Journal of Software for Algebra and Geometry

Implementing real polyhedral homotopy

KISUN LEE, JULIA LINDBERG AND JOSE ISRAEL RODRIGUEZ

vol 14 2024

JSAG **14** (2024), 59–71
https://doi.org/10.2140/jsag.2024.14.59

The Journal of Software for
Algebra and Geometry

Implementing real polyhedral homotopy

KISUN LEE, JULIA LINDBERG AND JOSE ISRAEL RODRIGUEZ

ABSTRACT: We implement a real polyhedral homotopy method using three functions. The first function provides a certificate that our real polyhedral homotopy is applicable to a given system; the second function generates binomial systems for a start system; and the third function outputs target solutions from the start system obtained by the second function. This work realizes the theoretical contributions of Ergür and Wolff (2023) as easy-to-use functions, allowing for further investigation into real homotopy algorithms.

1. INTRODUCTION. Finding all points in a zero-dimensional algebraic variety is an important problem in many applications in the sciences. This problem amounts to solving a polynomial system of equations with finitely many complex (real or nonreal) solutions. Many types of algorithms have been proposed using both symbolic and/or numerical techniques. A popular family of numerical algorithms are called *homotopy continuation* algorithms. These algorithms work by continuously deforming the solutions from an "easy" polynomial system into the desired one. While there exist many off-the-shelf homotopy continuation solvers that find all complex solutions, many applications, such as to power systems engineering [15], economics [12], and statistics [16], only require knowledge of the real solutions. In general, there are many more complex solutions than real ones, leading to wasted computation. For this reason, the problem of developing an efficient homotopy that finds only the real solutions to a polynomial system is an incredibly important open problem in the field of numerical algebraic geometry [1].

Recent work tackles this problem by presenting an algorithm that certifiably finds all real solutions as long as an inequality based on the geometry of the polynomial system is satisfied [8]. This work relies heavily on mathematical objects from tropical geometry [17]. We implement this algorithm in a `Julia` package, giving the first homotopy based software package that can provably find all real solutions to a patchworked polynomial system without first finding all complex solutions.

We review some of the mathematical concepts behind the algorithms for the functions in Section 2. In Section 3, we highlight the key proposition from [8] needed for construction of the *real polyhedral homotopy*. In Section 4, we describe our implementation of the real polyhedral homotopy, which relies on

The research of Rodriguez was supported by the Office of the Vice Chancellor for Research and Graduate Education at the University of Wisconsin Madison with funding from the Wisconsin Alumni Research Foundation.

MSC2020: primary 65H14; secondary 14P99.

Keywords: homotopy continuation, numerical algebraic geometry, real algebraic geometry, amoeba, discriminant, polyhedral homotopy.

`RealPolyhedralHomotopy` version 1.2.1

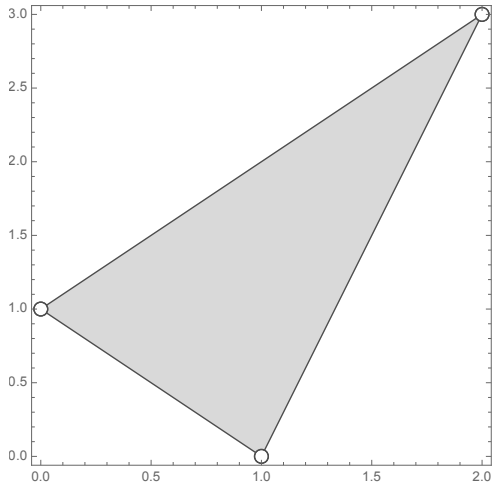

Figure 1. The polytope A^w from Example 2.1.

three functions: `certify_patchwork`, `generate_binomials`, and `rph_track`. The code is available as the `Julia` package `RealPolyhedralHomotopy.jl`.

2. PRELIMINARIES.

2A. ***Regular subdivisions and triangulations.*** Let $A \subset \mathbb{Z}^n$ be a set of integer lattice points with convex hull $Q = \mathrm{conv}(A)$. A function $w : A \to \mathbb{R}$ assigning a real number to each lattice point in A is called a *lifting function*. Denote the collection of lifted points $(a, w(a)) \in A \times \mathbb{R}$ by A^w. The convex hull of A^w is a polytope, and projecting the lower faces of $\mathrm{conv}(A^w)$ onto the first n-coordinates induces a *polyhedral subdivision* Δ_w of Q, i.e., all cells in Δ_w are polyhedral. When all cells of Δ_w are simplices, it is called a *triangulation*. If a polyhedral subdivision (or triangulation, respectively) is induced by a lifting function, we say that the subdivision (or triangulation, respectively) is *regular*. For a triangulation Δ_w of $Q = \mathrm{conv}(A)$, we define the *secondary cone* $\mathcal{C}(\Delta_w)$ of Δ_w as the collection of lifting functions that induce the same regular triangulation. Specifically,

$$\mathcal{C}(\Delta_w) = \{v \in \mathbb{R}^{|A|} \mid \Delta_v = \Delta_w\}.$$

Example 2.1. Consider $A = \{0, 1, 2\} \subset \mathbb{Z}$ with lifting function $w(0) = 1$, $w(1) = 0$, and $w(2) = 3$. A picture of A^w is given in Figure 1. The lower faces of $\mathrm{conv}(A^w)$ are the line segments $\mathrm{conv}\{(0, 1), (1, 0)\}$ and $\mathrm{conv}\{(1, 0), (2, 3)\}$, which induce a regular triangulation of A, namely $\Delta_w = \{\{0, 1\}, \{1, 2\}\}$. In this case, we know that any lifting function $w = (w_0, w_1, w_2)$ induces the same triangulation if $w_0 \geq w_1$ and $w_2 \geq w_1$ when w_0, w_1, w_2 are not all equal. Therefore, we have

$$\mathcal{C}(\Delta_w) = \left\{(w_0, w_1, w_2) \in \mathbb{R}^3 \mid w_0 \geq w_1 \text{ and } w_2 \geq w_1, \text{ where } w_0, w_1, w_2 \text{ are not all equal}\right\}.$$

