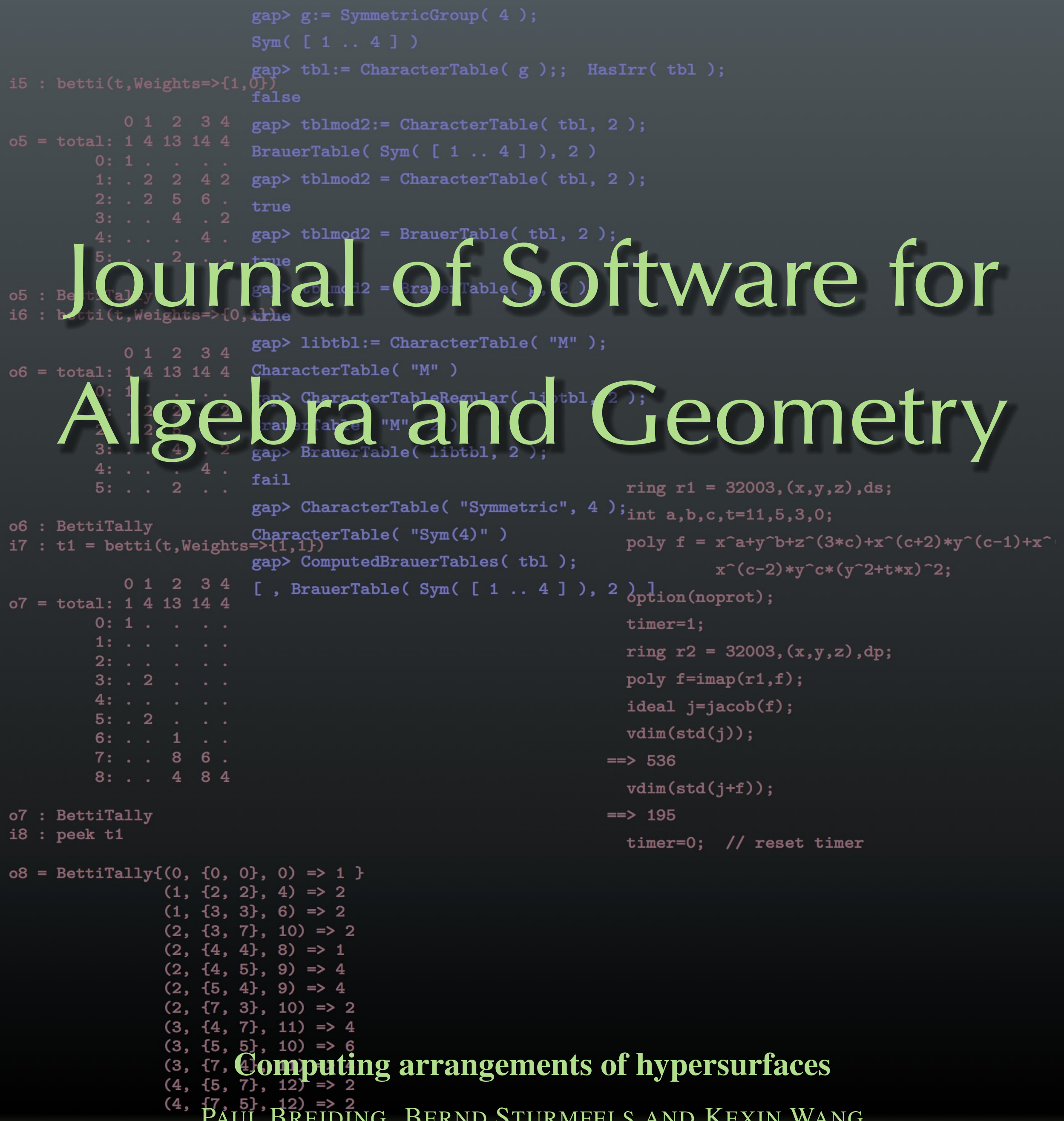
Journal of Software for Algebra and Geometry



```
gap> g:= SymmetricGroup( 4 );
Sym( [ 1 .. 4 ] )
gap> tbl:= CharacterTable( g );;  HasIrr( tbl );
false
gap> tblmod2:= CharacterTable( tbl, 2 );
BrauerTable( Sym( [ 1 .. 4 ] ), 2 )
gap> tblmod2 = CharacterTable( tbl, 2 );
true
gap> tblmod2 = BrauerTable( tbl, 2 );
true
gap> tblmod2 = BrauerTable( g, 2 );
true
gap> libtbl:= CharacterTable( "M" );
CharacterTable( "M" )
gap> CharacterTableRegular( libtbl, 2 );
BrauerTable( "M", 2 )
gap> BrauerTable( libtbl, 2 );
fail
gap> CharacterTable( "Symmetric", 4 );
CharacterTable( "Sym(4)" )
gap> ComputedBrauerTables( tbl );
[ , BrauerTable( Sym( [ 1 .. 4 ] ), 2 ) ]
```

```
i5 : betti(t,Weights=>{1,0})

            0 1  2  3 4
o5 = total: 1 4 13 14 4
         0: 1 .  .  . .
         1: . 2  2  4 2
         2: . 2  5  6 .
         3: . .  4  . 2
         4: . .  .  4 .
         5: . .  2  . .

o5 : BettiTally
i6 : betti(t,Weights=>{0,1})

            0 1  2  3 4
o6 = total: 1 4 13 14 4
         0: 1 .  .  . .
         1: . 2  2  4 2
         2: . 2  5  6 .
         3: . .  4  . 2
         4: . .  .  4 .
         5: . .  2  . .

o6 : BettiTally
i7 : t1 = betti(t,Weights=>{1,1})

            0 1  2  3 4
o7 = total: 1 4 13 14 4
         0: 1 .  .  . .
         1: . .  .  . .
         2: . .  .  . .
         3: . 2  .  . .
         4: . .  .  . .
         5: . 2  .  . .
         6: . .  1  . .
         7: . .  8  6 .
         8: . .  4  8 4

o7 : BettiTally
i8 : peek t1

o8 = BettiTally{(0, {0, 0}, 0) => 1 }
                (1, {2, 2}, 4) => 2
                (1, {3, 3}, 6) => 2
                (2, {3, 7}, 10) => 2
                (2, {4, 4}, 8) => 1
                (2, {4, 5}, 9) => 4
                (2, {5, 4}, 9) => 4
                (2, {7, 3}, 10) => 2
                (3, {4, 7}, 11) => 4
                (3, {5, 5}, 10) => 6
                (3, {7, 4}, 11) => 4
                (4, {5, 7}, 12) => 2
                (4, {7, 5}, 12) => 2
```

```
  ring r1 = 32003,(x,y,z),ds;
  int a,b,c,t=11,5,3,0;
  poly f = x^a+y^b+z^(3*c)+x^(c+2)*y^(c-1)+x^
           x^(c-2)*y^c*(y^2+t*x)^2;
  option(noprot);
  timer=1;
  ring r2 = 32003,(x,y,z),dp;
  poly f=imap(r1,f);
  ideal j=jacob(f);
  vdim(std(j));
==> 536
  vdim(std(j+f));
==> 195
  timer=0;  // reset timer
```

Computing arrangements of hypersurfaces

Paul Breiding, Bernd Sturmfels and Kexin Wang

vol 15 2025

JSAG **15** (2025), 11–27
https://doi.org/10.2140/jsag.2025.15.11

The Journal of Software for
Algebra and Geometry

Computing arrangements of hypersurfaces

PAUL BREIDING, BERND STURMFELS AND KEXIN WANG

ABSTRACT: We present a Julia package `HypersurfaceRegions.jl` for computing all connected components in the complement of an arrangement of real algebraic hypersurfaces in $\mathbb{R}^n$.

1. INTRODUCTION. Arrangements of real hyperplanes are ubiquitous in combinatorics, algebra and geometry. Their *regions* (i.e., connected components) are convex polyhedra, either bounded or unbounded, and their numbers are invariants of the underlying matroid [6; 13]. The number of bounded regions agrees with the Euler characteristic of the complex arrangement complement [7; 8], and also with the maximum likelihood degree (ML degree). The latter is the number of complex critical points of the associated models in statistics and physics [19].

Arrangements of nonlinear hypersurfaces are equally important, but they are studied much less. Polynomials of higher degree create features that are not seen when dealing with hyperplanes. The number of regions is no longer a combinatorial invariant, but it depends in a subtle way on the coefficients. Moreover, the regions are generally not contractible.

