, the maximal bitsize of the coefficients involved in the SOS decomposition of f is bounded from above by $O(k^2 + \tau)$, and this decomposition can be computed using an expected number of $\tilde{O}(k^4 + k^3 \tau)$ boolean operations.*

4. Nichtnegativstellensätze with perturbed polynomials

Here, we recall the algorithm given in [5, Section 5.2]. The description of this algorithm, denoted by `univsos2`, is given in Figure 2.

Input: non-negative polynomial $f \in K[X]$ of degree $n \geq 2$, with K a subfield of $\mathbb{R}$, $\varepsilon \in K$ such that $0 < \varepsilon < f_n$, precision $\delta \in \mathbb{N}$ for complex root isolation
Output: list `c_list` of numbers in K and list `s_list` of polynomials in $K[X]$

1: $(p, h) :=$ `sqrfree`(f) $\qquad \triangleright\ f = p\,h^2$
2: $n' := \deg p$, $k := n'/2$
3: $p_\varepsilon := p - \varepsilon \sum_{i=0}^{k} X^{2i}$
4: **while** `has_real_roots`(p_ε) **do**
5: $\quad \varepsilon := \frac{\varepsilon}{2}$, $p_\varepsilon := p - \varepsilon \sum_{i=0}^{k} X^{2i}$
6: **done**
7: $\varepsilon := \frac{\varepsilon}{2}$
8: $(s_1, s_2) :=$ `sum_two_squares`(p_ε, δ)
9: $\ell := f_n$, $u := p_\varepsilon - \ell s_1^2 - \ell s_2^2$, $u_{-1} := 0$, $u_{2k+1} := 0$ $\qquad \triangleright\ u = \sum_{i=0}^{2k-1} u_i X^i$
10: **while** $\varepsilon < \max_{0 \leq i \leq k} \{ \frac{|u_{2i+1}|}{4} - u_{2i} + |u_{2i-1}| \}$ **do**
11: $\quad \delta := 2\delta$, $(s_1, s_2) :=$ `sum_two_squares`(p_ε, δ), $u := p_\varepsilon - \ell s_1^2 - \ell s_2^2$
12: **done**
13: `c_list` $:= [\ell, \ell]$, `s_list` $:= [h\, s_1, h\, s_2]$
14: **for** $i = 0$ to $k - 1$ **do**
15: $\quad$ `c_list` $:=$ `c_list` $\cup \{|u_{2i+1}|\}$, `s_list` $:=$ `s_list` $\cup \{h\,(X^{i+1} + \frac{\operatorname{sgn}(u_{2i+1})}{2} X^i)\}$
16: $\quad$ `c_list` $:=$ `c_list` $\cup \{\varepsilon - \frac{|u_{2i+1}|}{4} + u_{2i} - |u_{2i-1}|\}$, `s_list` $:=$ `s_list` $\cup \{h\, X^i\}$
17: **done**
18: **return** `c_list` $\cup \{\varepsilon + u_n - |u_{n-1}|\}$, `s_list` $\cup \{h\, X^k\}$

Figure 2: `univsos2`: algorithm to compute SOS decompositions of non-negative univariate polynomials.

4.1. Algorithm `univsos2`

Given a subfield K of $\mathbb{R}$ and a non-negative polynomial $f = \sum_{i=0}^{n} f_i X^i \in K[X]$ of degree $n = 2k$, one first obtains the square-free decomposition of f, yielding $f = p\,h^2$ with $p > 0$ on $\mathbb{R}$ (see Step 1 of Figure 2).Then the idea is to find a positive number $\varepsilon > 0$ in K such that the perturbed polynomial $p_\varepsilon(X) := p(X) - \varepsilon \sum_{i=0}^{k} X^{2i}$ is also positive on $\mathbb{R}$ (see [5, Section 5.2.2] for more details). This number is computed thanks to the loop going from Step 4 to Step 6, and relies on the auxiliary procedure `has_real_roots`, which checks whether the polynomial p_ε has real roots using root isolation techniques. As mentioned in [5, Section 5.2.2], the number ε is divided by 2 again to allow a margin of safety (Step 7).

Note that one can always ensure that the leading coefficient $\ell := p_n$ of p is the same as the leading coefficient f_n of the input polynomial f.

We obtain an approximate weighted rational sum of two polynomial squares decomposition of the polynomial p_ε with the auxiliary procedure `sum_two_squares` (Step 8), relying on an arbitrary precision complex root finder. Recalling Theorem 1, this implies that the polynomial p can be approximated as closely as desired by a weighted sum of two polynomial squares in $\mathbb{Q}[X]$, that is $\ell s_1^2 + \ell s_2^2$.

Thus there exists a remainder polynomial $u := p_\varepsilon - \ell s_1^2 - \ell s_2^2$ with coefficients of arbitrarily small magnitude (as mentioned in [5, Section 5.2.3]). The magnitude of the coefficients converges to 0 as the precision δ of the complex root finder goes to infinity. The precision is increased thanks to the loop going from Step 10 to Step 12 until a condition between the coefficients of u and ε becomes true, ensuring that $\varepsilon \sum_{i=0}^{k} X^{2i} + u(X)$ also admits a weighted SOS decomposition. For more details, see [5, Section 5.2.4].