2B. ***Cayley configurations and mixed cells.*** For sets of lattice points $A_1, \ldots, A_m$ in $\mathbb{Z}^n$, consider the set

$$A_1 * \cdots * A_m := \left\{(a_i, e_{n+i}) \in \mathbb{Z}^{n+m} \mid a_i \in A_i, i = 1, \ldots, m\right\}$$

of lattice points in $\mathbb{Z}^{n+m}$, where e_{n+i} is the $(n+i)$-th canonical vector. The set $A = A_1 * \cdots * A_m$ is called a *Cayley configuration*. Given a Cayley configuration $A \in \mathbb{Z}^{n+m}$, we lift A with a lifting function w and construct a polyhedral subdivision Δ_w of $\mathrm{conv}(A)$. A cell σ in Δ_w is called *mixed* if σ has exactly two elements from each A_i. For a fixed cell σ, we are interested in all lifting functions w such that σ is a mixed cell of Δ_w. We call such set a *mixed cell cone of* σ, and it is formally defined by

$$M(\sigma) := \left\{ w \in \mathbb{R}^{|A|} \mid \sigma \text{ is a mixed cell in } \Delta_w \right\}.$$

For a triangulation Δ_w of A induced by w, we define the *mixed cell cone of* Δ_w as

$$M(\Delta_w) := \bigcap_{\{\sigma \text{ is a mixed cell in } \Delta_w\}} M(\sigma).$$

Example 2.2. Consider $A = \{0, 1, 2\}$ with lifting function $w = (1, 0, 3)$ as in Example 2.1. Since $m = n = 1$, the Cayley configuration of A is just a translation of A. We observe that Δ_w is mixed since each cell contains exactly two elements from A. Let $\sigma_1 = \{0, 1\} \in \Delta_w$ and $\sigma_2 = \{1, 2\} \in \Delta_w$. Then $M(\sigma_1) = M(\sigma_2)$, giving

$$M(\Delta_w) = \left\{ (w_0, w_1, w_2) \in \mathbb{R}^3 \mid w_0 \geq w_1,\ w_2 \geq w_1,\ w_0, w_1, w_2 \text{ are not all equal} \right\},$$

which is the same as $\mathcal{C}(\Delta_w)$.

2C. ***A-discriminant and its amoeba.*** For a sparse polynomial $f \in \mathbb{C}[x_1, \ldots, x_n]$, we define the (monomial) *support* A_f of f as the set of exponents of all monomials of f. Let $\mathbb{C}^{|A_f|}$ be the set of polynomials with complex coefficients supported on A_f. Then, a polynomial $f \in \mathbb{C}^{|A_f|}$ can be written as

$$f(x) = \sum_{a \in A_f} c_a x^a,$$

where x^a is the monomial $x_1^{a_1} \cdots x_n^{a_n}$ for $a = (a_1, \ldots, a_n)$. For a polynomial $f = \sum_{a \in A_f} c_a x^a$ supported on A_f, let $c_f := (c_a)_{a \in A_f}$ be the coefficient vector for f. For an n-tuple of vectors $C = (c_{f_1}, \ldots, c_{f_n})$, we let $F_C := \langle f_1, \ldots, f_n \rangle$ be a square polynomial system such that c_{f_i} is a coefficient vector for f_i for each $i = 1, \ldots, n$. For a Cayley configuration $A = A_{f_1} * \cdots * A_{f_n}$, we define the *A-discriminant*

$$\nabla_A := \left\{ C \in \mathbb{C}^{|A|} \mid F_C(x) \text{ has a singularity in } (\mathbb{C} \setminus \{0\})^n \right\}.$$

We say that ∇_A is *nondefective* if it is codimension one. We are particularly interested in $\nabla_A(\mathbb{R}) := \nabla_A \cap \mathbb{R}^{|A|}$ because in a connected component of the complement of $\nabla_A(\mathbb{R})$, the number of real solutions to the corresponding polynomial systems is constant.

When an A-discriminant is nondefective, we may consider the Newton polytope of its defining polynomial. For a vertex v of the Newton polytope, its normal cone is denoted by $\mathcal{N}(v)$. For a lifting function w for A, we note the following relations, proved in [8, Lemma 2.16], between the secondary cone $\mathcal{C}(\Delta_w)$, mixed cell cone $M(\Delta_w)$ and the normal cone $\mathcal{N}(v)$ of ∇_A:

$$\mathcal{C}(\Delta_w) \subseteq M(\Delta_w) \subseteq \mathcal{N}(v) \tag{2-1}$$

for the vertex v of the Newton polytope of ∇_A satisfying that $w \in \mathcal{N}(v)$.

