Journal of Software for Algebra and Geometry

The `GroebnerWalk.jl` package for `OSCAR`

Kamillo Ferry and Francesco Nowell

vol 16 2026

JSAG **16** (2026), 21–30
https://doi.org/10.2140/jsag.2026.16.21

The Journal of Software for
Algebra and Geometry

The `GroebnerWalk.jl` package for `OSCAR`

KAMILLO FERRY AND FRANCESCO NOWELL

ABSTRACT: Gröbner bases are a central tool in computational algebra, but it is well-known that their ease of computation rapidly deteriorates with increasing number of variables and/or degree of the input generators. Due to the connection between polyhedral geometry and Gröbner bases through the *Gröbner fan*, one can attempt an incremental approach to compute Gröbner bases. First computing a Gröbner basis with respect to an "easy" term ordering and transforming that result to a Gröbner basis with respect to the desired term ordering by using information about this polyhedral fan is done by a family of algorithms termed as *Gröbner walk*. We implemented two variants of the Gröbner walk in the computer algebra system `OSCAR` and compared their performance with classical Gröbner basis methods already found in `OSCAR`.

1. INTRODUCTION. The *Gröbner walk* is an approach to reduce the computational complexity of Gröbner basis computations first proposed in [4]. Algorithms of this type work by exploiting the geometry of the *Gröbner fan* which has been introduced by Mora and Robbiano [12]. This is a polyhedral fan associated to a polynomial ideal I, the maximal cones of which are in one-to-one correspondence with the Gröbner bases of I as one varies over all term orderings. These algorithms belong to the wider class of Gröbner basis methods which operate incrementally via subsequent *conversions*; e.g., FGLM [8] or Hilbert-driven Buchberger [20].

The aim of the Gröbner walk is to compute a Gröbner basis for an ideal I given two term orderings of a polynomial ring, the start ordering $<_s$ and the target ordering $<_t$. The algorithm starts by computing a *Gröbner basis* with respect to $<_s$ and the corresponding cone in the Gröbner fan. It then computes a boundary point in the direction of the cone corresponding to desired target ordering. This point ω is then used to retrieve a basis of the ideal of initial forms $\mathrm{in}_\omega(I)$, which in the Gröbner fan corresponds to the lower-dimensional cone on which ω lies. This basis is subsequently lifted to the Gröbner basis of I corresponding to the adjacent full-dimensional cone. This process is repeated until one obtains the target Gröbner basis after finitely many steps. Our package `GroebnerWalk.jl` implements the original algorithm described in [4] (which we refer to as the *standard walk*) as well as the *generic Gröbner walk* of [10], which combines methods of the standard walk with techniques from symbolic computation. It adds to the array of Gröbner basis algorithms already accessible in the Julia ecosystem via the `msolve/AlgebraicSolving.jl`, `OSCAR` [6; 14] and `Groebner.jl` packages.

MSC2020: primary 13P10; secondary 14T10.
Keywords: Gröbner walk, Gröbner fan, computer algebra system, implementation.
`GroebnerWalk.jl` version 1

2. BASICS OF GRÖBNER FANS. A *term ordering* on $R = k[x_1, \ldots, x_n]$ is a relation $<$ on the monomials of R which is a strict total well-ordering that satisfies

$$\text{for all } \alpha, \beta, \gamma \in \mathbb{N}^n : \quad x^\alpha < x^\beta \Longrightarrow x^{\alpha+\gamma} < x^{\beta+\gamma}.$$

For any non-zero polynomial $f \in R$ and term ordering $<$, there is a unique maximal term $c_\alpha x^\alpha$ with respect to $<$ which is called the *leading term* $\text{in}_<(f)$.

The *initial ideal* of a non-zero ideal $I \trianglelefteq R$ with respect to $<$ is

$$\text{in}_<(I) = \langle \text{in}_<(f) \mid f \in I \rangle.$$

Sometimes, this is also referred to as *leading ideal* in literature and also in OSCAR. A finite set $G = \{g_1, \ldots g_s\} \subset R$ of polynomials is called a *Gröbner basis* for an ideal I w.r.t $<$ if $I = \langle G \rangle$ and

$$\langle \text{in}_<(g_1) \ldots \text{in}_<(g_s) \rangle = \text{in}_<(I).$$

Example 1. We demonstrate the theory and the corresponding functions in OSCAR using the ideal

$$I = \langle y^4 + x^3 - x^2 + x, x^4 \rangle \triangleleft \mathbb{Q}[x, y] \tag{1}$$

as running example.