We present a practical software tool, called `HypersurfaceRegions.jl` and implemented in the programming language `Julia` [1], whose input consists of k polynomials in n variables

$$f_1, f_2, \dots, f_k \in \mathbb{R}[x_1, \dots, x_n]. \tag{1}$$

The output is a list of all regions C of the n-dimensional manifold

$$\mathcal{U} = \{u \in \mathbb{R}^n : f_1(u) \cdot f_2(u) \cdots f_k(u) \neq 0\}. \tag{2}$$

The list is grouped according to sign vectors $\sigma \in \{-1, +1\}^k$, where σ_i is the sign of f_i on C. Unlike in the case of hyperplane arrangements, each sign vector σ typically corresponds to multiple regions. For each region C we find the Euler characteristic via a Morse function.

Example 1 ($n = k = 3$). Figure 1 shows two concentric spheres that are pierced by an ellipse

$$f_1 = x_1^2 + x_2^2 + x_3^2 - 1, \quad f_2 = x_1^2 + x_2^2 + x_3^2 - 4, \quad f_3 = 100x_1^2 + 100x_2^2 + x_3^2 - 9.$$

The threefold $\mathcal{U}$ has eight regions. Five are contractible, with Euler characteristic $\chi = 1$. The central region has $\sigma = (-, -, -)$. The sign vectors $(+, -, -)$ and $(+, +, -)$ each contribute two regions. Two

MSC2020: 14Q30, 57R19, 68-04.

Keywords: hypersurface arrangement, Morse theory, Euler characteristic.
`HypersurfaceRegions.jl` version 1

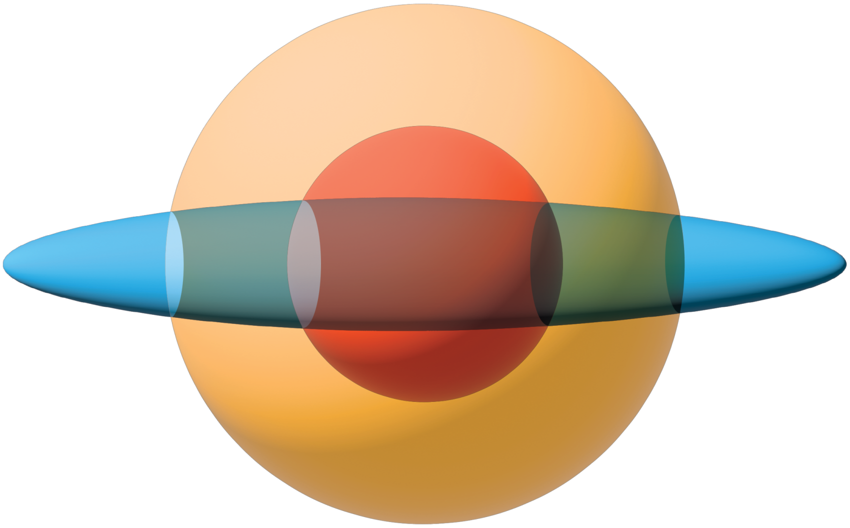


Figure 1. A skinny ellipse pierces two concentric spheres. This arrangement has 8 regions.

bounded regions are solid tori, with $\chi = 0$ and $\sigma = (-, -, +), (+, -, +)$. The unique unbounded region, with $\sigma = (+, +, +)$ and $\chi = 2$, is homotopic to the 2-sphere.

Our software `HypersurfaceRegions.jl` also features heuristics for deciding whether a region is bounded or unbounded, and which of the regions are fused when the hyperplane at infinity is added. In our Section 4, titled *How to use the software*, we explain how this works.

Example 2 ($n = 2$, $k = 21$). In del Pezzo geometry [11, Section 3], it is known that removing the 27 lines from a real cubic surface creates 130 regions. We can confirm this by applying `HypersurfaceRegions.jl` to the plane curves in the blow-up construction of the surface. Fix six general points in convex position in $\mathbb{R}^2$. Let $f_1, \ldots, f_{15}$ be the lines spanned by pairs of points and $f_{16}, \ldots, f_{21}$ the ellipses spanned by five of the points. This arrangement has 145 regions, all contractible, of which 115 are bounded and 30 are unbounded. Our software identifies these and fuses 15 pairs of unbounded regions. It outputs the 130 regions in $\mathbb{RP}^2$.

The method we present is an adaptation of the algorithm by Cummings, Hauenstein, Hong and Smyth [9], which in turn is inspired by [12]. Their setting is more general in that they allow semialgebraic sets defined by both equations and inequalities. We restrict ourselves to arrangement complements, defined by $f_1 \neq 0, \ldots, f_k \neq 0$. This enables us to offer tools in `Julia` [1] that are easy to use and widely applicable. Our implementation is based on the numerical algebraic geometry software `HomotopyContinuation.jl` [5] and on the software `DifferentialEquations.jl` [16] for solving differential equations.

This paper is organized as follows. In Section 2 we introduce a Morse function for our problem. This is a modified version of the log-likelihood function associated with $f_1, \ldots, f_k$. Every region of $\mathcal{U}$ contains at least one critical point. We compute the critical points using `HomotopyContinuation.jl` . These points determine the Euler characteristic of each region.

In Section 3 we turn to the mountain pass theorem [2; 14], which ensures that we obtain a connected graph in each region when tracking paths from index 1 critical points to index 0 critical points. This tracking procedure is realized in our software by solving an ODE using `DifferentialEquations.jl`. In short, our software is built on flows in Morse theory.

Section 4 explains how to run `HypersurfaceRegions.jl`. It is aimed at beginners with no prior experience with numerical software. We demonstrate that `HypersurfaceRegions.jl` can be used for a wide range of scenarios from the mathematical spectrum. In Section 5 we report on test runs of our software on generic instances, obtained by sampling random hypersurfaces and random spectrahedra. Our presentation emphasizes ease and simplicity.

2. LOG-LIKELIHOOD AS A MORSE FUNCTION. The algorithm in [9] rests on Morse theory. We review some basics from the textbook [15]. Let $\mathcal{M}$ be an n-dimensional manifold. A smooth function $g : \mathcal{M} \to \mathbb{R}$ is a *Morse function* if the Hessian matrix $(\partial^2 g/\partial x_i \partial x_j)$ is invertible at every critical point of g. The number of positive eigenvalues of the Hessian matrix is the *index* of the critical point. Let g be a Morse function which is *exhaustive*, i.e., the superlevel set $\{g \geq c\} = \{u \in \mathcal{M} : g(u) \geq c\}$ is compact for each $c \in \mathbb{R}$. If an interval $[a, b] \subset \mathbb{R}$ contains no critical value of g then the superlevel sets $\{g \geq c\}$ are diffeomorphic for $c \in [a, b]$. By contrast, suppose c is a critical value of g, corresponding to a unique critical point of index ℓ. Then, for sufficiently small ϵ, the superlevel set $\{g \geq c - \epsilon\}$ is diffeomorphic to $\{g \geq c + \epsilon\}$ with one ℓ-handle of dimension n attached. Since ℓ-handles can be shrunk to ℓ-cells, one arrives at the following result.

Theorem 3. *The manifold $\mathcal{M}$ is homotopy equivalent to a CW-complex with exactly one ℓ-cell for every critical point of index ℓ. In particular, the Euler characteristic of $\mathcal{M}$ equals*

$$\chi(\mathcal{M}) = \sum_{\ell=0}^{n} (-1)^{\ell} \mu_{\ell}. \tag{3}$$

where μ_ℓ is the number of index ℓ critical points of the exhaustive Morse function $g : \mathcal{M} \to \mathbb{R}$.