The *log-absolute value map* $\operatorname{Log}|\cdot| : (\mathbb{C}\setminus\{0\})^n \to \mathbb{R}^n$ is defined by

$$\operatorname{Log}|(x_1,\dots,x_n)| := (\log|x_1|,\dots,\log|x_n|). \tag{2-2}$$

For a Laurent polynomial $f \in \mathbb{C}[x_1^{\pm},\dots,x_n^{\pm}]$, the image of the variety $\mathcal{V}(f)$ under the log-absolute value map is called the *amoeba* of f and denoted by $\mathcal{A}(f)$. For a textbook reference, see [17, Chapter 1].

We remark that the complement of an amoeba consists of convex regions. For the real polyhedral homotopy algorithm, we will consider a specific lifting function w inducing a triangulation Δ_w such that mixed cell cone $M(\Delta_w)$ is contained in a complement of the amoeba of ∇_A.

2D. ***Homotopy continuation.*** For a system $F = \langle f_1,\dots,f_n\rangle$ of polynomials in n variables, finding all isolated solutions of the system is an important task. *Homotopy continuation* is a method to find numerical approximations of solutions of a system of polynomial equations. The main idea is to track solution paths from a system G called a *start system* whose solutions are known to the *target system* F. For a start system $G = \langle g_1,\dots,g_n\rangle$ with the same number of variables and equations of F, we construct a homotopy $H(x,t)$ such that $H(x,0) = G$ and $H(x,1) = F$. To track the solutions from $G = 0$ as t varies from zero to one, a common approach is to use *predictor-corrector* methods. These methods rely on numerically solving an ordinary differential equation, called the Davidenko equation, as well as using Newton iterations. For details, see [21, Chapter 2].

2E. ***Polyhedral homotopy continuation and Bernstein's theorem.*** In order to choose a start system G for a homotopy continuation algorithm, we want the number of solutions of $G(x) = 0$ to be roughly equal to the number of solutions of $F(x) = 0$. In this paper, the *polyhedral homotopy continuation* established by Huber and Sturmfels [10] is considered.

For polytopes P, Q in $\mathbb{R}^n$, the *Minkowski sum* of the polytopes is defined as

$$P + Q = \{p + q \mid p \in P,\ q \in Q\}.$$

For a nonnegative number a and polytope P, we define $aP := \{ap \mid p \in P\}$. The Euclidean volume of the Minkowski sum $\operatorname{Vol}(a_1Q_1 + \cdots + a_nQ_n)$ is a homogeneous polynomial in variables $a_1,\dots,a_n$. The coefficient of the mixed term $a_1a_2\cdots a_n$ of the polynomial $\operatorname{Vol}(a_1Q_1 + \cdots + a_nQ_n)$ is called the *mixed volume* of $Q_1,\dots,Q_n$ and is denoted by $\operatorname{MVol}(Q_1,\dots,Q_n)$. The following celebrated theorem relates the mixed volume of the Newton polytopes of a polynomial system to the number of isolated solutions in the torus.

Theorem 2.3 (Bernstein's theorem [2, Theorem A]). *Let F be a system of polynomials $f_1,\dots,f_n$ in $\mathbb{C}[x_1,\dots,x_n]$. For Newton polytopes Q_{f_i} for each f_i, we have*

$$\big(\textit{the number of isolated solutions of } F \textit{ in } (\mathbb{C}\setminus\{0\})^n\big) \le \operatorname{MVol}(Q_{f_1},\dots,Q_{f_n}).$$

Furthermore, for polynomials $f_1,\dots,f_n$ with generic coefficients the inequality is tight.

Polyhedral homotopy continuation tracks $\operatorname{MVol}(Q_{f_1},\dots,Q_{f_n})$ paths to find all solutions to $F = \langle f_1,\dots,f_n\rangle = 0$. The idea is to construct a collection of binomial start systems whose number of

solutions sum to $\mathrm{MVol}(Q_{f_1}, \ldots, Q_{f_n})$. We now describe how to find one such system, G. Consider a polynomial

$$f(x) = \sum_{a \in A} c_a x^a$$

supported on A. Consider a lifting function w. By multiplying each monomial of f by $t^{w(a)}$, we have the *lifted polynomial*

$$\bar{f}(x, t) = \sum_{a \in A} c_a x^a t^{w(a)}.$$

Suppose that a square polynomial system F consists of polynomials $f_1, \ldots, f_n$ supported on $A_{f_1}, \ldots, A_{f_n}$, respectively. Lifting all polynomials in F gives the lifted system $\bar{F}(x, t)$ which satisfies $\bar{F}(x, 1) = F$. The solutions of $\bar{F}$ can be expressed by $x(t) = (x_1(t), \ldots, x_n(t))$, where each $x_i(t)$ is a Puiseux series and

$$x_i(t) = t^{\alpha_i} y_i + (\text{higher order terms in } t)$$

for some rational number α_i and a nonzero constant y_i. Plugging $x(t)$ into polynomials gives

$$\bar{f}_j(x(t), t) = \sum_{a \in A_{f_j}} c_a y^a t^{\langle a, \alpha \rangle + w(a)} + (\text{higher order terms in } t),$$

where $y^a = y_1^{a_1} \cdots y_n^{a_n}$. For the minimum value of $\langle a, \alpha \rangle + w(a)$ over all $a \in A_{f_j}$, divide by $t^{\langle a, \alpha \rangle + w(a)}$ and set $t = 0$. Iterating this for each $\bar{f}_j$, we have a start system G. Note that for a fixed lifting function w, the minimum can be obtained from different monomials $a \in A_{f_j}$ depending on α. Therefore, we may have several start systems G and they induce a collection of start systems.

