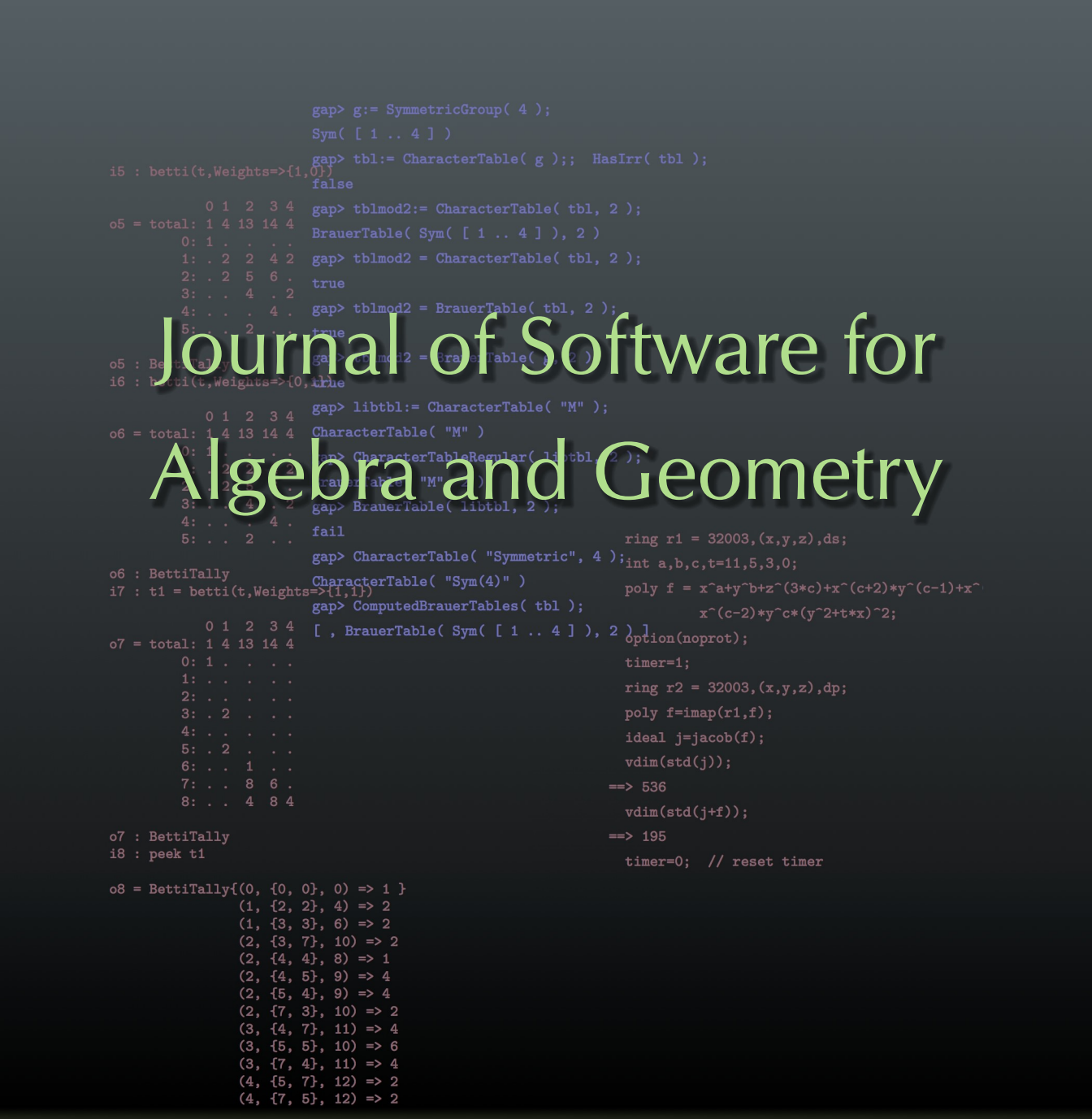

Journal of Software for Algebra and Geometry

vol 12 2022

JSAG **12** (2022), 1–10
https://doi.org/10.2140/jsag.2022.12.1

The Journal of Software for
Algebra and Geometry

Computing maximum likelihood estimates for Gaussian graphical models with Macaulay2

CARLOS AMÉNDOLA, LUIS DAVID GARCÍA PUENTE,
ROSER HOMS, OLGA KUZNETSOVA AND HARSHIT J. MOTWANI

ABSTRACT: We introduce the package *GraphicalModelsMLE* for computing the maximum likelihood estimates (MLEs) of a Gaussian graphical model in the computer algebra system *Macaulay*2. This package allows the computation of MLEs for the class of loopless mixed graphs. Additional functionality allows the user to explore the underlying algebraic structure of the model, such as its maximum likelihood degree and the ideal of score equations.

1. INTRODUCTION. The purpose of the package *GraphicalModelsMLE* is to extend the functionality of [Macaulay2] related to algebraic statistics, specifically allowing computations of maximum likelihood estimates of Gaussian graphical models. While *GraphicalModels* is an existing package that already provides useful information such as conditional independence ideals and vanishing ideals for such models, the fundamental statistical inference task of computing maximum likelihood estimates is missing. This package aims to fill this void and also extend the functionality of *GraphicalModels* to handle more general types of graphs, in particular, *loopless mixed graphs* (LMG). This class of graphs was introduced in [Sadeghi and Lauritzen 2014] in order to unify the Markov theory of several classical types of graphs such as undirected graphs, directed acyclic graphs, summary graphs and ancestral graphs [Lauritzen 1996].

The algebraic framework of *Macaulay*2 permits us to use both commutative algebra and numerical algebraic geometry to obtain a guaranteed global optimal solution by computing all critical points of the log-likelihood function. This is different from the classical statistical approach of the *R* package *ggm* [Marchetti 2006], and more in line with the recent numerical algebraic geometry approach from the package *LinearCovarianceModels.jl* in *Julia* [Sturmfels et al. 2020]. The package *GraphicalModelsMLE* is a complement to these two, handling some Gaussian graphical models not covered by them (*LinearCovarianceModels.jl* version 0.2 and *ggm* version 2.5). The capabilities of our package are limited by the feasibility of the Gröbner basis computations involved.

Given a data sample of n independent and identically distributed random vectors $X^{(1)}, \ldots, X^{(n)}$ that follow an m-dimensional multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$, the *maximum likelihood*

MSC2020: 62F10, 62H22, 62R01.
Keywords: algebraic statistics, Gaussian graphical models, loopless mixed graphs, MLE.
GraphicalModelsMLE version 1.0 is included in *Macaulay2* version 1.20 or as an online supplement with this paper.

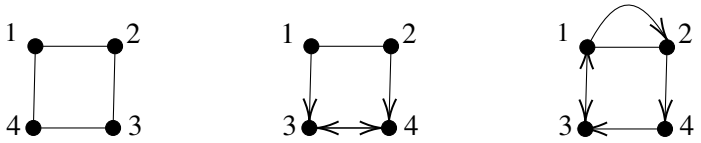

Figure 1. An undirected graph (left), a mixed graph with no directed cycles (centre), and a mixed graph with directed cycles (right).

estimate (MLE) for the covariance matrix Σ is the matrix that best explains the observed data, in the sense that it maximizes the likelihood function of the Gaussian model (see Section 3).

2. GRAPHICAL MODELS OF LOOPLESS MIXED GRAPHS. A *mixed graph* $G = (V, E)$ is a graph with undirected edges $i - j$, directed edges $i \to j$ and bidirected edges $i \leftrightarrow j$. A *directed cycle* is a cycle formed by directed edges after identifying the vertices that are connected by undirected or bidirected edges. A *loopless mixed graph* (LMG) is a mixed graph without loops or directed cycles. We allow double edges of the types directed-undirected and directed-bidirected. See Figure 1 (left and centre) for examples and Figure 1 (right) for a nonexample.

Following [Sullivant et al. 2010], we assume the nodes of G are partitioned as $V = U \cup W$, such that:

- If $i - j$ in G then $i, j \in U$.
- If $i \leftrightarrow j$ in G then $i, j \in W$.
- There is no directed edge $i \to j$ in G such that $i \in W$ and $j \in U$.

Our definition differs from the one in [Sadeghi and Lauritzen 2014] in that we do not allow multiple edges of the same type, which is due to the setup of the *Graphs* package. Also note that the partition of vertices excludes multiple edges of the undirected-bidirected type. In addition, we prohibit directed cycles, which ensures there is an ordering on the vertices such that all vertices in U come before vertices in W, and whenever $i \to j$ we have $i < j$. For simplicity of the algorithm, in our *Macaulay*2 implementation we will require the graph to be provided with such an ordering of the vertices (see more details in the description of `partitionLMG` in Section 6).

A Gaussian graphical model imposes constraints on the covariance matrix of a Gaussian distribution. More precisely, a loopless mixed graph $G = (V, E)$ gives rise to the space of covariance matrices $\Sigma \in \mathbb{R}^{|V|\times|V|}$ of the form [Sullivant et al. 2010, Section 2.3]

$$\Sigma \quad = \quad (I - \Lambda)^{-T} \begin{bmatrix} K^{-1} & \mathbf{0} \\ \mathbf{0} & \Psi \end{bmatrix} (I - \Lambda)^{-1}, \tag{2.1}$$

where

(i) $\Lambda = [\lambda_{ij}] \in \mathbb{R}^{|V|\times|V|}$ is such that $\lambda_{ij} = 0$ whenever $i \to j \notin E$;

(ii) $K = [k_{ij}] \in \mathbb{R}^{|U|\times|U|}$ is symmetric positive definite such that $k_{ij} = 0$ whenever $i - j \notin E$;

(iii) $\Psi = [\psi_{ij}] \in \mathbb{R}^{|W|\times|W|}$ is symmetric positive definite such that $\psi_{ij} = 0$ whenever $i \leftrightarrow j \notin E$.

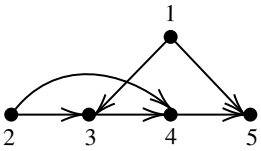

Figure 2. Verma graph.

3. MAXIMUM LIKELIHOOD ESTIMATES. Let the data sample consist of n independent identically distributed random vectors $X^{(1)}, \ldots, X^{(n)}$ sampled from an m-dimensional Gaussian distribution $\mathcal{N}_m(\mu, \Sigma)$. The parameter space of the corresponding statistical model is $\Theta = \mathbb{R}^m \times \Theta_2 \subseteq \mathbb{R}^m \times PD_m$, where Θ_2 is the space of covariance matrices Σ and PD_m is the cone of $m \times m$ positive-definite matrices. The maximum likelihood estimates for the covariance matrix are determined by maximizing the log-likelihood function

$$\ell(\Sigma) = -\frac{n}{2} \log \det \Sigma - \frac{n}{2} \operatorname{tr}(S\Sigma^{-1}) \tag{3.1}$$

over $\Sigma \in \Theta_2$ [Sullivant 2018, Proposition 7.1.9], where S is the sample covariance matrix. The function `solverMLE` allows us to compute this optimum when Θ_2 is induced by (2.1). It does so by calculating the critical points of the log-likelihood function and selecting the points corresponding to the maximum value in the cone of positive definite matrices. The default output is the maximum value of $\ell(\Sigma)$, the list of maximum likelihood estimates for the covariance matrix and the maximum likelihood degree of the model.

For undirected graphs, the MLE for the covariance matrix is known to be the unique positive definite critical point of the likelihood function. In particular, it is a positive definite matrix completion to the partial sample covariance matrix. See [Uhler 2012, Theorem 2.1] or [Drton et al. 2009, Theorem 2.1.14] for more details.

Example 3.2. We consider the directed acyclic graph G known as the Verma graph; see Figure 2 and [Drton et al. 2009, Example 3.3.14]. We take as sample data the columns of a real matrix U generated with the command `random` in *Macaulay*2 and compute the MLE for the covariance matrix that best explains the data within the graphical model given by G.

```
i1: loadPackage "GraphicalModelsMLE";

i2: G=digraph{{1,3},{1,5},{2,3},{2,4},{3,4},{4,5}};

i3: U=matrix {{.0137595, .983763, .963969, .152094, .0453326},
              {.527344, .597575, .777622, .97937, .112339},
              {.097922, .300712, .333058, .824002, .420228},
              {.849322, .594136, .114729,.69734, .98773},
              {.764547, .42209, .480193, .246573, .846734}};

i4: solverMLE(G,U)

o4 = (8.77485, | .115729     -6.32685e-18 -.0387187 .00115181 .102733   |, 1)
               | 6.10884e-18 .053294      .0392544  -.0356783 .00701454 |
               | -.0387187   .0392544     .0807822  -.0278223 -.0289767 |
               | .00115181   -.0356783    -.0278223 .105095   -.0196375 |
               | .102733     .00701454    -.0289767 -.0196375 .148723   |
```

Example 3.3. We compute the MLE for the covariance matrix of the graphical model associated to the undirected 4-cycle; see Figure 1 (left). We encode the sample data by a matrix U generated as in Example 3.2 and compute the sample covariance matrix $S = \frac{1}{n}UU^T$.

```
i1 : loadPackage "GraphicalModelsMLE";

i2 : G=graph{{1,2},{2,3},{3,4},{4,1}};

i3 : S=matrix {{.105409, -.0745495, -.0186132, .0621907},
               {-.0745495, .0783734,-.00844503,-.0872842},
               {-.0186132, -.00844503, .128307, .0230245},
               {.0621907, -.0872842, .0230245,.109849}};

i4 : solverMLE(G,S,SampleData=>false)

o4 = (6.62005, | .105409   -.0745495  .0124099   .0621907  |, 5)
               | -.0745495 .0783734   -.00844503 -.0439427 |
               | .0124099  -.00844503 .128307    .0230245  |
               | .0621907  -.0439427  .0230245   .109849   |
```

Note that all entries in the MLE for the covariance matrix coincide with the entries in the sample covariance matrix except for those corresponding to nonedges of the graph. See [Michałek and Sturmfels 2021, Example 12.16] for more on a positive definite matrix completion problem associated to the 4-cycle.

For more general types of graphs, uniqueness of the positive definite critical points is no longer guaranteed. In the mixed graph in Example 3.4, the optimization problem has a global maximum, but there are also local maxima; see Example 4.3.

Example 3.4. We compute the MLE for the covariance matrix of the graphical model associated to the loopless mixed graph with undirected edge $1-2$, directed edges $1 \to 3$, $2 \to 4$ and bidirected edge $3 \leftrightarrow 4$; see Figure 1 (centre). S is a sample covariance matrix computed from sample data encoded in a rational matrix obtained again with the command `random`.

```
i2 : G = mixedGraph(graph{{1,2}},digraph{{1,3},{2,4}},bigraph{{3,4}});

i3 : S=matrix {{34183/50000, 716539/10000000, 204869/250000, 12213/25000},
               {716539/10000000, 112191/500000, 309413/1000000, 1803/4000},
               {204869/250000, 309413/1000000, 3849/3125, 15172/15625},
               {12213/25000, 1803/4000, 15172/15625, 4487/4000}};

i4 : solverMLE(G,S,SampleData=>false)

o4 = (9.36624, {| .68366   .0716539 1.00282   .234375   |}, 5)
                | .0716539 .224382  .105105   .733937   |
                | 1.00282  .105105  1.76955   -.0700599 |
                | .234375  .733937  -.0700599 2.97432   |
```

4. IDEAL OF SCORE EQUATIONS. The critical points of the log-likelihood function $\ell(\Sigma)$ are the solutions to the system of equations obtained by taking partial derivatives of ℓ with respect to all variables in the entries of Σ from our construction in (2.1) and setting them to zero:

$$-\frac{\partial}{\partial(\cdot)} \det \Sigma - \det \Sigma \frac{\partial}{\partial(\cdot)} \operatorname{tr}(S\Sigma^{-1}) = 0, \tag{4.1}$$

where $\frac{\partial}{\partial(\cdot)}$ stands for the partial derivatives with respect to the variables $\lambda_{ij}, k_{ij}, \psi_{ij}$ in the entries of the covariance matrix Σ from (2.1). These polynomial equations are called *score equations*. The command `scoreEquations` returns the ideal generated by such polynomials, which lives in the polynomial

ring $\mathbb{Q}[\lambda_{ij}, k_{ij}, \psi_{ij}]$. From an algebraic perspective, this ideal is already of interest on its own; see [Sullivant 2018, Chapter 7].

Note that the log-likelihood function depends both on the sample covariance matrix and the graphical model. Therefore our implementation of `scoreEquations` requires as input the sample data along with information about the model. The latter is obtained via the command `gaussianRing` in the package *GraphicalModels* (see [García-Puente et al. 2013] and examples in Section 6), which produces a ring associated to the graph G that stores all relevant features of the graphical model.

Example 4.2. We compute the ideal of score equations associated to the 4-cycle after creating the graph G as in Example 3.3. We now consider as input data the sample data encoded in the columns of the integer matrix U below, obtained via the command `random`.

```
i5 : U=matrix{{3,5,9,5},{1,6,1,5},{2,9,6,6},{2,5,0,4}};
i6 : J=scoreEquations(gaussianRing G,U);
o6 : Ideal of QQ[k    , k    , k    , k    , k    , k    , k    , k    ]
                 1,1    2,2    3,3    4,4    1,2    1,4    2,3    3,4
i7 : dim J
o7 = 0
```

The ideal of the score equations J is generated by fourteen nonhomogeneous polynomials within $\mathbb{Q}[k_{1,1}, k_{1,2}, k_{1,4}, k_{2,2}, k_{2,3}, k_{3,3}, k_{3,4}, k_{4,4}]$: 4 linear polynomials and 10 quadratic polynomials such as $1312002k_{3,4}^2 - 387081k_{1,2} + 109860k_{1,4} + 1972025k_{2,3} - 898518k_{3,4} - 291556$. Since this ideal is zero-dimensional, the log-likelihood function $\ell(\Sigma)$ defined in (3.1) has finitely many complex critical points, as will be discussed in Section 5.