The reason why Algorithm `univsos2` terminates is the following: at first, one can always find a sufficiently small perturbation ε such that the perturbed polynomial p_ε remains positive. Next, one can always find sufficiently precise approximations of the complex roots of p_ε ensuring that the error between the initial polynomial p and the approximate SOS decomposition is compensated, thanks to the perturbation term.

The outputs of Algorithm `univsos2` are a list of numbers in K and a list of polynomials in $K[X]$, allowing one to retrieve a weighted SOS decomposition of f. The size r of both lists is equal to $2k + 3 = n' + 3 \leq n + 3$. If we write $c_r, \dots, c_1$ for the numbers belonging to `c_list` and $s_r, \dots, s_1$ for the polynomials belonging to `s_list`, one obtains the following SOS decomposition: $f = c_r s_r^2 + \cdots + c_1 s_1^2$.

Example 20. *Let us consider the same polynomial $f := \frac{1}{16}X^6 + X^4 - \frac{1}{9}X^3 - \frac{11}{10}X^2 + \frac{2}{15}X + 2 \in \mathbb{Q}[X]$ as in Example 13. We describe the different steps performed by Algorithm* `univsos2`*:*

- *The polynomial f is square-free, so we obtain $p = f$ (Step 1). After performing the loop from Step 4 to Step 6, Algorithm* `univsos2` *provides the value $\varepsilon = \frac{1}{32}$ at Step 7 as well as the polynomial $p_\varepsilon := p - \frac{1}{32}(1 + X^2 + X^4 + X^6)$, which has no real root.*
- *Next, after increasing three times the precision in the loop going from Step 10 to Step 12, the result of the approximate root computation yields $s_1 = X^3 - \frac{69}{8}X$ and $s_2 = 7X^2 - \frac{1}{4}X - \frac{63}{8}$.*

Applying Algorithm `univsos2`*, we obtain the following two lists of size $6 + 3 = 9$:*

$$\mathtt{c\_list} = \left[\frac{1}{32}, \frac{1}{32}, \frac{913}{15360}, \frac{731}{92160}, \frac{7}{1152}, \frac{1}{32}, \frac{79}{7680}, \frac{1}{576}, 0\right],$$
$$\mathtt{s\_list} = \left[X^3 - \frac{69}{8}X, 7X^2 - \frac{1}{4}X - \frac{63}{8}, 1, X, X^2, X^3, X + \frac{1}{2}, X(X - \frac{1}{2}), X^2(X + \frac{1}{2})\right],$$

yielding the following weighted SOS decomposition:

$$f = \frac{1}{32}\left(X^3 - \frac{69}{8}X\right)^2 + \frac{1}{32}\left(7X^2 - \frac{1}{4}X - \frac{63}{8}\right)^2 + \frac{913}{15360} + \frac{731}{92160}X^2 + \frac{7}{1152}X^4 + \frac{1}{32}X^6 + \frac{79}{7680}\left(X + \frac{1}{2}\right)^2 + \frac{1}{576}X^2\left(X - \frac{1}{2}\right)^2.$$

4.2. Bitsize of the output

First, we need the following auxiliary result:

Lemma 21. *Let $p \in \mathbb{Z}[X]$ be a positive polynomial over $\mathbb{R}$, with $\deg p = n$ and τ an upper bound on the bitsize of the coefficients of p. Then, one has*

$$\inf_{x \in \mathbb{R}} p(x) > (n 2^\tau)^{-n+2} 2^{-n \log_2 n - n\tau}.$$

Proof. Denoting by τ' the maximum bitsize of the coefficients of p' and picking an α such that $p(\alpha) = \inf_{x \in \mathbb{R}} p(x)$ (i.e., a global minimizer of p), Q with p and A with p' in the third item of [26, Lemma 3.2], one obtains

$$\inf_{x \in \mathbb{R}} p(x) > (n 2^\tau)^{-n+2} 2^{-n\tau'}.$$

Now note that $\tau' \leq \log_2 n + \tau$. Using this inequality in the one above concludes the proof. □

Lemma 22. *Let $p \in \mathbb{Z}[X]$ be a positive polynomial over $\mathbb{R}$, with $\deg p = n = 2k$ and let τ be an upper bound on the bitsize of the coefficients of p. Then there exists a positive integer $N = O(n \log_2 n + n\tau)$ such that for all $N' \geq N$ the following holds. For $\varepsilon(N') := \frac{1}{2^{N'}}$, the polynomial $p_{\varepsilon(N')} := p - \varepsilon(N') \sum_{i=0}^{k} X^{2i}$ is positive over $\mathbb{R}$.*

Proof. Let us first consider the polynomial $r := p - \frac{L}{2} \sum_{i=0}^{k} X^{2i}$, where L is the leading coefficient of p. Using Lemma 6, the absolute value of each real root of the polynomial $2r$ is bounded by $2^{\tau(2r)} + 1 \leq 2^{\tau+1}$. By defining $R := 2^{\tau+1}$, it follows that r is positive for $|x| > R$. In addition, for all positive $N \in \mathbb{N} \setminus \{0\}$ and $\varepsilon(N) = \frac{1}{2^N}$, one has $\varepsilon(N) \leq \frac{1}{2} \leq \frac{L}{2}$ and $p_{\varepsilon(N)} = p - \varepsilon(N) \sum_{i=0}^{k} X^{2i} \geq p - \frac{L}{2} \sum_{i=0}^{k} X^{2i} = r$, which implies that $p_{\varepsilon(N)}$ is also positive for $|x| > R$. Observe also that for all $N' \geq N$, $p_{\varepsilon(N')}$ is also positive for $|x| > R$.