We shall apply Theorem 3 to the manifold $\mathcal{U}$ defined in (2). This requires an exhaustive Morse function. On each region of $\mathcal{U}$, this role will be played by the rational function

$$g(x) = \frac{\prod_{i=1}^{k} |f_i(x)|^{s_i}}{q(x)^t}. \tag{4}$$

For the denominator $q(x)$ we take a generic polynomial of degree 2 that is strictly positive on $\mathbb{R}^n$. The exponents $s_1, \ldots, s_k, t$ are arbitrary positive integers which satisfy the inequality

$$\sum_{i=1}^{k} s_i \operatorname{degree}(f_i) < 2t. \tag{5}$$

Proposition 4. *The function $g(x)$ is an exhaustive Morse function. It is strictly positive on $\mathcal{U}$ and it is zero on each of the hypersurfaces $\{f_i = 0\}$. It also vanishes at infinity in $\mathbb{R}^n$.*

The Morse function $g(x)$ in (4) is called *master function* in the theory of arrangements [8] and *likelihood function* in algebraic statistics [19]. Its logarithm is the *log-likelihood function*

$$\log(g(x)) = \sum_{i=1}^{k} s_i \cdot \log|f_i(x)| - t \cdot \log(q(x)). \tag{6}$$

The critical points of $g(x)$ are the solutions in $\mathbb{C}^n$ of the equation $\nabla \log(g(x)) = 0$. The generic choice of the quadric $q(x)$ implies that all critical points have distinct critical values. The number of critical values is the Euler characteristic of the very affine complex variety

$$\{u \in \mathbb{C}^n : f_1(u) \cdot f_2(u) \cdots f_k(u) \cdot q(u) \neq 0\}. \tag{7}$$

The construction of $g(x)$ ensures that each region of $\mathcal{U}$ contains at least one critical point. Our algorithm computes all real critical points, and it connects them via the mountain pass theorem. This will be explained in Section 3. We now first return to our three ellipsoids.

Example 5. Let f_1, f_2, f_3 be as in Example 1, fix $s_1 = s_2 = s_3 = 1$ and $t = 4$, and define

$$q = (x_1+2)^2 + (x_2-3)^2 + (x_3-3)^2 + (2x_1+x_2)^2 + 4.$$

This quadric is positive on $\mathbb{R}^3$. Our Morse function for the complement of the three ellipses is

$$g = \frac{|f_1 f_2 f_3|}{q^4}.$$

The log-likelihood function $\log(g) = \log(f_1) + \log(f_2) + \log(f_3) - 4 \cdot \log(q)$ has 29 complex critical points, so the threefold (7) has ML degree 29. Among the critical points, 21 are real. These serve as landmark points for the eight regions. For instance, let $\mathcal{M}$ be the unbounded region. It contains ten critical points, and their indices are $0, 0, 1, 1, 1, 1, 2, 2, 2, 2$. In the notation of Theorem 3, we have $\mu_0 = 2$, $\mu_1 = 4$, $\mu_2 = 4$, and hence $\chi(\mathcal{M}) = \mu_0 - \mu_1 + \mu_2 = 2$.

The signed Euler characteristic of $\mathcal{U}$ is known as the maximum likelihood degree (*ML degree*) in algebraic statistics [7; 19]. This is the number of critical points we must compute. Explicitly, the logarithmic derivative of $g(x)$ translates into the rational function equations

$$\begin{array}{c}
\dfrac{s_1}{f_1}\dfrac{\partial f_1}{\partial x_1} + \cdots + \dfrac{s_k}{f_k}\dfrac{\partial f_k}{\partial x_1} + \dfrac{t}{q}\dfrac{\partial q}{\partial x_1} = 0, \\
\vdots \qquad\qquad \vdots \qquad\quad \vdots \\
\dfrac{s_1}{f_1}\dfrac{\partial f_1}{\partial x_n} + \cdots + \dfrac{s_k}{f_k}\dfrac{\partial f_k}{\partial x_n} + \dfrac{t}{q}\dfrac{\partial q}{\partial x_n} = 0.
\end{array}$$

The following a priori bound on the number of solutions was established in [7, Theorem 1].

Proposition 6. *Let $d_1, \dots, d_k$ be the degrees of the polynomials $f_1, \dots, f_k$. Then the ML degree of $\mathcal{U}$ is bounded above by the coefficient of z^n in the rational generating function*

$$\frac{(1-z)^n}{(1-zd_1)\cdots(1-zd_k)(1-2z)}. \tag{8}$$

This bound is attained when the polynomials f_i are generic relative to their degrees d_i.

Example 7. The ML degree in Proposition 6 equals $\sum_{i=1}^{k} d_i^2 + \sum_{i<j} d_i d_j + 1$ for $n = 2$, and it equals $\sum_{i\le j\le l} d_i d_j d_l - \sum_{i\le j} d_i d_j + 1$ for $n = 3$. See [7, Section 4] for some nice geometry.

Since every region of our manifold $\mathcal{U}$ contains at least one critical point, we conclude:

Corollary 8. *The number of regions of $\mathcal{U}$ is at most the ML degree in Proposition 6.*

The bound in Corollary 8 is tight for linear polynomials. This is proved in the subsequent article [17]. We close this section by sketching a proof for hyperplanes in general position.

Suppose $k \ge n$ and let $d_1 = \cdots = d_k = 1$. Then the generating function (8) becomes

$$\frac{1}{(1-z)^{k-n}\cdot(1-2z)}.$$

We find that the coefficient of z^n in the Taylor expansion of this rational function equals

$$1 + k + \binom{k}{2} + \cdots + \binom{k}{n}.$$

This agrees with the number of all regions in a general arrangement of k hyperplanes in $\mathbb{R}^n$. Thus, all complex critical points are real, and each critical point lies in its own region.

3. Mountain passes and path tracking.

The mountain pass theorem guarantees that all critical points in a region will be connected. This theorem originates in the calculus of variations. We learned about its importance for numerical computations in real algebraic geometry from the work of Cummings et al. [9].

We fix the Morse function g on $\mathcal{U}$ that is given in (4). Since the quadric q is generic, the function g has only finitely many critical points, and g takes distinct values at these critical points. Given any starting point $x_0 \in \mathcal{U}$, we will reach one of these critical points by numerical path tracking. This is done by solving the ODE that describes the gradient flow

$$\dot{x}(t) = \nabla g(x)(t), \quad x(0) = x_0. \tag{9}$$

If x_0 is chosen at random then the gradient flow will reach a critical point of index 0. Loosely speaking, hill climbing in the steepest direction will probably lead us to a mountain peak.

Let $p_1, p_2, \dots, p_d \in \mathcal{U}$ denote the real critical points of g. Our aim is to build a graph with vertex set $\{p_1, p_2, \dots, p_d\}$ whose connected components correspond to the regions of $\mathcal{U}$. For each critical point p_i, we compute the eigenvalues and eigenvectors of the Hessian H_{p_i}. This is the symmetric $n \times n$ matrix of second derivatives of $g(x)$ evaluated at $x = p_i$.

Note that $\det(H_{p_i}) \neq 0$ because g was constructed to be a Morse function. An eigenvector v of H_{p_i} is called *unstable* if the corresponding eigenvalue is positive. If this holds then

$$g(p_i + \epsilon v) > g(p_i), \quad \text{whenever } |\epsilon| \text{ is small.}$$

If $x_0 = p_i + \epsilon v$ is the starting point in (9) then the ODE will lead us to a critical point p_j of g that is different from p_i. Whenever this happens, our graph acquires the edge $p_i \to p_j$.

We now explain why this method connects all critical points in a fixed region. First of all, each p_i is connected to some critical point of index 0. Suppose we start at a critical point p_{i_1} with positive index. There is a path from p_{i_1} that limits to some critical point p_{i_2} with $g(p_{i_2}) > g(p_{i_1})$. If p_{i_2} has positive index, then the solution path from p_{i_2} limits to some critical point p_{i_3} with $g(p_{i_3}) > g(p_{i_2})$. Since there are only finitely many critical points, the process must terminate and the last critical point in the sequence $p_{i_1}, p_{i_2}, \ldots$ has index 0.

We are left to show that all index 0 critical points in the same region will be connected. To this end, we introduce one more tool from Morse theory. The *stable manifold* of p_i is

$$M(p_i) = \{p_i\} \cup \{x \in \mathcal{U} \mid \nabla g(x) \neq 0 \text{ and the ODE solution starting from } x \text{ limits to } p_i\}.$$

The dimension of $M(p_i)$ is the number of stable eigenvectors of H_{p_i}. The stable manifolds for critical points of index 0 have full dimension n in $\mathcal{U}$. So, for each region C of $\mathcal{U}$, we have

$$C = \bigcup_{p_i \in C \text{ index } 0} \overline{M(p_i)}, \tag{10}$$

where the closure is taken in $\mathcal{U}$. Consider any two critical points p_{i_α} and p_{i_β} of index 0 in C. Since C is connected, (10) implies that there is a sequence of index 0 critical points $p_{i_1}, \ldots, p_{i_s}$ such that the intersections $\overline{M(p_{i_\alpha})} \cap \overline{M(p_{i_1})}$ and $\overline{M(p_{i_s})} \cap \overline{M(p_{i_\beta})}$ are nonempty, and also $\overline{M(p_{i_j})} \cap \overline{M(p_{i_{j+1}})}$ is nonempty for $j = 1, \ldots, s-1$. Corollary 10 below tells us that each of these intersections contains at least one critical point of index 1. Thus any pair of index 0 critical points in C is connected through index 1 critical points. Therefore, the connectivity of the finite graph we are building for C will be ensured by Corollary 10.