Example 4.3. We want to obtain all local maxima of the log-likelihood function associated to the graphical model in Example 3.4. We write λ as l and ψ as p in the code. The score equations generate an ideal in $\mathbb{Q}[k_{1,1}, k_{2,2}, k_{1,2}, l_{1,3}, l_{2,4}, p_{3,3}, p_{4,4}, p_{3,4}]$ and we display their solutions in the *Macaulay*2 session below. We retrieve the covariance matrix Σ with rational entries in variables $k_{1,1}$, $k_{2,2}$, $k_{1,2}$, $l_{1,3}$, $l_{2,4}$, $p_{3,3}$, $p_{4,4}$, and $p_{3,4}$ by requiring the optional output `CovarianceMatrix` in `scoreEquations`.

```
i5 : R = gaussianRing G;
i6 : (J,Sigma)=scoreEquations(R,S,SampleData=>false,CovarianceMatrix=>true);
i7 : dim J, degree J
o7 = (0, 5)
i8 : sols=zeroDimSolve(J);netList sols
     +-------------------------------------------------------------------------+
o8 = |{1.51337, 4.61101, -.483277, 1.46684, 3.27093, .298576, .573665, -.41385} |
     +-------------------------------------------------------------------------+
     |{1.51337, 4.61101, -.483277, 1.39884+.440525*ii, 2.45466-.923165*ii,     |
     |.144129+.120574*ii,.0696297-.184692*ii,-.19668+.0553853*ii}              |
     +-------------------------------------------------------------------------+
     |{1.51337, 4.61101, -.483277, 1.39884-.440525*ii, 2.45466+.923165*ii,     |
     |.144129-.120574*ii, .0696297+.184692*ii, -.19668-.0553853*ii}            |
     +-------------------------------------------------------------------------+
     |{1.51337, 4.61101, -.483277, .684147, .979681, .430388, .453924, .381688} |
     +-------------------------------------------------------------------------+
     |{1.51337, 4.61101, -.483277, .988484, 1.64649, .279607, .245722, .0952865}|
     +-------------------------------------------------------------------------+
```

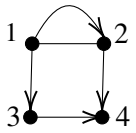

Figure 3. LMG with two types of edges between 1 and 2.

How many of the 3 real critical points correspond to positive definite matrices that are local maxima of the log-likelihood function? We first check that they correspond to positive definite matrices by substituting the three real solutions in the covariance matrix Σ.

```
i9 : checkPD(apply(sols,i->sub(Sigma,matrix{coordinates(i)})))

o9 = |.68366   .0716539 1.00282   .234375   |,| .68366   .0716539 .467724  .070198 |,
     |.0716539 .224382  .105105   .733937   | | .0716539 .224382  .0490218 .219823 |
     |1.00282  .105105  1.76955   -.0700599 | | .467724  .0490218 .75038   .429714 |
     |.234375  .733937  -.0700599 2.97432   | | .070198  .219823  .429714  .66928  |
                      | .68366   .0716539 .675787  .117978 |
                      | .0716539 .224382  .0708287 .369443 |
                      | .675787  .0708287 .947611  .211905 |
                      | .117978  .369443  .211905  .854009 |
```

The MLE for the covariance matrix obtained in Example 3.4 corresponds to the first positive definite matrix in the list above. The eigenvalues of the Hessian matrix computed below tell us which kind of critical point we have for each of the 3 real solutions — for a discussion about the properties of positive-semidefinite matrices; see [Parrilo 2013, Appendix A].

```
-- compute Jacobian matrix (i.e. score equations)
i10 : scoreEq=-1/det Sigma*jacobianMatrixOfRationalFunction(det Sigma)-
jacobianMatrixOfRationalFunction(trace(S*(inverse Sigma)));

-- compute Hessian matrix
i11 : Hessian=matrix for f in flatten entries scoreEq list
flatten entries jacobianMatrixOfRationalFunction(f);

-- compute eigenvalues of the Hessian matrix evaluated at real points in sols
i12 : apply({sols_0,sols_3,sols_4},i->eigenvalues sub(Hessian,matrix{coordinates(i)}))

o12 = {{-.516478   }, {-.516478 }, {-.516478 }}
       {-.271913   }  {-.271913 }  {-.271913 }
       {-.0464172  }  {-.0464172}  {-.0464172}
       {-9869730000}  {-414.15  }  {-59.7135 }
       {-128887    }  {-28.6352 }  {1.52598  }
       {-58261.1   }  {-6.1936  }  {-2.80504 }
       {-773.513   }  {-1.64689 }  {-11.5533 }
```

The first two points are local maxima and the last point is a saddle point. This shows that the log-likelihood function of this model is not a concave function, see [Drton 2006].

Example 4.4. Next we compute the ideal of score equations associated to a mixed graph that has two different types of edges connecting the same two vertices: directed edges $1 \to 3$, $1 \to 2$, $2 \to 4$, $3 \to 4$ and the undirected edge $1 - 2$; see Figure 3.

```
i2 : G = mixedGraph(digraph{{1,3},{1,2},{2,4},{3,4}},graph{{1,2}});

i3 : R = gaussianRing G;

i4 : U = random(RR^4,RR^4);

i5 : J=scoreEquations(R,U);

i6 : dim J

o6 = 1
```

Note that in this case, as opposed to Example 4.2, the log-likelihood function (3.1) has infinitely many complex critical points. Since our package evaluates the objective function in all critical points, in such a scenario the MLE cannot be computed.

5. MAXIMUM LIKELIHOOD DEGREE. The *maximum likelihood degree* (ML degree) of a model is defined as the number of complex critical points of the log-likelihood function $\ell(\Sigma)$ from (3.1) for generic sample data; see [Sullivant 2018, Definition 7.1.4]. For a more algebraic flavour of the notion of ML degree, see [Michałek et al. 2016, Definition 5.4].

Note that the ML degree is only well defined when the ideal of score equations is zero-dimensional. A typical way where this fails is where the model becomes nonidentifiable. See, for example, [Améndola et al. 2020], for some sufficient conditions to avoid nonidentifiability and preservation of dimension of the model in terms of the number of parameters.

It is important to observe that for generic data the solutions to score equations are all distinct; see [Améndola et al. 2021, Remark 2.1, Lemma 2.2]. Computing the algebraic degree of the zero-dimensional score equations ideal via the `degree` function in *Macaulay*2 is equivalent to computing the number of complex solutions - without multiplicity - to the score equations (4.1).

In our implementation of the `MLdegree` function in *Macaulay*2, a random sample matrix is used as sample data. Therefore, the ML degree of the graphical model we provide is correct with probability 1.

Example 5.1. The ML degree of the 4-cycle can be directly computed as follows:

```
i2 : G=graph{{1,2},{2,3},{3,4},{4,1}};
i3 : MLdegree(gaussianRing G)
o3 = 5
```

In the case of ideals of score equations with positive dimension, `MLdegree` will still compute the degree of the ideal but this no longer matches the number of solutions to the score equations.

Example 5.2. Continuing with Example 4.4, where the ideal of score equations is 1-dimensional, `MLdegree` does not provide a meaningful answer.

```
i2: G=mixedGraph(digraph{{1,3},{1,2},{2,4},{3,4}},graph{{1,2}});
i3: MLdegree(gaussianRing G)
error: the ideal of score equations has dimension 1 > 0,
so ML degree is not well defined. The degree of this ideal is 2.
```

6. UPDATES IN RELATED PACKAGES. *GraphicalModelsMLE* relies on the new package *StatGraphs* 0.1 and the updated packages *Graphs* 0.3.3 and *GraphicalModels* 2.0 (see [García-Puente et al. 2013] for version 1.0).

We created a dedicated package *StatGraphs* for graph theoretic functions relevant to algebraic statistics. It contains the functions `isCyclic`, `isSimple`, `isLoopless` and `partitionLMG` to deal with loopless mixed graphs.

The function `partitionLMG` computes the partition $V = U \cup W$ of vertices of a loopless mixed graph described in Section 2. Vertices in the input graph need to be ordered such that all vertices in U come before vertices in W, and if there is a directed edge $i \to j$, then $i < j$.

Example 6.1. The vertices of the loopless mixed graph in Example 3.4 are partitioned into $U = \{1, 2\}$ and $W = \{3, 4\}$.

```
i1 : loadPackage "StatGraphs";
i2 : G = mixedGraph(digraph {{1,3},{2,4}},bigraph{{3,4}},graph{{1,2}});
i3 : partitionLMG G
o3 = ({1, 2}, {3, 4})
o3 : Sequence
```

The central object in the implementation of our MLE algorithm is `gaussianRing` from the package *GraphicalModels*.

Example 6.2. We compute the ring associated to the graph in Example 6.1 and display the variables of the ring as entries of matrices. We write λ as l and ψ as p in the code.

```
i4 : loadPackage "GraphicalModels";
i5 : R=gaussianRing G;
i6 : undirectedEdgesMatrix R
o6 = | k_(1,1) k_(1,2) |
     | k_(1,2) k_(2,2) |
i7 : directedEdgesMatrix R
o7 = | 0 0 l_(1,3) 0       |
     | 0 0 0       l_(2,4) |
     | 0 0 0       0       |
     | 0 0 0       0       |
i8 : bidirectedEdgesMatrix R
o8 = | p_(3,3) p_(3,4) |
     | p_(3,4) p_(4,4) |
i9 : covarianceMatrix R
o9 = | s_(1,1) s_(1,2) s_(1,3) s_(1,4) |
     | s_(1,2) s_(2,2) s_(2,3) s_(2,4) |
     | s_(1,3) s_(2,3) s_(3,3) s_(3,4) |
     | s_(1,4) s_(2,4) s_(3,4) s_(4,4) |
```

In version 2.0 of *GraphicalModels*, we updated the functionality of the method `gaussianRing` and its related methods in order to accept loopless mixed graphs with undirected, directed and bidirected edges.

Note that mixed graphs that include undirected edges are required to have an ordering compatible with `partitionLMG`. For mixed graphs with only directed and bidirected edges this is no longer necessary, as in version 1.0 of *GraphicalModels*.

ACKNOWLEDGEMENTS. We thank Michael Stillman and Daniel Grayson for their support and help with the inner workings of *Macaulay*2 during the 2020 online Warwick *Macaulay*2 Workshop. We are also grateful to Elina Robeva and David Swinarski for creating the first draft of *GraphicalModelsMLE*, and to Piotr Zwiernik for useful discussions. Motwani was partially supported by UGent BOF/STA/201909/038 and FWO grant G0F5921N.

SUPPLEMENT. The online supplement contains version 1.0 of *GraphicalModelsMLE*.

REFERENCES.

[Améndola et al. 2020] C. Améndola, P. Dettling, M. Drton, F. Onori, and J. Wu, "Structure learning for cyclic linear causal models", pp. 999–1008 in *Proceedings of the* 36*th Conference on Uncertainty in Artificial Intelligence*, edited by J. Peters and D. Sontag, Proceedings of Machine Learning Research **124**, PMLR, 2020.

[Améndola et al. 2021] C. Améndola, L. Gustafsson, K. Kohn, O. Marigliano, and A. Seigal, "The maximum likelihood degree of linear spaces of symmetric matrices", *Matematiche* (*Catania*) **76**:2 (2021), 535–557. MR

[Drton 2006] M. Drton, "Computing all roots of the likelihood equations of seemingly unrelated regressions", *J. Symbolic Comput.* **41**:2 (2006), 245–254. MR Zbl

[Drton et al. 2009] M. Drton, B. Sturmfels, and S. Sullivant, *Lectures on algebraic statistics*, Oberwolfach Seminars **39**, Birkhäuser Verlag, Basel, 2009. MR Zbl

[García-Puente et al. 2013] L. D. García-Puente, S. Petrović, and S. Sullivant, "Graphical models", *J. Softw. Algebra Geom.* **5** (2013), 1–7. MR Zbl

[Lauritzen 1996] S. L. Lauritzen, *Graphical models*, Oxford Statistical Science Series **17**, Oxford University Press, 1996. MR Zbl

[Macaulay2] D. R. Grayson and M. E. Stillman, "Macaulay2: a software system for research in algebraic geometry", available at http://www.math.uiuc.edu/Macaulay2.

[Marchetti 2006] G. M. Marchetti, "Independencies Induced from a Graphical Markov Model After Marginalization and Conditioning: The R Package ggm", *Journal of Statistical Software* **15**:6 (2006), 1–15.

[Michałek and Sturmfels 2021] M. Michałek and B. Sturmfels, *Invitation to nonlinear algebra*, Graduate Studies in Mathematics **211**, American Mathematical Society, Providence, RI, 2021. MR Zbl

[Michałek et al. 2016] M. Michałek, B. Sturmfels, C. Uhler, and P. Zwiernik, "Exponential varieties", *Proc. Lond. Math. Soc.* (3) **112**:1 (2016), 27–56. MR Zbl

[Parrilo 2013] P. A. Parrilo, "Semidefinite optimization", pp. 3–46 in *Semidefinite optimization and convex algebraic geometry*, MOS-SIAM Ser. Optim. **13**, SIAM, Philadelphia, PA, 2013. MR Zbl

[Sadeghi and Lauritzen 2014] K. Sadeghi and S. Lauritzen, "Markov properties for mixed graphs", *Bernoulli* **20**:2 (2014), 676–696. MR Zbl

[Sturmfels et al. 2020] B. Sturmfels, S. Timme, and P. Zwiernik, "Estimating linear covariance models with numerical nonlinear algebra", *Algebr. Stat.* **11**:1 (2020), 31–52. MR Zbl

[Sullivant 2018] S. Sullivant, *Algebraic statistics*, Graduate Studies in Mathematics **194**, American Mathematical Society, Providence, RI, 2018. MR Zbl

[Sullivant et al. 2010] S. Sullivant, K. Talaska, and J. Draisma, "Trek separation for Gaussian graphical models", *Ann. Statist.* **38**:3 (2010), 1665–1685. MR Zbl

[Uhler 2012] C. Uhler, "Geometry of maximum likelihood estimation in Gaussian graphical models", *The Annals of Statistics* **40**:1 (2012), 238 – 261. Zbl

RECEIVED: 22 Dec 2020 REVISED: 30 Nov 2021 ACCEPTED: 14 Mar 2022

CARLOS AMÉNDOLA:

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

LUIS DAVID GARCÍA PUENTE:

lgarciapuente@coloradocollege.edu
Department of Mathematics and Computer Science, Colorado College, Colorado Springs, CO, United States

ROSER HOMS:

roser.homs@tum.de
Department of Mathematics, Technical University of Munich, Munich, Germany

OLGA KUZNETSOVA:

kuznetsova.olga@gmail.com
Department of Mathematics and Systems Analysis, Aalto University, Aalto, Finland

HARSHIT J. MOTWANI:

harshitjitendra.motwani@ugent.be
Department of Mathematics: Algebra and Geometry, Ghent University, Ghent, Belgium

JSAG **12** (2022), 11–16
https://doi.org/10.2140/jsag.2022.12.11

The Journal of Software for
Algebra and Geometry

Linear truncations package for Macaulay2

LAUREN CRANTON HELLER AND NAVID NEMATI

ABSTRACT: We introduce the *Macaulay*2 package `LinearTruncations` for finding and studying the truncations of a multigraded module over a standard multigraded ring that have linear resolutions.

1. INTRODUCTION AND PRELIMINARIES. Castelnuovo–Mumford regularity is a fundamental invariant in commutative algebra and algebraic geometry. Roughly speaking, it measures the complexity of a module or sheaf. Let S be a polynomial ring with the standard grading and let M be a finitely generated S-module. In this case, Castelnuovo–Mumford regularity is typically defined in terms of either the graded Betti numbers of M or the vanishing of local cohomology modules $H^i_{\mathfrak{m}}(M)$, where $\mathfrak{m}$ is the maximal homogeneous ideal of S. Eisenbud and Goto [1984] showed that the Castelnuovo–Mumford regularity of M is the minimum degree where the truncation of M has a linear resolution.

Extensions of Castelnuovo–Mumford regularity were introduced for bigraded modules by Hoffman and Wang [2004], independently for multigraded modules by Maclagan and Smith [2004], then later in a more general setting by Botbol and Chardin [2017]. The multigraded regularity of a module is a region in $\mathbb{Z}^r$ rather than an integer. For a polynomial ring with a standard $\mathbb{Z}^r$-grading, multigraded regularity is invariant under positive translations and thus can be described by its minimal elements. An affirmative answer to the following open question would reduce this to a finite computation.

Question 1.1. Can the minimal elements of the regularity of M be bounded in terms of S and the Betti numbers of M?

In analogy to the singly graded case, one may ask about the relation between multigraded Castelnuovo–Mumford regularity and the multidegrees where the truncation of a module has a linear resolution, which we call the *linear truncation region*. (See Definitions 1.2 and 1.3.) In the multigraded setting these regions can differ, but a bound on the linear truncations would answer Question 1.1 by [Eisenbud et al. 2015, Proposition 4.11]. Our goal is to compute the minimal elements of the linear truncation region within a specified finite region of $\mathbb{Z}^r$.

We introduce the *LinearTruncations* package for [Macaulay2], which provides tools for studying the resolutions of truncations of modules over rings with standard multigradings. Given a module and a bounded range of multidegrees, our package can identify all linear truncations in the range. The algorithm

MSC2020: primary 13-04, 13D02; secondary 13P20, 14M25.
Keywords: multigraded regularity, truncation.
`LinearTruncations` version 1.0

uses a search function that is also applicable to other properties of modules described by sets of degrees. The examples here were computed using version 1.18 of *Macaulay*2 and version 1.0 of *LinearTruncations*.

In Section 2, we describe the main algorithms of this package, `findRegion` and `linearTruncations`. In Section 3, we discuss the relation between the linear truncation region and the multigraded regularity and we introduce `regularityBound` and `linearTruncationsBound` as faster methods for calculating subsets of the multigraded regularity and linear truncation regions, respectively.