Since $R = 2^{\tau+1} > 1$, one has $1 + R^2 + R^4 + \cdots + R^n < nR^n$. Let us choose the smallest positive integer N such that $nR^n \leq 2^N \inf_{|x| \leq R} p$. This implies that $\varepsilon(N) < \frac{\inf_{|x|\leq R} p}{1+R^2+R^4+\cdots+R^n}$, which ensures that the polynomial p_ε is also positive for all $|x| \leq R$. Note also that for all $N' \geq N$, $p_{\varepsilon(N')}$ is also positive for all $|x| \leq R$ since $\varepsilon(N') \leq \varepsilon(N)$. .

Now, applying Lemma 21, we obtain the following upper bound:

$$2^N \leq nR^n (n2^\tau)^{n-2} 2^{n \log_2 n + n\tau} = n2^{n(\tau+1)} (n2^\tau)^{n-2} 2^{n \log_2 n + n\tau}.$$

We straightforwardly deduce that $N = O(n \log_2 n + n\tau)$. □

In the sequel, we denote by $z_1, \ldots, z_n$ the (not necessarily distinct) complex roots of the polynomial p_ε. Assuming that we approximate each complex root with a relative precision of δ, we shall say that $\hat{z_1}, \ldots, \hat{z_n}$ are approximations to the roots of p_ε if we can write $\hat{z_i} = z_i(1 + e_i)$, with $|e_i| \leq 2^{-\delta}$, for all $i = 1, \ldots, n$.

Theorem 23. *Let $f \in \mathbb{Z}[X]$ be a positive polynomial over $\mathbb{R}$, with $\deg f = n$ and τ an upper bound on the bitsize of the coefficients of f. Then the maximal bitsize of the weights and coefficients involved in the weighted SOS decomposition of f obtained with Algorithm* `univsos2` *is bounded from above by $O(n^3 + n^2\tau)$.*

Proof. Let p be the square-free part of the polynomial f (see Step 1 of Algorithm `univsos2`). Then by using Lemma 2, one has $\tau(p) \leq n + \tau + \log_2(n + 1) = O(n + \tau)$.

Let $\varepsilon = \frac{1}{2^N}$ be as in Lemma 22, so that the polynomial $p_\varepsilon = p - \varepsilon \sum_{i=0}^{k} X^{2i}$ is positive over $\mathbb{R}$. By Lemma 22, one can take $N = C(n^2 + n\tau)$ for any large enough constant $C > 1$. Let us write $p_\varepsilon = \sum_{i=0}^{n} a_i X^i$ with $a_n = \ell$ and prove that a precision of $\delta := N + \log_2(5n\|p\|_\infty) = C(n^2 + n\tau) + \log_2(5n\tau)$ is large enough to ensure that the coefficients of u satisfy $\varepsilon \geq \frac{|u_{2i+1}|}{4} - u_{2i} + |u_{2i-1}|$, for all $i = 0, \ldots, k$. First, note that $e := 2^{-\delta} < \frac{1}{\delta} < \frac{1}{Cn(n+\tau)} < \frac{1}{n(n+1)}$ holds. By using Vieta's formulas provided in Lemma 8, one has for all $j = 1, \ldots, n$:

$$\sum_{1 \leq i_1 < \cdots < i_j \leq n} z_{i_1} \cdots z_{i_j} = (-1)^j \frac{a_{n-j}}{\ell}.$$

Then one has for all $j = 1, \ldots, n$:

$$u_{n-j} = \ell \sum_{1 \leq i_1 < \cdots < i_j \leq n} (z_{i_1} \cdots z_{i_j} - \hat{z}_{i_1} \cdots \hat{z}_{i_j}) = \ell \sum_{1 \leq i_1 < \cdots < i_j \leq n} z_{i_1} \cdots z_{i_j} \big(1 - (1 + e_{i_1}) \cdots (1 + e_{i_j})\big).$$

Since $e < \frac{1}{n}$, one can apply [16, Lemma 3.3], which yields $\prod_{1\leq i_1<\cdots<i_j\leq n}(1+e_{i_j}) \leq 1+\theta_j$, with $|\theta_j| \leq \frac{je}{1-je}$. In addition, one has $(j+1)e - \frac{je}{1-je} = \frac{e(1-j(j+1)e)}{1-je} \geq 0$ since $e < \frac{1}{n(n+1)} \leq \frac{1}{j(j+1)}$, for all $j = 1,\ldots,n$. Hence, one has $|u_{n-j}| \leq |a_{n-j}|(j+1)e \leq e\|p_\varepsilon\|_\infty(j+1)$, for all $j = 1,\ldots,n$.