The following code snippet defines this ideal and calculates the initial ideal with respect to the lexicographic ordering $<_{\text{lex}}$:

```
julia> using Oscar
julia> R, (x,y) = QQ[:x, :y]; I = ideal([y^4+ x^3-x^2+x,x^4]);
julia> leading_ideal(I; ordering=lex(R))
Ideal generated by
  y^16
  x
```

Given a vector $\omega \in \mathbb{R}^n_{\geq 0}$ and a term ordering $<$, we can define the *weight ordering*

$$\alpha <_\omega \beta : \iff \langle \omega, \alpha \rangle < \langle \omega, \beta \rangle \text{ or } (\langle \omega, \alpha \rangle = \langle \omega, \beta \rangle \text{ and } x^\alpha < x^\beta).$$

This is also a term ordering, sometimes referred to as the *refinement* of the weight vector ω by $<$. Upon relaxing the comparison with $<$, we obtain the *partial weight ordering* $\prec_\omega$:

$$\alpha \prec_\omega \beta :\Longleftrightarrow \langle \alpha, \omega \rangle < \langle \beta, \omega \rangle,$$

which in general is not a term ordering. For example, in the setting of Example 1 a refinement of the weight vector $(2, 1)$ by $<_{\text{lex}}$ may be obtained in OSCAR as follows:

```
julia> weight_ordering([2,1], lex(R))
matrix_ordering([x, y], [2 1])*lex([x, y])
```

The *initial form* $\mathrm{in}_\omega(f)$ of f with respect to ω is defined as the sum of the terms of f which are maximal with respect to $\prec_\omega$. The *generalized initial ideal* of I with respect to ω is the ideal generated by the initial forms

$$\mathrm{in}_\omega(I) = \langle \mathrm{in}_\omega(f) \mid f \in I \rangle.$$

Note that $\mathrm{in}_\omega(I)$ is not a monomial ideal in general. For example, for the ideal I from Example 1, the weight vector $(4, 3) \in \mathbb{R}^2$ yields a partial weight ordering $\prec_\omega$ such that $\mathrm{in}_\omega(I) = \langle x^3 + y^4, x^4 \rangle$ which is not monomial.

For a fixed ideal I and term ordering $<$ we say that a weight vector ω *represents* $<$ if $\mathrm{in}_\omega(I) = \mathrm{in}_<(I)$. If a Gröbner basis G w.r.t. $<$ is given, a necessary and sufficient condition for ω to represent $<$ is that $\mathrm{in}_\omega(g) = \mathrm{in}_<(g)$ holds for all $g \in G$. The crucial connection to polyhedral geometry is that the set of all weight vectors representing a given term ordering $<$ lie in the relative interior of a full-dimensional polyhedral cone in $\mathbb{R}^n$ with integer generators. Upon taking the closure of this cone, and then the union of all such cones varying over all term orderings, one obtains a full-dimensional rational polyhedral fan, called the *Gröbner fan* of I. We denote this by $\mathbb{G}(I)$. Another key notion is that of a *marked Gröbner basis*, which is a Gröbner basis $G = \{g_1, \dots g_s\}$ for I w.r.t. $<$ with the following additional properties:

(i) G is minimal w.r.t inclusion, i.e.,

$$\mathrm{in}_<(G \setminus \{g_i\}) \subsetneq \mathrm{in}_<(I) \text{ for all } i \in \{1, \dots, s\}.$$

(ii) G is monic, i.e., the coefficient of $\mathrm{in}_<(g_i)$ is equal to 1 for all $i \in \{1, \dots, s\}$.

(iii) G is reduced, i.e.,

$$\text{for all } i, j \in \{1, \dots, s\} \text{ with } i \neq j, \text{ no term of } g_i \text{ is divisible by } \mathrm{in}_<(g_j).$$

(iv) The leading terms of elements of G are *marked*, in the sense that each $g \in G$ is formally encoded as the pair (g, x^α), where $x^\alpha = \mathrm{in}_<(g)$.