To set our notation, let k be a field and let

$$S = k[x_{i,j} \mid 1 \le i \le r,\ 0 \le j \le n_i]$$

be a $\mathbb{Z}^r$-graded polynomial ring with $\deg x_{ij} = \boldsymbol{e}_i$, the i-th standard basis vector in $\mathbb{Z}^r$, for all j (so that S is the coordinate ring of a product $\mathbb{P}^{n_1} \times \cdots \times \mathbb{P}^{n_r}$ of projective spaces). The function `multigradedPolynomialRing` produces such rings:

```
i1 : needsPackage "LinearTruncations"
o1 = LinearTruncations
o1 : Package
i2 : S = multigradedPolynomialRing {1,2}
o2 = S
o2 : PolynomialRing
i3 : degrees S
o3 = {{1, 0}, {1, 0}, {0, 1}, {0, 1}, {0, 1}}
o3 : List
```

Let M be a finitely generated $\mathbb{Z}^r$-graded S-module. For a multidegree $\boldsymbol{d} = (d_1, \ldots, d_r) \in \mathbb{Z}^r$, write $\bar{\boldsymbol{d}}$ for the total degree $d_1 + \cdots + d_r$ of $\boldsymbol{d}$ and $M_{\ge \boldsymbol{d}}$ for the truncation $\bigoplus_{\boldsymbol{d}' \ge \boldsymbol{d}} M_{\boldsymbol{d}'}$ of M at $\boldsymbol{d}$, where $\boldsymbol{d}' \ge \boldsymbol{d}$ if this inequality is true for each coordinate.

Definition 1.2. A homogeneous chain complex

$$0 \leftarrow G_0 \leftarrow G_1 \leftarrow \cdots \leftarrow G_k \leftarrow 0$$

of free S-modules is *linear* if $G_0 \simeq \bigoplus S(-\boldsymbol{d})$ for some $\boldsymbol{d} \in \mathbb{Z}^r$ and for each free summand $S(-\boldsymbol{d}')$ of G_i we have $\bar{\boldsymbol{d}}' = \bar{\boldsymbol{d}} + i$.

The function `isLinearComplex` checks this condition. To print the degrees appearing in the complex, use `supportOfTor`:

```
i4 : B = irrelevantIdeal S
o4 = ideal (x   x   , x   x   , x   x   , x   x   , x   x   , x   x   )
             0,1 1,2   0,0 1,2   0,1 1,1   0,0 1,1   0,1 1,0   0,0 1,0
o4 : Ideal of S
i5 : F = res comodule B
          1       6       9       5       1
o5 = S  <-- S  <-- S  <-- S  <-- S  <-- 0
     0      1       2       3       4       5
o5 : ChainComplex
```

```
i6 : netList supportOfTor F
     +------+------+
o6 = |{0, 0}|      |
     +------+------+
     |{1, 1}|      |
     +------+------+
     |{2, 1}|{1, 2}|
     +------+------+
     |{2, 2}|{1, 3}|
     +------+------+
     |{2, 3}|      |
     +------+------+

i7 : isLinearComplex F

o7 = false
```

Definition 1.3. The *linear truncation region* of M is

$$\{\boldsymbol{d} \mid M_{\geq \boldsymbol{d}} \text{ has a linear resolution with generators in degree } \boldsymbol{d}\} \subset \mathbb{Z}^r.$$

Remark 1.4. Our definitions imply that the nonzero entries in the differential matrices of a linear resolution will have total degree 1. We also require that the generators have degree $\boldsymbol{d}$ so that a linear resolution for $M_{\geq \boldsymbol{d}}$ implies the existence of a linear resolution for $M_{\geq \boldsymbol{d}'}$ whenever $\boldsymbol{d}' \geq \boldsymbol{d}$. We can thus describe the linear truncation region by giving its minimal elements.

2. FINDING LINEAR TRUNCATIONS. Eisenbud, Erman, and Schreyer [Eisenbud et al. 2015] proved that the linear truncation region of M is nonempty. In particular it contains the output of the function `coarseMultigradedRegularity` from their package [TateOnProducts]. However, in general this degree is neither a minimal element itself nor greater than all the minimal elements. (See Example 2.1.)

The function `linearTruncations` searches for multidegrees where the truncation of M has a linear resolution by calling the function `findRegion`, which implements Algorithm 1. Since we do not know of a bound on the total degree of the minimal elements in the linear truncation region given the Betti numbers of M, `linearTruncations` is not guaranteed to produce all generators as a module over the semigroup $\mathbb{N}^r$. By default it searches above the componentwise minimum of the degrees of the generators of M and below the degree with all coordinates equal to $c+1$, where c is the output of `regularity`. Otherwise the range is taken as a separate input.

Example 2.1. Let $S = k[x_{0,0}, x_{0,1}, x_{0,2}, x_{1,0}, x_{1,1}, x_{1,2}, x_{1,3}]$ be the Cox ring of $\mathbb{P}^2 \times \mathbb{P}^3$. For each $d \geq 2$, let $\phi_d : S(-d,-d)^6 \to S(0,-d)^2 \oplus S(-d,0)^4$ be given by

$$\begin{pmatrix} x_{0,0}^d & x_{0,1}^d & x_{0,2}^d & 0 & 0 & 0 \\ 0 & 0 & 0 & x_{0,1}^d & x_{0,0}^d & x_{0,2}^d \\ x_{1,0}^d & 0 & 0 & x_{1,0}^d & 0 & 0 \\ 0 & x_{1,1}^d & 0 & 0 & x_{1,1}^d & 0 \\ 0 & 0 & x_{1,2}^d & 0 & 0 & x_{1,2}^d \\ 0 & 0 & 0 & x_{1,3}^d & 0 & 0 \end{pmatrix},$$

and define $M^{(d)} := \operatorname{coker} \phi_d$. The `coarseMultigradedRegularity` of $M^{(3)}$ is $\{3,3\}$, the `regularity` of $M^{(3)}$ is 5, and $\{3,3\}$ and $\{8,2\}$ are minimal elements of the linear truncation region. Since $\{8,2\}$ is

Input : a module M, a Boolean function f that takes M as input, and a range $(\boldsymbol{a}, \boldsymbol{b})$
Output : minimal elements between $\boldsymbol{a}$ and $\boldsymbol{b}$ where M satisfies f

$A := \varnothing$
$K := \{\boldsymbol{a}\}$
while $K \neq \varnothing$ **do**
  $\boldsymbol{d} :=$ first element of K
  $K = K \setminus \{\boldsymbol{d}\}$
  if $\boldsymbol{d} \notin A + \mathbb{N}^r$ **then**
    if *M satisfies f at* $\boldsymbol{d}$ **then**
      $A = A \cup \{\boldsymbol{d}\}$
    else
      for $1 \leq i \leq r$ **do**
        if $\boldsymbol{d} + \boldsymbol{e}_i \leq \boldsymbol{b}$ **then**
          $K = K \cup \{\boldsymbol{d} + \boldsymbol{e}_i\}$
return *minimal elements of A*

Algorithm 1. `findRegion`

not below $\{5+1, 5+1\}$, it will not be returned by the `linearTruncations` function with the default options:

```
i8 : (S,E) = productOfProjectiveSpaces{2,3};

i9 : d = 3;

i10 : M = coker(map(S^{{0,-d},{0,-d},{-d,0},{-d,0},{-d,0},{-d,0}},
        S^{{-d,-d},{-d,-d},{-d,-d},{-d,-d},{-d,-d},{-d,-d}},
        {{x_(0,0)^d,x_(0,1)^d,x_(0,2)^d,0,0,0}, {0,0,0,x_(0,1)^d,x_(0,0)^d,x_(0,2)^d},
        {-x_(1,0)^d,0,0,-x_(1,0)^d,0,0}, {0,-x_(1,1)^d,0,0,-x_(1,1)^d,0},
        {0,0, -x_(1,2)^d,0,0, -x_(1,2)^d}, {0,0,0, -x_(1,3)^d,0,0}}));

i11 : linearTruncations M

o11 = {{3, 3}}

o11 : List

i12 : linearTruncations({{0,0},{8,6}},M)

o12 = {{3, 3}, {8, 2}}

o12 : List
```

Based on the computations from $M^{(d)}$ for $2 \leq d \leq 10$ we expect that for $d \geq 2$ the module $M^{(d)}$ will have `coarseMultigradedRegularity` equal to $\{d, d\}$, with $\{d, d\}$ and $\{3d-1, d-1\}$ both minimal elements of the linear truncation region.

At each step of Algorithm 1 the set A contains degrees satisfying f and the set K contains the minimal degrees remaining to be checked. There are options to initialize A and K differently — degrees in A will be assumed to satisfy f, and degrees below those in K will be excluded from the search (and thus assumed not to satisfy f). Supplying such prior knowledge can decrease the length of the computation by limiting the number of times the algorithm calls f.

The pseudocode in Algorithm 1 masks the fact that A and K are stored as monomial ideals in a temporary singly graded polynomial ring. Similarly, the function `findMins` will convert a list of multidegrees to a monomial ideal in order to calculate its minimal elements via a Gröbner basis.

3. RELATION TO REGULARITY. As discussed above, the minimal element of the linear truncation region of a singly graded module agrees with its Castelnuovo–Mumford regularity, which can be determined from its Betti numbers. In the multigraded case these concepts are still linked, but their relationship is more complicated. For instance, the following inclusion is strict:

Theorem 3.1. *If $H^0_B(M) = 0$, then the linear truncation region of M is a subset of the multigraded regularity region* reg *M of M, as defined in* [Maclagan and Smith 2004].

Proof. See [Berkesch et al. 2020, Theorem 2.9] or [Eisenbud et al. 2015, Proposition 4.11]. □

Unfortunately the multigraded Betti numbers of M do not determine either its regularity or its linear truncations. However, the functions `regularityBound` and `linearTruncationsBound` compute subsets of these regions using only the twists appearing in the minimal free resolution of M. In many examples they produce the same outputs as `multigradedRegularity` (from the package *VirtualResolutions* [Almousa et al. 2020]) and `linearTruncations`, respectively, without computing sheaf cohomology or truncating the module.

The algorithms for `linearTruncationsBound` and `regularityBound` are based on the following theorem (from [Bruce et al. 2021]):

Theorem 3.2. *If $H^0_B(M) = 0$ and $H^1_B(M) = 0$ then*

$$\bigcap_{\mathrm{Tor}_i(M,k)_{\boldsymbol{b}}\neq 0} \bigcup_{\sum \lambda_j = i} [\boldsymbol{b} - \lambda_1 \boldsymbol{e}_i - \cdots - \lambda_r \boldsymbol{e}_r + \mathbb{N}^r]$$

is a subset of the `linearTruncations` *of M, and*

$$\bigcap_{\mathrm{Tor}_i(M,k)_{\boldsymbol{b}}\neq 0} \bigcup_{\sum \lambda_j = i-1} [\boldsymbol{b} - \mathbf{1} - \lambda_1 \boldsymbol{e}_i - \cdots - \lambda_r \boldsymbol{e}_r + \mathbb{N}^r]$$

is a subset of the `multigradedRegularity` *of M.*

The function `partialRegularities` calculates the Castelnuovo–Mumford regularity in each component of a multigrading.

Remark 3.3. In the bigraded case, Theorem 3.2 implies that $\boldsymbol{d}$ is in `linearTruncations M` if $\boldsymbol{d} \geq$ `partialRegularities M` and $\bar{\boldsymbol{d}} \geq$ `regularity M`.

For some modules, `linearTruncationsBound` gives a proper subset of the linear truncations:

```
i13 : S = multigradedPolynomialRing 2;
i14 : M = coker(map(S^{{-1,0},{0,-1},{0,-1}},S^{{-1,-1},{-1,-1}},
          {{x_(1,0),x_(1,1)},{-x_(0,0),0},{0,-x_(0,1)}}));
i15 : multigraded betti res M

            0   1
o15 = 1: a+2b   .
      2:    . 2ab
i16 : linearTruncations M
o16 = {{0, 2}, {1, 1}}
i17 : linearTruncationsBound M
o17 = {{1, 1}}
```

4. ACKNOWLEDGMENTS. The authors would like to thank David Eisenbud, who is also an author of the package. Cranton Heller was partially supported by the NSF under grant no. 1502209. Nemati would like to thank the organizers of the *Macaulay*2 workshop in Leipzig (June 2018), during which the algorithms were first implemented.

SUPPLEMENT. The online supplement contains version 1.0 of `LinearTruncations`.

REFERENCES.

[Almousa et al. 2020] A. Almousa, J. Bruce, M. Loper, and M. Sayrafi, "The virtual resolutions package for Macaulay2", *J. Softw. Algebra Geom.* **10**:1 (2020), 51–60. MR Zbl

[Berkesch et al. 2020] C. Berkesch, D. Erman, and G. G. Smith, "Virtual resolutions for a product of projective spaces", *Algebr. Geom.* **7**:4 (2020), 460–481. MR Zbl

[Botbol and Chardin 2017] N. Botbol and M. Chardin, "Castelnuovo Mumford regularity with respect to multigraded ideals", *J. Algebra* **474** (2017), 361–392. MR Zbl

[Bruce et al. 2021] J. Bruce, L. Cranton Heller, and M. Sayrafi, "Characterizing multigraded regularity on products of projective spaces", preprint, 2021. arXiv 2110.10705

[Eisenbud and Goto 1984] D. Eisenbud and S. Goto, "Linear free resolutions and minimal multiplicity", *J. Algebra* **88**:1 (1984), 89–133. MR Zbl

[Eisenbud et al. 2015] D. Eisenbud, D. Erman, and F.-O. Schreyer, "Tate resolutions for products of projective spaces", *Acta Math. Vietnam.* **40**:1 (2015), 5–36. MR Zbl

[Hoffman and Wang 2004] J. W. Hoffman and H. H. Wang, "Castelnuovo–Mumford regularity in biprojective spaces", *Adv. Geom.* **4**:4 (2004), 513–536. MR Zbl

[Macaulay2] D. R. Grayson and M. E. Stillman, "Macaulay2: a software system for research in algebraic geometry", available at http://www.math.uiuc.edu/Macaulay2.

[Maclagan and Smith 2004] D. Maclagan and G. G. Smith, "Multigraded Castelnuovo–Mumford regularity", *J. Reine Angew. Math.* **571** (2004), 179–212. MR Zbl

[TateOnProducts] D. Eisenbud, D. Erman, and F.-O. Schreyer, "TateOnProducts", Macaulay2 package, available at https://faculty.math.illinois.edu/Macaulay2/doc/Macaulay2/share/doc/Macaulay2/TateOnProducts/html/index.html.

RECEIVED: 1 Jul 2021 REVISED: 14 Mar 2022 ACCEPTED: 18 May 2022

LAUREN CRANTON HELLER:

lch@math.berkeley.edu
Department of Mathematics, University of California, Berkeley, Berkeley, CA, United States

NAVID NEMATI:

navid.nemati@inria.fr
Université Côte d'Azur, Inria, Sophia Antipolis, France

JSAG **12** (2022), 17–26
https://doi.org/10.2140/jsag.2022.12.17

The Journal of Software for
Algebra and Geometry

RationalMaps, a package for Macaulay2

C. J. BOTT, SEYED HAMID HASSANZADEH, KARL SCHWEDE AND DANIEL SMOLKIN

ABSTRACT: This paper describes the `RationalMaps` package for Macaulay2. This package provides functionality for computing several aspects of rational maps.

1. INTRODUCTION. This package aims to compute several things about rational maps between varieties, including

- the base locus of a rational map,
- whether a rational map is birational,
- the inverse of a birational map,
- whether a map is a closed embedding.

Our functions have numerous options which allow them to run much more quickly in certain examples if configured correctly. Setting the option `Verbosity` to a value ≥ 1 will mean that functions will provide hints as to the best ways to run them. This paper discusses `RationalMaps` version 1.0.

A rational map $\mathfrak{F} : X \subseteq \mathbb{P}^n \dashrightarrow Y \subseteq \mathbb{P}^m$ between projective varieties is presented by $m+1$ forms $\boldsymbol{f} = \{f_0, \dots f_m\}$ of the same degree in the coordinate ring of X, denoted by R. The idea of looking at the syzygies of the forms $\boldsymbol{f}$ to detect the geometric properties of $\mathfrak{F}$ goes back at least to [Hulek et al. 1992] in the case where $X = \mathbb{P}^n$, $Y = \mathbb{P}^m$ and $m = n$ (see also [Semple and Tyrrell 1969]). Russo and Simis [2001] developed this method to handle the case $X = \mathbb{P}^n$ and $m \geq n$. Simis [2004] pushed the method further to the study of general rational maps between two integral projective schemes in arbitrary characteristic by an extended ideal-theoretic method emphasizing the role of the Rees algebra associated to the ideal generated by $\boldsymbol{f}$. Doria, Hassanzadeh, and Simis [Doria et al. 2012] applied these Rees algebra techniques to study the birationality of $\mathfrak{F}$. Our core functions, particularly those related to computing inverse maps, rely heavily on this work.

Hassanzadeh was supported by CNPq-bolsa de Produtividade and by the MathAmSud project "ALGEO". Schwede was supported in part by the NSF FRG Grant DMS #1265261/1501115, NSF CAREER Grant DMS #1252860/1501102, NSF Grants DMS #1840190 and #2101800. Smolkin was supported in part by the NSF FRG Grant DMS #1265261/1501115, NSF Career Grant DMS #1252860/1501102 and NSF Grant DMS #1801849.
MSC2010: 14E05, 14E25.
Keywords: rational map, birational map, inverse map, closed embedding.
`RationalMaps` version 1.0

2. BASE LOCI. We begin with the problem of computing the base locus of a map to projective space. Let X be a projective variety over any field k and let $\mathfrak{F} : X \to \mathbb{P}^m_k$ be a rational map from X to projective space. Then we choose a representative $(f_0, \ldots, f_m)$ of $\mathfrak{F}$, where each f_i is the i-th coordinate of $\mathfrak{F}$. A priori, each f_i is in $K = \operatorname{frac} R$, where R is the coordinate ring of X. However, we can get another representative of $\mathfrak{F}$ by clearing denominators. (Note that this does not enlarge the base locus of $\mathfrak{F}$ since $\mathfrak{F}$ is undefined whenever the denominator of any of the f_i vanishes.) Thus we assume that $f_i \in R$ for all i, and that all the f_i are homogeneous of the same degree.