This implies that for all $i = 0,\ldots,k$:

$$\frac{|u_{2i+1}|}{4} - u_{2i} + |u_{2i-1}| \leq e\|p_\varepsilon\|_\infty(\frac{n-2i}{4} + (n-(2i-1)) + (n-(2i-2)) \leq 5ne\|p_\varepsilon\|_\infty \leq 5ne\|p\|_\infty .$$

Since $\delta = N + \log_2(5n\|p\|_\infty)$, one has $5ne\|p\|_\infty = \varepsilon$. Thus, for all $i = 0,\ldots,k$, $\varepsilon \geq \frac{|u_{2i+1}|}{4} - u_{2i} + |u_{2i-1}|$ holds with $\delta = O(n^2 + n\tau + \log_2 n + n + \tau) = O(n^2 + n\tau)$.

Choosing $e_j = e = 2^{-\delta}$ and $\hat{z}_j = z_j(1+2^{-\delta})$ yields $|u_{n-j}| = |a_{n-j}||1-(1+2^{-\delta})^j|$, for all $j = 1,\ldots,n$. Next, we bound the size of the weighted SOS decomposition. One has $\tau(\delta) = O(n^2 + n\tau)$, and for all $i = 1,\ldots,n$, $\tau(a_{n-i}) \leq \tau(\varepsilon) = O(n^2+n\tau)$. Therefore, for all $j = 1,\ldots,n$, $\tau(u_{n-j}) \leq O(n^2 + n\tau + j(n^2+n\tau))$ and the maximal bitsize of the coefficients of u is bounded by $O(n^3 + n^2\tau)$.

From Lemma 6, one has $|\hat{z}_j| = |z_j|(1+2^{-\delta}) \geq \frac{1}{2^{\tau(p_\varepsilon)+1}}(1+2^{-\delta}) \geq \frac{1}{2^{\tau(p_\varepsilon)+\delta+1}}$, so that it is enough to perform root isolation for the polynomial p_ε with a precision bounded from above by $O(\tau(p_\varepsilon)+\delta) = O(n^2+n\tau)$.

Finally, the weighted SOS decomposition of f has weights and coefficients of maximal bitsize bounded by $O(n^3+n^2\tau)$, as claimed. □

4.3. Bit complexity analysis

Theorem 24. *Let $f \in \mathbb{Z}[X]$ be a positive polynomial over $\mathbb{R}$, with $\deg f = n = 2k$ and τ an upper bound on the bitsize of the coefficients of f. Then, on input f, Algorithm `univsos2` runs in $\widetilde{O}(n^4 + n^3\tau)$ boolean operations.*

Proof. By Lemma 3, the square-free decomposition of f can be computed using an expected number of $\widetilde{O}(n^2\tau)$ boolean operations. Checking that the polynomial p_ε has no real root can be performed using an expected number of $\widetilde{O}(n^2\cdot\tau(\varepsilon)) = \widetilde{O}(n^3\tau)$ boolean operations while relying on Sylvester-Habicht Sequences [23, Corollary 5.2].

As seen in the proof of Theorem 23, the complex roots of p_ε must be approximated with isolating intervals (resp. disks) of radius less than $2^{-\tau(p_\varepsilon)-\delta}$. Thus, by Lemma 7, all real (resp. complex) roots of p_ε can be computed in $\widetilde{O}(n^3 + n^2\tau(p_\varepsilon) + n(\delta+\tau(p_\varepsilon)) = \widetilde{O}(n^4+n^3\tau)$ boolean operations.

As in the proof of Theorem 23, one can select $|u_{n-j}| = |a_{n-j}||1-(1+2^{-\delta})^j|$, for all $j = 1,\ldots,n$. This implies that the computation of each coefficient of u can be performed with at most $\widetilde{O}(n\cdot\tau(\delta)) = \widetilde{O}(n^3+n^2\tau)$ boolean operations. Eventually, we obtain a bound of $\widetilde{O}(n^4+n^3\tau)$ for the computation of all coefficients of u, which yields the desired result. □

We state now the complexity result for checking the SOS certificates output by Algorithm `univsos2`. As for the output of Algorithm `univsos1`, this is done through evaluation of the output at $n+1$ distinct values where n is the degree of the output.

Theorem 25. *Let $f \in \mathbb{Z}[X]$ be a positive polynomial over $\mathbb{R}$, with $\deg f = n = 2k$ and τ an upper bound on the bitsize of the coefficients of f. Then one can check the correctness of the weighted SOS decomposition of f obtained with Algorithm `univsos2` using $\widetilde{O}(n^4+n^3\tau)$ bit operations.*

Proof. From [8, Corollary 8.27], the cost of multiplying polynomials in $\mathbb{Z}[X]$ of degree less than n with coefficients of bitsize less than l is bounded by $\tilde{O}(n \cdot l)$. By Theorem 23, the maximal coefficient bitsize of the SOS decomposition of f obtained with Algorithm `univsos2` is bounded from above by $l = O(n^3 + n^2\tau)$. Therefore, from [8, Corollary 10.8], the cost of the evaluation of this decomposition at n points can be performed using at most $\tilde{O}(n \cdot (n^3 + n\tau))$ boolean operations, as claimed. ☐

5. Practical experiments

Now we present experimental results obtained by applying Algorithm `univsos1` and Algorithm `univsos2`, respectively, presented before in Sections 3 and 4. Both algorithms have been implemented in a tool, called `univsos`, written in Maple version 16. The interested reader can find more details about installation and benchmark execution on the dedicated webpage.[3] This tool is integrated into the RAGlib Maple package[4]. We obtained all results on an Intel Core i7-5600U CPU (2.60 GHz) with 16Gb of RAM. SOS decomposition (resp. verification) times are provided after averaging over five (resp., one thousand) runs.

