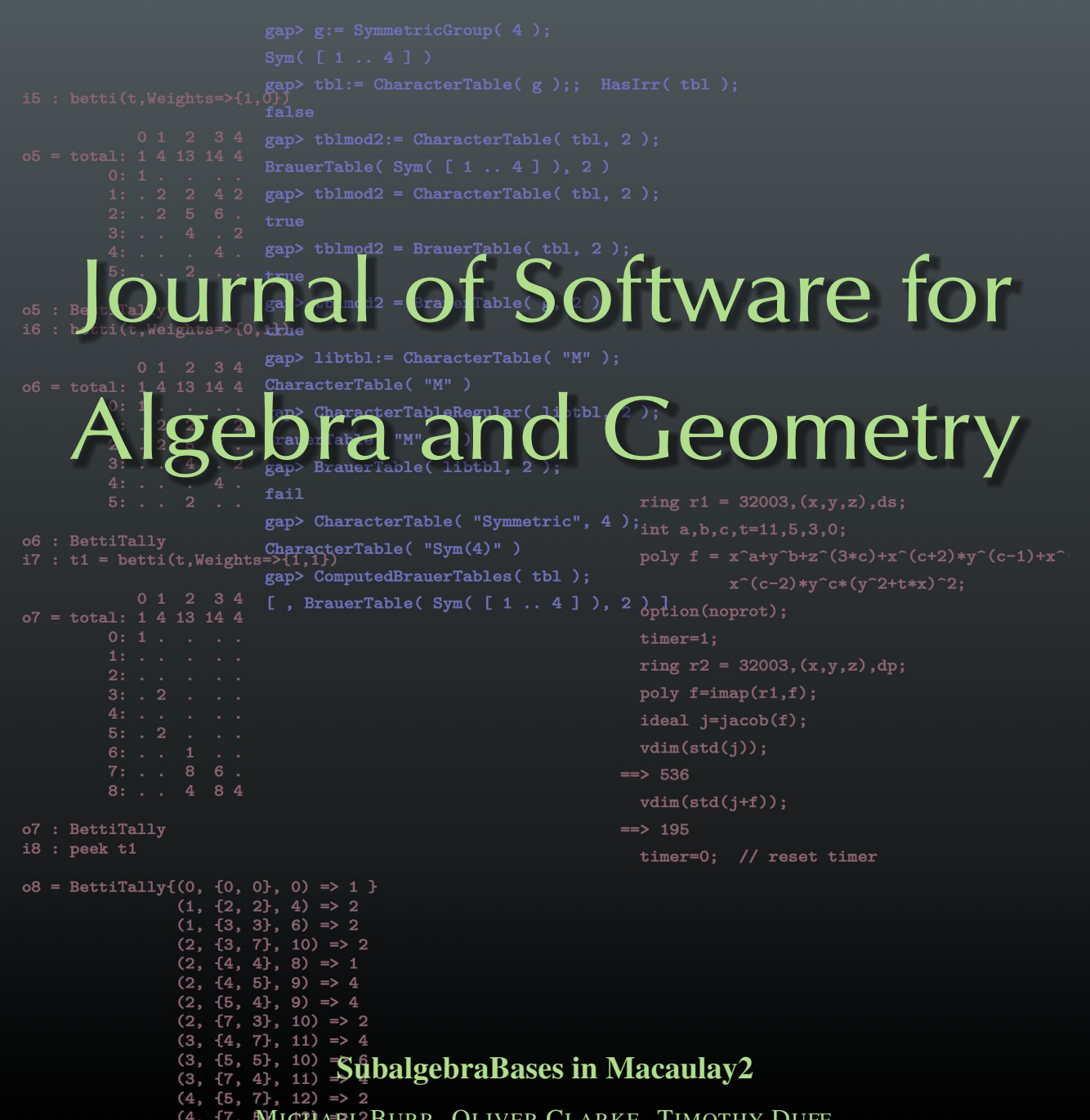
Journal of Software for Algebra and Geometry



SubalgebraBases in Macaulay2

Michael Burr, Oliver Clarke, Timothy Duff, Jackson Leaman, Nathan Nichols and Elise Walker

vol 14 2024

JSAG **14** (2024), 97–109
https://doi.org/10.2140/jsag.2024.14.97

The Journal of Software for
Algebra and Geometry

SubalgebraBases in Macaulay2

MICHAEL BURR, OLIVER CLARKE, TIMOTHY DUFF,
JACKSON LEAMAN, NATHAN NICHOLS AND ELISE WALKER

ABSTRACT: We describe a recently revived version of the software package `SubalgebraBases`, which is distributed in the *Macaulay*2 computer algebra system. The package allows the user to compute and manipulate subalgebra bases — which are also known as SAGBI bases, or canonical bases, and form a special class of Khovanskii bases — for polynomial rings and their quotients. We provide an overview of the design and functionality of the `SubalgebraBases` package and demonstrate how the package works on several motivating examples.

1. INTRODUCTION. In the field of computational algebraic geometry, the notion of a Gröbner basis plays an instrumental role in solving many basic problems involving polynomial ideals. For computations involving polynomial *algebras*, there is an analogous, but more subtle, notion of a *subalgebra basis*. This notion was introduced independently by Kapur and Madlener [19] and Robbiano and Sweedler [25], where the names *canonical basis* and *SAGBI basis* were also proposed. In more recent years, the name *Khovanskii basis* has also been adopted [20] to describe a more general notion for valued algebras.