The Morse function $g : C \to \mathbb{R}$ satisfies the *Palais–Smale condition*. This means that every sequence $\{x_j\}$ in C which satisfies $\lim_{j\to\infty} g(x_j) = \alpha \in \mathbb{R}$ and $\lim_{j\to\infty} \nabla g(x_j) = 0$ has a convergent subsequence. The limit of such a convergent subsequence is a critical point in C with critical value α. The Palais–Smale condition holds in our situation because g is positive on C, it tends to zero on the boundary, and $g^{-1}([\alpha_1, \alpha_2])$ is compact for $0 < \alpha_1 < \alpha_2$.

A *path* between two points a and b in C is a continuous function $\gamma : [0, 1] \mapsto C$ with $\gamma(0) = a$ and $\gamma(1) = b$. We write $\Gamma_{a,b}$ for the set of all such paths. A set $S \subset C$ *separates* a and b if every path $\gamma \in \Gamma_{a,b}$ contains some point in S. We write $g(\gamma) = \{g(\gamma(t)) : t \in [0, 1]\}$. The Palais–Smale condition for g is a hypothesis for the next result, which is [14, Theorem 3].

Theorem 9 (mountain pass theorem). *Fix a closed subset $S \subset C$ which separates a and b and satisfies* $\max(g(x) : x \in S) < \min(g(a), g(b))$. *Then $\omega = \sup(\min(g(\gamma)) : \gamma \in a, b)$ is a critical value of g, and its preimage $g^{-1}(\omega)$ contains a unique critical point of index* 1.

Corollary 10. *Let p_i and p_j be index* 0 *critical points of the Morse function g on C, and suppose that $S = \overline{M(p_i)} \cap \overline{M(p_j)}$ is nonempty. Then S contains a critical point of index* 1.

Proof. The set S satisfies the hypotheses of Theorem 9 for $a = p_i$ and $b = p_j$. Let p_ℓ be the critical point which is promised by Theorem 9. The point p_ℓ necessarily lies in S. □

We have shown that the regions C of $\mathcal{U}$ can be identified by gradient ascent from index 1 critical points. Here we start in the two directions determined by the unique unstable eigenvector. The resulting graph on the index 0 critical points is guaranteed to be connected. We now summarize our approach. Algorithm 1 forms the basis of `HypersurfaceRegions.jl`.

```
Input: Polynomials $f_1, f_2, \dots, f_k$ in $\mathbb{R}[x_1, \dots, x_n]$.
Output: All sign patterns in $\{-, +\}^k$ that are realized by $f_1, \dots, f_k$, and each region with that
        sign pattern, along with its Euler characteristic.
Calculate the critical points $\{p_1, \dots, p_d\}$ of $\log g(x)$ and record the sign vectors
    $\sigma = (\mathrm{sign}(f_1(p_i)), \dots, \mathrm{sign}(f_k(p_i))) \in \{-1, +1\}^k$   for $i = 1, \dots, d$.
Calculate the Hessian matrix $H_{p_i}$ of each critical point $p_i$, verify that it is invertible, and compute
its index and the unstable eigenvectors.
foreach sign pattern $\sigma$ do
    Identify the set $\{p_{i_1}, \dots, p_{i_j}\}$ of all critical points with sign pattern $\sigma$.
    Initialize a graph $G_\sigma$ which has this set as its vertex set.
    if there is only one vertex $p_{i_r}$ of index 0 in $G_\sigma$ then
        Add edges between all index $> 0$ critical points and $p_{i_r}$ to $G_\sigma$.
    else
        For each index 1 critical point $p_{i_\ell}$, identify the unstable eigenvector $v$, and solve the ODE (9)
        starting from $p_{i_\ell} + \epsilon v$ and $p_{i_\ell} - \epsilon v$ for small $\epsilon > 0$.
        Compute the limit points (one or two). Add edge(s) from $p_{i_\ell}$ to these in $G_\sigma$.
        For each $p_{i_\ell}$ of index $> 1$, pick an unstable eigenvector $v$, solve (9) from $p_{i_\ell} + \epsilon v$ for small
        $\epsilon > 0$, and add to $G_\sigma$ the edge from $p_{i_\ell}$ to the limit point.
    Identify the connected components of the graph $G_\sigma$. Compute the Euler characteristic of the
    corresponding regions using the formula in (3).
end
```

Algorithm 1: Computing the regions of an arrangement of k hypersurfaces in $\mathbb{R}^n$.

Correctness proof of Algorithm 1. Each region in $\mathcal{U}$ has constant sign pattern for the evaluation of $(f_1, \ldots, f_k)$, so it suffices to track the ODE paths between critical points with the same sign pattern. Tracking the ODE along the positive and negative unstable eigenvector directions of index 1 critical points connects all index 0 critical points in the same region.

A sign pattern σ is realizable if and only if it is attained by some critical point. We now fix σ, and we consider all critical points with sign pattern σ. For every index > 1 critical point p_{j_1}, the ODE path connects it with some critical point p_{j_2} with $g(p_{j_2}) > g(p_{j_1})$. If again p_{j_2} has index > 1, then the ODE path from p_{j_2} limits to some critical point p_{j_3} with $g(p_{j_3}) > g(p_{j_2})$. Since there are only finitely many critical points, the process terminates, and p_{j_1} is connected to some critical point of index 0 or index 1 via a sequence of paths. Therefore, the connected components of the graph G_σ correspond exactly to the regions in $\mathcal{U}$ with sign pattern σ. The statement about the Euler characteristic follows from Theorem 3. □

Frequently one is interested in the regions of an arrangement in projective space $\mathbb{RP}^n$. Algorithm 2 addresses this point. To begin with, we take the same input as before, namely $f_1, f_2, \ldots, f_k \in \mathbb{R}[x_1, \ldots, x_n]$. We further assume that Algorithm 1 has already been run.

Correctness proof of Algorithm 2. All unbounded regions of $\mathcal{U}$ intersect the hyperplane at infinity, denoted $\{x_0 = 0\}$. If two of them are connected in $\mathbb{RP}^n$, then their intersections with $\{x_0 = 0\}$ share a region of $\mathcal{U}_\infty$. We sample points q_i from each region of $\mathcal{U}_\infty$. The choice of λ guarantees that $\lambda(1, q_i)$ and $-\lambda(1, q_i)$ lie in unbounded regions of $\mathbb{R}^n$ which are adjacent to the region of q_i at infinity. In particular, $\lambda(1, q_i)$ and $-\lambda(1, q_i)$ lie in the same projective region, and this region contains the unbounded affine regions

```
Input: All data that were used and computed in Algorithm 1.
Output: A list of all regions in projective space $\mathbb{RP}^n$, grouped by realizable sign pattern pairs
        in $\{-, +\}^k$.
Homogenize $f_1, \ldots, f_k \in \mathbb{R}[x_1, \ldots, x_n]$ to $f_1'(x_0, x_1, \ldots, x_n), \ldots, f_k'(x_0, x_1, \ldots, x_n)$.
Compute the regions of $\mathcal{U}_\infty = \{u \in \mathbb{R}^{n-1} : f_1'(0, 1, u) \cdots f_k'(0, 1, u) \neq 0\}$ and record the list
$\{q_1, \ldots, q_{d'}\}$ of representatives from each region.
foreach $q_i$ do
    Find the largest absolute value among all zeros of the polynomials $f_j(t, tq_i)$ for $i = 1, \ldots, d$
    and $j = 1, \ldots, k$. Fix a real number $\lambda$ larger than that value.
    Solve the ODE (9) with starting points $\lambda(1, q_i)$ and $-\lambda(1, q_i)$. Record the limiting critical
    points. The regions of these points are connected in $\mathbb{RP}^n$.
end
The regions not visited in the previous step are either bounded or undecided.
```

Algorithm 2: Regions of a projective arrangement of k hypersurfaces in $\mathbb{RP}^n$.

that are identified by the ODE (9). One unbounded region can get matched multiple times in this process. Hence, even three or more regions in $\mathbb{R}^n$ can get fused to the same region in $\mathbb{RP}^n$. □

In our implementation of Algorithm 2 we discard critical points that are close to the hypersurface $\mathbb{R}^{n-1}\backslash\mathcal{U}_\infty$. These points cause numerical problems when solving the corresponding ODE. Hence, that part of our software is based on a heuristic.