For most choices of α, the procedure outlined above gives a start system consisting of monomials, which is not useful since monomial systems of equations have no solutions in the torus. Instead, one chooses α carefully to get a binomial start system since general binomial systems of polynomial equations have solutions in the torus and can be solved efficiently using linear algebra [7]. Such an α and the corresponding binomial system can be obtained from a mixed cell of a triangulation Δ_w of $\mathrm{conv}(A)$ induced by w.

We summarize this section with the following definition:

Definition 2.4 (polyhedral homotopy). For a polynomial system $F = \langle f_1, \ldots, f_n \rangle$, with $f_i = \sum_{a \in A_{f_i}} c_a x^a$, define the *polyhedral homotopy* as

$$H_m(x, t) := \begin{cases} \sum_{a \in A_{f_1}} t^{m_{1,a}} c_a x^a, \\ \quad \vdots \\ \sum_{a \in A_{f_n}} t^{m_{n,a}} c_a x^a, \end{cases}$$

where each element of $m = \{m_{i,a} \mid i \in [n] \text{ and } a \in A_{f_i}\}$ is in $\mathbb{Q}_{\geq 0}$ and for each $i \in [n]$, the minimum of $\{m_{i,a} \mid a \in A_{f_i}\}$ equals zero and is obtained precisely twice. We call m the *lifting of the polyhedral homotopy*.

By substitution, we have $H_m(x, 1) \equiv F(x)$, which is the target system. Assuming genericity of the lifting m, and system F, the polyhedral homotopy $H_m(x, t)$ has smooth, nonintersecting paths

in $(\mathbb{C} \setminus \{0\})^n \times (0, 1]$, parameterized by t. The starting points of these paths at $t = 0$ can be obtained by solving binomial systems. The polyhedral homotopy continuation is implemented in software `HOM4PS2` [13], `PHCPack` [24], `Pss5` [18], and `HomotopyContinuation.jl` [4].

2F. ***Certification of numerical solutions.*** Practical implementations of homotopy continuation rely on heuristics to compute numerical approximations of solutions to a system of polynomial equations. Therefore, *a posteriori* certification for the correctness of these approximations may be necessary. We say a numerical approximation of a solution of a system F is *certified* if it can be refined up to arbitrary precision to an actual solution of F that is uniquely contained in a neighborhood of the approximation by applying a suitable operator (e.g., the Newton operator). Smale's α-theory [3, Chapter 8] and the Krawczyk method [19, Chapter 8] are commonly used for certification. Numerical root certification algorithms are implemented in the software `alphaCertified` [9], `NumericalCertification.m2` [11], and the function `certify` in `HomotopyContinuation.jl` [5].

3. FINDING A REAL POLYHEDRAL HOMOTOPY. The real polyhedral homotopy algorithm introduced in [8] provides a framework for finding all real solutions of a polynomial system devised as a variation of the standard polyhedral homotopy. The main idea stems from Viro's patchworking for complete intersections [22]. This result establishes a homeomorphic relation between the set of real solutions of a polynomial system and the intersection of the positive part of tropical varieties; see [8, Section 2.2] for a rigorous statement. We say a system is *patchworked* if the real solution set of the system is homeomorphic to the union of the intersection points of the subcomplexes obtained by Viro's patchworking. In other words, we are interested in constructing a homotopy whose number of real solutions does not change while tracking.

Sections 2A, 2B, and 2C, provide the preliminaries for the following proposition motivating the real polyhedral homotopy algorithm in [8]:

Proposition 3.1 [8, Proposition 2.19]. *Consider a polynomial system $F_C = \langle f_1, \dots, f_n \rangle$ with a coefficient vector C and support sets $A_{f_1}, \dots, A_{f_n}$ such that $\dim(A_{f_i}) = n$ for each i. Suppose that $u = (u_1, \dots, u_n)$, where $u_i = (u_a)_{a \in A_{f_i}}$ is a vector satisfying that*

(1) *the vector u is not on the boundary of any secondary cone of the Cayley configuration $A = A_{f_1} * \cdots * A_{f_n}$,*

(2) *the ray $\operatorname{Log}|C| + \lambda u$ does not intersect $\mathcal{A}(\nabla_A(\mathbb{R}))$ for any $\lambda \in [0, \infty)$.*

Then, the tuple of real Puiseux series $x(t) = (x_1(t), \dots, x_n(t))$ where $x_i(t)$ is a real Puiseux series and

$$x_i(t) = t^{\alpha_i} y_i + (\textit{higher order terms in } t)$$

is a solution to the system F_C only if $(\alpha_1, \dots, \alpha_n, 1)$ is an outer normal to a lower facet of $\sum_{i=1}^n \operatorname{conv}(A_i^{u_i})$.

The solution $x(t)$ in Proposition 3.1 is derived in a similar way to the polyhedral homotopy method outlined in Section 2E.