In this setting, one might naively think that the map $\mathfrak{F}$ is undefined exactly when all of the f_i vanish, and thus the base locus is the vanishing set of the ideal $(f_0, \ldots, f_m)$. However, this yields a base locus that is too big. Indeed, to find the base locus of a rational map, we must consider all possible representatives of the map and find where none of them are defined. To do this, we use the following result.

Proposition 2.1 [Simis 2004, Proposition 1.1]. *Let $\mathfrak{F} : X \dashrightarrow \mathbb{P}^m$ be a rational map and let $\boldsymbol{f} = \{f_0, \ldots, f_m\}$ be a representative of $\mathfrak{F}$ with $f_i \in R$ homogeneous of degree d for all i. Set $I = (f_0, \ldots, f_m)$. Then the set of such representatives of $\mathfrak{F}$ corresponds bijectively to the homogeneous vectors in the rank* 1 *graded R-module* $\operatorname{Hom}_R(I, R) \cong (R :_K I)$.

The bijection comes from multiplying our fixed representative $\boldsymbol{f}$ of $\mathfrak{F}$ by $h \in (R :_K I)$. Now, in the setting of Proposition 2.1, let

$$\bigoplus_s R(-d_s) \xrightarrow{\varphi} R(-d)^{m+1} \xrightarrow{[f_0,\ldots,f_m]} I \to 0$$

be a free resolution of I. Then we get

$$0 \to \operatorname{Hom}_R(I, R) \to (R(-d)^{m+1})^\vee \xrightarrow{\varphi^t} \left(\bigoplus_s R(d_s)\right)^\vee$$

where φ^t is the transpose of φ and $R^\vee$ is the dual module of R. Thus, we get that $\operatorname{Hom}_R(I, R) \cong \ker \varphi^t$, and so each representative of $\mathfrak{F}$ corresponds to a vector in $\ker \varphi^t$. The correspondence takes a representative $(hf_0, \ldots, hf_m)$ to the map that multiplies vectors in R^{m+1} by $[hf_0, \ldots, hf_m]$ on the left.

The base locus of $\mathfrak{F}$ is the intersection of the sets $V(f_0^i, \ldots, f_m^i)$ as $\boldsymbol{f}^i = (f_0^i, \ldots, f_m^i)$ ranges over all the representatives of $\mathfrak{F}$. The above implies that this is the same as the intersection of the sets $V(w_0^i, \ldots, w_m^i)$ as $\boldsymbol{w}^i = (w_0^i, \ldots, w_m^i)$ ranges over the vectors in $\ker \varphi^t$. Now, given any $a, f, g \in R$, we have $V(af) \supseteq V(f)$ and $V(f+g) \supseteq V(f) \cap V(g)$. Thus, it is enough to take a generating set $\boldsymbol{w}^1, \ldots, \boldsymbol{w}^n$ of $\ker \varphi^t$ and take the intersection over this generating set.

The base locus of $\mathfrak{F}$ is then the variety cut out by the ideal generated by all the entries of all of the $\boldsymbol{w}^i$. Our function `baseLocusOfMap` returns this ideal. It can be applied either to our new type `RationalMapping` or to a `RingMap` between the homogeneous coordinate rings which represents the rational map.

```
i1 : loadPackage "RationalMaps";
i2 : R = QQ[x,y,z];
```

```
i3 : f = rationalMapping(R, R, {x^2*y, x^2*z, x*y*z})
                                 2    2
o3 = Proj R - - - > Proj R   {x y, x z, x*y*z}
o3 : RationalMapping
i4 : baseLocusOfMap(f)
o4 = ideal (y*z, x*z, x*y)
o4 : Ideal of R
```

If the `SaturateOutput` option is set to `false`, our function `baseLocusOfMap` will not saturate the output.

3. BIRATIONALITY AND INVERSE MAPS. Again, a rational map $\mathfrak{F} : X \subseteq \mathbb{P}^n \dashrightarrow Y \subseteq \mathbb{P}^m$ between projective spaces is defined by $m+1$ forms $\boldsymbol{f} = \{f_0, \dots f_m\}$ of the same degree in the coordinate ring of X, denoted by R. R is a standard graded ring in $n+1$ variables. Here we assumed the varieties are defined over a field k and $\dim R \geq 1$. Our goal is to find a ring theoretic criterion for birationality and, on top of that, to find the inverse of a rational map. To do this, we study the Rees algebra of the ideal $I = (\boldsymbol{f})$ in R. To that end set $R \simeq k[x_0, \dots, x_n] = k[\boldsymbol{X}]/\mathfrak{a}$ with $k[\boldsymbol{X}] = k[X_0, \dots, X_n]$ and $\mathfrak{a}$ a homogeneous ideal. The Rees algebra is defined by the polynomial relations among $\{f_0, \dots f_m\}$ in R. To this end, we consider the polynomial extension $R[\boldsymbol{Y}] = R[Y_0, \dots, Y_m]$. To keep track of the variables by degrees, we set the standard bigrading $\deg(X_i) = (1, 0)$ and $\deg(Y_j) = (0, 1)$. Mapping $Y_j \mapsto f_j t$ yields a presentation $R[\boldsymbol{Y}]/\mathcal{J} \simeq \mathcal{R}_R((\boldsymbol{f}))$, with $\mathcal{J}$ a bihomogeneous *presentation ideal*. $\mathcal{J}$ is a bigraded ideal that depends only on the rational map defined by $\boldsymbol{f}$ and not on this particular representative.

$$\mathcal{J} = \bigoplus_{(p,q)\in\mathbb{N}^2} \mathcal{J}_{(p,q)},$$

where $\mathcal{J}_{(p,q)}$ denotes the k-vector space of forms of bidegree (p, q). Every piece of this ideal contains information about the rational map. For example, $\mathcal{J}_{0,*}$ determines the dimension of the image of the map. For birationality, the following bihomogeneous piece is important:

$$\mathcal{J}_{1,*} := \bigoplus_{q\in\mathbb{N}} \mathcal{J}_{1,q}$$

with $\mathcal{J}_{1,q}$ denoting the bigraded piece of $\mathcal{J}$ spanned by the forms of bidegree $(1, q)$ for all $q \geq 0$. Now, a form of bidegree $(1, *)$ can be written as $\sum_{i=0}^{n} Q_i(\boldsymbol{Y})\, x_i$, for suitable homogeneous $Q_i(\boldsymbol{Y}) \in R[\boldsymbol{Y}]$ of the same degree.

One then goes on to construct a matrix that measures the birationality of the map. The first step is to lift the polynomials $Q_i(\boldsymbol{Y}) \in R[\boldsymbol{Y}]$ into $k[\boldsymbol{X}, \boldsymbol{Y}]$. Since the $\{y_0, \dots, y_m\}$ are indeterminates over R, each pair of such representations of the same form gives a syzygy of $\{x_0, \dots, x_n\}$ with coefficients in k. This is where one must take into account whether $X \subseteq \mathbb{P}^n$ is minimally embedded or not. To measure this, one can easily check the vector space dimension of $\mathfrak{a}_1$, the degree-1 part of $\mathfrak{a}$; if it is zero then $X \subseteq \mathbb{P}^n$ is nondegenerated.

Next, one can pick a minimal set of generators of the ideal $(\mathcal{J}_{1,*})$ consisting of a finite number of forms of bidegree $(1, q)$, for various q's. Let us assume $X \subseteq \mathbb{P}^n$ is nondegenerated. Let $\{P_1, \ldots, P_s\} \subset k[\boldsymbol{X}, \boldsymbol{Y}]$ denote liftings of these biforms and consider the Jacobian matrix of the polynomials $\{P_1, \ldots, P_s\}$ with respect to $\{x_0, \ldots, x_n\}$. This is a matrix with entries in $k[\boldsymbol{Y}]$. Write ψ for the corresponding matrix over $S = k[\boldsymbol{Y}]/\mathfrak{b}$, the coordinate ring of Y. This matrix is called the *weak Jacobian dual matrix* associated to the given set of generators of $(\mathcal{J}_{1,*})$. Note that a weak Jacobian matrix ψ is not uniquely defined due to the lack of uniqueness in the expression of an individual form and to the choice of bihomogeneous generators. However, it is shown in [Doria et al. 2012, Lemma 2.13] that if the weak Jacobian matrix associated to one set of bihomogeneous minimal generators of $(\mathcal{J}_{1,*})$ has rank over S then the weak Jacobian matrix associated to any other set of bihomogeneous minimal generators of $(\mathcal{J}_{1,*})$ has rank over S and the two ranks coincide.

The following criterion is [Doria et al. 2012, Theorem 2.18]. In the package, we consider only the cases where $\boldsymbol{X}$ is irreducible, i.e., R is a domain.

Theorem 3.1. *Let $X \subseteq \mathbb{P}^n$ be nondegenerate. Then $\mathfrak{F}$ is birational onto $\boldsymbol{Y}$ if and only if* $\operatorname{rank}(\psi) = \operatorname{edim}(R) - 1 (= n)$. *Moreover*:

(i) *We get a representative for the inverse of $\mathfrak{F}$ by taking the coordinates of any homogeneous vector of positive degree in the* (*rank* 1) *null space of ψ over S for which these coordinates generate an ideal containing a regular element.*

(ii) *If*, *further*, *R is a domain*, *the representative of $\mathfrak{F}$ in* (i) *can be taken to be the set of the* (*ordered*, *signed*) $(\operatorname{edim}(R)-1)$*-minors of an arbitrary* $(\operatorname{edim}(R) - 1) \times \operatorname{edim}(R)$ *submatrix of ψ of rank* $\operatorname{edim}(R) - 1$.

As expected, the most expensive part of applying this theorem is computing the Rees ideal $\mathcal{J}$. In the package `RationalMaps`, we use `ReesStrategy` to compute the Rees equations. The algorithm is the standard elimination technique. However, we do not use the `ReesAlgebra` [Eisenbud 2018] package, since verifying birationality according to Theorem 3.1 only requires computing a small part of the Rees ideal, namely elements of first-degree 1. This idea is applied in the `SimisStrategy`. More precisely, if the given map $\mathfrak{F}$ is birational, then the Jacobian dual rank will attain its maximum value of $\operatorname{edim}(R) - 1$ after computing the Rees equations up to degree $(1, N)$ for N sufficiently large. This allows us to compute the inverse map. The downside of `SimisStrategy` is that if $\mathfrak{F}$ is not birational, the desired number N cannot be found and the process never terminates. To provide a definitive answer for birationality, we use `HybridStrategy`, which is a hybrid of `ReesStrategy` and `SimisStrategy`. The default strategy is `HybridStrategy`.

`HybridLimit` is an option to switch `SimisStrategy` to `ReesStrategy`, if the computations up to degree $(1, \texttt{HybridLimit})$ do not lead to $\operatorname{rank}(\psi) = \operatorname{edim}(R) - 1$. The default value for `HybridLimit` is 15. The change from `SimisStrategy` to `ReesStrategy` is done in such a way that the generators of the Rees ideal computed in the `SimisStrategy` phase are not lost; the program computes other generators of the Rees ideal while keeping the generators it found before attaining `HybridLimit`.

There is yet another method for computing the Rees ideal called `SaturationStrategy`. In this option, the whole Rees ideal is computed by saturating the defining ideal of the symmetric algebra with respect to a nonzero element in R (we assume R to be a domain). This strategy appears to be slower in some examples, though one might be able to improve this option in the future by stopping the computation of the saturation at a certain step.

Computing inverse maps is the most important function of this package and is done by the function `inverseOfMap` (or by running `RationalMap^-1`). According to Theorem 3.1, there are two ways to compute the inverse of a map: (1) by finding any syzygy of the Jacobian dual matrix, and (2) by finding a submatrix of ψ of rank $\operatorname{edim}(R) - 1$. Each way has its benefits. Method (1) is quite fast in many cases; however, method (2) is very useful if the rank of the Jacobian dual matrix ψ is relatively small compared to the degrees of the entries of ψ. Our function `inverseOfMap` starts by using the second method and later switches to the first method if the second method does not work. The timing of this transition from the first method to the second method is controlled by the option `MinorsLimit`. Setting `MinorsLimit` to zero will mean that no minors are checked and the inverse map is computed just by looking at the syzygies of ψ. If `MinorsLimit` is left as null (the default value), these functions will determine a value using a heuristic that depends on the varieties involved.

To improve the speed of the function `inverseOfMap`, we also have two other options, `AssumeDominant` and `CheckBirational`. If `AssumeDominant` is set to be `true`, then `inverseOfMap` assumes that the map from X to Y is dominant and does not compute the image of the map; this is time-consuming in certain cases as it computes the kernel of a ring map. However, this function goes through a call to `idealOfImageOfMap` which first checks whether the ring map is injective (at least if the target is a polynomial ring) using the method described in [Simis 2003, Proposition 1.1]. Similarly, if `CheckBirational` is set to be `false`, `inverseOfMap` will not check birationality although it still computes the Jacobian dual matrix. The option `QuickRank` is available to many functions. At various points, the rank of a matrix is computed, and sometimes it is faster to compute the rank of an interesting-looking submatrix (using the tools of the package `FastMinors` [Martinova et al. 2020]). Turning `QuickRank` off will make showing that certain maps are birational slower, but will make showing that certain maps are *not* birational faster. There is a certain amount of randomness in the functions of `FastMinors`, and so occasionally rerunning a slow example will result in a massive speedup.

In general, as long as `Verbosity` is `>= 1`, the function will make suggestions as to how to run it more quickly. For example:

```
i1 : loadPackage "RationalMaps";
i2 : Q=QQ[x,y,z,t,u];
i3 : f = map(Q,Q,matrix{{x^5,y*x^4,z*x^4+y^5,t*x^4+z^5,u*x^4+t^5}});
o3 : RingMap
i4 : phi=rationalMapping(f)
                                5   4    5    4    5    4    5    4
o4 = Proj Q - - - > Proj Q   {x , x y, y  + x z, z  + x t, t  + x u}
o4 : RationalMapping
```

```
i5 : time inverseOfMap(phi, CheckBirational=>false, Verbosity => 1);
inverseOfMapSimis: About to find the image of the map.
                   If you know the image, you may want to use the AssumeDominant
                   option if this is slow.
inverseOfMapSimis: About to check rank, if this is very slow,
                   you may want to try turning QuickRank=>false.
inverseOfMapSimis: About to check rank, if this is very slow,
                   you may want to try turning QuickRank=>false.
inverseOfMapSimis: About to check rank, if this is very slow,
                   you may want to try turning QuickRank=>false.
inverseOfMapSimis: About to check rank, if this is very slow,
                   you may want to try turning QuickRank=>false.
inverseOfMapSimis: About to check rank, if this is very slow,
                   you may want to try turning QuickRank=>false.
inverseOfMapSimis: We give up.  Using the previous computations,
                   we compute the whole Groebner basis of the Rees ideal.
                   Increase HybridLimit and rerun to avoid this.
inverseOfMapSimis: Looking for a nonzero minor.
                   If this fails, you may increase the attempts with MinorsLimit => #
inverseOfMapSimis: We found a nonzero minor.
                   -- used 0.189563 seconds

o5 : RationalMapping

i6 : ident = rationalMapping map(Q,Q);

o6 : RationalMapping

i7 : o5*phi == ident

o7 = true
```

Using the `RationalMap^-1` syntax to compute inverses of maps will always suppress such output:

```
i6 : time phi^-1;
    -- used 0.192791 seconds

o6 : RationalMapping

i7 : o4 == o7

o7 = true
```

4. EMBEDDINGS. Our package also checks whether a rational map $\mathfrak{F} : X \to Y$ is a closed embedding. The strategy is quite simple:

(a) We first check whether $\mathfrak{F}$ is regular (by checking if its base locus is empty).

(b) We next invert the map (if possible).

(c) Finally, we check whether the inverse map is also regular.

If all three conditions are met, then the map is a closed embedding and the function returns `true`. Otherwise, `isEmbedding` returns `false`. In the following example which illustrates this, we take a plane quartic, choose a point Q on it, and take the map associated with the divisor $12Q$. This map is an embedding by [Hartshorne 1977, Chapter IV, Corollary 3.2], which we now verify:

```
i1 : needsPackage "Divisor"; --used to quickly define a map

i2 : C = ZZ/101[x,y,z]/(x^4+x^2*y*z+y^4+z^3*x);

i3 : Q = ideal(y,x+z);

o3 : Ideal of C
```

```
i4 : f2 = mapToProjectiveSpace(12*divisor(Q));

                     ZZ
o4 : RingMap C <--- ---[YY ..YY  ]
                     101   1    10

i5 : needsPackage "RationalMaps";

i6 : time isEmbedding(f2)

isEmbedding: About to find the image of the map.  If you know the image,
             you may want to use the AssumeDominant option if this is slow.
inverseOfMapSimis: About to check rank, if this is very slow,
                   you may want to try turning QuickRank=>false
inverseOfMapSimis: rank found, we computed enough of the Groebner basis.
        -- used 0.140107 seconds

o6 = true
```

Notice that `MinorsLimit => 0` by default for `isEmbedding`. This is because the expressions defining the inverse map obtained from an appropriate minor frequently are more complicated than the expressions for the inverse map obtained via the syzygies. Complicated expressions can sometimes slow down the checking of whether the inverse map is regular.

5. FUNCTIONALITY OVERLAP WITH OTHER PACKAGES. We note that our package has some overlaps in functionality with other packages.

The `Parametrization` [Böhm 2010] package focuses mostly on curves, but also includes a function `invertBirationalMap` that has the same functionality as `inverseOfMap`. On the other hand, these two functions were implemented differently so sometimes one function can be substantially faster than the other.