The `SubalgebraBases` package in *Macaulay2* was initially developed around 1997 by Stillman and Tsai [16]. In subsequent years, this package was inaccessible to most users due to internal changes in *Macaulay2*, until version 1.18 was released in mid-2021. Our work before and after this rerelease has focused on restoring the package's original functionality, implementing additional algorithms, and designing new data structures that facilitate working with larger examples and admit new features in the package.

Other software packages exist for computing subalgebra bases. In recent work of Bruns and Conca [6], they develop new methods for computing subalgebra bases and implement them in Singular [13]. They also compare the performance of their package to existing software packages for computing subalgebra bases. In particular, they compare their package to a previous version of our `SubalgebraBases` package in Macaulay2, a preexisting package in Singular [13], and the package [3] in an upcoming release

Burr was partially supported by NSF grant DMS-1913119 and SIMONS collaboration grant #964285.
The work was started when Oliver Clarke was at the University of Bristol supported by the EPSRC Doctoral Training Partnership award EP/N509619/1 and continued while being an overseas researcher under a postdoctoral fellowship of the Japan Society for the Promotion of Science.
Duff acknowledges support from an NSF Mathematical Sciences Postdoctoral Research Fellowship (DMS-2103310).
MSC2020: 14-04, 68W30.
Keywords: SAGBI basis, Newton–Okounkov body, computer algebra, Macaulay2, symbolic computation.
`SubalgebraBases` version 1.3

of CoCoA [1]. Certain computations may be faster, slower, or infeasible with different software packages. As far as we are aware, the `SubalgebraBases` package is currently the only package that works with quotients of polynomial rings.

Outline. In Section 2, we provide some background on subalgebra bases, with examples illustrating the basic usage of our package. In Section 3A, we introduce the main data structures: the `Subring` and `SAGBIBasis` types, which allow for resuming partial subalgebra basis computations, and describe the various methods that they enable. Section 3B summarizes several options that may be useful for computing subalgebra bases, and Section 3C covers some additional functionality. Finally, in Section 4, we provide some more sophisticated examples of our package in action. The examples appearing in this paper may be found in the file `accompanyingCode.m2`.

2. BACKGROUND AND BASIC COMPUTATIONS. For a field k, let $R := k[x_1, \ldots, x_n]$ be a polynomial ring, which we equip with a monomial order $<$. Unless otherwise specified, we follow the *Macaulay2* convention for variable ordering, in which $x_n < \cdots < x_1$. Consider a subalgebra $A := k[f_i \mid i \in I] \subset R$, where I is some index set. The *initial algebra* $\mathrm{in}_<(A)$ is generated by the initial terms of all elements in A:

$$\mathrm{in}_<(A) = k[\mathrm{in}_<(f) \mid f \in A].$$

Definition 2.1. A set $\{g_j\}_{j\in J} \subseteq A$ is a *subalgebra basis* for A, with respect to the order $<$, if

$$\mathrm{in}_<(A) = k\big[\{\mathrm{in}_<(g_j)\}_{j\in J}\big].$$

Just as Gröbner bases enable many computations with polynomial ideals, the knowledge of a finite subalgebra basis allows us to answer several basic questions about a given algebra. A basic application is *algebra membership*: if $g_1, \ldots, g_s \in A$ form a subalgebra basis for A, then any polynomial $f \in R$ has an associated *normal form* $r \in R$: for some polynomial $q \in k[y_1, \ldots, y_s]$,

$$f = q(g_1, \ldots, g_s) + r, \tag{1}$$

with the property that either none of the monomials of r are contained in $\mathrm{in}_<(A)$, which implies r is unique. The polynomials q and normal form r appearing in (1) can be computed using an analogue of multivariate polynomial division known as the *subduction algorithm*. A description of the subduction algorithm may be found in [25, Algorithm 1.5] or [27, Algorithm 11.1], for example.

Unlike Gröbner bases, the theory of subalgebras bases is not, strictly speaking, algorithmic. For instance, as illustrated in Example 2.3, a finitely generated polynomial algebra need not have a finite subalgebra basis with respect to a given term order. Nevertheless, there are many interesting examples for which finite subalgebra bases exist and can be computed. We consider a classical and well-known example from invariant theory.

Example 2.2. Let $R = \mathbb{Q}[x_1, x_2, x_3]$ and consider the first three power-sums in R,

$$f_1 = \underline{x_1} + x_2 + x_3, \quad f_2 = \underline{x_1^2} + x_2^2 + x_3^2, \quad f_3 = \underline{x_1^3} + x_2^3 + x_3^3.$$

The underlined monomials above are the initial terms with respect to the GRevLex order on R. The algebra $A = \mathbb{Q}[f_1, f_2, f_3]$ is the ring of invariants for the standard action of the symmetric group $S_3 \curvearrowright R$, induced by $\sigma \cdot x_i = x_{\sigma(i)}$ for each i. A subalgebra basis for A consists of the three elementary symmetric polynomials,

$$g_1 = \underline{x_1} + x_2 + x_3, \quad g_2 = \underline{x_1x_2} + x_1x_3 + x_2x_3, \quad g_3 = \underline{x_1x_2x_3},$$

hence $\mathrm{in}_<(A) = \mathbb{Q}[x_1, x_1x_2, x_1x_2x_3]$. We verify these assertions in *Macaulay2* with the code below.

```
needsPackage "SubalgebraBases";
R = QQ[x_1..x_3];
A = subring {x_1+x_2+x_3, x_1^2+x_2^2+x_3^2, x_1^3+x_2^3+x_3^3};
SB = sagbi A
gens SB
isSAGBI SB
```