Since the conditions from the proposition above are not satisfied in general, a certification procedure for checking if a polynomial system is patchworked is suggested in [8]. Note that the containment (2-1) shows that the mixed cell cone $M(\Delta_w)$ is contained in the corresponding normal cone $\mathcal{N}(v)$ of ∇_A. Therefore, a proper choice of a lifting function w might result in the mixed cell cone $M(\Delta_w)$ that is contained in the complement of $\mathcal{A}(\nabla_A(\mathbb{R}))$. When such a coefficient vector is given, costly computation of the amoeba $\mathcal{A}(\nabla_A(\mathbb{R}))$ can be replaced by a mixed cell cone computation. Based on the argument above, the following proposition establishes a sufficient condition for a polynomial system to be patchworked:

Proposition 3.2 [8, Proposition 3.1]. *Let $F_C = \langle f_1, \ldots, f_n\rangle$ be a system of sparse polynomials with coefficient vector C with support sets $A_{f_1}, \ldots, A_{f_n}$. Let Δ_w be the triangulation of the Cayley configuration $A = A_{f_1} * \cdots * A_{f_n}$ induced by the lifting $w = \operatorname{Log}|C|$. Consider the corresponding dual mixed cell cone $M(\Delta_w)^\circ$ and its generating vectors $\zeta_1, \ldots, \zeta_L$. Then,*

$$\langle \operatorname{Log}|C|, \zeta_i\rangle > \|\zeta_i\|_1 \log(|A|) \tag{3-1}$$

for all $i = 1, \ldots, L$ implies that the system F_C is a patchworked system. Also, for any $v \in M(\Delta_w)$, the ray $\operatorname{Log}|C| + \lambda v$ for $\lambda \in [0, \infty)$ does not intersect $\mathcal{A}(\nabla_A(\mathbb{R}))$.

Definition 3.3 (real polyhedral homotopy). We say a lifting m induces a *real polyhedral homotopy* of $F = \langle f_1, \ldots, f_n\rangle$ if

- $H_m(x, 1) \equiv F(x) \subset \mathbb{R}[x]$,
- $\{(x, t) \in (\mathbb{R}\setminus\{0\})^n \times (0, 1] \mid H_m(x, t) = 0, \text{ and } t \in (0, 1]\}$ defines smooth nonintersecting paths in $(\mathbb{R}\setminus\{0\})^m \times (0, 1]$, parameterized by t, and emanating from isolated $(\mathbb{R}\setminus\{0\})$-zeros of $F(x) \equiv H(x, 1)$ and continue toward $t = 0$,
- the starting points of these paths as $t = 0$ are obtained by solving binomial systems coming from mixed cells and the $(\mathbb{R}\setminus\{0\})$-zeros of the target system $F(x)$ can be found by tracking these paths over the real numbers.

Proposition 3.2 gives a method to certify when the lifting $m = \log(|C|)$ induces a real polyhedral homotopy.

Example 3.4. If $f_1 = -1 - 24000y + x^3$ and $f_2 = -9 + 50xy - y^2$, then the $\operatorname{Log}|C|$ lifting corresponds to the lifted polynomials

$$-1 - 24000t^{\log 24000}y + x^3 \quad \text{and} \quad -9t^{\log 9} + 50t^{\log 50}xy - y^2.$$

Denoting the real Puiseux series solutions by $(x(t), y(t))$, these lifted polynomials give two homotopies with binomial start systems induced by mixed cells. The homotopies are

$$h_1(x, y; t) = \langle -t^{a_1} - 24000y + x^3, -9t^{a_2} + 50xy - y^2\rangle,$$
$$h_2(x, y; t) = \langle -t^{b_1} - 24000y + x^3, -9 + 50xy - t^{b_2}y^2\rangle,$$

where $a_1, a_2, b_1, b_2 \in \mathbb{Z}_{>0}$. We implement a way to certify that this is a (well-chosen) real polyhedral homotopy and outline our implementation in the next section.

4. REAL POLYHEDRAL HOMOTOPY IMPLEMENTATION. This section describes our `Julia` implementation in detail. We assume we are given a polynomial system $F = \langle f_1, \ldots, f_n \rangle$ with finitely many complex solutions. Let C denote the vector of coefficients of the system F.

4A. ***certify_patchwork.*** This function certifies if a given system is patchworked so that all real solutions can be found using the real polyhedral homotopy. It checks if inequality (3-1) holds for each mixed cell. It returns the value 1 if the system F is certified to be patchworked according to the inequality. Otherwise, 0 is returned.

```
@var x y
f1 = -1 - 24000*y + x^3
f2 = -9 + 50*x*y - y^2
F = System([f1, f2])
certify_patchwork(F)
    1
```

Remark 4.1. As an optional argument we have `Number_Real_Solutions`. When this optional argument is set to true (default is false) we return $(1, k)$ where k is number of real solutions to the target system when the target system is patchworked. It works by solving the binomial systems B (discussed in Section 4B) by using Smith normal forms as outlined in [8, Section 2.5]. For additional details on solving binomial systems see [7]. Otherwise, we return 0.

```
certify_patchwork(F; Number_Real_Solutions = true)
    (1,4)
```

4B. ***generate_binomials.*** This function takes as an input a polynomial system $F = \langle f_1, \ldots, f_n \rangle$ in n variables and outputs an object called `Binomial_system_data` that consists of 4 objects: a collection of binomial systems, their normal vectors, the lifting function, and the mixed cells. Contents in the object `Binomial_system_data` stem from the mixed cells induced by the $\mathrm{Log}\,|C|$-lifting mentioned in Proposition 3.2. Each value in `Binomial_system_data` can be called by its name `binomial_system`, `normal_vectors`, `lifts`, and `cells` as follows:

```
# Continued from above.
B = generate_binomials(F);
B.binomial_system
    2-element Vector{Any}:
    Expression[-24000*y + x^3, 50*x*y - y^2]
    Expression[-24000*y + x^3, -9 + 50*x*y]
B.lifts
    2-element Vector{Vector{Int64}}:
    [0, -10085809, 0]
    [-3912023, 0, -2197225]
```