As mentioned in [5, Section 6], the SOS decomposition performed by Algorithm `univsos2` has been implemented using the PARI/GP software tool[5] and is freely available (see [5]). To ensure fair comparison, we have rewritten this algorithm in Maple. To compute approximate complex roots of univariate polynomials, we rely on the PARI/GP procedure `polroots` through an interface with our Maple library. We also tried to use the Maple procedure `fsolve`, but the `polroots` routine from Pari/GP yielded significantly better performance for the polynomials involved in our examples.

The nine polynomial benchmarks presented in Table 1 allow to approximate some given mathematical functions, considered in [5, Section 6]. Computation and verification of SOS certificates are a mandatory step required to validate the supremum norm of the difference between such functions and their respective approximation polynomials on given closed intervals. This boils down to certifying two inequalities of the form $\forall x \in [b, c], p(x) \geq 0$, with $p \in \mathbb{Q}[X]$, $b, c \in \mathbb{Q}$ and $\deg p = n$. As explained in [5, Section 5.2.5], this latter problem can be addressed by computing a weighted SOS decomposition of the polynomial $q(Y) := (1 + Y^2)^n \, p\Big(\frac{b+cY^2}{1+Y^2}\Big)$, with either Algorithm `univsos1` or Algorithm `univsos2`. For each benchmark, we indicate in Table 1 the degree n and the bitsize τ of the input polynomial, the bitsize τ_1 of the weighted SOS decomposition provided by Algorithm `univsos1` as well as the corresponding computation (resp. verification) time t_1 (resp. t'_1) in milliseconds. Similarly, we display τ_2, t_2, t'_2 for Algorithm `univsos2`. The table results show that for all other eight benchmarks, Algorithm `univsos2` yields better certification and verification performance, together with more concise SOS certificates. This observation confirms what we could expect after comparing the theoretical complexity results from Sections 3 and 4.

The five benchmarks from Table 2 are related to problems arising in verification of digital filters against frequency specifications (see [38, Section III B)]). As for the problems from Table 1, computation and verification of SOS certificates are mandatory to show the non-negativity of a polynomial, which allows one in turn to validate the bounds of a rational function. By

---

[3] `https://github.com/magronv/univsos`
[4] `http://www-polsys.lip6.fr/~safey/RAGLib/`
[5] `http://pari.math.u-bordeaux.fr`

Table 1: Comparison results of output size and performance between Algorithm `univsos1` and Algorithm `univsos2` for non-negative polynomial benchmarks from [5].

Id	n	τ (bits)	`univsos1`			`univsos2`		
			τ_1 (bits)	t_1 (ms)	t'_1 (ms)	τ_2 (bits)	t_2 (ms)	t'_2 (ms)
# 1	13	22 682	3 403 218	2 723	0.40	51 992	824	0.14
# 3	32	269 958	11 613 480	13 109	1.18	580 335	2 640	0.68
# 4	22	47 019	1 009 507	4 063	1.45	106 797	1 776	0.31
# 5	34	117 307	8 205 372	102 207	20.1	265 330	5 204	0.60
# 6	17	26 438	525 858	1 513	0.74	59 926	1 029	0.21
# 7	43	67 399	62 680 827	217 424	48.1	152 277	11 190	0.87
# 8	22	27 581	546 056	1 979	0.77	63 630	1 860	0.38
# 9	20	30 414	992 076	964	0.44	68 664	1 605	0.25
# 10	25	42 749	3 146 982	1 100	0.38	98 926	2 753	0.39

Table 2: Comparison results of output size and performance between Algorithm `univsos1` and Algorithm `univsos2` for non-negative polynomial benchmarks from [38].

Id	n	τ (bits)	`univsos1`			`univsos2`		
			τ_1 (bits)	t_1 (ms)	t'_1 (ms)	τ_2 (bits)	t_2 (ms)	t'_2 (ms)
# A		290 265	579 515	1 184	2.27	294 745	7 553	1.14
# B		290 369	579 720	1 008	2.25	294 803	7 543	0.99
# C	40	282 964	539 693	428	1.01	589 939	9 080	6.21
# D		289 630	552 702	500	1.14	596 604	8 902	0.62
# E		279 304	19 389 110	17 024	1.26	604 918	20 161	0.69

contrast with the comparison results from Table 1, Algorithm `univsos1` is faster for all examples. In addition, Algorithm `univsos1` produces output certificates of smaller size, compared to Algorithm `univsos2`, on the two benchmarks # C and # D. For all three other benchmarks, Algorithm `univsos2` provides more concise certificates. The slower performance of Algorithm `univsos2` is due to the time spent to obtain accurate approximations of the polynomial roots.