The three lines without semicolons produce the following output, informing us that SB is an instance of the type SAGBIBasis, and that our computation terminated successfully:

```
i4 : SB = sagbi A
o4 = SAGBIBasis Computation Object with 3 generators, Limit = 20.
o4 : SAGBIBasis
i5 : gens SB
o5 = | x_1+x_2+x_3 x_1x_2+x_1x_3+x_2x_3 x_1x_2x_3 |
                1       3
o5 : Matrix R  <-- R
i6 : isSAGBI SB
o6 = true
```

To verify $f = x_1^4 + x_2^4 + x_3^4 \in A$, we may compute q and r appearing in (1) as follows:

```
i5 : A = subring(gens SB, GeneratorSymbol => g);
i6 : f = x_1^4 + x_2^4 + x_3^4;
i7 : q = f // A
                4      2 2    2      2
o7 = g_1^4 - 4g_1^2g_2 + 2g_2  + 4g_1g_3
o7 : QQ[g_1..g_3]
i8 : r = f % A
o8 = 0
o8 : R
```

For a given set of algebra generators $\{f_i\}_{i\in I}$, the *binomial syzygies* on the set of initial terms $\{\mathrm{in}_<(f_i)\}_{i\in I}$, also known as *tête-a-têtes* [25], provide an analogue of S-polynomials from Gröbner basis theory. The binomial syzygies $y^\alpha - y^\beta$ generate the kernel of the monomial map

$$\varphi_{\mathrm{in}_<(f)} : k[y_i \mid i \in I] \mapsto k[\mathrm{in}_<(f_i) \mid i \in I], \quad y_i \mapsto \mathrm{in}_<(f_i).$$

If we consider a presentation of the algebra A given by

$$\varphi_f : k[y_i \mid i \in I] \mapsto A, \quad y_i \mapsto f_i,$$

then the initial term of the "S-polynomial" $S_{\alpha,\beta} = \varphi_f(y^\alpha - y^\beta) \in A$ is strictly less than $\text{in}_<\big(\varphi_f(y^\alpha)\big) = \text{in}_<\big(\varphi_f(y^\beta)\big)$. Applying the subduction algorithm, we obtain a normal form $r_{\alpha,\beta} \in A$ for $S_{\alpha,\beta}$ analogous to r in (1). If every $r_{\alpha,\beta}$ is zero, then the set $\{f_i\}_{i\in I}$ forms a subalgebra basis. Otherwise, we enlarge the set of generators to include all nonzero $r_{\alpha,\beta}$ and restart the computation.

Example 2.3. Consider the algebra $A = \mathbb{Q}[x_1 + x_2, x_1x_2, x_1x_2^2] \subseteq \mathbb{Q}[x_1, x_2]$, and let $<$ be the `GRevLex` order. The initial algebra $\text{in}_<(A) = \mathbb{Q}[x_1, x_1x_2, x_1x_2^2, x_1x_2^3, \dots]$ is not finitely generated. However, we can still compute a partial subalgebra basis in *Macaulay2* as follows:

```
i3 : R = QQ[x_1,x_2];
i4 : SB = subalgebraBasis({x_1+x_2, x_1*x_2, x_1*x_2^2}, Limit => 7)
o4 = | x_1+x_2 x_1x_2 x_1x_2^2 x_1x_2^3 x_1x_2^4 x_1x_2^5 x_1x_2^6 |
                1       15
o4 : Matrix R  <-- R
```

The option `Limit` (whose default value is 20) allows the procedure outlined above to terminate. The final S-polynomial computed is

$$S_{\alpha,\beta} = (x_1 + x_2)(x_1x_2^5) - (x_1x_2^2)(x_1x_2^4) = x_1x_2^6,$$

of degree 7. Using `isSAGBI`, we can check that the computation is incomplete.

```
i3 : isSAGBI SB
o3 = false
```

Remark 2.4. It is important to note that setting the `Limit` option does not guarantee that all subalgebra basis generators up to that degree have been found. During the subalgebra basis computation, it is possible for low-degree subalgebra basis generators to arise from high-degree S-polynomials. For example, consider the following algebra with respect to the `Lex` term order with $x > y$:

$$S = \mathbb{Q}[x^2{+}y,\ x^2y,\ x^2y^2,\ x^2y^{10}{+}x] \subseteq \mathbb{Q}[x, y].$$

We observe that, for each $i \geq 3$, there is a degree $i{+}3$ S-polynomial which shows that $x^2y^i \in S$. For instance, we have $x^2y^3 = (x^2{+}y)(x^2y^2) - (x^2y)^2$, and, by using x^2y^3, we have $x^2y^4 = (x^2{+}y)(x^2y^3) - (x^2y)(x^2y^2)$, and so on. In particular, we have $x^2y^{10} \in S$. However, the lowest degree S-polynomial which shows that $x \in S$ is degree 12 and is given by $x = (x^2y^{10} + x) - (x^2y^{10})$. So, if we compute a subalgebra basis in *Macaulay*2 with the `Limit` option set to any value less than 12, then the result will not include x among the generators.

Various extensions of Definition 2.1 may be found in the literature. For instance, in [21], a Noetherian integral domain with identity replaces the coefficient field k. Building upon this, the authors of [26] extend the theory of subalgebra bases to quotients of these polynomial rings. Currently, our package can be used to compute a (partial) subalgebra basis for a finitely-generated subalgebra $A \subseteq R/I$, where I is a polynomial ideal. In this setting, the initial algebra $\text{in}_<(A)$ is defined to be a subalgebra of $R/\,\text{in}_<(I)$; see, for example, [20; 26]. We provide an example illustrating this functionality below.