In our implementation, we use the `Julia` package `MixedSubdivisions.jl` [23] to compute the mixed cells. The package requires a lifting function to take on integer values. So in our implementation, we take $10^6 \cdot \mathrm{Log}\,|C|$ rounded to the nearest integer as our lifting. Using this scaled and rounded lifting is suitable because uniformly scaling a lifting function gives the same triangulation. Therefore, rounding the lifting function will preserve the desired triangulation so long as $\mathrm{Log}\,|C|$ is of distance at least 10^{-6} from the boundary of its mixed cell cone. We emphasize that this is a heuristic and for some systems, one may need to take $10^k \cdot \mathrm{Log}\,|C|$ for $k > 6$. In all of our experiments, $k = 6$ was sufficient to preserve the desired triangulation.

4C. ***rph_track.*** Our function `rph_track` takes an object `Binomial_system_data` B and a polynomial system F as its input. It returns the output of tracking the real solutions of the binomial systems to the target system. For our running example, the function returns all real solutions to $F = 0$ and no nonreal solutions.

```
rph_track(B,F)
    4-element Vector{Vector{Float64}}:
    [-1095.4451129504978, -54772.25548320812]
    [1095.4451137838312, 54772.255524874796]
    [8.111114476617955, 0.02219298606763958]
    [-8.103507635567631, -0.022213821121964985]
```

In contrast, the default solve command in `HomotopyContinuation.jl` finds all six complex solutions of which two are nonreal and the four real solutions are the solutions found above.

```
S = HomotopyContinuation.solve(F)
solutions(S)
    6-element Vector{Vector{ComplexF64}}:
    [-0.003803837191831332 - 8.10709081578636im,
                        -1.0415806358129898e-5 + 0.022201564813120536im]
    [-0.003803837191831332 + 8.10709081578636im,
                        -1.0415806358129898e-5 - 0.022201564813120536im]
    ...
```

We implement our function in two steps: The first step is to use Smith normal forms to find all real solutions to the binomial systems as mentioned in Remark 4.1. The second step of this function tracks the real solutions from the binomial systems to the target system using `HomotopyContinuation.jl`.

Remark 4.2. In the event `certify_patchwork` function returns 0, `rph_track` function will still run. In this situation, there is no guarantee that the real solutions of the start system will converge to real solutions of the target system nor that every real solution of the target system will be returned. The following example shows how the real polyhedral homotopy behaves for a non-patchworked system:

```
@var x y
F2 = System([ -1 - 240*y + x^3,  -9 + 50*x*y - y^2 ])
certify_patchwork(F2)
    0

B2 = generate_binomials(F2);
R = rph_track(B2,F2)
    4-element Vector{Vector{Float64}}:
    [-109.54445340262997, -5477.221026962461]
    [109.54453673601331, 5477.225193632879]
    [2.601807483849416, 0.06921950525397731]
    [-2.525776652013741, -0.07130546925252307]
```

As a consequence, for non-patchworked system we can view our implementation as a heuristic for finding real solutions to a polynomial system.

4D. ***An rph_track option.*** The optional argument `Certification`, certifies all real solutions to a patchworked system F found through the aforementioned real polyhedral homotopy algorithm are in fact good approximations to $F(x) = 0$. Given correct path tracking, the real polyhedral homotopy guarantees to find all real solutions when the system is patchworked. To guarantee that the path tracking was done correctly, we use an *a posteriori* certification. For a patchworked system F, when the real polyhedral

homotopy root-finding is certified, the function returns a list of solutions to $F(x) = 0$ and 1; otherwise, it returns 0.

```
rph_track(B,F; Certification = true)
    ([[-1095.4451129504978, -54772.25548320812],...], 1)
```

When the option `Certification` is set to `true`, two extra steps are done in `rph_track`. First, it certifies the real start solutions of a binomial system in B are true solutions to B, which provides the real root count of $F = 0$. For the second additional step, the numerical approximations for solutions of F produced by the real polyhedral homotopy tracking are certified using the Krawczyk method. The certification relies on `certify` in `HomotopyContinuation.jl` with details found in [5].

4E. ***Reliability.*** For those who are not frequent users of numerical methods in algebraic geometry, we add this subsection to clearly explain the reliability of our package and the meaning of certification of the output. Our package relies on the path-tracking heuristics in `HomotopyContinuation.jl`. It is well known that numerical path-tracking methods sometimes miss solutions. While in theory such an event happens with probability zero, in practice, we work with floating point arithmetic so may observe this behavior.

Both verifying (like a trace test [14; 6]) and certifying the completeness of the set of real solutions still remain open, to the best of the authors' knowledge. The certification functionality in the package is limited to checking if all numerical solutions that are computed are certified in the context of Section 2F. When the system is not patchworked, our implementation does not guarantee finding all real solutions to a given system, even if path-tracking was perfect. Nevertheless, our package can still run the real polyhedral homotopy and get an output that is a (potentially proper) subset of the real solution set. When the certification option is turned on the output consists of certified real solutions with no guarantee of completeness of the real solution set.