We now explain the distinction between *bounded* and *undecided*, alluded to in step 7. It is a feature of numerical computations that tangencies and singularities are hard to detect.

Example 11 ($n = 2, k = 1$). Fix a real number ϵ and consider the plane curves given by

$$f_\epsilon(x, y) = y^2 - (x-1)(x^2+\epsilon) \quad \text{and} \quad g_\epsilon(x, y) = y^2 + \epsilon x^2 - x.$$

Each curve is viewed as an arrangement in $\mathbb{R}^2$ with $k = 1$. The arrangements $\{f_\epsilon\}$ and $\{g_\epsilon\}$ have two regions for $\epsilon \geq 0$ and three regions for $\epsilon < 0$. Note the topology changes when we pass from $\epsilon > 0$ to $\epsilon = 0$. The cubic f_0 acquires an isolated point, and the conic g_0 becomes tangent to the line at infinity. Both regions of $\{g_0\}$ are unbounded, but only one of them is recognized as unbounded by Algorithm 2. The interior of the parabola remains undecided.

We identify bounded regions as follows. We fix a small parameter $\delta > 0$. A region in step 7 is declared *bounded* if it is contained in $-1/\delta < x_1 < 1/\delta$. In practice, we compute critical points of our Morse function on the hyperplanes $x_1 = 1/\delta$ and $x_1 = -1/\delta$. We process these critical points similar to what is done in Algorithm 2. Namely, for each critical point q on the hyperplane $x_1 = 1/\delta$, we find the largest value among all zeros of the polynomials $f_j(t, tq)$ for $j = 1, \ldots, k$ that are smaller than $1/\delta$. Fix a real number λ larger than that value, but smaller than $1/\delta$. We then solve the ODE (9) with starting points $\lambda \cdot (1, q)$ and record the limiting critical points. We proceed similarly for the other hyperplane $x_1 = -1/\delta$.

If a region in $\mathbb{R}^n$ is not visited during this process, then it does not intersect either hyperplane. If its critical points satisfy $-1/\delta < x_1 < 1/\delta$, then it is bounded. If a region is not bounded or unbounded, then it is declared *undecided*. Thus, undecided regions are close to being tangent to the hyperplane at infinity, where "close" refers to the tolerance δ.

4. How to use the software.

Our software is easy to use, even for beginners. We here offer a step-by-step introduction.[1] The first step is to download the programming language `Julia`. One navigates to the website https://julialang.org/downloads/ for the latest version. After it is downloaded and installed, we start `Julia` and enter the package manager by pressing the `]` button. Once in the package manager, type `add HypersurfaceRegions` and hit enter. This installs `HypersurfaceRegions.jl`. One leaves the package manager by pressing the back space button. To load our software into the current Julia session, type the following command:

```
using HypersurfaceRegions
```

Now, we are ready to use the implementation. Let's compute our first arrangement!!

[1] For a complete overview over our implementation we refer to the online documentation, which is available at https://juliaalgebra.github.io/HypersurfaceRegions.jl/.

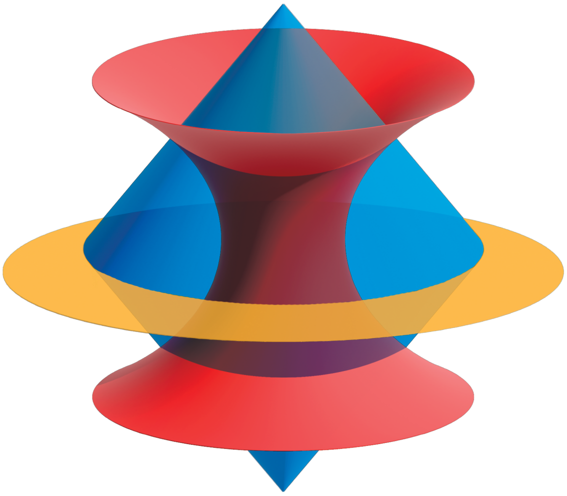


Figure 2. A hyperboloid and two paraboloids. The paraboloids intersect in a circle that spans a plane. This arrangement of four surfaces has 12 regions in $\mathbb{R}^3$. Four are contractible.

Being ambitious, we start with an example in $\mathbb{R}^3$. Consider the four surfaces in Figure 2. The yellow plane is defined by $z = 0$. The three quadrics are invariant under rotation around the z-axis. The two paraboloids are defined by $z = x^2 + y^2 - 1$ and $z = -x^2 - y^2 + 1$. The hyperboloid is defined by $x^2 + y^2 - \frac{3}{2}z^2 = \frac{1}{4}$. We input this arrangement into our software:

```
@var x y z
f = [z, x^2 + y^2 - 1 + z, x^2 + y^2 - 1 - z, x^2 + y^2 - 1/4 - 3/2*z^2]
regions(f)
```

The output for this input is shown on the left in Figure 3. We see that $\mathcal{U}$ has 12 regions, and each of them is uniquely identified by its sign pattern $\sigma \in \{-,+\}^4$. Six regions lie outside the hyperboloid, which means that $\sigma_4 = +$. Each of them is homotopy equivalent to a circle, so $\chi = 0$. The other six regions lie inside the hyperboloid, which means $\sigma_4 = -$.

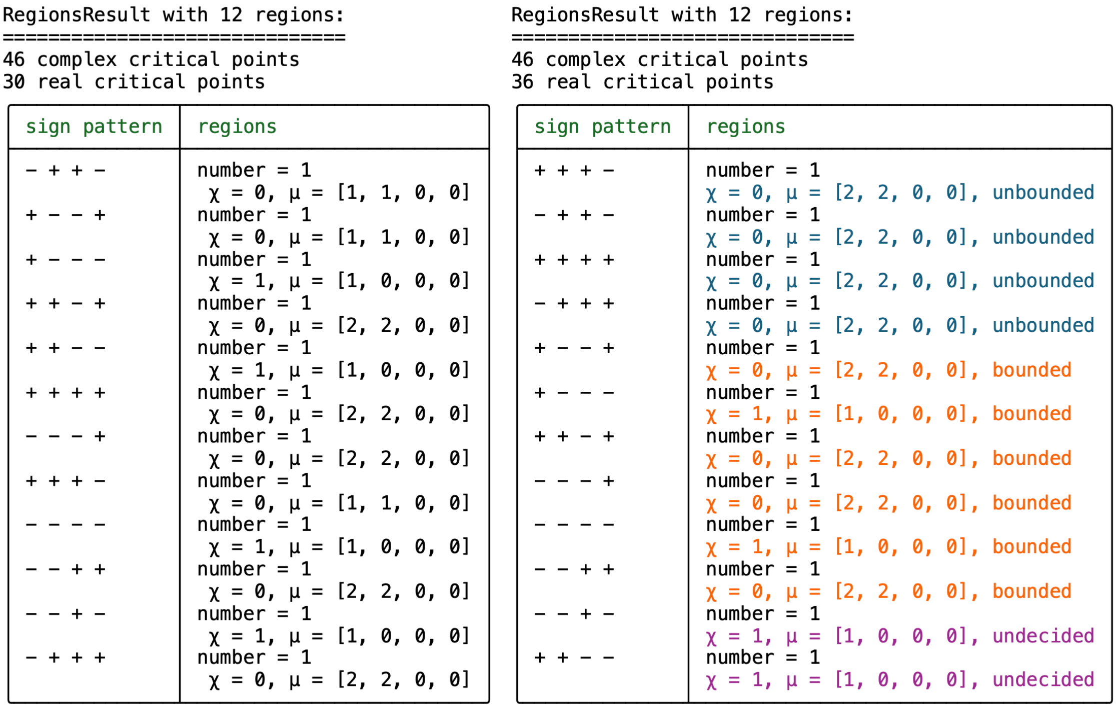
Two of the regions inside the hyperboloid are unbounded and also circular ($\chi = 0$). Finally, there are two bounded regions and two regions tangent to the hyperplane at infinity. They are all contractible ($\chi = 1$). We compute this information by setting a flag as follows:

```
regions(f; bounded_check = true)
```

The output of `HypersurfaceRegions.jl` for this input is shown on the right in Figure 3. This also reminds us that the number of real critical points depends on $q(x)$, which is random.