The comparison results available in Table 3 are obtained for power sums of increasing degrees. For a given natural number $n = 2k$ with $10 \leq n \leq 500$, we consider the polynomial $P_n := 1 + X + \cdots + X^n$. The roots of this polynomial are the $(n+1)$-st roots of unity, thus yielding the following SOS decomposition with real coefficients: $P_n := \prod_{j=1}^{k}((X - \cos\theta_j)^2 + \sin^2\theta_j)$, with $\theta_j := \frac{2j\pi}{n+1}$, for each $j = 1, \ldots, k$. By contrast with the benchmarks from Table 1, Table 3 shows that Algorithm `univsos1` produces output certificates of much smaller size compared to Algorithm `univsos2`, with a bitsize ratio lying between 6 and 38 for values of n between 10 and 200. This is due to the fact that Algorithm `univsos1` outputs a value of t equal to 0 at each step. The execution performance of Algorithm `univsos1` is also much better in this case. The lack of efficiency of Algorithm `univsos2` is due to the computational bottleneck occurring when obtaining an accurate approximation of the relatively close roots $\cos\theta_j \pm i \sin\theta_j$, $j = 1, \ldots, k$. For $n \geq 300$, the execution of Algorithm `univsos2` did not succeed after two hours of computation, as indicated by the symbol $-$ in the corresponding line.

Table 3: Comparison results of output size and performance between Algorithm `univsos1` and Algorithm `univsos2` for non-negative power sums of increasing degrees.

n	`univsos1`			`univsos2`		
	τ_1 (bits)	t_1 (ms)	t'_1 (ms)	τ_2 (bits)	t_2 (ms)	t'_2 (ms)
10	84	7	0.03	567	264	0.03
20	195	10	0.05	1 598	485	0.06
40	467	26	0.09	6 034	2 622	0.18
60	754	45	0.14	12 326	6 320	0.32
80	1 083	105	0.18	21 230	12 153	0.47
100	1 411	109	0.26	31 823	19 466	0.69
200	3 211	444	0.48	120 831	171 217	2.08
300	5 149	1 218	0.74			
400	7 203	2 402	0.95	–	–	–
500	9 251	4 292	1.19			
1000	20 483	30 738	2.56			

Further experiments are summarized in Table 4 for modified Wilkinson polynomials W_n of increasing degrees $n = 2k$ with $10 \leq n \leq 600$ and $W_n := 1 + \prod_{j=1}^{k}(X - j)^2$. The roots $j = 1, \dots, k$ of $W_n - 1$ are relatively close (i.e.,the difference between two consecutive roots is small by comparison with the size of the coefficients), which yields again significantly slower performance of Algorithm `univsos2`. As observed in the case of power sums, timeout behaviors occur for $n \geq 60$. In addition, the bitsize of the SOS decompositions returned by Algorithm `univsos1` are much smaller. This is a consequence of the fact that in this case, $a = 1$ is the smallest global minimizer of W_n. Hence the algorithm always terminates at the first iteration by returning the trivial quadratic approximation $f_t = f_a = 1$ together with the square-free decomposition of $W_n - f_t = \prod_{j=1}^{k}(X - j)^2$.

Table 4: Comparison results of output size and performance between Algorithm `univsos1` and Algorithm `univsos2` for modified Wilkinson polynomials of increasing degrees.

n	τ (bits)	`univsos1`			`univsos2`		
		τ_1 (bits)	t_1 (ms)	t'_1 (ms)	τ_2 (bits)	t_2 (ms)	t'_2 (ms)
10	140	47	17	0.01	2 373	751	0.03
20	737	198	31	0.01	12 652	3 569	0.08
40	3 692	939	35	0.01	65 404	47 022	0.17
60	9 313	2 344	101	0.01			
80	17 833	4 480	216	0.01			
100	29 443	7 384	441	0.01			
200	137 420	34 389	3 249	0.01			
300	335 245	83 859	11 440	0.01	–	–	–
400	628 968	157 303	34 707	0.02			
500	1 022 771	255 767	73 522	0.02			
600	1 519 908	380 065	149 700	0.04			

Finally, we consider experimentation performed on modified Mignotte polynomials defined by $M_{n,m} := X^n + 2(101X - 1)^m$ and $N_n := (X^n + 2(101X - 1)^2)(X^n + 2((101 + \frac{1}{101})X - 1)^2)$, for even integers n and $m \geq 2$. The corresponding results are displayed in Table 5 for $M_{n,m}$ with $m = 2$ and $10 \leq n \leq 10000$, $m = n - 2$ and $10 \leq n \leq 100$ as well as for N_n with $10 \leq n \leq 100$. Note that similar benchmarks are used in [36] to anayze the efficiency of (real) root isolation techniques for polynomials with close roots. As for modified Wilkinson polynomials, Algorithm `univsos2` can only handle instances of small size, due to the limited scalability of the `polroots` procedure. In this case, Algorithm `univsos1` computes the approximation $t = \frac{1}{100}$ of the unique global minimizer of $M_{n,2}$. Thus, Algorithm `univsos1` always outputs weighted SOS decompositions of polynomials $M_{n,2}$ within a single iteration by first computing the quadratic polynomial $f_t = 2(101X - 1)^2$ and the trivial square-free decomposition $W_n - f_t = X^n$. In the absence of such minimizers, Algorithm `univsos1` can only handle instances of polynomials $M_{n,n-2}$ and N_n with moderate degree (less than 100).

Table 5: Comparison results of output size and performance between Algorithm `univsos1` and Algorithm `univsos2` for modified Mignotte polynomials of increasing degrees.