Example 2.5 [26, Example 2]. Fix a positive integer n. If we let

$$S = \mathbb{Q}[a, b, c, d, u_1, \ldots, u_n, v_1, \ldots, v_n]/(ad - bc - 1),$$

and $A \subseteq S$ be the subalgebra generated by $G = \{au_i + bv_i, cu_i + dv_i\}_{1 \le i \le n} \subseteq Q$, then

$$G \cup \{u_i v_j - u_j v_i\}_{1 \le i < j \le n}$$

forms a subalgebra basis for A with respect to `Lex`. We verify this below when $n = 3$.

```
i1 : n = 3;
i2 : R = QQ[a,b,c,d,u_1..u_n, v_1..v_n, MonomialOrder => Lex];
i3 : S = R / ideal(a*d - b*c - 1);
i4 : G = flatten for i from 1 to n list {a*u_i + b*v_i, c*u_i + d*v_i};
i5 : SB = subalgebraBasis G
o5 = | cu_3+dv_3 cu_2+dv_2 cu_1+dv_1 au_3+bv_3 au_2+bv_2 au_1+bv_1 u_2v_3-u_3v_2
                                                              u_1v_3-u_3v_1 u_1v_2-u_2v_1 |
                1       9
o5 : Matrix S <-- S
i6 : isSAGBI SB
o6 = true
```

3. Design and functionality.

3A. ***Data structures and resuming computations.*** We provide two main data structures for working with subalgebras and subalgebra bases: the `Subring` and `SAGBIBasis` types. An instance of the type `Subring`, which we call a *subring*, represents a subalgebra A of either a polynomial ring or a quotient ring. Subrings are designed to behave similarly to the core data type `Ideal`. The other main type `SAGBIBasis` captures the state of a subalgebra basis computation, making it similar to a `GroebnerBasis` computation object. In particular, this type is designed to keep track of the already-computed elements of the subalgebra basis, and we refer to an instance of the type `SAGBIBasis` as a *computation object*.

A subring `A` is a light-weight object that keeps track of the generators of the algebra A, and may be used as an argument for the main functions that compute subalgebra bases; `subalgebraBasis` and `sagbi`. A subring also keeps track of the furthest-advanced computation object that has been constructed during a subalgebra basis computation. To access or create a computation object, we use the function `sagbi`. The generators associated with a computation object are the currently-known elements of a (partial) subalgebra basis. Applying the method function `isSAGBI` to a subring checks whether its generators are a subalgebra basis, exploiting any previous computations. On the other hand, applying `isSAGBI` to a computation object will check if the currently-known (partial) subalgebra basis is a complete subalgebra basis for the original algebra.

Algebra generators associated with an instance of `Subring` or `SAGBIBasis` can be recovered using `gens`. For example, if `S` is a subring and `isSAGBI sagbi S` returns `true`, then a complete subalgebra basis for `S` has been computed. In this case, the subalgebra basis may be accessed with `gens sagbi S`, or equivalently `subalgebraBasis S`. Note that this is different to performing `isSAGBI S`, which, if `true`, means that `gens S` is a subalgebra basis.

Example 3.1. Let $A = \mathbb{Q}[x+y, x^6, y^6] \subseteq \mathbb{Q}[x, y]$. The following code computes a partial GRevLex subalgebra basis:

```
i1 : R = QQ[x,y];
i2 : A = subring {x+y, x^6, y^6}
o2 = subring of R with 3 generators
o2 : Subring
i3 : SB = sagbi(A, Limit => 5)
o3 = Partial SAGBIBasis Computation Object with 1 generators, Limit = 5.
o3 : SAGBIBasis
i4 : isSAGBI SB
o4 = false
```

The computation object SB above records the state of a subalgebra basis computation accounting for binomial syzygies of degree up to 5. This computation does not compute the S-polynomial $(x+y)^6-(x^6)$, whose initial term is needed to generate the initial algebra. Since the subring A has recorded the progress of the last computation, we may resume computation with a higher value for Limit.

```
i5 : SB' = sagbi(A, Limit => 100)
o5 = SAGBIBasis Computation Object with 3 generators, Limit = 100.
o5 : SAGBIBasis
i6 : isSAGBI SB'
o6 = true
i7 : gens SB'
o7 = | x+y y6 x5y+5/2x4y2+10/3x3y3+5/2x2y4+xy5 |
                 1       3
o7 : Matrix R <-- R
```

Remark 3.2. Subring and SAGBIBasis are both immutable types. However, a Subring keeps track of the progress of computations by caching partial results. Both Subring and SAGBIBasis have generators and ambientRing values, which can be accessed by their respective methods. However, the SAGBIBasis type has many additional keys reserved for internal use. One key of particular interest is SAGBIdata, which stores user-readable information such as the original generators of the subring, the partially computed subalgebra basis, and whether the subalgebra basis computation is complete.

3B. ***Computation options.*** When calling sagbi or subalgebraBasis, several options for fine-tuning computations may be used. These options typically carry over when resuming computations. Two exceptions are the options Limit, which is always taken to be the specified or default value, and PrintLevel, which is described below and controls the verbosity of the computation. However, by using the option RenewOptions, the options may be modified. The option Recompute is used to completely restart a subalgebra basis computation. We catalog several other options below, which may be useful in various settings:

PrintLevel: This option takes a nonnegative integer and controls the verbosity of the function. For successively higher values, the function will print more data related to the computation.