Marked Gröbner bases are unique in the sense that if two term orderings give rise to the same initial ideal, then their marked Gröbner bases coincide.

Some key theoretical results behind the Gröbner walk are stated below. Detailed proofs and additional context may be found in Chapters 1 and 2 of [13].

Theorem 2. *For a non-zero ideal $I \lhd R$, the following sets are in one-to-one correspondence*:

$$\{\mathrm{in}_<(I), < \textit{ is a term ordering}\} \leftrightarrow \{\textit{marked Gröbner bases of } I\} \leftrightarrow \{\textit{full-dimensional cones of } \mathbb{G}(I)\}.$$

The first correspondence in Theorem 2 is immediate, whilst the second correspondence is a consequence of [17, Theorem 1.11]: a marked Gröbner basis G encodes a facet description of the corresponding cone. Each element $(g, x^\alpha) \in G$ prescribes a leading term of g with respect to $<$ which translate to linear integral inequalities for the weight vectors ω given by

$$\alpha\omega \geq \beta\omega, \quad \text{where } x^\alpha = \mathrm{in}_<(g) \text{ and } \beta \in \mathrm{supp}(g) \setminus \{\alpha\} \text{ for every } g \in G,$$

where $\mathrm{supp}(g)$ denotes the set of exponent vectors $\alpha \in \mathbb{Z}^n$ such that x^α has non-zero coefficient in g.

Lower-dimensional cones in $\mathbb{G}(I)$ correspond to generalized initial ideals $\text{in}_\omega(I)$, where ω is any weight vector in the relative interior of said cone. Generically, such ideals are "almost monomial", and may be retrieved with the help of the following lemma.

Lemma 3. *Let G be a marked Gröbner basis of I with respect to $<$ and $\omega \in \mathbb{Q}^n_{\geq 0}$ be a weight vector on the boundary of the corresponding cone in $\mathbb{G}(I)$. The set*

$$\text{in}_\omega(G) = \{\text{in}_\omega(g),\, g \in G\}$$

is a marked Gröbner basis of $\text{in}_\omega(I)$ *with respect to $<$, where the marking of* $\text{in}_\omega(g)$ *is taken to be that of g.*

At every step of the Gröbner walk, a basis of this form converted with Buchberger's algorithm and then lifted to the basis of I corresponding to the adjacent full-dimensional cone, which corresponds to the refinement $<'_\omega$. Recall that the *normal form* of a non-zero polynomial $f \in R$ with respect to a Gröbner basis G is the unique remainder obtained upon dividing f by G with the multivariate division algorithm; see [5, page 83]. We denote this by $\bar{f}^G$.

Lemma 4. *Let H be the Gröbner basis of I with respect to a term ordering $<$ and let $\omega \in \mathbb{Q}^n$ lie on the boundary of the cone in $\mathbb{G}(I)$ corresponding to $<$.*

If $M = \{m_1, \ldots, m_r\}$ is the marked Gröbner basis of $\text{in}_\omega(I)$ *with respect to the refinement ordering $<'_\omega$, then*

$$G := \{m_1 - \bar{m}_1^H, \ldots, m_r - \bar{m}_r^H\}$$

is a Gröbner basis of I with respect to $<'_\omega$.

The Gröbner basis G that is obtained from M is referred to as the *lift* of M by $<$; it is not a marked Gröbner basis since it is not necessarily reduced.

In the setting of the Gröbner walk, the ordering $<$ from Lemma 4 is taken to be the start ordering, with respect to which a marked Gröbner basis is already known. The marked Gröbner basis which can be computed by reducing the lift Gröbner basis G corresponds to a full-dimensional cone in $\mathbb{G}(I)$ adjacent to the cone of $<$ such that ω lies in their common intersection. As ω is chosen to be a point on the line segment from the cone of $<$ to that of the target ordering, the cone corresponding to $<'_\omega$ will be "closer" to the cone of the target ordering in the Gröbner fan. Thus, the process of subsequent passing to the generalized initial ideal and lifting to the adjacent basis may be repeated finitely many times to obtain a basis with respect to the target ordering.

3. FUNCTIONALITY. Our implementation of the Gröbner walk ships with `OSCAR` since version 1.2.0, thus it suffices to load `OSCAR`. There is a straightforward interface through the function `groebner_walk`.