Id	n	τ (bits)	`univsos1`			`univsos2`		
			τ_1 (bits)	t_1 (ms)	t_1' (ms)	τ_2 (bits)	t_2 (ms)	t_2' (ms)
$M_{n,2}$	10	27	23	2	0.01	4 958	1 659	0.04
	10^2			3		−	−	−
	10^3			85				
	10^4			3 041				
$M_{n,n-2}$	10	288	25 010	21	0.03	6 079	2 347	0.04
	20	1 364	182 544	138	0.04	26 186	10 922	0.06
	40	5 936	1 365 585	1 189	0.13	−	−	−
	60	13 746	4 502 551	4 966	0.33			
	100	39 065	20 384 472	38 716	1.66			
N_n	10	212	25 567	27	0.04	−	−	−
	20		189 336	87	0.05			
	40		5 027 377	1 704	0.17			
	60		16 551 235	8 075	0.84			
	100		147 717 572	155 458	11.1			

6. Conclusion and perspectives

We presented and analyzed two different algorithms, `univsos1` and `univsos2`, to compute weighted sum of squares (SOS) decompositions of non-negative univariate polynomials. When the input polynomial has rational coefficients, one feature shared by both algorithms is their ability to provide non-negativity certificates whose coefficients are also rational. Our study shows that the complexity analysis of Algorithm `univsos1` yields an upper bound that is exponential w.r.t. the input degree, while the complexity of Algorithm `univsos2` is polynomial. However, comparison benchmarks emphasize the need for both algorithms to handle various classes of non-negative polynomials, e.g., in the presence of rational global minimizers or when root isolation can be performed efficiently.

A first direction of further research is a variant of Algorithm `univsos2` where one would compute approximate SOS decompositions of perturbed positive polynomials by using semidefinite programming (SDP) instead of root isolation. Preliminary experiments yield very promising results when the bitsize of the polynomials is small, e.g., for power sums of degree up to 1000. However, the performance decreases when the bitsize becomes larger, either for polynomial benchmarks from [5] or modified Wilkinson polynomials. At the moment, we are not able to provide any SOS decomposition for all such benchmarks. Our SDP-based algorithm relies on the high-precision solver SDPA-GMP ([28]), but it is still challenging to obtain precise values of eigenvalues/vectors of SDP output matrices. Another advantage of this technique is its ability to perform global polynomial optimization. A topic of interest would be to obtain the same feature with the two current algorithms. We also plan to develop extensions to the non-polynomial case.

References

[1] Basu, S., Pollack, R., Roy, M.-F., 1996. On the combinatorial and algebraic complexity of quantifier elimination. Journal of the ACM (JACM) 43 (6), 1002–1045.

[2] Basu, S., Pollack, R., Roy, M.-F., 2006. Algorithms in Real Algebraic Geometry (Algorithms and Computation in Mathematics). Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[3] Boudaoud, F., Caruso, F., Roy, M.-F., 2008. Certificates of Positivity in the Bernstein Basis. Discrete & Computational Geometry 39 (4), 639–655.

[4] Cauchy, A. L. B., 1832. Calcul des indices des fonctions. Journal de l'Ecole Polytechnique 15 (25), 176 – 229.

[5] Chevillard, S., Harrison, J., Joldes, M., Lauter, C., 2011. Efficient and accurate computation of upper bounds of approximation errors. Theoretical Computer Science 412 (16), 1523 – 1543, `https://hal.archives-ouvertes.fr/ensl-00445343v2`.

[6] Collins, G. E., 1975. Quantifier elimination for real closed fields by cylindrical algebraic decompostion. In: Automata Theory and Formal Languages 2nd GI Conference Kaiserslautern, May 20–23, 1975. Springer, pp. 134–183.

[7] Coq, 2016. The Coq Proof Assistant. `http://coq.inria.fr/`.

[8] Gathen, J., Gerhard, J., 1999. Modern Computer Algebra. Cambridge University Press, New York, NY, USA.

[9] Girard, A., 1629. Invention nouvelle en l'algébre. Blauew.

[10] Greuet, A., Safey El Din, M., 2014. Probabilistic algorithm for polynomial optimization over a real algebraic set. SIAM Journal on Optimization 24 (3), 1313–1343.
URL `https://doi.org/10.1137/130931308`

[11] Guo, Q., Safey El Din, M., Zhi, L., 2013. Computing Rational Solutions of Linear Matrix Inequalities. In: Proceedings of the 38th International Symposium on Symbolic and Algebraic Computation. ISSAC '13. ACM, New York, NY, USA, pp. 197–204.

[12] Hales, T., Adams, M., Bauer, G., Dat, D. T., Harrison, J., Truong, H. L., Kaliszyk, C., Magron, V., Mclaughlin, S., Thang, N. T., Truong, N. Q., Nipkow, T., Obua, S., Pleso, J., Rute, J., Solovyev, A., An, T. T. H., Trung, T. N., Diep, T. T., Urban, J., Ky, V. K., Zumkeller, R., 2015. A Formal Proof of the Kepler Conjecture. Submitted.

[13] Harrison, J., 1996. HOL Light: A Tutorial Introduction. In: Srivas, M. K., Camilleri, A. J. (Eds.), FMCAD. Vol. 1166 of Lecture Notes in Computer Science. Springer, pp. 265–269.