The package `Cremona` [Staglianò 2018] focuses on fast probabilistic computation in general cases and fast deterministic computation for special kinds of maps from projective space. In particular, in `Cremona`,

- `isBirational` gives a probabilistic answer to the question of whether a map between varieties is birational. Furthermore, if the source is projective space, then `degreeOfRationalMap` with `MathMode=>true` gives a deterministic answer that can be faster than what our package provides with `isBirationalMap`;
- `inverseMap` gives a very fast computation of the inverse of a birational map if the source is projective space and the map has maximal linear rank. If this function is passed a map where the domain is not projective space, then it calls a modified, improved version of `invertBirationalMap` originally from `Parametrization`. Even in some cases with maximal linear rank, our `inverseOfMap` function appears to be quite competitive, however.

The package `ReesAlgebra` [Eisenbud 2018] includes a function `jacobianDual` which computes the jacobian dual matrix. We also have a function `jacobianDualMatrix` which computes a weak form of this same matrix.

6. COMMENTS AND COMPARISONS ON FUNCTION SPEEDS. We begin with a comparison using examples with maximal linear rank where `Cremona` excels. These examples were executed using version 5.1 of `Cremona` and version 1.0 of `RationalMaps` running Macaulay2 1.19.1.1 on Ubuntu 20.04.

Indeed, in this example (taken from Cremona's documentation), Cremona is substantially faster.

```
i1 : loadPackage "Cremona"; loadPackage "RationalMaps";

i3 : ringP20=QQ[t_0..t_20];

i4 : phi=map(ringP20,ringP20,{t_10*t_15-t_9*t_16+t_6*t_20,t_10*t_14-t_8*t_16+t_5*t_20,
         t_9*t_14-t_8*t_15+t_4*t_20,t_6*t_14-t_5*t_15+t_4*t_16,
         t_11*t_13-t_16*t_17+t_15*t_18-t_14*t_19+t_12*t_20,
         t_3*t_13-t_10*t_17+t_9*t_18-t_8*t_19+t_7*t_20,
         t_10*t_12-t_2*t_13-t_7*t_16-t_6*t_18+t_5*t_19,
         t_9*t_12-t_1*t_13-t_7*t_15-t_6*t_17+t_4*t_19,
         t_8*t_12-t_0*t_13-t_7*t_14-t_5*t_17+t_4*t_18,t_10*t_11-t_3*t_16+t_2*t_20,
         t_9*t_11-t_3*t_15+t_1*t_20,t_8*t_11-t_3*t_14+t_0*t_20,
         t_7*t_11-t_3*t_12+t_2*t_17-t_1*t_18+t_0*t_19,t_6*t_11-t_2*t_15+t_1*t_16,
         t_5*t_11-t_2*t_14+t_0*t_16,t_4*t_11-t_1*t_14+t_0*t_15,t_6*t_8-t_5*t_9+t_4*t_10,
         t_3*t_6-t_2*t_9+t_1*t_10,t_3*t_5-t_2*t_8+t_0*t_10,
         t_3*t_4-t_1*t_8+t_0*t_9,t_2*t_4-t_1*t_5+t_0*t_6});

o4 : RingMap ringP20 <--- ringP20

i5 : time inverseOfMap(phi, Verbosity=>0);-- Function from "RationalMaps"
        -- used 0.118508 seconds

o5 : RationalMapping

i6 : time inverseMap phi;
        -- used 0.0370978 seconds

o6 : RingMap ringP20 <--- ringP20

i7 : o5 == rationalMapping o6

o7 = true
```

However, sometimes the RationalMaps function is faster, even in examples with maximal linear rank (a good source of examples where different behaviors can be seen can be found in the documentation of Cremona). We now include an example where the map does not have the maximal linear rank.

```
i1 : loadPackage "Cremona"; loadPackage "RationalMaps";

i3 : Q=QQ[x,y,z,t,u];

i4 : phi=map(Q,Q,matrix{{x^5,y*x^4,z*x^4+y^5,t*x^4+z^5,u*x^4+t^5}});

o4 : RingMap Q <--- Q

i5 : (time g = inverseOfMap(phi, Verbosity=>0));
     -- used 0.233111 seconds

i6 : (time f = inverseOfMap(phi, Verbosity=>0, MinorsLimit=>0));
     -- used 60.1969 seconds

i7 : (time h = inverseMap(phi)); -- Function from "Cremona"
     -- used 49.2842 seconds

o7 : RingMap Q <--- Q

i8 : f == rationalMapping h

o8 = true

i9 : g == rationalMapping h

o9 = true
```

In the previous example, setting MinorsLimit=>0 makes inverseOfMap much slower – approximately the same speed as the corresponding command from Cremona. The takeaway for the user should be that changing the options Strategy, HybridLimit, MinorSize, and QuickRank, can make a large difference in performance.

We conclude with discussions of the limits of this package. A work attributed to O. Gabber [Bass et al. 1982, Theorem 1.5] shows that if $f : \mathbb{P}^n \to \mathbb{P}^n$ is defined by forms of degree d, then its inverse can be defined by forms of degree d^{n-1}. This bound is sharp, as the map

$$(x_0^d : x_1x_0^{d-1} : x_2x_0^{d-1} - x_1^d : \cdots : x_nx_0^{d-1} - x_{n-1}^d)$$

has inverse given by forms of degree d^{n-1}; see [Hassanzadeh and Simis 2017]. Thus we might expect that this family of maps would be good to explore to see the limits of `RationalMaps`. We ran these examples with the following code:

```
R = ZZ/101[x_0..x_n];
L = {x_0^d, x_1*x_0^(d-1)} | toList(apply(2..n, i -> (x_i*x_0^(d-1) + x_(i-1)^d)));
psi = map(R, R, L);
time inv = inverseOfMap(psi, AssumeDominant=>true, CheckBirational=>false, Verbosity=>0);
```

For $n = 3$ (we are working on $\mathbb{P}^3$), we include a table showing the computation time, in seconds, to find the inverse map for various values of d. The degrees are those we would expect in this example (when $d = 100$, the degree of the forms in the inverse is 10000). Note that `Cremona` has very similar performance for these examples in $\mathbb{P}^3$ ($n = 3$), but seems substantially slower than `RationalMaps` as we increase the dimension:

d	5	10	20	40	60	80	100
seconds	0.0925	0.0958	0.1402	1.0667	7.2652	37.4577	135.915

However, as the size of projective space increases, this becomes much slower. Here is a table for $n = 4$:

d	5	8	10	11	12	13	14	15
seconds	0.1523	1.3115	7.4682	14.9912	28.8554	57.1229	120.778	217.706

We conclude with a table for $n = 5$:

d	3	4	5	6
seconds	0.2619	4.8770	134.424	2713.56

Note the $d = 6$ case took more than 45 minutes.

Finally, Zhuang He and Lei Yang, working under the direction of Ana-Maria Castravet, communicated to us that they used `RationalMaps` to help understand and compute the inverse of a rational map from $\mathbb{P}^3$ to $\mathbb{P}^3$; see [He and Lei ≥ 2022]. Quoting Zhuang He, this rational map is "induced by a degree 13 linear system with the base locus at 6 very general points in P3 and 9 lines through them". From a computational perspective, this map was given by 4 degree 13 forms, with 485, 467, 467, and 467 terms respectively. Computing the inverse of this map took several hours, but it was successful.

ACKNOWLEDGEMENTS. Much of the work on this package was completed during the Macaulay2 workshop held at the University of Utah in May 2016. We especially thank David Eisenbud, Aron Simis, Greg Smith, Giovanni Staglianò, and Mike Stillman for valuable conversations. We also thank the referees for numerous helpful comments on both the package and the paper.

SUPPLEMENT. The online supplement contains version 1.0 of `RationalMaps`.

REFERENCES.

[Bass et al. 1982] H. Bass, E. H. Connell, and D. Wright, "The Jacobian conjecture: reduction of degree and formal expansion of the inverse", *Bull. Amer. Math. Soc.* (*N.S.*) **7**:2 (1982), 287–330. MR Zbl

[Böhm 2010] J. Böhm, *Parametrisierung rationaler Kurven*, Ph.D. thesis, 2010, https://www.mathematik.uni-kl.de/~boehm/diplom%20janko%20boehm.pdf.

[Doria et al. 2012] A. V. Doria, S. H. Hassanzadeh, and A. Simis, "A characteristic-free criterion of birationality", *Adv. Math.* **230**:1 (2012), 390–413. MR Zbl

[Eisenbud 2018] D. Eisenbud, "The ReesAlgebra package in Macaulay2", *J. Softw. Algebra Geom.* **8** (2018), 49–60. MR Zbl

[Hartshorne 1977] R. Hartshorne, *Algebraic geometry*, Graduate Texts in Mathematics **52**, Springer, 1977. MR Zbl

[Hassanzadeh and Simis 2017] S. H. Hassanzadeh and A. Simis, "Bounds on degrees of birational maps with arithmetically Cohen–Macaulay graphs", *J. Algebra* **478** (2017), 220–236. MR Zbl

[He and Lei ≥ 2022] Z. He and Y. Lei, "The Mori dream space property of blow-ups of projective spaces at points and lines", preprint.

[Hulek et al. 1992] K. Hulek, S. Katz, and F.-O. Schreyer, "Cremona transformations and syzygies", *Math. Z.* **209**:3 (1992), 419–443. MR Zbl

[Martinova et al. 2020] B. Martinova, M. Robinson, K. Schwede, and Y. Yao, "FastMinors package for Macaulay2", 2020. arXiv 2002.05758

[Russo and Simis 2001] F. Russo and A. Simis, "On birational maps and Jacobian matrices", *Compositio Math.* **126**:3 (2001), 335–358. MR Zbl

[Semple and Tyrrell 1969] J. G. Semple and J. A. Tyrrell, "The Cremona transformation of S_6 by quadrics through a normal elliptic septimic scroll ${}^1R^7$", *Mathematika* **16** (1969), 89–97. MR Zbl

[Simis 2003] A. Simis, "Two differential themes in characteristic zero", pp. 195–204 in *Topics in algebraic and noncommutative geometry* (Luminy, MD, 2001), Contemp. Math. **324**, Amer. Math. Soc., Providence, RI, 2003. MR Zbl

[Simis 2004] A. Simis, "Cremona transformations and some related algebras", *J. Algebra* **280**:1 (2004), 162–179. MR Zbl

[Staglianò 2018] G. Staglianò, "A Macaulay2 package for computations with rational maps", *J. Softw. Algebra Geom.* **8** (2018), 61–70. MR Zbl

RECEIVED: 29 Aug 2016 REVISED: 15 Mar 2022 ACCEPTED: 17 Jul 2022

C. J. BOTT:

cbott2@math.tamu.edu
Department of Mathematics, Texas A&M University, College Station, TX, United States

SEYED HAMID HASSANZADEH:

hamid@im.ufrj.br
Instituto de Matemática, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil

KARL SCHWEDE:

schwede@math.utah.edu
Department of Mathematics, The University of Utah, Salt Lake City, UT, United States

DANIEL SMOLKIN:

smolkin@math.utah.edu
Department of Mathematics, The University of Michigan, Ann Arbor, MI, United States

JSAG **12** (2022), 27–32
https://doi.org/10.2140/jsag.2022.12.27

The Journal of Software for
Algebra and Geometry

Primary decomposition of squarefree pseudomonomial ideals

ALAN VELIZ-CUBA

ABSTRACT: A *squarefree pseudomonomial* in $K[x_1, x_2, \ldots, x_n]$ is a polynomial of the form $z_1 z_2 \cdots z_r$ such that $z_j \in \{x_{i_j}, x_{i_j} - 1\}$ for some $i_j \in \{1, \ldots, n\}$ and the i_j's are distinct. A *squarefree pseudomonomial ideal* is an ideal generated by squarefree pseudomonomials. Here we present the package *PseudomonomialPrimaryDecomposition* for the computation of the primary decomposition of squarefree pseudomonomial ideals. Internally, the package performs computation using bitwise logical operations on integers.

1. INTRODUCTION. A squarefree pseudomonomial of a ring $K[x_1, x_2, \ldots, x_n]$ is a polynomial of the form $z_1 z_2 \cdots z_r$ such that $z_j \in \{x_{i_j}, x_{i_j} - 1\}$ for some $i_j \in \{1, \ldots, n\}$ and the i_j's are distinct. A squarefree pseudomonomial ideal is an ideal that is generated by squarefree pseudomonomials. Squarefree pseudomonomial ideals appear in the study of problems such as reverse engineering of finite dynamical systems and in the study of neural coding, where the primary decomposition of these ideals contains important combinatorial information [Jarrah et al. 2007; Veliz-Cuba 2012; Curto et al. 2013].

In this manuscript we present the theory, algorithm, and implementation of the primary decomposition computation for squarefree pseudomonomial ideals. We use the structure of squarefree pseudomonomial ideals to design an efficient algorithm to compute their primary decomposition, similar to existing algorithms for monomial ideals.

2. THEORY. The theory behind the algorithm is based on the work presented in [Curto et al. 2013] following similar properties of squarefree monomial ideals [Miller and Sturmfels 2005; Sturmfels 2002]. Here we summarize those results. In this section K denotes a field.

Lemma 2.1. *Consider a squarefree pseudomonomial ideal $J \subset K[x_1, \ldots, x_n]$. If f is a squarefree pseudomonomial such that $f \in \langle J, z\rangle$, where $z = x_t$ or $z = 1 - x_t$ for some $t \in \{1, \ldots, n\}$, then $f \in \langle z\rangle$ or $(1 - z) f \in J$.*

Proof. Adapted from [Curto et al. 2013]. Since f is a squarefree pseudomonomial, we have that up to a sign $f = z_1 z_2 \cdots z_r$ such that $z_j \in \{x_{i_j}, 1 - x_{i_j}\}$ and the i_j's are distinct. Since J is a squarefree pseudomonomial ideal, we can write

$$J = \langle z g_1, \ldots, z g_k, (1 - z) f_1, \ldots, (1 - z) f_l, h_1, \ldots, h_m\rangle,$$

MSC2010: primary 13P05, 14Q99, 68W30; secondary 92B05, 92B20.
Keywords: primary decomposition, minimal primes, monomials, pseudomonomials.
PseudomonomialPrimaryDecomposition version 0.3

where g_j, f_j, and h_j are squarefree pseudomonomials that contain no factor z or $1-z$. Thus, f is of the form

$$f = \sum_{j=1}^{k} u_j z g_j + (1-z)\sum_{j=1}^{l} v_j f_j + \sum_{j=1}^{m} w_j h_j + yz,$$

for some $u_j, v_j, w_j, y \in K[x_1, \ldots, x_n]$. Setting $z = 0$ (i.e., $x_t = 0$ if $z = x_t$ and $x_t = 1$ if $z = 1 - x_t$) and multiplying by $1 - z$ yields

$$(1-z)f|_{z=0} = (1-z)\sum_{j=1}^{l} v_j|_{z=0} f_j + (1-z)\sum_{j=1}^{m} w_j|_{z=0} h_j,$$

where "$|_{z=0}$" means evaluated at $z = 0$. Note that since $(1-z)f_j, h_j \in J$, we have that $(1-z)f|_{z=0} \in J$. Now we have 3 cases.

If z is a factor of f, then $f \in \langle z \rangle$.

If $1-z$ is a factor of f, say $f = (1-z)z_2 \cdots z_r$, then $f = (1-z)f|_{z=0} \in J$.

If neither z or $1-z$ is a factor of f, then $(1-z)f = (1-z)f|_{z=0} \in J$.

In all cases we obtain $f \in \langle z \rangle$ or $(1-z)f \in J$. □

Proposition 2.2. *Consider a squarefree pseudomonomial ideal $J \subseteq K[x_1, \ldots, x_n]$ and let $z_1 \cdots z_r$ be a squarefree pseudomonomial. Then, $\left\langle J, \prod_{j=1}^{r} z_i \right\rangle = \bigcap_{i=1}^{r} \langle J, z_i \rangle$.*

Proof. Adapted from [Curto et al. 2013]. Note that we can consider $z_j \in \{x_{i_j}, 1 - x_{i_j}\}$.

It is clear that $\left\langle J, \prod_{i=1}^{r} z_i \right\rangle \subseteq \bigcap_{i=1}^{r} \langle J, z_i \rangle$.

We now consider $f \in \bigcap_{i=1}^{r} \langle J, z_i \rangle$. We have three cases.

If $f \in \langle z_i \rangle$ for all i, then $f \in \left\langle \prod_{i=1}^{r} z_i \right\rangle \subseteq \left\langle J, \prod_{i=1}^{r} z_i \right\rangle$.

If $f \notin \langle z_i \rangle$ for some i's, then without loss of generality we consider $f \notin \langle z_i \rangle$ for $i = 1, \ldots, m$ and $f \in \langle z_i \rangle$ for $i = m+1, \ldots, r$. We can write

$$f = (1-z_1)f + z_1(1-z_2)f + \cdots + z_1 \cdots z_{m-1}(1-z_m)f + z_1 \cdots z_m f.$$

From Lemma 2.1, $(1-z_j)f \in J$ for $j = 1, \ldots, m$, since $f \in \langle J, z_i \rangle$ and $f \notin \langle z_i \rangle$. Then, the first m terms are in J. Also, since $f \in \langle z_i \rangle$ for $i = m+1, \ldots, r$, it follows that $f \in \langle z_{m+1} \cdots z_r \rangle$ and then $z_1 \cdots z_m f \in \langle z_1 \cdots z_r \rangle$. Thus $f \in \left\langle J, \prod_{i=1}^{r} z_i \right\rangle$. □

Proposition 2.2 provides a concrete way to write an ideal as an intersection of simpler ideals. Using this result recursively, we will obtain at the end an intersection of ideals of the form $\langle x_{i_1} - a_1, \ldots, x_{i_k} - a_k \rangle$, where $a_j \in \{0, 1\}$, which provide the primary decomposition. We remark that Proposition 2.2 is similar to Lemma 2.1 in [Sturmfels 2002, monomial ideals chapter]. Also, since each ideal in the intersection is prime, the ideal is radical, and so the primary decomposition is the set of minimal primes of the ideal. Before describing the algorithm we show two examples.