`SubductionMethod`: This option controls whether subduction is performed by top-level code `"Top"` or by engine-level compiled code `"Engine"`. The engine-level code can be faster for computations that require many subduction steps. One advantage of `"Top"` is that one can view intermediate results (e.g., from subduction) if a high enough `PrintLevel` is used.

`Strategy`: This option controls how the computation object is modified after new generators are added to the subalgebra basis. There are two primary strategies: `"DegreeByDegree"` and `"Incremental"`. The strategy `"DegreeByDegree"` computes a partial Gröbner basis for the *reduction ideal*, whereas the strategy `"Incremental"` computes a full Gröbner basis. However, the strategy `"DegreeByDegree"` computes the partial Gröbner basis from scratch, whereas the strategy `"Incremental"` makes use of the previously computed full Gröbner basis to speed up the computation of the next full Gröbner basis. Both strategies have strengths and weaknesses: `"DegreeByDegree"` is suited to computations where a large number of new generators are added at a particular degree, and `"Incremental"` is well-suited to computations where very few generators are added at each degree. The default option is the strategy `"Master"` which heuristically blends between the `"DegreeByDegree"` and `"Incremental"` strategies in order to gain performance benefits from each method.

`AutoSubduceOnPartialCompletion`: This option controls whether the subalgebra basis generators are subducted against each other. More precisely, if no new subalgebra basis generators are found at a particular degree, then the current subalgebra basis generators are subducted against each other. This produces a reduced set of generators. We intend for this option to be used when the supplied generators are suspected to be a subalgebra basis; reducing the number of generators speeds up the computation of the subsequent binomial syzygies.

3C. ***Other functionality.*** The function `subduction` takes a collection of polynomials $g_1, \ldots, g_s$ and subducts a given polynomial $f \in R$ (or matrix of polynomials) against them. When $g_1, \ldots, g_s$ are encoded using a subring `A` or computation object, then $g_1, \ldots, g_s$ are taken to be the generators of `S` or (partial) subalgebra basis generators of `A`, respectively.

Example 3.3. Fix the polynomial ring $\mathbb{Q}[x, y]$, and let $G = \{x^2 + x, y^2 + y\}$. We subduct the polynomial $f = x^2y^2 + x^3y$ against G with respect to `GRevLex` as follows:

```
i1 : R = QQ[x,y];
i2 : G = {x^2 + x, y^2 + 1};
i3 : subduction(G, x^2*y^2 + x^3*y)
      3      2
o3 = x y - x*y
o3 : R
```

As in Example 2.2, we may compute normal forms using the operator `%`. Analogous to its use with ideals and Gröbner bases, the syntax `f%S` works for `S` of class either `Subring` or `SAGBIBasis`. If no complete subalgebra basis for `S` of class `Subring` is known, then `f%S` falls back on an *extrinsic method* using Gröbner bases to compute the normal form. On the other hand, if either a complete

subalgebra basis has been computed for S or S is of class SAGBIBasis, then f % S subducts f against the (partial) subalgebra basis associated with S.

Example 3.4. Consider the subalgebra $A = \mathbb{C}[x+y, xy, xy^2]$ in Example 2.3. Let

$$f = xy^3 + xy^4 + xy^5 + xy^6 \in A.$$

We illustrate the different behaviors of % in the code below.

```
i1 : R = QQ[x,y];
i2 : A = subring {x+y, x*y, x*y^2};
i3 : SB = sagbi(A, Limit => 5);
i4 : f = x*y^3 + x*y^4 + x*y^5 + x*y^6;
i5 : f % A
o5 = 0
o5 : R
i6 : f % SB
          6     5
o6 = x*y + x*y
o6 : R
i7 : SB = sagbi(A, Limit => 7);
i8 : f % SB
o8 = 0
o8 : R
```

The generators associated with SB, namely

$$\{x+y, xy, xy^2, xy^3, xy^4, xy^5, xy^6\},$$

form a partial subalgebra basis for A. To subduct f against these generators, we may use f % SB. However, the generators of SB do not form a complete subalgebra basis for A, so the operation f % A falls back on the extrinsic method.

The function groebnerMembershipTest is an extrinsic membership test for elements of a subring. It can potentially be used in cases where a subring has an intractable subalgebra basis. If a sufficiently large partial subalgebra basis has already been computed, then it is recommended to use the operator % or the function subduction.

Example 3.5. Following from Example 3.4, we test the membership of f in A as follows:

```
i9 : groebnerMembershipTest(f, A)
o9 = true
```

The function intersect computes the intersection of two subrings using an analogous method to that of intersecting ideals via Gröbner bases [10]. The output of the function is an instance of IntersectedSubring, which is a type that inherits from Subring. To check whether the output is guaranteed to be equal to the full intersection of the subrings, we use the function isFullIntersection. If the function isFullIntersection returns true then the generators of the intersected subring are a subalgebra basis for the intersection.