5. OUTLOOK. There may be potential improvement available for the real polyhedral homotopy algorithm described in this paper by constructing homotopy paths that avoid the real discriminant locus. The current verification using inequality (3-1) does not give both necessary and sufficient conditions for detecting if a polynomial system is patchworked and as a result is a conservative way to certify this. To illuminate the potential of the real polyhedral homotopy algorithm described in this paper, we provide an example of a polynomial system from game theory where the real polyhedral homotopy algorithm finds all real solutions even though the system is not certified to be patchworked.

Example 5.1 (a modified version of [20]). Consider a three-player game such that each player has 3 strategies. For $i = 1, 2, 3$, define the i-th player's payoff by a $3 \times 3 \times 3$-tensor $P^{(i)}$, and let $p_j^{(i)}$ be the probability that the i-th player chooses the j-th strategy. In this case, the i-th player's payoff a_i can be computed as

$$a_i = \sum_{j,k,l=1}^{3} P^{(i)}_{j,k,l} \cdot p_j^{(1)} p_k^{(2)} p_l^{(3)} \quad \text{for } i = 1, 2, 3.$$

Assuming that all players do not change their initial strategy, the vector of probabilities for strategies of players optimizing output of their own is called a *Nash equilibrium*. The problem of finding all

Nash equilibria can be modeled as a constrained optimization problem. Constrained optimization problems with smooth critical points can be solved using the method of Lagrange multipliers. This method then reduces to solving a system of polynomial equations. For example, all Nash equilibria for the three-player game described above appear as real solutions to the following system of 12 polynomial equations in 12 unknowns $\left(p_1^{(1)}, p_2^{(1)}, p_3^{(1)}, p_1^{(2)}, p_2^{(2)}, p_3^{(2)}, p_1^{(3)}, p_2^{(3)}, p_3^{(3)}, a_1, a_2, a_3\right)$:

$$\sum_{j=1}^{3} p_j^{(1)} = \sum_{j=1}^{3} p_j^{(2)} = \sum_{j=1}^{3} p_j^{(3)} = 1,$$

$$p_j^{(1)}\left(a_1 - \sum_{k,l=1}^{3} P_{j,k,l}^{(1)} \cdot p_k^{(2)} p_l^{(3)}\right) = 0 \quad \text{for } j = 1, 2, 3,$$

$$p_k^{(2)}\left(a_2 - \sum_{j,l=1}^{3} P_{j,k,l}^{(2)} \cdot p_j^{(1)} p_l^{(3)}\right) = 0 \quad \text{for } k = 1, 2, 3,$$

$$p_l^{(3)}\left(a_3 - \sum_{j,k=1}^{3} P_{j,k,l}^{(3)} \cdot p_j^{(1)} p_k^{(2)}\right) = 0 \quad \text{for } l = 1, 2, 3.$$

For randomly chosen payoff tensors

$$P^{(1)} = \left[\begin{array}{ccc|ccc|ccc} 34 & 162 & 70 & 140 & 174 & -183 & 104 & 68 & 18 \\ 136 & 2001 & 72 & 166 & 140 & -10541 & 1261 & 72 & 20 \\ 80 & 32 & 10 & 132 & 176 & 17 & 174 & 86 & 94 \end{array}\right],$$

$$P^{(2)} = \left[\begin{array}{ccc|ccc|ccc} 56 & -6 & 22 & 5712 & -150 & 74 & 150 & 10 & 150 \\ -28 & 80 & 1196 & 182 & 116 & 22 & 44 & 186 & 60 \\ 12 & 48 & 22 & 64 & 11 & 46 & 80 & 1192 & 82 \end{array}\right],$$

$$P^{(3)} = \left[\begin{array}{ccc|ccc|ccc} 104 & 162 & 655 & 2116 & 130 & 559 & 134 & 138 & -4 \\ 172 & 152 & -12 & 25 & 188 & 162 & 1168 & 162 & 178 \\ 134 & 124 & 170 & 118 & 38 & 130 & 113 & 800 & 152 \end{array}\right],$$

we solve this system using `RealPolyhedralHomotopy.jl` and see that it returns all eight real solutions.

```
8-element Vector{Vector{Float64}}:
 [0.023564140817086864, -0.003578459398610543,
                        ... , 17.00815164148862, 150.9733200702087]
 [0.1690458198092814, 0.94412282597982,
                        ... , 40.84440318232266, 199.80455930749537]
 ...
```

In this case, the first nine coordinates of solutions to the polynomial system correspond to probabilities. By restricting the real solutions to those with positive first nine coordinates we find one Nash equilibrium.

```
valid_real_solutions = filter(s -> all(s[1:9] .> 0), rsols)
    [0.38164510348087044, 0.3858370607366621,
                        ... , 221.7388487672952, 210.9466957898507]
```

The success of Example 5.1 motivates the future development of heuristic methods for finding a complete real solution set. In addition to applications in economics, as seen in the previous example, we hope that problems in reaction networks, statistics, and optimization, where polyhedral geometry has played a role in counting complex solutions, are ripe for applying real polyhedral homotopy.

ACKNOWLEDGMENT. The authors are grateful to Paul Breiding, Timo de Wolff and Christopher O'Neill for helpful discussions.

SUPPLEMENT. The online supplement contains version 1.2.1 of `RealPolyhedralHomotopy`.