Example 2.3. Consider $I = \langle x_1, x_2(1-x_3), (1-x_2)x_4 \rangle$. Using Proposition 2.2 we obtain

$$\begin{aligned} I &= \langle x_1, x_2(1-x_3), 1-x_2 \rangle \cap \langle x_1, x_2(1-x_3), x_4 \rangle \\ &= \langle x_1, x_2, 1-x_2 \rangle \cap \langle x_1, 1-x_3, 1-x_2 \rangle \cap \langle x_1, x_2, x_4 \rangle \cap \langle x_1, 1-x_3, x_4 \rangle. \end{aligned}$$

Since the first ideal in the last equality contains 1, it is not proper, thus we obtain

$$I = \langle x_1, 1 - x_3, 1 - x_2\rangle \cap \langle x_1, x_2, x_4\rangle \cap \langle x_1, 1 - x_3, x_4\rangle.$$

Example 2.4. Consider $I = \langle x_1(1 - x_2), (1 - x_1)x_2, x_1x_2\rangle$. Using Proposition 2.2 we obtain

$$\begin{aligned} I &= \langle x_1(1 - x_2), (1 - x_1)x_2, x_1\rangle \cap \langle x_1(1 - x_2), (1 - x_1)x_2, x_2\rangle \\ &= \langle (1 - x_1)x_2, x_1\rangle \cap \langle x_1(1 - x_2), x_2\rangle \\ &= \langle 1 - x_1, x_1\rangle \cap \langle x_2, x_1\rangle \cap \langle x_1, x_2\rangle \cap \langle 1 - x_2, x_2\rangle = \langle x_1, x_2\rangle. \end{aligned}$$

3. ALGORITHM. The algorithm uses Proposition 2.2 recursively as well as some simplifications seen in Examples 2.3 and 2.4.

Algorithm for the computation of the primary decomposition of squarefree pseudomonomial ideals. The algorithm is adapted from [Curto et al. 2013].

Input: A squarefree pseudomonomial ideal $J \subseteq K[x_1, \ldots, x_n]$.
Output: Primary decomposition of J, a list $\mathcal{P}$ of primary ideals such that $J = \cap_{I\in\mathcal{P}} I$.

Step 1. Set $\mathcal{P} = \varnothing$ and $\mathcal{D} = \{J\}$. Remove redundant generators.

Step 2. For each ideal $I \in \mathcal{D}$ define $\mathcal{D}_I = \{\langle I, z_1\rangle, \ldots, \langle I, z_m\rangle\}$, where $z_1z_2 \cdots z_m$ is one of the generators of I of degree greater than 1 and $z_j \in \{x_{i_j}, x_{i_j} - 1\}$.

Step 3. For each $\langle I, z\rangle$ found in Step 2, remove redundant generators. Also, remove ideals that have 1 as a generator.

Step 4. Define $\mathcal{T} = \bigcup_{I\in\mathcal{D}} \mathcal{D}_I$ and set $\mathcal{P} := \mathcal{P} \cup \{I \in \mathcal{T} : I$ has linear generators only$\}$ and $\mathcal{D} := \{I \in \mathcal{T} : I$ has nonlinear generators$\}$.

Step 5. Repeat Steps 2–4 until $\mathcal{D} = \varnothing$.

Step 6. Remove redundant ideals of $\mathcal{P}$.

The algorithm is guaranteed to give the primary decomposition because:

- Proposition 2.2 guarantees that $I = \bigcap_{K\in\mathcal{D}_I} K$ in Step 2.
- At every step we have $J = \bigcap_{I\in\mathcal{P}} I \cap \bigcap_{I\in\mathcal{D}} I$.
- Step 2 reduces the number of nonlinear generators, so the algorithm will terminate after a finite number of iterations.
- Step 6 results in $J = \cap_{I\in\mathcal{P}} I$ such that each ideal in $\mathcal{P}$ has linear generators only and hence is primary.

4. IMPLEMENTATION AND EXAMPLES. We implemented the algorithm described in Section 3 using bitwise logical operations on integers in [Macaulay2]. A squarefree pseudomonomial

$$P = x_{i_1} \cdots x_{i_r}(x_{k_1} - 1) \cdots (x_{k_m} - 1)$$

is encoded as a list of two integers $B_P := \left\{\sum_{j=1}^{r} 2^{i_j}, \sum_{j=1}^{m} 2^{k_j}\right\}$. Note that the integer $\sum_{j=1}^{r} 2^{i_j}$ represents

the binary string with 1's at positions i_j and 0's elsewhere, and the integer $\sum_{j=1}^{m} 2^{k_j}$ represents the binary string with 1's at positions k_j and 0's elsewhere. A similar approach was used to compute the Gröbner basis for Boolean polynomials [Hinkelmann and Arnold 2010]. The advantage of using bitwise representation is that comparison between polynomials can be done quickly. For example, consider a polynomial P with bitwise representation $B_P = \{a, b\}$ and a polynomial Q with bitwise representation $B_Q = \{c, d\}$. Then, to check whether P divides Q it is enough to check whether $a\&c = a$ and $b\&d = b$, where "&" is the bitwise AND operator.

The package *PrimaryDecompositionPseudomonomial* uses bitwise logical operations on integers. However, the input and output of the package are in polynomial form, so the user does not need to manipulate polynomials in bitwise form. To ensure a correct implementation of the package, we computed the primary decomposition of over 10,000,000 squarefree pseudomonomial ideals and compared our results with those of the current implementation of `primaryDecomposition` in Macaulay2. In all cases, our results were correct. The package *PrimaryDecompositionPseudomonomial* exports two functions: `primaryDecompositionPseudomonomial` and `isSquarefreePseudomonomialIdeal`; the first computes the primary decomposition and the second determines whether an ideal is a squarefree pseudomonomial ideal. Below are some examples in Macaulay2.

Example 4.1.

```
i1 : needs "PseudomonomialPrimaryDecomposition.m2"
i2 : R = QQ[x1,x2,x3,x4];
i3 : I = ideal(x1,x2*(1-x3),(1-x2)*x4);
o3 : Ideal of R
i4 : primaryDecomposition I
o4 = {ideal (x4, x3 - 1, x1), ideal (x3 - 1, x2 - 1, x1), ideal (x4, x2, x1)}
o4 : List
i5 : primaryDecompositionPseudomonomial I
o5 = {ideal (x1, x4, x2), ideal (x1, x3 - 1, x2 - 1), ideal (x1, x3 - 1, x4)}
o5 : List
```

Note that the order of ideals and generators used by `primaryDecompositionPseudomonomial` and `primaryDecomposition` may be different.

Example 4.2.

```
i1 : needs "PseudomonomialPrimaryDecomposition.m2"
i2 : R = ZZ/3[x1,x2];
i3 : I = ideal(x1*(1-x2),(1-x1)*x2, x1*x2);
o3 : Ideal of R
i4 : primaryDecomposition I
o4 = {ideal (x2, x1, x1*x2)}
o4 : List
i5 : primaryDecompositionPseudomonomial I
o5 = {ideal (x2, x1)}
o5 : List
```

Note that `primaryDecompositionPseudomonomial` finds a minimal set of generators for each ideal.

Example 4.3.

```
i1 : needs "PseudomonomialPrimaryDecomposition.m2"
i2 : R = ZZ/3[x1,x2];
i3 : I = ideal(x1*(1-x2),(1-x1)*x2, x1*x2,(x1-1)*(x2-1));
o3 : Ideal of R
i4 : primaryDecomposition I
o4 = {}
o4 : List
i5 : primaryDecompositionPseudomonomial I
o5 = {}
o5 : List
```

The ideal is not proper, so both functions return the empty set.

Example 4.4.

```
i1 : needs "PseudomonomialPrimaryDecomposition.m2"
i2 : R = ZZ/3[x1,x2];
i3 : I = ideal(x1*(1-x1),(1-x1)*x2, x1*x2);
o3 : Ideal of R
i4 : primaryDecompositionPseudomonomial I
stdio:4:1:(3): error: Not a squarefree pseudomonomial ideal.
i5 : isSquarefreePseudomonomialIdeal I
o5 = false
```

The algorithm works only with squarefree pseudomonomial ideals.

Example 4.5.

```
i1 : needs "PseudomonomialPrimaryDecomposition.m2"
i2 : R = ZZ/2[x1,x2,x3,x4,x5,x6,x7,x8,x9];
i3 : I = ideal( (x8+1)*(x7+1)*(x4+1)*x3, (x7+1)*(x6+1)*(x4+1)*x3*(x1+1),
     (x8+1)*(x4+1)*x2, (x7+1)*x3*(x1+1), x5*(x4+1)*x3*x1, x5*(x2+1)*x1,
     x8*(x6+1)*x5*(x4+1)*x3, x8*(x6+1)*x5*(x2+1), x9*(x7+1)*x5,
     x9*(x8+1)*x5*(x4+1)*(x3+1)*x2*x1, x6*(x4+1)*(x3+1)*x2*x1, x8*(x7+1)*x6,
     (x8+1)*x5*(x4+1), (x7+1)*x5*x3*(x2+1)*(x1+1), x7*x5*(x4+1)*x2*x1,
     x8*x5*x3, x1*(x2+1)*(x3+1)*(x4+1)*(x5+1)*(x6+1)*x7*x8*(x9+1),
     x7*x5*(x3+1)*x1, x8*x5*x4*(x2+1) );
o3 : Ideal of R
i4 : timing(primaryDecompositionPseudomonomial I;)
o4 = -- .415975 seconds
o4 : Time
i5 : timing(primaryDecomposition I;)
o5 = -- 42.7676 seconds
o5 : Time
```

In some cases `primaryDecompositionPseudomonomial` can be orders of magnitude faster than the standard algorithm.

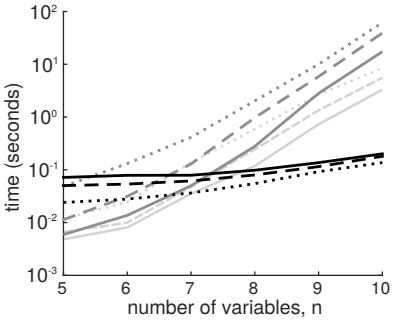

Figure 1. Timings of the function `primaryDecompositionPseudomonomial` (black) and `primaryDecomposition` (dark gray). Timings for `minimalPrimes` (light gray) are shown as well, since the ideals are known to be radical. The data was generated by calculating the average time of the computation of the primary decomposition of squarefree pseudomonomial ideals $I = \langle p_1, p_2, \ldots, p_m \rangle$ of $\mathbb{Z}_2[x_1, \ldots, x_n]$, where p_i is a randomly generated squarefree pseudomonomial. The curves show the timings as a function of n for $m = 10$ (solid), $m = 20$ (dashed), and $m = 30$ (dotted). The sample size for each n and m was 1000. Some computations were done on the Ohio Supercomputer Center.

Figure 1 shows a comparison between `primaryDecompositionPseudomonomial` and the current implementation of `primaryDecomposition`. For a small number of variables the current implementation is faster on squarefree pseudomonomial ideals than the implementation we describe in this paper. However, as the number of variables increases, our implementation is faster for that class of ideals.

SUPPLEMENT. The online supplement contains version 0.3 of *PseudomonomialPrimaryDecomposition*.

REFERENCES.

[Curto et al. 2013] C. Curto, V. Itskov, A. Veliz-Cuba, and N. Youngs, "The neural ring: an algebraic tool for analyzing the intrinsic structure of neural codes", *Bull. Math. Biol.* **75**:9 (2013), 1571–1611. MR Zbl

[Hinkelmann and Arnold 2010] F. Hinkelmann and E. Arnold, "Fast Gröbner basis computation for Boolean polynomials", 2010. arXiv 1010.2669

[Jarrah et al. 2007] A. S. Jarrah, R. Laubenbacher, B. Stigler, and M. Stillman, "Reverse-engineering of polynomial dynamical systems", *Adv. in Appl. Math.* **39**:4 (2007), 477–489. MR

[Macaulay2] D. R. Grayson and M. E. Stillman, "Macaulay2: a software system for research in algebraic geometry", available at http://www.math.uiuc.edu/Macaulay2.

[Miller and Sturmfels 2005] E. Miller and B. Sturmfels, *Combinatorial commutative algebra*, vol. 227, Springer, 2005. MR Zbl

[Sturmfels 2002] B. Sturmfels, "Ideals, varieties and Macaulay 2", pp. 3–15 in *Computations in algebraic geometry with Macaulay* 2, edited by D. Eisenbud et al., Algorithms Comput. Math. **8**, Springer, 2002. MR Zbl

[Veliz-Cuba 2012] A. Veliz-Cuba, "An algebraic approach to reverse engineering finite dynamical systems arising from biology", *SIAM J. Appl. Dyn. Syst.* **11**:1 (2012), 31–48. MR Zbl

RECEIVED: 4 Apr 2016 REVISED: 5 Jan 2021 ACCEPTED: 18 Jul 2022

ALAN VELIZ-CUBA:
avelizcuba1@udayton.edu
Department of Mathematics, University of Dayton, Dayton, OH, United States

JSAG **12** (2022), 33–41
https://doi.org/10.2140/jsag.2022.12.33

The Journal of Software for
Algebra and Geometry

Noetherian operators in Macaulay2

JUSTIN CHEN, YAIRON CID-RUIZ, MARC HÄRKÖNEN,
ROBERT KRONE AND ANTON LEYKIN

ABSTRACT: A primary module over a polynomial ring can be described by an algebraic variety and a finite set of Noetherian operators, which are vectors of differential operators with polynomial coefficients. We implement both symbolic and numerical algorithms to produce such a description in various scenarios, as well as routines for studying affine schemes and coherent sheaves through the prism of Noetherian operators and Macaulay dual spaces.

1. INTRODUCTION. The idea of describing ideals in polynomial rings via systems of differential operators has been brewing since the beginning of the twentieth century. Macaulay [1916] brought forth the notion of an *inverse system*, a system of differential conditions that describes a *modular system* (a system of polynomials, or a polynomial ideal, in the modern language).

It was apparent to the contemporaries of Macaulay that a finite number of differential conditions should suffice to describe a 0-dimensional affine or projective scheme. Gröbner [1938] derived explicit characterizations for ideals that are primary to a rational maximal ideal [Gröbner 1970, p. 174–178]. Moreover, he suggested that the same program can be carried out for any primary ideal [Gröbner 1952, §1].

Despite this early algebraic interest, a complete description of primary ideals and modules in terms of differential operators was first obtained by analysts in the fundamental principle of Ehrenpreis and Palamodov [Ehrenpreis 1970; Palamodov 1970]. At the core of this fundamental principle, one has the following theorem by Palamodov:

Theorem 1.1 (Palamodov). *Let R be a polynomial ring $R = \mathbb{C}[x_1, \ldots, x_n]$ over the complex numbers, $P \subseteq R$ be a prime ideal, and $Q \subseteq R^k$ be a P-primary R-module. Then there exist vectors of differential operators $A_1, \ldots, A_m \in R\langle \partial_{x_1}, \ldots, \partial_{x_n}\rangle^k$ such that $Q = \{f \in R^k \mid A_i \bullet f \in P \text{ for } 1 \leq i \leq m\}$.*

Following the terminology of Palamodov, the differential operators $A_1, \ldots, A_m$ are commonly called *Noetherian operators for the P-primary submodule $Q \subseteq R^k$*. Subsequent algebraic and computational approaches to characterize primary ideals and modules with the use of differential operators have been given in [Brumfiel 1978; Oberst 1999; Damiano et al. 2007; Cid-Ruiz 2021; Chen et al. 2022; Cid-Ruiz et al. 2021; Chen and Cid-Ruiz 2022; Ait El Manssour et al. 2021].

The research of Chen, Härkönen, and Leykin was partially supported by NSF DMS-2001267.

MSC2020: primary 14-04; secondary 13N05, 14Q15, 65D05, 65L80.

Keywords: commutative algebra, computational algebraic geometry, differential algebra, Noetherian operators, dual spaces, numerical algebraic geometry.

`NoetherianOperators` version 2.2.1

The purpose of this note is to present a Macaulay2 software package that implements the algorithms for Noetherian operators introduced in [Chen et al. 2022; Cid-Ruiz et al. 2021; Chen and Cid-Ruiz 2022; Ait El Manssour et al. 2021] as well as the algorithms for (Macaulay) dual spaces addressed in [Krone and Leykin 2017a; 2017b; Krone 2013]. While some of these algorithms rely on exact symbolic computation, the others employ numerical approximations using paradigms of numerical algebraic geometry.

2. COMPUTING NOETHERIAN OPERATORS FROM MODULES. The main method within the package `NoetherianOperators` is `noetherianOperators`, which contains implementations of symbolic algorithms. To represent Noetherian operators, this package introduces a new type called `DiffOp`, representing elements in $R\langle\partial_{x_1},\dots,\partial_{x_n}\rangle$. The type `DiffOp` is a wrapper around `Vector`, with the added feature that instances of `DiffOp` in $R\langle\partial_{x_1},\dots,\partial_{x_n}\rangle^k$ operate on elements of R^k.

The symbolic backbone of the default Noetherian operator computation routine rests on [Chen and Cid-Ruiz 2022, Theorem 3.2], which can be seen as a "representation theorem" that parametrizes primary modules via three closely related objects (points in the punctual Quot scheme, differentially closed vector spaces, and bisubmodules of the Weyl–Noether module).

For a prime ideal P of codimension c, let $\mathbb{F}$ be the field of fractions of the integral domain R/P. Up to a linear change of coordinates, we may (and do) assume that $\{x_{c+1},\dots,x_n\}$ is a maximal independent set of variables modulo P, to simplify notation.