Example 3.6. Consider the subrings $A_1 = \mathbb{Q}[x^2, xy]$ and $A_2 = \mathbb{Q}[x, y^2]$ of the quotient ring $S = \mathbb{Q}[x, y]/\langle x^3 + xy^2 + y^3\rangle$. We compute the intersection $A_1 \cap A_2$ as follows:

```
i1 : R = QQ[x,y];
i2 : I = ideal(x^3 + x*y^2 + y^3);
i3 : S = R/I;
i4 : A1 = subring {x^2, x*y};
i5 : A2 = subring {x, y^2};
i6 : A = intersect(A1, A2)
o6 = QQ[p_0..p_5], subring of S
o6 : IntersectedSubring
i7 : gens A
o7 = | x2 x2y2+xy3 y4 xy3 y6 xy5 |
             1      6
o7 : Matrix S  <-- S
i8 : isFullIntersection A
o8 = true
```

Since `isFullIntersection A` returns `true`, the generators of the intersected subring `A` are in fact guaranteed to generate the entire ring $A_1 \cap A_2$. Moreover the generators are a subalgebra basis for the intersection with respect to the monomial order `GRevLex`.

If `isFullIntersection A` returns `false` for an intersected subring `A`, then the generators of `A` may or may not generate the entire intersection. When this occurs, we suggest experimenting with options of `intersect` such as `Limit` or `SAGBILimitType`.

4. EXAMPLES. Subalgebra bases have a number of applications. A short sample of the literature, including several recent papers which use the `SubalgebraBases` package, follows: [2; 4; 5; 11; 12; 15; 17; 18; 22; 26; 28].

Example 4.1. Following [11, Section 2], we describe the adjoint action of SO(3) on its Lie algebra and its ring of invariants. We write

$$(R, \boldsymbol{t}) \in \mathrm{SO}(3) \ltimes \mathbb{R}^3 = \mathrm{SE}(3)$$

for an element of the Special Euclidean group of rigid motions of $\mathbb{R}^3$. To describe the adjoint action of SE(3) on its Lie algebra $\mathfrak{se}(3)$, it is convenient to define, for each $\boldsymbol{t} = (t_1, t_2, t_3)^T \in \mathbb{R}^3$, a skew-symmetric matrix

$$[\boldsymbol{t}]_\times = \begin{pmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{pmatrix}.$$

We identify elements of $\mathfrak{se}(3)$ with their Plücker coordinates $(\boldsymbol{w}, \boldsymbol{v}) \in \mathfrak{se}(3)$. The adjoint action of SE(3) on $\mathfrak{se}(3)$ is given by

$$(R, \boldsymbol{t}) \cdot \begin{pmatrix} \boldsymbol{w} \\ \boldsymbol{v} \end{pmatrix} = \begin{pmatrix} R & 0 \\ [\boldsymbol{t}]_\times R & R \end{pmatrix} \begin{pmatrix} \boldsymbol{w} \\ \boldsymbol{v} \end{pmatrix} = \begin{pmatrix} R\boldsymbol{w} \\ [\boldsymbol{t}]_\times R\boldsymbol{w} + R\boldsymbol{v} \end{pmatrix}. \tag{2}$$

Consider the polynomial algebra $A \subseteq \mathbb{Q}[t_i, w_i, v_i]_{i\in\{1,2,3\}}$ generated by the entries of the matrix in (2) with the rotation R set to be the 3×3 identity matrix. Explicitly, the subring is given by

$$A = \mathbb{Q}[w_1, w_2, w_3, -t_3w_2+t_2w_3+v_1, t_3w_1-t_1w_3+v_2, -t_2w_1+t_1w_2+v_3].$$

The ring of invariants of $\mathfrak{se}(3)$ under the action of the translational subgroup $\mathbb{R}^3 \triangleleft \mathrm{SE}(3)$ is given by the intersection $A \cap \mathbb{Q}[w_i, v_i]_{i\in\{1,2,3\}}$. We compute this intersection using a monomial order that eliminates the variables t_1, t_2, t_3.

```
i1 : R = QQ[t_1..t_3, w_1..w_3, v_1..v_3, MonomialOrder =>{Eliminate 3, Lex}];
i2 : SB = sagbi {w_1, w_2, w_3, -t_3*w_2+t_2*w_3+v_1, t_3*w_1-t_1*w_3+v_2,
                                                          -t_2*w_1+t_1*w_2+v_3};
i3 : isSAGBI SB
o3 = true
i4 : SB' = selectInSubring (1, gens SB)
o4 = | w_3 w_2 w_1 w_1v_1+w_2v_2+w_3v_3 |
             1       4
o4 : Matrix R  <-- R
```

It follows from this computation that the algebra of translational invariants is simply $\mathbb{Q}[w_1, w_2, w_3, \boldsymbol{w}\cdot\boldsymbol{v}]$, which confirms the computation in [11, Section 5.1]. The action above naturally extends to an action on $(\boldsymbol{w}_1, \boldsymbol{v}_1, \ldots, \boldsymbol{w}_n, \boldsymbol{v}_n) \in \mathfrak{se}(3)^n$, called a multiscrew. In the file `accompanyingCode.m2`, calling `screwsExample n` will attempt to compute a subalgebra basis for the translational invariants for any number n. In particular, it is straightforward to verify the computation when $n = 2$, which is used in [11, Section 5.2] as a step in computing full invariant ring.

Besides invariant-theoretic applications such as Examples 2.2 and 4.1, computing subalgebra bases is also a key operation for constructing *toric degenerations*. A toric degeneration is a particular type of flat family whose generic fibers are some variety of interest and whose special fiber is a toric variety. Various properties of the variety of interest that are preserved under flat limits (e.g., dimension and degree) can then be studied via toric degenerations by passing to the toric fiber. Newton–Okounkov bodies are a closely-related construction, whose applications include counting the number of solutions to classes of polynomial systems; see, e.g., [5; 7; 14; 24]. Using subalgebra bases, toric degenerations have been constructed for many families of varieties, including Grassmannians, flag varieties, and Cox–Nagata rings [4; 9; 23; 28]. We conclude with two representative examples.