Example 5. Continuing from example Example 1, we can calculate a Gröbner basis of the ideal

$$I = \langle y^4 + x^3 - x^2 + x,\, x^4 \rangle \triangleleft \mathbb{Q}[x, y]$$

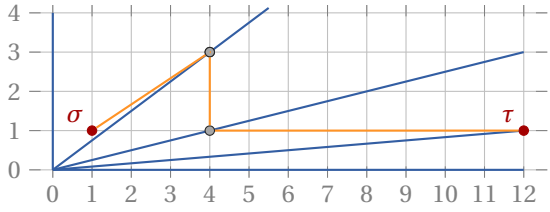

Figure 1. The Gröbner fan of the ideal $\langle y^4+x^3-x^2+x, x^4\rangle$. Each point denotes an intermediate weight vector for which a Gröbner basis is computed. For implementation reasons, we choose an integer weight vector in every intermediate step.

with respect to $<_{\text{lex}}$ by starting from a Gröbner basis for the *graded reverse lexicographic ordering* $<_{\text{degrevlex}}$. In OSCAR, every polynomial ring comes with an internal ordering that is used for computations involving orderings. By default, this internal ordering is $<_{\text{degrevlex}}$. We adopt this interface and take the internal ordering as default if no other start ordering is specified.

Thus, it suffices to call the Gröbner walk in the following way to calculate a Gröbner basis for $<_{\text{lex}}$ on $\mathbb{Q}[x, y]$ from a Gröbner basis with respect to $<_{\text{degrevlex}}$:

```
julia> groebner_walk(I, lex(R))
Grobner basis with elements
  1: x + y^12 - y^8 + y^4
  2: y^16
with respect to the ordering
  lex([x, y])
```

The corresponding computation path in the Gröbner fan is shown in Figure 1.

When calling `groebner_walk`, we construct the necessary initial information for the algorithm in the background and start the computation. This way, the user faces a single, uniform interface to each implemented algorithm. The selection of the algorithm used is done by an (optional) *keyword argument* `algorithm=`.

Our implementation also offers additional diagnostic output to trace the computation. For example this allows us to inspect the intermediate steps of the computation in Example 5 which are shown in Figure 1:

```
julia> set_verbosity_level(:groebner_walk, 1);
julia> groebner_walk(I, lex(R))
Results for standard_walk
Crossed Cones in:
ZZRingElem[1, 1]
ZZRingElem[4, 3]
ZZRingElem[4, 1]
ZZRingElem[12, 1]
Cones crossed: 4
```

```
Grobner basis with elements
  1: x + y^12 - y^8 + y^4
  2: y^16
with respect to the ordering
  lex([x, y])
```

The default choice is the variant by [4] (which can also be specified by setting `algorithm=:standard`). Another supported choice is `:generic` for the *generic walk* by [10].

The default choices for starting and target ordering are the internal ordering of `R` and the lexicographical ordering respectively. If not specified further, this internal ordering is usually the *degree reverse lexicographical ordering*. In general, it is not obvious though what is a sensible choice for a starting ordering. One possible way to alleviate this is to use *Gröbner basis detection methods*, which we discuss in Section 6.

As a word of caution, since we rely on the implementation of Gröbner bases in `OSCAR` which in turn is based on the facilities for Gröbner bases in Singular, we also inherit the restrictions on computations with Gröbner bases. Namely, the backend in Singular only supports weights less than $2^{31} - 1$. This means that Gröbner walk computations which involve a significant blowing up of weight vectors throw an `InexactError`. At present, we do not handle this limitation.

4. TECHNICAL CONTRIBUTION. We implement two variants of Gröbner walk algorithms, the standard walk by [4] and the generic walk by [10]. For the generic walk, we also provided a naive implementation of marked Gröbner bases.

Problems of Gröbner basis conversion arise in a wide variety of contexts. By default, the `OSCAR` [14] function `groebner_basis` computes a Gröbner basis using Buchberger's algorithm. This approach did not terminate in reasonable time in our larger examples.