One application of our method is the study of discriminantal arrangements. Such arrangements arise whenever one is interested in the topology of real varieties that depend on parameters. The one-dimensional case of this is known as *real root classification*, where one expresses the number of real roots of a zero-dimensional system as a function of parameters.

To use the current version of `HypersurfaceRegions.jl` in this context, it is assumed that the discriminant is known and that it breaks up into smaller factors. Here is an example.



```
RegionsResult with 12 regions:
==============================
46 complex critical points
30 real critical points
┌──────────────┬─────────────────────────────┐
│ sign pattern │ regions                     │
├──────────────┼─────────────────────────────┤
│ - + + -      │ number = 1                  │
│              │  χ = 0, μ = [1, 1, 0, 0]    │
│ + - - +      │ number = 1                  │
│              │  χ = 0, μ = [1, 1, 0, 0]    │
│ + - - -      │ number = 1                  │
│              │  χ = 1, μ = [1, 0, 0, 0]    │
│ + + - +      │ number = 1                  │
│              │  χ = 0, μ = [2, 2, 0, 0]    │
│ + + - -      │ number = 1                  │
│              │  χ = 1, μ = [1, 0, 0, 0]    │
│ + + + +      │ number = 1                  │
│              │  χ = 0, μ = [2, 2, 0, 0]    │
│ - - - +      │ number = 1                  │
│              │  χ = 0, μ = [2, 2, 0, 0]    │
│ + + + -      │ number = 1                  │
│              │  χ = 0, μ = [1, 1, 0, 0]    │
│ - - - -      │ number = 1                  │
│              │  χ = 1, μ = [1, 0, 0, 0]    │
│ - - + +      │ number = 1                  │
│              │  χ = 0, μ = [2, 2, 0, 0]    │
│ - - + -      │ number = 1                  │
│              │  χ = 1, μ = [1, 0, 0, 0]    │
│ - + + +      │ number = 1                  │
│              │  χ = 0, μ = [2, 2, 0, 0]    │
└──────────────┴─────────────────────────────┘
```

```
RegionsResult with 12 regions:
==============================
46 complex critical points
36 real critical points
┌──────────────┬──────────────────────────────────────┐
│ sign pattern │ regions                              │
├──────────────┼──────────────────────────────────────┤
│ + + + -      │ number = 1                           │
│              │ χ = 0, μ = [2, 2, 0, 0], unbounded   │
│ - + + -      │ number = 1                           │
│              │ χ = 0, μ = [2, 2, 0, 0], unbounded   │
│ + + + +      │ number = 1                           │
│              │ χ = 0, μ = [2, 2, 0, 0], unbounded   │
│ - + + +      │ number = 1                           │
│              │ χ = 0, μ = [2, 2, 0, 0], unbounded   │
│ + - - +      │ number = 1                           │
│              │ χ = 0, μ = [2, 2, 0, 0], bounded     │
│ + - - -      │ number = 1                           │
│              │ χ = 1, μ = [1, 0, 0, 0], bounded     │
│ + + - +      │ number = 1                           │
│              │ χ = 0, μ = [2, 2, 0, 0], bounded     │
│ - - - +      │ number = 1                           │
│              │ χ = 0, μ = [2, 2, 0, 0], bounded     │
│ - - - -      │ number = 1                           │
│              │ χ = 1, μ = [1, 0, 0, 0], bounded     │
│ - - + +      │ number = 1                           │
│              │ χ = 0, μ = [2, 2, 0, 0], bounded     │
│ - - + -      │ number = 1                           │
│              │ χ = 1, μ = [1, 0, 0, 0], undecided   │
│ + + - -      │ number = 1                           │
│              │ χ = 1, μ = [1, 0, 0, 0], undecided   │
└──────────────┴──────────────────────────────────────┘
```

Figure 3. The output of `HypersurfaceRegions.jl` for the arrangement shown in Figure 2.

Example 12 (polynomial of degree 8). Consider the following polynomial in one variable t:

$$f(t) = t^8 + x \cdot t^7 + (x+y) \cdot t^6 + t^5 + (x+y) \cdot t^4 + (y+1) \cdot t^3 + t^2 + (x+y) \cdot t.$$

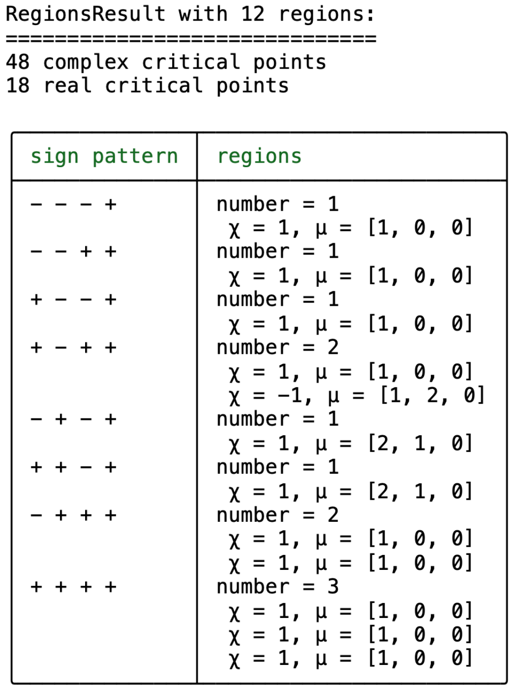
The coefficients depend linearly on two parameters x and y. The discriminant of $f(t)$ is a polynomial of degree 14 in x and y. This discriminant factors into four irreducible factors. Two of these have multiplicity two. We input the four factors into our software as follows:

```
@var x y
f = [x + y,
     23x^6 + 60x^5*y + 50x^4*y^2 + 16x^3*y^3 + 3x^2*y^4 - 78x^5
        - 336x^4*y - 478x^3*y^2 - 284x^2*y^3 - 76x*y^4 - 12y^5 - 87x^4
        - 144x^3*y + 54x^2*y^2 + 180x*y^3 + 68y^4 + 28x^3 + 24x^2*y
        - 58x*y^2 - 56y^3 - 87x^2 - 300x*y - 208y^2 - 78x - 72y + 23,
    x + 3y + 1,
    5x^2 + 4x*y + y^2 - 6x - 4y + 5]
regions(f)
```

The output is shown in Figure 4. Eight of the 16 sign patterns are realizable. The parameter plane is divided into 12 regions. For instance, $\sigma = (+, -, +, +)$ contributes two regions, one contractible and one with two holes ($\chi = -1$). The latter parametrizes polynomials with only two real roots. One sample point in that region is $(x, y) = (-1, 5)$.

You are now invited to run the following code and to match its output with Figure 1:

```
@var x y z
f = [x^2 + y^2 + z^2 - 1, x^2 + y^2 + z^2 - 4, 100*x^2 + 100*y^2 + z^2 - 9]
R = regions(f, bounded_check = true)
```



```
RegionsResult with 12 regions:
==============================
48 complex critical points
18 real critical points

┌──────────────┬───────────────────────┐
│ sign pattern │ regions               │
├──────────────┼───────────────────────┤
│ - - - +      │ number = 1            │
│              │  χ = 1, μ = [1, 0, 0] │
│ - - + +      │ number = 1            │
│              │  χ = 1, μ = [1, 0, 0] │
│ + - - +      │ number = 1            │
│              │  χ = 1, μ = [1, 0, 0] │
│ + - + +      │ number = 2            │
│              │  χ = 1, μ = [1, 0, 0] │
│              │  χ = -1, μ = [1, 2, 0]│
│ - + - +      │ number = 1            │
│              │  χ = 1, μ = [2, 1, 0] │
│ + + - +      │ number = 1            │
│              │  χ = 1, μ = [2, 1, 0] │
│ - + + +      │ number = 2            │
│              │  χ = 1, μ = [1, 0, 0] │
│              │  χ = 1, μ = [1, 0, 0] │
│ + + + +      │ number = 3            │
│              │  χ = 1, μ = [1, 0, 0] │
│              │  χ = 1, μ = [1, 0, 0] │
│              │  χ = 1, μ = [1, 0, 0] │
└──────────────┴───────────────────────┘
```

Figure 4. Output for the discriminantal arrangement in Example 12.