We now recall the steps of [Chen and Cid-Ruiz 2022, Algorithm 4.1]. The main idea of this algorithm is to reduce the study of arbitrary P-primary modules over R to a zero-dimensional setting over the function field $\mathbb{F}$. This reduction is made by parametrizing P-primary modules with the punctual Quot scheme $\mathrm{Hilb}^m(\mathbb{F}[\![y_1,\dots,y_c]\!]^k)$. This is a quasiprojective scheme over the function field $\mathbb{F}$. Its classical points are $\mathbb{F}[\![y_1,\dots,y_c]\!]$-submodules $V\subseteq\mathbb{F}[\![y_1,\dots,y_c]\!]^k$ of colength m. If $k=1$, the punctual Quot scheme is known as the *punctual Hilbert scheme*. For more details regarding punctual Hilbert and Quot schemes the reader is referred to [Iarrobino 1972; Baranovsky 2000]. We define the inclusion map

$$\gamma : R \hookrightarrow \mathbb{F}[y_1,\dots,y_c] \qquad \begin{aligned} x_i &\mapsto y_i+\overline{x_i}, && \text{for } 1\le i\le c,\\ x_j &\mapsto \overline{x_j}, && \text{for } c+1\le j\le n,\end{aligned}$$

where $\overline{x_i}$ denotes the class of x_i in $\mathbb{F}$, $1\le i\le n$. With this, we can give the explicit bijective correspondence

$$\begin{Bmatrix} P\text{-primary } R\text{-submodules of } R^k \\ \text{with multiplicity } m \text{ over } P\end{Bmatrix} \longleftrightarrow \big\{\text{points in } \mathrm{Quot}^m(\mathbb{F}[\![y_1,\dots,y_c]\!]^k)\big\},$$
$$Q \longrightarrow V=\gamma(Q)+\langle y_1,\dots,y_c\rangle^m\mathbb{F}[\![y_1,\dots,y_c]\!]^k,$$
$$Q=\gamma^{-1}(V) \longleftarrow V.$$

After computing the point $V\subseteq\mathrm{Quot}^m(\mathbb{F}[\![y_1,\dots,y_c]\!]^k)$ corresponding to a P-primary ideal Q of multiplicity m over P, the inverse system $V^\perp$ of V is computed. Lastly, an $\mathbb{F}$-basis of $V^\perp$ is lifted to a set of Noetherian operators for the module Q.

While this is the default strategy used to compute Noetherian operators, it can also be explicitly called by specifying `Strategy => "PunctualQuot"`, as shown below:

```
i1 : needsPackage "NoetherianOperators";
i2 : S = QQ[x_1, x_2, x_3];
i3 : Q = image matrix{{x_1, x_2^2, 0}, {x_3, x_3^2, x_2^2-x_1*x_3}}
o3 = image | x_1 x_2^2 0            |
           | x_3 x_3^2 x_2^2-x_1x_3 |
                                     2
o3 : S-module, submodote of S
i4 : associatedPrimes comodule Q
                 2
o4 = {ideal(x  - x x )}
             2    1 3
o4 : List
i5 : noetherianOperators(Q, Strategy => "PunctualQuot")
o5 = {| -x_3 |}
      |  x_1 |
o5 : List
```

Nonprimary modules can be studied via *differential primary decompositions* [Chen and Cid-Ruiz 2022, Algorithm 4.9], implemented in `differentialPrimaryDecomposition`. The output is a list of pairs whose first entry is an associated prime and the second entry is a list of Noetherian operators corresponding to that primary component. Our implementation returns the minimal number of Noetherian operators required to describe a module, namely the *arithmetic multiplicity* of the module (`amult`) [Cid-Ruiz and Sturmfels 2021, Theorem 4.6]:

```
i6 : Q = image matrix{{x_1^2, x_2*x_1, 0},{x_3^2,0,x_1^2}};
i7 : amult Q
o7 = 8
i8 : netList differentialPrimaryDecomposition Q
     +--------------+-----------------------------------------------+
o8 = |ideal x       |{| 1 |}                                        |
     |       1      | | 0 |                                        |
     +--------------+-----------------------------------------------+
     |ideal (x , x )|{| dx_1 |, | x_3^2dx_1^2 |, | x_3^2dx_1^3 |}|
     |        2   1 | |   0  |  |      -2     |  |    -6dx_1   | |
     +--------------+-----------------------------------------------+
     |ideal (x , x )|{| 0 |, |   0  |, |   0  |, |     0    |}   |
     |        3   1 | | 1 |  | dx_1 |  | dx_3 |  | dx_1dx_3 |    |
     +--------------+-----------------------------------------------+
```

One application of Noetherian operators is in solving systems of PDE with constant coefficients. The fundamental principle of Ehrenpreis and Palamodov asserts that every distributional solution to a system of PDE can be represented as an integral of exponential-polynomial solutions with respect to suitable measures supported on algebraic varieties. The exponential-polynomial solutions correspond to Noetherian operators, and the varieties correspond to the varieties of the associated primes of the module in question. See [Ait El Manssour et al. 2021] and the function `solvePDE` for more details on this viewpoint.

3. COMPUTING NOETHERIAN OPERATORS FROM IDEALS. In this section we discuss additional algorithms that can be used to compute a set of Noetherian operators for primary ideals. Specifying a value for the option `Strategy` in the method `noetherianOperators` allows the user to choose which symbolic algorithm to use. The method `numericalNoetherianOperators` implements a numerical algorithm, which can deal with approximate input. In addition to the algorithm described in Section 2, we implement the following three algorithms:

(1) *Symbolic algorithm via dual spaces*: this algorithm computes Noetherian operators as bases of Macaulay dual spaces.

(2) *Numerical algorithm via interpolation*: this algorithm interpolates Noetherian operators from their specializations at several general points sampled on the underlying variety.

(3) *Hybrid symbolic/numerical*: this approach optimizes the approach in (1) by using information obtained from applying the numerical algorithm (2) at one general point on the underlying variety.

We now discuss each algorithm and illustrate its use in the package. Throughout the article, let $\mathbb{K}$ denote a field of characteristic zero, $R = \mathbb{K}[x_1, \ldots, x_n]$ a polynomial ring over $\mathbb{K}$, and $P \subseteq R$ a prime ideal in R. We typically use Q to denote a P-primary ideal, and I to denote a general (not necessarily primary) ideal which has P as a minimal prime.

3.1. ***Symbolic algorithm via dual spaces.*** The next algorithm to compute a set of Noetherian operators is a direct approach which reduces the problem to linear algebra. For convenience, write $\boldsymbol{t}$ for a maximal set of independent variables modulo P, and $\boldsymbol{x} := \{x_1, \ldots, x_n\} \setminus \boldsymbol{t}$ as the dependent variables. Then in the localization $S := \mathbb{K}(\boldsymbol{t})[\boldsymbol{x}]$ of R, the extension of I to S is zero-dimensional. If now I is a zero-dimensional P-primary ideal, then a set of Noetherian operators for I is the same as a basis for the dual space of I at P (as will be discussed in Section 5). This in turn can be computed as the kernel of a *Macaulay matrix*, which is a matrix over R/P with columns indexed by differential monomials and rows indexed by elements of I, whose entries are the result of applying a differential monomial to an element of I. As the numbers of rows and columns increase, the kernel eventually stabilizes, at which point the result is returned.

This is the default strategy used to compute Noetherian operators, when the input is a pair of ideals, the second of which should be a minimal prime of the first. It can also be explicitly called by specifying `Strategy => "MacaulayMatrix"`:

```
i9 : needsPackage "K3Carpets";
i10 : I = carpet(2, 2, Characteristic => 0);
o10 : Ideal of QQ[x ..x , y ..y ]
                  0   2   0   2
i11 : R = ring I;
i12 : noetherianOperators(I, Strategy => "MacaulayMatrix")
o12 = {| 1 |, | 2y_1dy_0+y_2dy_1 |}
o12 : List
```

Calling `noetherianOperators` with a minimal prime ideal P of I as the second argument will compute a set of Noetherian operators for the P-primary component of I. If I is unmixed, then the result of applying this method to all associated primes is a differential primary decomposition of I.

```
i13 : (P1, P2) = (radical I, ideal(R_0 + R_1));
i14 : J = intersect(I, P2^2);
o14 : Ideal of R
i15 : noetherianOperators(J, P1)
o15 = {| 1 |, | 2y_1dy_0+y_2dy_1 |}
o15 : List
```

```
i16 : noetherianOperators(J, P2)
o16 = {| 1 |, | dx_0 |}
o16 : List
```

3.2. ***Numerical algorithm via interpolation.*** We also provide algorithms to compute Noetherian operators purely from numerical data, bypassing the need to compute Gröbner bases. This is based on computing a set of *specialized Noetherian operators*, i.e., the result of evaluating (at some point) all polynomial coefficients in a set of Noetherian operators. The key observation is that one can obtain a set of specialized Noetherian operators by suitably slicing the variety [Chen et al. 2022, Theorem 4.1]. More precisely, for a P-primary ideal $Q \subseteq \mathbb{C}[\boldsymbol{t}, \boldsymbol{x}]$ and a point $p = (\boldsymbol{t}_0, \boldsymbol{x}_0) \in V(P)$, a minimal set of specialized Noetherian operators corresponds to a basis of the dual space of the zero-dimensional ideal $Q + (\boldsymbol{t} - \boldsymbol{t}_0)$ at the point p. The function `specializedNoetherianOperators` can be used to perform this computation.

```
i17 : p = point{{1,1,1,1,1,1}};
i18 : specializedNoetherianOperators(I, p, DependentSet => {R_1, R_3, R_4})
o18 = {| 1 |, | 2dy_0+dy_1 |}
o18 : List
```

Once a set of specialized Noetherian operators has been computed at a single general point, subsequent computations at other points can be sped up as the monomial support of a valid set of Noetherian operators is known (this fact also underlies the hybrid approach in Section 3.3). After specialized Noetherian operators are computed at sufficiently many general points on the variety, the original set of Noetherian operators can be recovered from their specializations via interpolation of rational functions; see [Chen et al. 2022, Algorithm 5].

This is the preferred strategy when the input is inexact, although it can also be used for exact input, as shown here. Note that the value of `DependentSet` must be specified:

```
i19 : numericalNoetherianOperators(I, DependentSet => {R_1, R_3, R_4})
-- warning: experimental computation over inexact field begun
--          results not reliable (one warning given per session)
           1y
             1
o19 = {1, ----dy  + 1dy }
          .5y   0      1
            2
o19 : List
```

By default, `Bertini` is used to sample points on $V(\sqrt{I})$. The user can specify their own sampling function with the option `Sampler`. The sampler should be a function that takes an integer n and the ideal I as input, and returns a `List` of n points on the variety.

```
i20 : needsPackage"NumericalImplicitization";
i21 : numericalNoetherianOperators(I, DependentSet => {R_1, R_3, R_4},
Sampler => (n,I) -> apply(n, i -> point sub(matrix realPoint radical I, CC)))
           y
             1
o21 = {1, ----dy  + 1dy }
          .5y   0      1
            2
o21 : List
```

3.3. ***Hybrid symbolic/numerical.*** In Section 3.1, Noetherian operators are found by computing the kernel of a Macaulay matrix with entries in the function field $\mathbb{K}(\boldsymbol{t})$, but computations in this field can be expensive. The numerical approach in Section 3.2 instead specializes the independent variables to random values. This allows computations to be done in $\mathbb{K}$ (typically with $\mathbb{K} = \mathbb{C}$) which is cheaper but the result consists of specializations of the Noetherian operators. A hybrid approach can combine the best of both strategies. In essence, the information revealed from running the numerical algorithm at a single point (without performing interpolation) can be used to trim the Macaulay matrix down to a smaller (optimal) size, without changing the kernel. For more details, we refer the interested reader to [Chen et al. 2022, Section 4.1].

Specifying `Strategy => "Hybrid"` with the method `noetherianOperators` calls this strategy. On larger examples, this strategy can greatly outperform the approach in Section 3.1.

```
i22 : noetherianOperators(I, Strategy=>"Hybrid")
o22 = {| 1 |, | 2y_1dy_0+y_2dy_1 |}
o22 : List
```

As in the numerical algorithm, the user may also specify a sampling function to find a general point.

4. COMPUTING MODULES FROM NOETHERIAN OPERATORS. In this section, we discuss a procedure that can be seen as the inverse of the process of computing a set of Noetherian operators. First, note that for any R-bisubmodule $\mathcal{E} \subseteq R\langle \partial_{x_1}, \dots, \partial_{x_n} \rangle^k$, the set

$$\{f \in R^k \mid A \bullet f \in P \text{ for all } A \in \mathcal{E}\}$$

is always a P-primary R-submodule of R^k (see [Cid-Ruiz 2021, Proposition 3.5]). We now consider the following problem:

Problem 4.1. Given an R-bisubmodule $\mathcal{E} \subseteq R\langle \partial_{x_1}, \dots, \partial_{x_n} \rangle^k$, compute (generators for) the P-primary submodule $\{f \in R^k \mid A \bullet f \in P \text{ for all } A \in \mathcal{E}\}$.

This is accomplished by [Chen and Cid-Ruiz 2022, Algorithm 4.3]. The idea is to use the explicit maps provided in [Chen and Cid-Ruiz 2022, Theorem 3.2] in inverse order to how they appear in [Chen and Cid-Ruiz 2022, Algorithm 4.1] (i.e., as discussed in Section 2). It should be noted that our implementation solves the following effective version of the problem above:

Problem 4.1′. Given $A_1, \dots, A_m \in R\langle \partial_{x_1}, \dots, \partial_{x_n} \rangle^k$, compute the P-primary submodule

$$\{f \in R^k \mid A \bullet f \in P \text{ for all } A \in \mathcal{E}\},$$

where $\mathcal{E} \subseteq R\langle \partial_{x_1}, \dots, \partial_{x_n} \rangle^k$ is the R-bisubmodule generated by $A_1, \dots, A_m$.

The function `getIdealFromNoetherianOperators` implements [Cid-Ruiz et al. 2021, Algorithm 8.2]. Below we show an example for $k = 1$, in which, given a P-primary ideal Q, we compute a set of Noetherian operators for Q, and then we recover Q from its Noetherian operators along with P. In general, this process may result in a different set of generators for Q.

```
i23 : R = QQ[x_1,x_2,x_3];
i24 : Q = ideal(x_1^2, x_2^2, x_3^2, x_1*x_2 + x_1*x_3 +x_2*x_3);
o24 : Ideal of R
i25 : L = noetherianOperators Q
o25 = {| 1 |, | dx_1 |, | dx_2 |, | dx_3 |, | dx_1dx_2-dx_1dx_3 |, | dx_1...
o25 : List
i26 : Q' = getIdealFromNoetherianOperators(L, radical Q)
                2   2                          2
o26 = ideal (x , x , x x  + x x  + x x , x )
              3   2   1 2    1 3    2 3   1
o26 : Ideal of R
i27 : Q == Q'
o27 = true
```

5. DUAL SPACES AND LOCAL HILBERT FUNCTIONS. In Section 3, dual spaces were used in service of computing Noetherian operators. However, dual spaces can also directly provide information about polynomial ideals. Suppose P is the maximal ideal corresponding to a $\mathbb{K}$-rational point $p \in \mathbb{K}^n$, and $I \subseteq R$ is an ideal with $p \in V(I)$. The *dual space of I at P* is

$$D_P[I] := \{A \in \mathbb{K}[\partial_{x_1}, \ldots, \partial_{x_n}] \mid (A \bullet f)(p) = 0 \text{ for all } f \in I\}.$$

The dual space is a subspace of the space of differential operators on R with constant coefficients and finite support which uniquely determines IR_P, where R_P denotes the localization of R at P.

The following dual space algorithms work with numerical data, for example if $\mathbb{K} = \mathbb{C}$ and the point associated to P is known only approximately. This is in contrast to symbolic algorithms relying on Gröbner bases. The methods described in this section were previously part of a package titled `NumericalHilbert` which has now been incorporated into `NoetherianOperators` due to an overlap in functionality.

If P is a minimal prime of I, then the dual space is finite-dimensional, and a basis of the dual space is a set of Noetherian operators for the P-primary component of I. Otherwise the dual space is infinite-dimensional, and only a truncation up to a specified degree can be computed. The methods `zeroDimensionalDual` and `truncatedDual` compute a basis for the dual space in these two cases. As in Section 3.1, these dual spaces are computed as kernels of Macaulay matrices. From a basis of the dual space truncated to degree d, it is straightforward to compute the local Hilbert function of R/I up to degree d, and this is implemented as applying `hilbertFunction` to the generators of a dual space.

```
i28 : R = CC[x_1,x_2];
i29 : I = ideal{x_1^2 + x_2^2 - 4, (x_1 - 1)^2};
o29 : Ideal of R
i30 : p = point{{1.0, 1.7320508}};
i31 : D = zeroDimensionalDual(p, I)
o31 = | 1 -1.73205x_1+x_2 |
o31 : DualSpace
i32 : apply(3, i -> hilbertFunction(i, D))
o32 = {1, 1, 0}
o32 : List
```

Another way of truncating the dual space of a positive-dimensional ideal is with respect to a local elimination order instead of a degree order. In [Krone and Leykin 2017a], this is referred to as an

eliminating dual space. Assume $\{x_{c+1}, \dots, x_n\}$ is a maximal set of independent variables for R/I, and choose an order that eliminates $V = \{x_1, \dots, x_c\}$. An eliminating dual for I with respect to V truncated to degree d, denoted $E_P^d[I, V]$, is defined as the set of dual operators whose lead terms with respect to the monomial order have degree at most d in the variables V, and is computed with the method `eliminatingDual`. When $V = \{x_1\}$, such a truncated dual space allows one to find the dual space of the colon ideal $I : x_1$ directly (without requiring the potentially expensive symbolic computation of finding $I : x_1$):

$$E_P^d[I : x_1, \{x_1\}] = x_1 \cdot E_0^{d+1}[I, \{x_1\}]$$

where $x_1 \cdot A$ represents the right action of $x_1 \in R$ on $A \in \mathbb{K}[\partial_{x_1}, \dots, \partial_{x_n}]$ (for example $x_1 \cdot \partial_{x_1}^2 = 2\partial_{x_1}$). Representations of these colon ideals are needed in [Krone and Leykin 2017a, Algorithm 5.1] for identifying embedded primes on a curve.