Several other Gröbner basis algorithms have been implemented in `OSCAR` [7; 8; 16], and may be used via the keyword `algorithm=` when calling `groebner_basis`. While all of these methods are improvements on Buchberger's algorithm, they each come with their own limitations; for example, FGLM [8] is only applicable to zero-dimensional ideals, whereas the current `OSCAR` implementation of the F4 algorithm [7] may be called for ideals over the rationals or finite fields of machine-size characteristic, but only calculates Gröbner bases for $<_{\text{degrevlex}}$. In contrast to this, the Gröbner walk (and our implementation in `OSCAR`) works in full generality; it may be called on ideals over $\mathbb{Q}$ or $\mathbb{F}_p$ of arbitrary dimension and for arbitrary term orderings.

This last fact makes it especially well-suited for problems of elimination. Furthermore, due to the variable and unpredictable performance of Gröbner basis computations on generic ideals, it is advantageous to have a variety of options for these tasks.

Our implementation is included in version 1.5.0 of `OSCAR` as an *experimental package*, which means our implementation is be shipped with `OSCAR` as a submodule. Alternatively, the implementation is provided as self-contained Julia package with dependency on `OSCAR` which can be found at [9]. Calling

polynomial systems			characteristics				
name	description	ref	$<_\sigma$	$<_\tau$	$\lvert G\rvert$	$\lvert G_\sigma\rvert$	$\lvert G_\tau\rvert$
cyclic5	the cyclic 5-roots problem	[2]	degrevlex	lex	5	20	30
cyclic6	the cyclic 6-roots problem				6	45	70
katsura6	a problem of magnetism in physics	[11]	degrevlex	lex	7	41	64
katsura7					8	74	128
katsura8					9	143	256
agk4	a parametric Bézier surface	[1]	(2)	(2)	3	3	29
newell	the Newell teapot	[19]	(2)	(2)	3	12	39
tran3.3	[18, Example 3.3]	[18]	degrevlex	lex	2	5	10

Table 1. Overview of the polynomial systems $I = \langle G\rangle$ chosen for the comparison with a summary of characteristics.

the function `GroebnerWalk.groebner_walk` calls the GroebnerWalk.jl version of our implementation, which may differ from the OSCAR version in the future.

5. COMPARISON TO CLASSICAL GRÖBNER BASIS ALGORITHMS. For our comparisons we choose two types of problems and ran computations over $\mathbb{Q}$ and $\mathbb{F}_p$ for $p = 11863279$. The first kind are computations of lexicographic Gröbner bases of zero-dimensional ideals for solving systems of polynomial equations. The chosen systems are from Jan Verschelde's database and included in the software `PHCpack` [21].[1] They are commonly used in the benchmarks of polynomial solvers. The second kind are computations of Gröbner bases of ideals of dimension ≥ 1 with respect to *elimination term orderings*. These computations arise in problems of *implicitization* of surfaces given in parametric form, such as in the `agk` [1] and `newell` [19] examples.

Following [19] we chose the start and target orderings for those two problems as represented by the matrices

$$<_\sigma = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad <_\tau = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}. \tag{2}$$

We ran the comparisons on a MacBook Pro with an 2.4GHzApple M2 Max. Each computation was allotted a maximum of 16GBof memory. We used macOS 26.0 with Julia 1.10.10 and `OSCAR` 1.5.0. The results of the comparison are shown in Table 2. The data and code to run the benchmarks can be found at https://zenodo.org/records/17473425.

Owing to the high upper bounds in Gröbner basis computations and the resulting unpredictability, we report mixed results in our comparisons. While the implementation of classical Gröbner basis algorithms

[1]The database can be found at http://homepages.math.uic.edu/ jan/demo.html.

	runtime					
system	standard walk		generic walk		groebner_basis	
	$\mathbb{Q}$	$\mathbb{F}_p$	$\mathbb{Q}$	$\mathbb{F}_p$	$\mathbb{Q}$	$\mathbb{F}_p$
cyclic5	8.35ms	5.29ms	0.13s	0.23s	3.82ms	0.68ms
cyclic6	0.17s	0.04s	3.42s	7.67s	0.08s	0.04s
katsura6	0.20s	0.06s	13.23s	10.93s	20.72s	-
katsura7	2.37s	0.58s	365.80s	156.02s	59.80s	-
katsura8	23.35s	4.92s	-	-	-	-
agk4	3.98s	0.71s	39.79s	23.95s	0.55s	0.10s
newell	20.42s	3.17s	-	833.53s	808.14s	0.16s
tran3.3	0.40s	0.18s	3.40s	0.66s	0.05s	3.35ms