REFERENCES.

[1] D. J. Bates, J. D. Hauenstein, A. J. Sommese, and C. W. Wampler, *Numerically solving polynomial systems with Bertini*, Software, Environments, and Tools **25**, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2013. MR Zbl

[2] D. N. Bernstein, "The number of roots of a system of equations", *Funkcional. Anal. i Priložen.* **9**:3 (1975), 183–185. MR Zbl

[3] L. Blum, F. Cucker, M. Shub, and S. Smale, *Complexity and real computation*, Springer, New York, 1998. MR Zbl

[4] P. Breiding and S. Timme, "HomotopyContinuation.jl: A package for homotopy continuation in Julia", pp. 458–465 in *Mathematical software*: *ICMS* 2018 (South Bend, IN), edited by J. H. Davenport et al., Lecture Notes in Computer Science **10931**, Springer, 2018. Zbl

[5] P. Breiding, K. Rose, and S. Timme, "Certifying zeros of polynomial systems using interval arithmetic", *ACM Trans. Math. Software* **49**:1 (2023), art. id. 11. MR Zbl

[6] T. Brysiewicz and M. Burr, "Sparse trace tests", *Math. Comp.* **92**:344 (2023), 2893–2922. MR Zbl

[7] T. Chen and T.-Y. Li, "Solutions to systems of binomial equations", *Ann. Math. Sil.* **28** (2014), 7–34. MR Zbl

[8] A. A. Ergür and T. de Wolff, "A polyhedral homotopy algorithm for real zeros", *Arnold Math. J.* **9**:3 (2023), 305–338. MR Zbl

[9] J. D. Hauenstein and F. Sottile, "Algorithm 921: alphaCertified: certifying solutions to polynomial systems", *ACM Trans. Math. Software* **38**:4 (2012), art. id. 28. MR Zbl

[10] B. Huber and B. Sturmfels, "A polyhedral method for solving sparse polynomial systems", *Math. Comp.* **64**:212 (1995), 1541–1555. MR Zbl

[11] K. Lee, "Certifying approximate solutions to polynomial systems on `Macaulay2`", *ACM Commun. Comput. Algebra* **53**:2 (2019), 45–48. MR Zbl

[12] K. Lee and X. Tang, "On the polyhedral homotopy method for solving generalized Nash equilibrium problems of polynomials", *J. Sci. Comput.* **95**:1 (2023), art. id. 13. MR Zbl

[13] T. L. Lee, T. Y. Li, and C. H. Tsai, "HOM4PS-2.0: A software package for solving polynomial systems by the polyhedral homotopy continuation method", *Computing* **83**:2–3 (2008), 109–133. MR Zbl

[14] A. Leykin, J. I. Rodriguez, and F. Sottile, "Trace test", *Arnold Math. J.* **4**:1 (2018), 113–125. MR Zbl

[15] J. Lindberg, A. Zachariah, N. Boston, and B. Lesieutre, "The distribution of the number of real solutions to the power flow equations", *IEEE Trans. Power Syst.* **38**:2 (2022), 1058–1068. Zbl

[16] J. Lindberg, N. Nicholson, J. I. Rodriguez, and Z. Wang, "The maximum likelihood degree of sparse polynomial systems", *SIAM J. Appl. Algebra Geom.* **7**:1 (2023), 159–171. MR Zbl

[17] D. Maclagan and B. Sturmfels, *Introduction to tropical geometry*, Graduate Studies in Mathematics **161**, American Mathematical Society, Providence, RI, 2015. MR Zbl

[18] G. Malajovich, "Pss5: Polynomial system solver", 2019, available at https://sourceforge.net/projects/pss5/. Version 5.1. Zbl

[19] R. E. Moore, R. B. Kearfott, and M. J. Cloud, *Introduction to interval analysis*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2009. MR Zbl

[20] I. Portakal, "Polynomial systems arising from Nash equilibria: A story about how game theory meets multilinear systems and product of simplices", electronic reference, 2021, available at https://www.JuliaHomotopyContinuation.org/examples/nash/. Online; accessed 10 March 2023. Zbl

[21] A. J. Sommese and C. W. Wampler, II, *The numerical solution of systems of polynomials arising in engineering and science*, World Scientific, Hackensack, NJ, 2005. MR Zbl

[22] B. Sturmfels, "Viro's theorem for complete intersections", *Ann. Scuola Norm. Sup. Pisa Cl. Sci.* (4) **21**:3 (1994), 377–386. MR Zbl

[23] S. Timme, "MixedSubdivisions.jl: Package for computing a (fine) mixed subdivision and the mixed volume of lattice polytopes", available at https://github.com/saschatimme/MixedSubdivisions.jl. Version 0.5. Zbl

[24] J. Verschelde, "Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation", *ACM Trans. Math. Softw.* **25**:2 (1999), 251–276. Zbl

RECEIVED: 11 Jul 2022 REVISED: 9 Jan 2024 ACCEPTED: 24 Jan 2024

KISUN LEE:

kisunl@clemson.edu
School of Mathematical and Statistical Science, Clemson University, Clemson, SC, United States

JULIA LINDBERG:

julia.lindberg@math.utexas.edu
Department of Mathematics, University of Texas-Austin, Austin, TX, United States

JOSE ISRAEL RODRIGUEZ:

jose@math.wisc.edu
Department of Mathematics, University of Wisconsin, Madison, WI, United States