Example 4.2. Consider the matrix

$$A = \begin{bmatrix} 1 & 0 & 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & -1 \end{bmatrix},$$

and let $G = \ker(A)$. The columns of A are six points in $\mathbb{P}^2$, which are the points of intersection of four generic lines. Following [28, Example 2.6], the Cox–Nagata ring is given by

$$R^G = \mathbb{Q}[x_1, \ldots, x_6, L_{124}, L_{135}, L_{236}, L_{456}, M_{16}, M_{25}, M_{34}] \subseteq \mathbb{Q}[x_1, \ldots, x_6, y_1, \ldots, y_6],$$

where the polynomials $L_\bullet$ and $M_\bullet$ are described below. We verify that the generators indeed form a subalgebra basis as follows:

```
i1  : R = QQ[x_1..x_6, y_1..y_6];
i2  : L124 = y_3*x_5*x_6 + x_3*y_5*x_6 - x_3*x_5*y_6;
i3  : L135 = y_2*x_4*x_6 - x_2*y_4*x_6 + x_2*x_4*y_6;
i4  : L236 = y_1*x_4*x_5 + x_1*y_4*x_5 - x_1*x_4*y_5;
i5  : L456 = y_1*x_2*x_3 + x_1*y_2*x_3 + x_1*x_2*y_3;
i6  : M16 = y_2*x_3*x_4*x_5 + x_2*y_3*x_4*x_5 - x_2*x_3*y_4*x_5 + x_2*x_3*x_4*y_5;
i7  : M25 = y_1*x_3*x_4*x_6 + x_1*y_3*x_4*x_6 + x_1*x_3*y_4*x_6 - x_1*x_3*x_4*y_6;
i8  : M34 = y_1*x_2*x_5*x_6 + x_1*y_2*x_5*x_6 - x_1*x_2*y_5*x_6 + x_1*x_2*x_5*y_6;
i9  : RG = subring {x_1..x_6, L124, L135, L236, L456, M16, M25, M34};
i10 : isSAGBI RG
o10 = true
i11 : leadTerm gens RG
o11 = | x_1 x_2 x_3 x_4 x_5 x_6 x_5x_6y_3 x_4x_6y_2 x_4x_5y_1 x_2x_3y_1 x_3x_4x_5y_2
                                                                x_3x_4x_6y_1 x_2x_5x_6y_1 |
```

The special fiber of the toric degeneration from this construction is the algebra generated by the above monomials.

Example 4.3. Let $X = (x_{i,j})$ be a 3×6 matrix of variables and $R = \mathbb{Q}[x_{i,j}]$ be a ring in those variables. Under the Plücker embedding, the coordinate ring of the Grassmannian $\mathrm{Gr}(3,6)$ is generated by the maximal minors of X. The maximal minors do not form a subalgebra basis with respect with respect to a weight vector associated to a *hexagonal matching field* [8; 9; 23].

```
i1 : R = QQ[x_(1,1)..x_(3,6), MonomialOrder => {Weights => {0,0,0,0,0,0,
                                                            0,15,3,12,9,6,0,7,14,21,28,35}}];
i2 : X = transpose genericMatrix (R, 6, 3);
                3      6
o2 : Matrix R  <-- R
i3 : A = subring for s in subsets (6, 3) list det X_s;
i4 : isSAGBI A
o4 = false
i5 : SB = sagbi(A, Limit => 100, SubductionMethod => "Engine")
o5 = Partial SAGBIBasis Computation Object with 21 generators, Limit = 100.  o5 : SAGBIBasis
i6 : isSAGBI SB
o6 = true
```

Since `isSAGBI A` is `false`, the 20 maximal minors that generate `A` do not form a subalgebra basis, and a twenty-first generator is required for a complete subalgebra basis.

ACKNOWLEDGEMENTS. We wish to thank the organizers of two virtual *Macaulay2* workshops held in 2020 through Cleveland State University and the University of Warwick. These workshops provided first introductions for several of the authors. We also thank several working group participants for their help during early stages of redeveloping this package. We would like to thank Aldo Conca for pointing

out some bugs in the previous versions of our package, and the referees for their suggestions which greatly improved both this paper and the associated software package.

This article has been coauthored by an employee of National Technology & Engineering Solutions of Sandia, LLC under contract no. DE-NA0003525 with the U.S. Department of Energy (DOE). The employee co-owns right, title and interest in and to the article and is responsible for its contents. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a nonexclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this article or allow others to do so, for United States Government purposes. The DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan .

SUPPLEMENT. The online supplement contains version 1.3 of `SubalgebraBases`.

REFERENCES.

[1] J. Abbott, A. M. Bigatti, and L. Robbiano, "CoCoA, a system for doing computations in commutative algebra", available at http://cocoa.dima.unige.it.