Table 2. Average runtimes for the Gröbner basis computations in Table 1 using our implementations of the Gröbner walk as well as OSCAR's built-in groebner_basis function. Missing entries indicate a computation that timed out. The cutoff was 3000s, $p = 11863279$.

in OSCAR performs reasonably well on the zero-dimensional cyclic ideals and some instances from implicitization, there are also examples where the Gröbner walk performed better or produced a Gröbner basis at all. We believe that the poor performance of the generic walk despite its theoretical advantages is due to the currently sub-optimal implementation. For example, the reduction steps are still performed using naive polynomial long division without any of the possible algorithmic improvements. Comparison with the Macaulay2 implementation [15] yielded mixed results, despite the implementations being very similar on a theoretical level. The results are presented in Table 3. Interestingly, none of the computations involving the katsura polynomial systems terminated in Macaulay2. We attribute the generally superior performance of the OSCAR implementation of the standard walk to the more optimized methods for polyhedral geometry and Gröbner basis computation via the Polymake and Singular backends respectively. The fact that the Macaulay2 implementation of the generic walk performs better than OSCAR in certain instances is likely due to the native markedGB data structure, which allows for the encoding of marked Gröbner bases in Macaulay2 without the explicit specification of a term ordering.

6. FUTURE DIRECTIONS. To compare the performance of the standard walk to the generic walk, we analyzed profiling data for the tran3.3 and agk4 instances. Those examples have a comparably big difference in computation time for standard and generic walk while the runtime of the generic walk is still low enough to reasonably compute traces.

In the generic walk computations the most significant bottleneck is the calculation of normal forms with respect to the symbolic intermediate orderings. This is due to the fact that we had to implement division with remainder naively as the existing implementations require at least a weight ordering with known weight vector. The next step would be to adopt a linear algebra approach to computing normal forms using Macaulay matrices. However, there is currently no user-facing OSCAR function for this and

	avg. runtime over $\mathbb{Q}$					
system	standard walk		generic walk		groebner_basis / gb	
	OSCAR	M2	OSCAR	M2	OSCAR	M2
cyclic5	8.35ms	0.02s	0.13s	0.25s	3.82ms	16.2ms
cyclic6	0.17s	0.69s	3.42s	4.29s	0.08s	0.55s
katsura6	0.20s	-	13.23s	-	20.72s	-
agk4	3.98s	0.93 s	39.79s	2.19s	0.55s	0.75s
	avg. runtime (s.) over $\mathbb{F}_p$					
cyclic5	5.29ms	0.27s	0.23s	4.16s	0.68ms	3.23ms
cyclic6	0.04s	0.27s	7.67s	4.19s	0.04s	0.12s
katsura6	0.06s	-	10.93s	-	-	-
agk4	0.71s	2.97s	23.95s	1.69s	0.10s	0.12s

Table 3. Side-by-side comparison of the standard and generic walks in OSCAR and Macaulay2 over $\mathbb{Q}$ and $\mathbb{F}_p$ ($p = 11863279$, cutoff 3000s)

in any case, such a method would have to be adapted to be compatible with our marked Gröbner basis structure. Also, both walks ultimately rely on the Singular for Gröbner basis computations, which at the time of writing does not support computations with weight orderings with entries larger than $2^{31} - 1$. Currently, the only work-around would be to write new Gröbner basis functions in OSCAR (which itself supports arbitrary precision integers) without any interaction with Singular, the implementation of which is a worthwhile avenue itself to extend the functionality of OSCAR but lies beyond our scope.

An important point in any variant of the Gröbner walk is the choice of starting term ordering to compute a first Gröbner basis. While rare, the specified generators of an ideal might already form a Gröbner basis with respect to some term ordering. There exist criteria for *Gröbner basis detection*; see, e.g., Chapter 3 in [17]. By employing such criteria, we can avoid the first computation of a Gröbner basis. A detection criterion has been implemented in Julia by [3].

ACKNOWLEDGMENTS. We would like to thank Carlos Améndola, Anne Frühbis-Krüger, Ben Hollering, Michael Joswig, Yue Ren, as well as an anonymous referee for helpful comments, guidance and pointers on how to improve our implementation.