[14] Henrion, D., Naldi, S., Din, M. S. E., 2016. Exact Algorithms for Linear Matrix Inequalities. SIAM Journal on Optimization 26 (4), 2512–2539.

[15] Henrion, D., Naldi, S., Safey El Din, M., 2016. Spectra-a maple library for solving linear matrix inequalities in exact arithmetic. arXiv preprint arXiv:1611.01947.

[16] Higham, N., 2002. Accuracy and Stability of Numerical Algorithms: Second Edition. SIAM.

[17] Hong, H., Safey El Din, M., 2012. Variant quantifier elimination. Journal of Symbolic Computation 47 (7), 883–901.

[18] Harrison, J., 2007. Verifying Nonlinear Real Formulas via Sums of Squares. In: Proceedings of the 20th International Conference on Theorem Proving in Higher Order Logics. TPHOLs'07. Springer-Verlag, Berlin, Heidelberg, pp. 102–118.

[19] Kaltofen, E. L., Li, B., Yang, Z., Zhi, L., 2012. Exact certification in global polynomial optimization via sums-of-squares of rational functions with rational coefficients. Journal of Symbolic Computation 47 (1), 1 – 15.

[20] Knapp, A. W., 2006. Basic Algebra, 1st Edition. Birkhäuser Basel.

[21] Landau, E., 1906. ber die darstellung definiter funktionen durch quadrate. Mathematische Annalen 62, 272–285.
URL `http://eudml.org/doc/158257`

[22] Lasserre, J.-B., 2001. Global optimization with polynomials and the problem of moments. SIAM Journal on Optimization 11 (3), 796–817.

[23] Lickteig, T., Roy, M.-F., 2001. Sylvesterhabicht sequences and fast cauchy index computation. Journal of Symbolic Computation 31 (3), 315 – 341.

[24] Magron, V., Allamigeon, X., Gaubert, S., Werner, B., 2015. Formal proofs for Nonlinear Optimization. Journal of Formalized Reasoning 8 (1), 1–24.

[25] Mehlhorn, K., Sagraloff, M., Wang, P., Jan. 2015. From Approximate Factorization to Root Isolation with Application to Cylindrical Algebraic Decomposition. J. Symb. Comput. 66, 34–69.

[26] Melczer, S., Salvy, B., 2016. Symbolic-Numeric Tools for Analytic Combinatorics in Several Variables. In: Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation. ISSAC '16. ACM, New York, NY, USA, pp. 333–340.
URL `http://doi.acm.org/10.1145/2930889.2930913`

[27] Mignotte, M., 1992. Mathematics for Computer Algebra. Springer-Verlag New York, Inc., New York, NY, USA.

[28] Nakata, M., Sept 2010. A numerical evaluation of highly accurate multiple-precision arithmetic version of semidefinite programming solver: SDPA-GMP, -QD and -DD. In: Computer-Aided Control System Design (CACSD), 2010 IEEE International Symposium on. pp. 29–34.

[29] Parrilo, P., 2000. Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization. Ph.D. thesis, California Institute of Technology.

[30] Peyrl, H., Parrilo, P., 2008. Computing sum of squares decompositions with rational coefficients. Theor. Comput. Sci. 409 (2), 269–281.

[31] Pourchet, Y., 1971. Sur la représentation en somme de carrés des polynômes à une indéterminée sur un corps de nombres algébriques. Acta Arithmetica 19 (1), 89–104.
URL `http://eudml.org/doc/205020`

[32] Prestel, A., Delzell, C., 2001. Positive Polynomials: From Hilbert's 17th Problem to Real Algebra. Springer Monographs in Mathematics. Springer Berlin Heidelberg.
URL `https://books.google.fr/books?id=29grNpTVWX8C`

[33] Rantzer, A., Parrilo, P. A., 2000. On convexity in stabilization of nonlinear systems. In: Proceedings of the 39th IEEE Conference on Decision and Control. Vol. 3. pp. 2942–2945.

[34] Safey El Din, M., Zhi, L., 2010. Computing rational points in convex semialgebraic sets and sum of squares decompositions. SIAM Journal on Optimization 20 (6), 2876–2889.

[35] Schweighofer, M., 1999. Algorithmische Beweise für Nichtnegativ- und Positivstellensätze. Master's thesis, Diplomarbeit an der Universität Passau.

[36] Strzebonski, A., Tsigaridas, E., 2011. Univariate Real Root Isolation in an Extension Field. In: Proceedings of the 36th International Symposium on Symbolic and Algebraic Computation. ISSAC '11. ACM, New York, NY, USA, pp. 321–328.
URL `http://doi.acm.org/10.1145/1993886.1993934`

[37] Swokowski, W., 1989, pages 216 - 221. Fundamentals of College Algebra. PWS-Kent Pub. Co.

[38] Volkova, A., Lauter, C., Hilaire, T., 2017. Reliable verification of digital implemented filters against frequency specifications. In: 24th IEEE Symposium on Computer Arithmetic. IEEE.

[39] Yun, D. Y., 1976. On Square-free Decomposition Algorithms. In: Proceedings of the Third ACM Symposium on Symbolic and Algebraic Computation. SYMSAC '76. ACM, New York, NY, USA, pp. 26–35.
URL `http://doi.acm.org/10.1145/800205.806320`