[2] M. Belotti and M. Panizzut, "Discrete geometry of Cox rings of blow-ups of $\mathbb{P}^3$", preprint, 2022. arXiv 2208.05258

[3] A. M. Bigatti and L. Robbiano, "Saturations of subalgebras, SAGBI bases, and U-invariants", available at https://www.dima.unige.it/~bigatti/data/ComputingSaturationsOfSubalgebras/. Zbl

[4] L. Bossinger, S. Lamboglia, K. Mincheva, and F. Mohammadi, "Computing toric degenerations of flag varieties", pp. 247–281 in *Combinatorial algebraic geometry*, edited by G. G. Smith and B. Sturmfels, Fields Inst. Commun. **80**, Fields Inst. Res. Math. Sci., Toronto, ON, 2017. MR Zbl

[5] P. Breiding, M. Michałek, L. Monin, and S. Telen, "The algebraic degree of coupled oscillators", preprint, 2022. arXiv 2208.08179

[6] W. Bruns and A. Conca, "SAGBI combinatorics of maximal minors and a SAGBI algorithm", *J. Symbolic Comput.* **120** (2024), art. id. 102237. MR Zbl

[7] M. Burr, F. Sottile, and E. Walker, "Numerical homotopies from Khovanskii bases", *Math. Comp.* **92**:343 (2023), 2333–2353. MR Zbl

[8] O. Clarke, "Matching fields in Macaulay2", preprint, 2023. Zbl arXiv 2306.09693

[9] O. Clarke, F. Mohammadi, and F. Zaffalon, "Toric degenerations of partial flag varieties and combinatorial mutations of matching field polytopes", *J. Algebra* **638** (2024), 90–128. MR Zbl

[10] 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, Cham, 2015. MR Zbl

[11] D. Crook and P. Donelan, "Polynomial invariants and SAGBI bases for multi-screws", preprint, 2020. Zbl arXiv 2001.05417v2

[12] J. P. Dalbec, "Straightening Euclidean invariants", *Ann. Math. Artificial Intelligence* **13**:1–2 (1995), 97–108. MR Zbl

[13] W. Decker, G.-M. Greuel, G. Pfister, and H. Schönemann, "SINGULAR 4-3-0, a computer algebra system for polynomial computations", available at http://www.singular.uni-kl.de.

[14] T. Duff, N. Hein, and F. Sottile, "Certification for polynomial systems via square subsystems", *J. Symbolic Comput.* **109** (2022), 367–385. MR Zbl

[15] M. Göbel, "Visualizing properties of comprehensive SAGBI bases — two examples", *Appl. Algebra Engrg. Comm. Comput.* **12**:5 (2001), 429–435. MR Zbl

[16] D. R. Grayson and M. E. Stillman, "Macaulay2, a software system for research in algebraic geometry", available at https://macaulay2.com/. Zbl

[17] M. Gross, P. Hacking, S. Keel, and M. Kontsevich, "Canonical bases for cluster algebras", *J. Amer. Math. Soc.* **31**:2 (2018), 497–608. MR Zbl

[18] B. Huber, F. Sottile, and B. Sturmfels, "Numerical Schubert calculus", *J. Symbolic Comput.* **26**:6 (1998), 767–788. MR Zbl

[19] D. Kapur and K. Madlener, "A completion procedure for computing a canonical basis for a k-subalgebra", pp. 1–11 in *Computers and mathematics* (Cambridge, MA, 1989), edited by E. Kaltofen and S. M. Watt, Springer, New York, 1989. MR Zbl

[20] K. Kaveh and C. Manon, "Khovanskii bases, higher rank valuations, and tropical geometry", *SIAM J. Appl. Algebra Geom.* **3**:2 (2019), 292–336. MR Zbl

[21] J. L. Miller, "Analogs of Gröbner bases in polynomial rings over a ring", *J. Symbolic Comput.* **21**:2 (1996), 139–153. MR Zbl

[22] P. Misra and S. Sullivant, "Gaussian graphical models with toric vanishing ideals", *Ann. Inst. Statist. Math.* **73**:4 (2021), 757–785. MR Zbl

[23] F. Mohammadi and K. Shaw, "Toric degenerations of Grassmannians from matching fields", *Algebr. Comb.* **2**:6 (2019), 1109–1124. MR Zbl

[24] N. K. Obatake and E. Walker, "Newton–Okounkov bodies of chemical reaction systems", *Adv. in Appl. Math.* **155** (2024), art. id. 102672. MR Zbl

[25] L. Robbiano and M. Sweedler, "Subalgebra bases", pp. 61–87 in *Commutative algebra* (Salvador, Brazil, 1988), Lecture Notes in Mathematics **1430**, Springer, Heidelberg, 1990. Zbl

[26] M. Stillman and H. Tsai, "Using SAGBI bases to compute invariants: Effective methods in algebraic geometry", *J. Pure Appl. Algebra* **139**:1-3 (1999), 285–302. MR Zbl

[27] B. Sturmfels, *Gröbner bases and convex polytopes*, University Lecture Series **8**, American Mathematical Society, Providence, RI, 1996. MR Zbl

[28] B. Sturmfels and Z. Xu, "Sagbi bases of Cox–Nagata rings", *J. Eur. Math. Soc.* **12**:2 (2010), 429–459. MR Zbl

RECEIVED: 23 Feb 2023 REVISED: 12 Mar 2024 ACCEPTED: 18 Mar 2024

MICHAEL BURR:

burr2@clemson.edu
Clemson University, Clemson, SC, United States

OLIVER CLARKE:

oliver.clarke@ed.ac.uk
University of Edinburgh, Edinburgh, United Kingdom

TIMOTHY DUFF:

timduff@uw.edu
Department of Mathematics, University of Washington, Seattle, WA, United States

JACKSON LEAMAN:

jleaman@g.clemson.edu
Clemson University, Clemson, SC, United States

NATHAN NICHOLS:

nathannichols454@gmail.com
Department of Mathematics and Statistical Sciences, Marquette University, Milwaukee, WI, United States

ELISE WALKER:

eawalke@sandia.gov
Center for Computing Research, Sandia National Laboratories, Albuquerque, NM, United States