A range of additional features is available in `HypersurfaceRegions.jl`. For instance, one finds the critical points in the i-th region of the output by running the following command:

```
Ri = regions(R)[i]
critical_points(Ri)
```

One can also locate the region to which a given point (e.g., the origin) belongs, as follows:

```
p = [0, 0, 0]
membership(R, p)
```

We next discuss a few implementation details. We use *monodromy* [10], which was implemented in `HomotopyContinuation.jl` [5], for computing the critical points of $g(x)$. The ODE solver from `DifferentialEquations.jl` [16] is used for the Morse flows (9) between critical points.

To find critical points, we solve the rational function equations $\nabla \log(g(x)) = 0$ in (4). These are n equations in n variables $x_1, \dots, x_n$ and $k+1$ parameters $u = (s_1, \dots, s_k, t)$. For the monodromy, we need a start pair (x_0, u_0) such that x_0 is a solution for $\nabla \log(g(x)) = 0$ with parameters u_0. Since $\nabla \log(g(x))$ is linear in u, we can use linear algebra to get such a start pair. Indeed, we can sample a random point x_0 and compute u_0 by solving linear equations. However, this works only if $k+1 > n$, and if $\nabla \log(g(x_0))$ has full rank. To avoid these issues, we use the following trick. If $k+1 \leq n$, we define new auxiliary parameters $v = (v_1, \dots, v_{n-k})$. Using monodromy, we solve the system $\nabla \log(g(x)) - (Au + Bv) = 0$, where $A \in \mathbb{C}^{n\times(k+1)}$ and $B \in \mathbb{C}^{n\times(n-k)}$ are random matrices. Afterwards, we track the homotopy

$\nabla \log(g(x)) - \lambda \cdot (Au + Bv) = 0$ from $\lambda = 1$ to $\lambda = 0$ using the solutions we have just computed. The *parameter continuation theorem* [18; 4] asserts that this approach works.

We modified the default options in `HomotopyContinuation.jl` for the monodromy loops. In `HypersurfaceRegions.jl`, a computation finishes if the last 10 new monodromy loops have not provided any new solutions. The number 10 is a conservative parameter here. Its purpose is to increase the probability of finding all critical points. The tolerance parameter δ for bounded regions, as described in the end of Section 3, is set to the default value $\delta = 10^{-5}$.

5. EXPERIMENTS. This final section is dedicated to systematic experiments with our software. We experimented by running `HypersurfaceRegions.jl` on random instances, drawn from two classes:

(1) Arrangements defined by random polynomials.

(2) Arrangements defined by random spectrahedra.

The first class is self-explanatory. To create (1), we choose f_i at random from some probability distribution on the space of inhomogeneous polynomials of degree d in n variables.

The second class requires an explanation. We fix positive integers n and m, and we draw $n + 1$ samples $A_0, A_1, \ldots, A_n$ from the $\binom{m+1}{2}$-dimensional space of symmetric $m \times m$ matrices. We consider the arrangement defined by the $k = 2^m - 1$ principal minors of the matrix

$$A(x) = A_0 + x_1 A_1 + \cdots + x_n A_n. \tag{11}$$

The distinguished region where all k principal minors are positive is the *spectrahedron*.

The stratification of symmetric matrices by signs of principal minors was studied by Boege, Selover and Zubkov in [3]. We explore random low-dimensional slices of the *Boege–Selover–Zubkov stratification*. How many regions get created, and what is their topology?

Example 13 ($n = m = 3$). A prominent spectrahedron is the *elliptope*, which is given by

$$A(x, y, z) = \begin{bmatrix} 1 & x & y \\ x & 1 & z \\ y & z & 1 \end{bmatrix}.$$

The 2×2 minors of this matrix give the six facet equations of the 3-dimensional cube $[-1, 1]^3$. Our arrangement lives in $\mathbb{R}^3$, and it consists of these six planes plus the cubic surface

$$\det(A) = 2xyz - x^2 - y^2 - z^2 + 1. \tag{12}$$

For an illustration see Figure 5. Only four of the six facets are shown for better visibility.

We run `HypersurfaceRegions.jl` on this $k = 7$ instance. The arrangement has 43 regions. All have Euler characteristic 1. The six facet planes of the cube divide $\mathbb{R}^3$ into $27 = 1 + 8 + 12 + 6$ regions. One is bounded (the cube itself) and 26 are unbounded cones. The surface (12) divides the cube into 5 regions, and it divides four of the cones into 4 unbounded regions, for a total of $27 + 4 + 12 = 43$ regions. The projective arrangement has 39 regions.

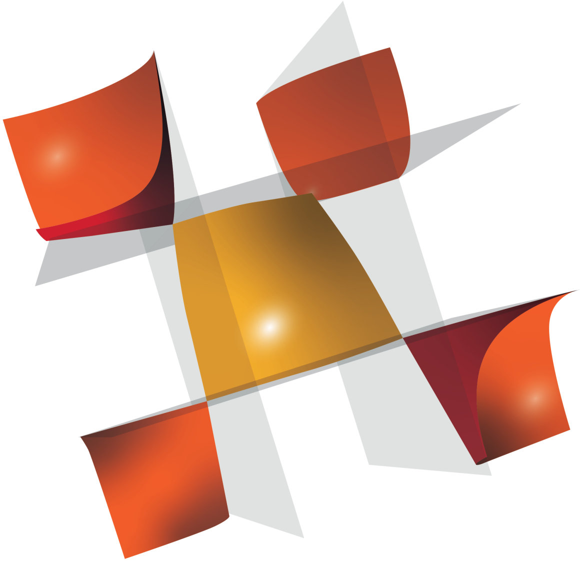


Figure 5. Arrangement given by the elliptope and the facet planes of the surrounding cube.

n	k	time	min-max #reg	min-max #σ	max #reg/σ	min-max χ
3	8	15.5	46, 325	30, 196	11	−5, 2
3	9	28.9	72, 462	41, 319	7	−2, 2
3	10	68.1	108, 612	83, 437	8	−3, 1
4	4	15.4	9, 38	8, 16	7	−6, 6
4	5	32.6	19, 87	15, 32	7	−7, 3
4	6	72.3	47, 192	32, 64	9	−7, 2
5	5	142.9	34, 67	24, 32	8	−8, 4

Table 1. Random affine arrangements defined by k quadrics in $\mathbb{R}^n$.

We now describe our experiments with arrangements of generic hypersurfaces. We run the program `HypersurfaceRegions.jl` on k polynomials in n variables of degree $d = 2$ and $d = 3$, where the coefficients are drawn independently from the standard Gaussian. Each experiment is carried out N times, where N is inverse proportional to the running time for each instance.

Our findings are presented in Table 1 for $d = 2$ and in Table 2 for $d = 3$. For each row we fix the parameters n and k. The time is the average running time per instance. This is measured in seconds. The fourth column concerns the total number of regions. We report the minimum number and the maximum number across all instances. The fifth column similarly reports the range for the number of realizable sign vectors σ. The sixth column gives the maximal number of regions per fixed sign pattern. And, finally, in the last column we report the minimum and maximum observed for the Euler characteristics χ of the regions.

n	k	time	min-max #reg	min-max #σ	max #reg/σ	min-max χ
3	5	25.0	34, 135	27, 32	8	−2, 2
3	6	59.5	67, 186	52, 64	7	−2, 2
3	7	107.5	123, 280	87, 126	7	−2, 2
4	4	127.4	19, 47	16, 16	6	−4, 3
4	5	348.5	46, 119	32, 32	8	−4, 3
5	3	290.9	8, 11	8, 8	2	−7, 4

Table 2. Random affine arrangements defined by k cubics in $\mathbb{R}^n$.

Let us discuss the first row in Table 1. The code runs 15.5 seconds, and it identifies ρ distinct regions in $\mathbb{R}^3$, where $46 \leq \rho \leq 325$. The number of realizable sign patterns is at most 196. This is less than the number $2^k = 2^8 = 256$ of all sign patterns. There were up to 11 regions per sign pattern. The Euler characteristic of any region was between -5 and 2.

Table 2 presents analogous results for cubics in $\mathbb{R}^n$ where $n = 3, 4, 5$. The second-last row concerns quintuples of cubics in $\mathbb{R}^4$. It takes less than six minutes to identify all regions, of which there are up to 119. We observed a narrow range $\{-4, \dots, 3\}$ of Euler characteristics.