KF is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy — The Berlin Mathematics Research Center MATH+ (EXC-2046/1, project ID: 390685689). FN was supported by the SPP 2458 "Combinatorial Synergies", funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) — project ID: 539875257.

SUPPLEMENT. The online supplement contains version 1 of GroebnerWalk.jl.

REFERENCES.

[1] B. Amrhein, O. Gloor, and W. Küchlin, "On the walk", *Theoret. Comput. Sci.* **187**:1-2 (1997), 179–202. MR

[2] G. Björck and R. Fröberg, "A faster way to count the solutions of inhomogeneous systems of algebraic equations, with applications to cyclic n-roots", *Journal of Symbolic Computation* **12**:3 (1991), 329–336.

[3] V. Borovik and T. Duff, "SAGBI and Gröbner bases detection", preprint, 2024. arXiv 2404.16796

[4] S. Collart, M. Kalkbrener, and D. Mall, "Converting bases with the Gröbner walk", *J. Symbolic Comput.* **24**:3-4 (1997), 465–469. MR

[5] D. A. Cox, J. Little, and D. O'Shea, *Ideals, varieties, and algorithms: An introduction to computational algebraic geometry and commutative algebra*, 4th ed., Springer, 2015. MR

[6] W. Decker, C. Eder, C. Fieker, M. Horn, and M. Joswig (editors), *The computer algebra system OSCAR—algorithms and examples*, Algorithms and Computation in Mathematics **32**, Springer, 2025. MR

[7] J.-C. Faugére, "A new efficient algorithm for computing Gröbner bases (F_4)", *J. Pure Appl. Algebra* **139**:1-3 (1999), 61–88. MR

[8] J. C. Faugère, P. Gianni, D. Lazard, and T. Mora, "Efficient computation of zero-dimensional Gröbner bases by change of ordering", *J. Symbolic Comput.* **16**:4 (1993), 329–344. MR

[9] K. Ferry and F. Nowell, "GroebnerWalk.jl", 2024, https://github.com/ooinaruhugh/GroebnerWalk.jl.

[10] K. Fukuda, A. N. Jensen, N. Lauritzen, and R. Thomas, "The generic Gröbner walk", *J. Symbolic Comput.* **42**:3 (2007), 298–312. MR

[11] S. Katsura, "Spin glass problem by the method of integral equation of the effective field", *New Trends in Magnetism* (1990), 110–121.

[12] T. Mora and L. Robbiano, "The Gröbner fan of an ideal", *J. Symbolic Comput.* **6**:2-3 (1988), 183–208. MR

[13] F. Nowell, *The Gröbner walk revisited*, master's thesis, Technische Universität Berlin, 2025.

[14] OSCAR, "OSCAR: Open Source Computer Algebra Research system, Version 1.4.1", 2025, https://www.oscar-system.org.

[15] D. Peifer, "dylanpeifer/GroebnerWalk", 2019, https://github.com/dylanpeifer/GroebnerWalk.

[16] B. Simões, "An Hilbert-driven strategy for signature-based Gröbner basis algorithms", pp. 13–37 in *Future vision and trends on shapes, geometry and algebra*, edited by R. De Amicis and G. Conti, Springer Proc. Math. Stat. **84**, Springer, 2014. MR

[17] B. Sturmfels, *Gröbner bases and convex polytopes*, University Lecture Series **8**, Amer. Math. Soc., Providence, RI, 1996. MR

[18] Q.-N. Tran, "A fast algorithm for Gröbner basis conversion and its applications", *Journal of Symbolic Computation* **30**:4 (2000), 451–467.

[19] Q.-N. Tran, "Efficient Groebner walk conversion for implicitization of geometric objects", *Comput. Aided Geom. Design* **21**:9 (2004), 837–857. MR

[20] C. Traverso, "Hilbert functions and the Buchberger algorithm", *J. Symbolic Comput.* **22**:4 (1996), 355–376. MR

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

RECEIVED: 16 Mar 2025 REVISED: 29 Oct 2025 ACCEPTED: 10 Feb 2026

KAMILLO FERRY:

ferry@math.tu-berlin.de
Institute of Mathematics, Technische Universität Berlin, Berlin, Germany

FRANCESCO NOWELL:

nowell@tu-berlin.de
Institute of Mathematics, Technische Universität Berlin, Berlin, Germany