Generic instances have no tangencies to the hyperplane at infinity. To experiment with that issue, we can try k paraboloids in $\mathbb{R}^n$. Here is an instance of $k = 4$ paraboloids for $n = 3$:

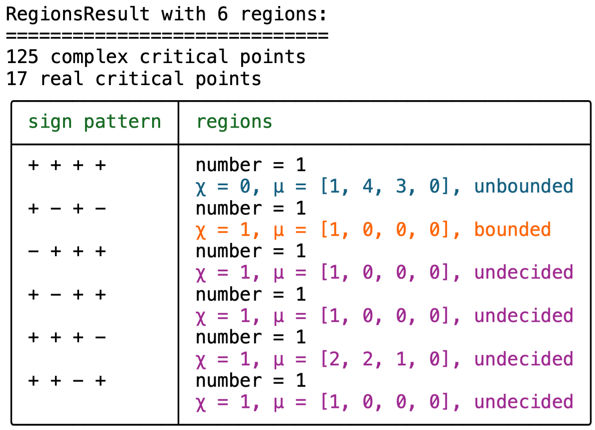
```
@var x y z
f = [3 + x + 3*y - z + (1 + 2*x + 4*y - 4*z)^2 + (2 + 3*x + 2*y + 3*z)^2,
     3 + x + 3*z + (3 - 3*x - 2*z)^2 + (3 + 3*x + 3*y + 4*z)^2,
     2 - 2*x - 2*y - 3*z + (2 - x + 4*z)^2 +  (2 + 3*x + y + 2*z)^2,
     1 - 3*x + 3*y - 3*z + (1 - 2*y + 2*z)^2 + (2 + x + 4*y)^2 ]
regions(f, bounded_check = true)
```

Each paraboloid contributes one undecided region touching the hyperplane at infinity. The output in Figure 6 consists of six regions, each with a unique sign pattern. Notice that the label "undecided" does *not* mean we claim this region touches the hyperplane at infinity. It means that our algorithm could not decide whether this region is bounded or unbounded.

We now turn to random spectrahedra. We sampled symmetric $m \times m$ matrices $A_0, \dots, A_n$ where the entries are independent standard Gaussians. The principal minors of the matrix $A(x)$ define an arrangement of $k = 2^m - 1$ hypersurfaces in the affine space $\mathbb{R}^n$. We ran `HypersurfaceRegions.jl` on this input. Our results are summarized in Table 3. For each row we fix the parameters n and m. The columns have the same meaning as before. For instance, the fifth column reports the range for the number of realizable sign vectors σ.

We also performed computations for $m \geq 4$, but we do not report them because we ran into numerical issues. Frequently, the Hessian of (6) at some critical point is almost singular.

SUPPLEMENT. The online supplement contains version 1 of `HypersurfaceRegions.jl`.



```
RegionsResult with 6 regions:
=============================
125 complex critical points
17 real critical points
┌──────────────┬──────────────────────────────────────────┐
│ sign pattern │ regions                                  │
├──────────────┼──────────────────────────────────────────┤
│ + + + +      │ number = 1                               │
│              │ χ = 0, μ = [1, 4, 3, 0], unbounded       │
│ + - + -      │ number = 1                               │
│              │ χ = 1, μ = [1, 0, 0, 0], bounded         │
│ - + + +      │ number = 1                               │
│              │ χ = 1, μ = [1, 0, 0, 0], undecided       │
│ + - + +      │ number = 1                               │
│              │ χ = 1, μ = [1, 0, 0, 0], undecided       │
│ + + + -      │ number = 1                               │
│              │ χ = 1, μ = [2, 2, 1, 0], undecided       │
│ + + - +      │ number = 1                               │
│              │ χ = 1, μ = [1, 0, 0, 0], undecided       │
└──────────────┴──────────────────────────────────────────┘
```

Figure 6. Output of `HypersurfaceRegions.jl` for four paraboloids in $\mathbb{R}^3$.

n	m	time	min-max #reg	min-max $\#\sigma$	max $\#\text{reg}/\sigma$	min-max χ
2	2	3.3	4, 8	4, 6	3	1, 1
2	3	4.4	38, 58	24, 35	9	−1, 1
3	3	13.4	80, 122	34, 38	21	−2, 1
4	3	46.3	117, 150	38, 38	26	−2, 1

Table 3. Random arrangements in $\mathbb{R}^n$ defined by $n+1$ symmetric $m \times m$ matrices.

References.

[1] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: a fresh approach to numerical computing", *SIAM Rev.* **59**:1 (2017), 65–98. MR Zbl

[2] J. Bisgard, "Mountain passes and saddle points", *SIAM Rev.* **57**:2 (2015), 275–292. MR Zbl

[3] T. Boege, J. Selover, and M. Zubkov, "Sign patterns of principal minors of real symmetric matrices", 2024. arXiv 2407.17826

[4] V. Borovik and P. Breiding, "A short proof for the parameter continuation theorem", *J. Symbolic Comput.* **127** (2025), art. id. 102373. MR Zbl

[5] P. Breiding and S. Timme, "HomotopyContinuation.jl: A package for homotopy continuation in Julia", pp. 458–465 in *Mathematical software — ICMS* 2018, edited by J. H. Davenport et al., Springer, Cham, 2018. Zbl

[6] T. Brysiewicz, H. Eble, and L. Kühne, "Computing characteristic polynomials of hyperplane arrangements with symmetries", *Discrete Comput. Geom.* **70**:4 (2023), 1356–1377. MR Zbl

[7] F. Catanese, S. Hoşten, A. Khetan, and B. Sturmfels, "The maximum likelihood degree", *Amer. J. Math.* **128**:3 (2006), 671–697. MR Zbl

[8] D. Cohen, G. Denham, M. Falk, and A. Varchenko, "Critical points and resonance of hyperplane arrangements", *Canad. J. Math.* **63**:5 (2011), 1038–1057. MR Zbl

[9] J. Cummings, J. D. Hauenstein, H. Hong, and C. D. Smyth, "Smooth connectivity in real algebraic varieties", *Numerical algorithms* (2024).

[10] T. Duff, C. Hill, A. Jensen, K. Lee, A. Leykin, and J. Sommars, "Solving polynomial systems via homotopy continuation and monodromy", *IMA J. Numer. Anal.* **39**:3 (2019), 1421–1446. MR Zbl

[11] N. Early, A. Geiger, M. Panizzut, B. Sturmfels, and C. H. Yun, "Positive del Pezzo geometry", preprint, 2023. To appear in *Annales de l'Institut Henri Poincaré Comb. Phys. Interact.* arXiv 2306.13604

[12] H. Hong, J. Rohal, M. S. E. Din, and E. Schost, "Connectivity in semi-algebraic sets, I", 2020. arXiv 2011.02162

[13] L. Kastner and M. Panizzut, "Hyperplane arrangements in `polymake`", pp. 232–240 in *Mathematical software—ICMS 2020*, edited by A. M. Bigatti et al., Lecture Notes in Comput. Sci. **12097**, Springer, 2020. MR Zbl

[14] J. J. Moré and T. S. Munson, "Computing mountain passes and transition states", *Math. Program.* **100**:1 (2004), 151–182. MR Zbl

[15] L. Nicolaescu, *An invitation to Morse theory*, 2nd ed., Springer, 2011. MR Zbl

[16] C. Rackauckas and Q. Nie, "Differentialequations.jl — a performant and feature-rich ecosystem for solving differential equations in Julia", *Journal of Open Research Software* **5**:1 (2017), art. id. 15.

[17] B. Reinke and K. Wang, "Hypersurface arrangements with generic hypersurfaces added", 2024. arXiv 2412.20869

[18] 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

[19] B. Sturmfels and S. Telen, "Likelihood equations and scattering amplitudes", *Algebr. Stat.* **12**:2 (2021), 167–186. MR Zbl

RECEIVED: 20 Sep 2024 REVISED: 30 Dec 2024 ACCEPTED: 27 Feb 2025

PAUL BREIDING:

pbreiding@uni-osnabrueck.de
University of Osnabrück, Osnabrück, Germany

BERND STURMFELS:

bernd@mis.mpg.de
Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany

KEXIN WANG:

kexin_wang@g.harvard.edu
Kexin Wang, Harvard University, Cambridge, MA, United States