Using [Krone 2013, Algorithm 23], computing truncated dual spaces up to a certain degree provides a numerical algorithm for finding a full set of generators for the initial ideal of I with respect to a local degree order, and this is implemented by `gCorners`. In the process, a standard basis can be computed by specifying `StandardBasis => true`.

```
i33 : gCorners(point p, I)
o33 = | x_2 x_1^2 |
                1       2
o33 : Matrix R  <--- R
```

Finding the initial ideal via approximate numerical methods is an essential part of `isPointEmbedded`, which implements a numerical algorithm for the detection of an embedded component developed in [Krone and Leykin 2017b, Algorithm 4.2].

SUPPLEMENT. The online supplement contains version 2.2.1 of `NoetherianOperators`.

REFERENCES.

[Ait El Manssour et al. 2021] R. Ait El Manssour, M. Härkönen, and B. Sturmfels, "Linear PDE with constant coefficients", *Glasgow Mathematical Journal* (2021), 1–26.

[Baranovsky 2000] V. Baranovsky, "Moduli of sheaves on surfaces and action of the oscillator algebra", *J. Differential Geom.* **55**:2 (2000), 193–227. MR Zbl

[Brumfiel 1978] G. Brumfiel, "Differential operators and primary ideals", *J. Algebra* **51**:2 (1978), 375–398. MR Zbl

[Chen and Cid-Ruiz 2022] J. Chen and Y. Cid-Ruiz, "Primary decomposition of modules: a computational differential approach", *J. Pure Appl. Algebra* **226**:10 (2022), art. id. 107080. MR Zbl

[Chen et al. 2022] J. Chen, M. Härkönen, R. Krone, and A. Leykin, "Noetherian operators and primary decomposition", *J. Symbolic Comput.* **110** (2022), 1–23. MR Zbl

[Cid-Ruiz 2021] Y. Cid-Ruiz, "Noetherian operators, primary submodules and symbolic powers", *Collect. Math.* **72**:1 (2021), 175–202. MR Zbl

[Cid-Ruiz and Sturmfels 2021] Y. Cid-Ruiz and B. Sturmfels, "Primary decomposition with differential operators", 2021. arXiv 2101.03643

[Cid-Ruiz et al. 2021] Y. Cid-Ruiz, R. Homs, and B. Sturmfels, "Primary ideals and their differential equations", *Found. Comput. Math.* **21**:5 (2021), 1363–1399. MR Zbl

[Damiano et al. 2007] A. Damiano, I. Sabadini, and D. C. Struppa, "Computational methods for the construction of a class of Noetherian operators", *Experiment. Math.* **16**:1 (2007), 41–53. MR Zbl

[Ehrenpreis 1970] L. Ehrenpreis, *Fourier analysis in several complex variables*, Pure and Applied Mathematics, Vol. XVII, Wiley-Interscience [A division of John Wiley & Sons], New York-London-Sydney, 1970. MR Zbl

[Gröbner 1938] W. Gröbner, "Über eine neue idealtheoretische Grundlegung der algebraischen Geometrie", *Math. Ann.* **115**:1 (1938), 333–358. MR Zbl

[Gröbner 1952] W. Gröbner, "La théorie des idéaux et la géométrie algébrique", pp. 129–144 in *Deuxième Colloque de Géométrie Algébrique*, *Liège*, 1952, Georges Thone, Liège; Masson & Cie, Paris, 1952. MR Zbl

[Gröbner 1970] W. Gröbner, *Algebraische Geometrie. 2. Teil*: *Arithmetische Theorie der Polynomringe*, B. I. Hochschultaschenbücher **737/737**, Bibliographisches Institut, Mannheim-Vienna-Zurich, 1970. MR Zbl

[Iarrobino 1972] A. Iarrobino, "Punctual Hilbert schemes", *Bull. Amer. Math. Soc.* **78** (1972), 819–823. MR Zbl

[Krone 2013] R. Krone, "Numerical algorithms for dual bases of positive-dimensional ideals", *J. Algebra Appl.* **12**:6 (2013), art. id. 1350018. MR Zbl

[Krone and Leykin 2017a] R. Krone and A. Leykin, "Eliminating dual spaces", *J. Symbolic Comput.* **79**:part 3 (2017), 609–622. MR Zbl

[Krone and Leykin 2017b] R. Krone and A. Leykin, "Numerical algorithms for detecting embedded components", *J. Symbolic Comput.* **82** (2017), 1–18. MR Zbl

[Macaulay 1916] F. S. Macaulay, *The algebraic theory of modular systems*, Cambridge Mathematical Library, Cambridge University Press, Cambridge, 1916. MR Zbl

[Oberst 1999] U. Oberst, "The construction of Noetherian operators", *J. Algebra* **222**:2 (1999), 595–620. MR Zbl

[Palamodov 1970] V. P. Palamodov, *Linear differential operators with constant coefficients*, Grundl. Math. Wissen. **168**, Springer, 1970. MR Zbl

RECEIVED: 4 Jan 2021 REVISED: 8 Jul 2022 ACCEPTED: 26 Sep 2022

JUSTIN CHEN:

jchen@math.berkeley.edu
School of Mathematics, Georgia Institute of Technology, Atlanta, GA, United States

YAIRON CID-RUIZ:

Yairon.CidRuiz@UGent.be
Department of Mathematics: Algebra and Geometry, Ghent University, Gent, Belgium

MARC HÄRKÖNEN:

harkonen@gatech.edu
School of Mathematics, Georgia Institute of Technology, Atlanta, GA, United States

ROBERT KRONE:

rkrone@math.ucdavis.edu
Department of Mathematics, UC Davis, Davis, CA, United States

ANTON LEYKIN:

leykin@math.gatech.edu
School of Mathematics, Georgia Institute of Technology, Atlanta, GA, United States

JSAG **12** (2022), 43–49
https://doi.org/10.2140/jsag.2022.12.43

The Journal of Software for
Algebra and Geometry

Computing with jets

FEDERICO GALETTO AND NICHOLAS IAMMARINO

ABSTRACT: We introduce a Macaulay2 package for working with jet schemes. The main method constructs jets of ideals, polynomial rings and their quotients, ring homomorphisms, affine varieties, and (hyper)graphs. The package also includes additional methods to compute principal components and radicals of jets of monomial ideals.

1. INTRODUCTION.

Roughly speaking, the scheme of s-jets of a scheme X is the collection of order s Taylor approximations at points of X. More formally, let X be a scheme over a field $\Bbbk$. Following [Ein and Mustaţă 2009, §2], we call a scheme $\mathcal{J}_s(X)$ over $\Bbbk$ the scheme of s-jets of X, if for every $\Bbbk$-algebra A there is a functorial bijection

$$\mathrm{Hom}(\mathrm{Spec}(A), \mathcal{J}_s(X)) \cong \mathrm{Hom}(\mathrm{Spec}(A[t]/\langle t^{s+1}\rangle), X).$$

This means that the A-points of $\mathcal{J}_s(X)$ are in bijection with the $A[t]/\langle t^{s+1}\rangle$-points of X. It follows that $\mathcal{J}_0(X) \cong X$, and $\mathcal{J}_1(X)$ is the total tangent scheme of X, in line with the definition of tangent space using dual numbers [Hartshorne 1977, II, Exercise 2.8]. Jet schemes play an important role in the study of singularities, as initially suggested by J. Nash [1995], and in connection with other related topics, such as motivic integration and birational geometry [Denef and Loeser 2001; Mustaţă 2001; 2002; Ein and Mustaţă 2009].

The existence of jet schemes is proved in detail in [Ein and Mustaţă 2009, §2]. We recall an essential step, which is the construction of jets of an affine variety. Let X be an affine variety over $\Bbbk$. Consider a closed embedding of X into an affine space $\mathbb{A}^n$ over $\Bbbk$. Let $I = \langle f_1, \ldots, f_r\rangle$ be the ideal of $R = \Bbbk[x_1, \ldots, x_n]$ corresponding to this embedding. For $s \in \mathbb{N}$, define the polynomial ring

$$\mathcal{J}_s(R) = \Bbbk[x_{i,j} \mid i = 1, \ldots, n,\ j = 0, \ldots, s].$$

For each $k = 1, \ldots, n$, perform the substitution

$$x_k \mapsto x_{k,0} + x_{k,1}t + x_{k,2}t^2 + \cdots + x_{k,s}t^s = \sum_{j=0}^{s} x_{k,j}t^j$$

MSC2020: primary 13-04, 14-04; secondary 05C25, 13F55, 14M12.
Keywords: jets, Macaulay2, monomial ideals, graphs, determinantal varieties.
`Jets` version 1.1

taking elements of R to elements of $\mathcal{J}_s(R)[t]$. This substitution is the "universal s-jet" corresponding to the identity map on $\mathcal{J}_s(X)$ in the functorial bijection above. Applying this substitution to a generator f_i of I gives the decomposition

$$f_i\left(\sum_{j=0}^{s} x_{1,j}t^j, \ldots, \sum_{j=0}^{s} x_{n,j}t^j\right) = \sum_{j\geqslant 0} f_{i,j}t^j,$$

where the coefficients $f_{i,j}$ are polynomials in $\mathcal{J}_s(R)$. The *ideal of s-jets* of $I = \langle f_1, \ldots, f_r\rangle$ is the ideal of $\mathcal{J}_s(R)$ defined by

$$\mathcal{J}_s(I) = \langle f_{i,j} \mid i = 1, \ldots, r,\ j = 0, \ldots, s\rangle.$$

The scheme of s-jets of X is $\operatorname{Spec}(\mathcal{J}_s(R)/\mathcal{J}_s(I))$.

This paper introduces the `Jets` package[1] for [Macaulay2], streamlining the process of constructing ideals of jets as indicated above. We adopt the following notation: the variables in the polynomial rings containing the equations of jets have the names of the variables of the original equations with the order of the jets appended to them, and the same subscripts. Moreover, the rings containing the equations of jets are constructed incrementally as towers.

Ideals of jets are computed via the `jets` method applied to objects of type `Ideal`. In addition, the `jets` method can also be applied to objects of type `QuotientRing`, `RingMap`, and `AffineVariety`, with the effects one would expect from applying jet functors. For more information, including grading options, we invite the reader to consult the documentation of the package. Each of the following sections consists of the package being demonstrated in different contexts.

2. JETS OF MONOMIAL IDEALS. As observed in [Goward and Smith 2006], the ideal of jets of a monomial ideal is typically not a monomial ideal.

```
i1 : needsPackage "Jets";
i2 : R=QQ[x,y,z];
i3 : I=ideal(x*y*z);
o3 : Ideal of R
i4 : J2I=jets(2,I);
o4 : Ideal of QQ[x0, y0, z0][x1, y1, z1][x2, y2, z2]
i5 : netList J2I_*
     +-------------------------------------------------------------------+
o5 = |y0*z0*x2 + x0*z0*y2 + x0*y0*z2 + z0*x1*y1 + y0*x1*z1 + x0*y1*z1|
     +-------------------------------------------------------------------+
     |y0*z0*x1 + x0*z0*y1 + x0*y0*z1                                     |
     +-------------------------------------------------------------------+
     |x0*y0*z0                                                           |
     +-------------------------------------------------------------------+
```

However, by [Goward and Smith 2006, Theorem 3.1], the radical is always a (squarefree) monomial ideal. In fact, the proof of [Goward and Smith 2006, Theorem 3.2] shows that the radical is generated by the individual terms of the generators $f_{i,j}$ described in the introduction. This observation provides an alternative algorithm for computing radicals of jets of monomial ideals, which can be faster than the

[1] Available as a supplement to this paper or at https://github.com/galettof/Jets.

default radical computation in Macaulay2.

```
i6 : jetsRadical(2,I);

o6 : Ideal of QQ[x0, y0, z0][x1, y1, z1][x2, y2, z2]

i7 : netList pack(5,oo_*)

     +--------+--------+--------+--------+--------+
o7 = |y0*z0*x2|x0*z0*y2|x0*y0*z2|z0*x1*y1|y0*x1*z1|
     +--------+--------+--------+--------+--------+
     |x0*y1*z1|y0*z0*x1|x0*z0*y1|x0*y0*z1|x0*y0*z0|
     +--------+--------+--------+--------+--------+
```

For a monomial hypersurface, [Goward and Smith 2006, Theorem 3.2] describes the minimal primes of the ideal of jets. Moreover, the main theorem in [Yuen 2006] counts the multiplicity of the jet scheme of a monomial hypersurface along its minimal primes (see also [Yuen 2007b]). We compute the minimal primes, then use Sayfrafi et al.'s `LocalRings` package to compute their multiplicities in the second jet scheme of the example above.

```
i8 : P=minimalPrimes J2I;

i9 : --flatten ring to use LocalRings package
     (A,f)=flattenRing ring J2I;

i10 : needsPackage "LocalRings";

i11 : --quotient by jets ideal as a module
      M=cokernel gens f J2I;

i12 : --compute the multiplicity of the jets along each component
      mult=for p in P list (
          Rp := localRing(A,f p);
          length(M ** Rp)
          );

i13 : netList(pack(4,mingle{P,mult}),HorizontalSpace=>1)

      +--------------------+---+--------------------+---+
o13 = | ideal (z0, y0, x0) | 6 | ideal (z0, y0, z1) | 3 |
      +--------------------+---+--------------------+---+
      | ideal (z0, y0, y1) | 3 | ideal (z0, x0, z1) | 3 |
      +--------------------+---+--------------------+---+
      | ideal (z0, x0, x1) | 3 | ideal (z0, z1, z2) | 1 |
      +--------------------+---+--------------------+---+
      | ideal (y0, x0, y1) | 3 | ideal (y0, x0, x1) | 3 |
      +--------------------+---+--------------------+---+
      | ideal (y0, y1, y2) | 1 | ideal (x0, x1, x2) | 1 |
      +--------------------+---+--------------------+---+
```

3. JETS OF GRAPHS. Jets of graphs were introduced in [Galetto et al. 2021]. Starting with a finite, simple graph G, one may construct a quadratic squarefree monomial ideal $I(G)$ (known as the *edge ideal* of the graph) by converting edges to monomials (see for example [Van Tuyl 2013]). One may then consider the radical of the ideal of s-jets of $I(G)$, which is again a quadratic squarefree monomial ideal. The graph corresponding to this ideal is the graph of s-jets of G, denoted $\mathcal{J}_s(G)$.

Jets of graphs and hypergraphs can be obtained by applying the `jets` method to objects of type `Graph` and `HyperGraph` from the Macaulay2 `EdgeIdeals` package [Francisco et al. 2009] (which is automatically loaded by the `Jets` package). Consider, for example, the graph in Figure 1.

```
i1 : needsPackage "Jets";

i2 : R=QQ[a..e];

i3 : G=graph({{a,c},{a,d},{a,e},{b,c},{b,d},{b,e},{c,e}});
```

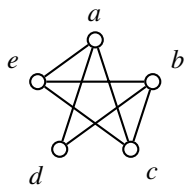

Figure 1. The graph G.

We compute the first and second order jets, and list their edges.

```
i4 : J1G=jets(1,G); netList pack(7,edges J1G)

     +--------+--------+--------+--------+--------+--------+--------+
o5 = |{c1, a0}|{d1, a0}|{e1, a0}|{c1, b0}|{d1, b0}|{e1, b0}|{a1, c0}|
     +--------+--------+--------+--------+--------+--------+--------+
     |{b1, c0}|{e1, c0}|{a0, c0}|{b0, c0}|{a1, d0}|{b1, d0}|{a0, d0}|
     +--------+--------+--------+--------+--------+--------+--------+
     |{b0, d0}|{a1, e0}|{b1, e0}|{c1, e0}|{a0, e0}|{b0, e0}|{c0, e0}|
     +--------+--------+--------+--------+--------+--------+--------+

i6 : J2G=jets(2,G); netList pack(7,edges J2G)

     +--------+--------+--------+--------+--------+--------+--------+
o7 = |{a1, c1}|{b1, c1}|{a1, d1}|{b1, d1}|{a1, e1}|{b1, e1}|{c1, e1}|
     +--------+--------+--------+--------+--------+--------+--------+
     |{c2, a0}|{d2, a0}|{e2, a0}|{c1, a0}|{d1, a0}|{e1, a0}|{c2, b0}|
     +--------+--------+--------+--------+--------+--------+--------+
     |{d2, b0}|{e2, b0}|{c1, b0}|{d1, b0}|{e1, b0}|{a2, c0}|{b2, c0}|
     +--------+--------+--------+--------+--------+--------+--------+
     |{e2, c0}|{a1, c0}|{b1, c0}|{e1, c0}|{a0, c0}|{b0, c0}|{a2, d0}|
     +--------+--------+--------+--------+--------+--------+--------+
     |{b2, d0}|{a1, d0}|{b1, d0}|{a0, d0}|{b0, d0}|{a2, e0}|{b2, e0}|
     +--------+--------+--------+--------+--------+--------+--------+
     |{c2, e0}|{a1, e0}|{b1, e0}|{c1, e0}|{a0, e0}|{b0, e0}|{c0, e0}|
     +--------+--------+--------+--------+--------+--------+--------+
```

As predicted in [Galetto et al. 2021, Theorem 3.1], all jets have the same chromatic number.

```
i8 : apply({G,J1G,J2G},chromaticNumber)

o8 = {3, 3, 3}

o8 : List
```

By contrast, jets may not preserve the property of being cochordal.

```
i9 : apply({G,J1G,J2G},x -> isChordal complementGraph x)

o9 = {true, true, false}

o9 : List
```

Using Fröberg's theorem [1990], we deduce that although the edge ideal of a graph may have a linear free resolution, the edge ideals of its jets may not have linear resolutions.

Finally, we compare minimal vertex covers of the graph and of its second order jets.

```
i10 : vertexCovers G

o10 = {a*b*c, a*b*e, c*d*e}

o10 : List

i11 : netList pack(2,vertexCovers J2G)

      +-------------------------+-------------------------+
o11 = |a2*b2*c2*a1*b1*c1*a0*b0*c0|a2*b2*e2*a1*b1*e1*a0*b0*e0|
      +-------------------------+-------------------------+
      |a2*b2*a1*b1*c1*a0*b0*c0*e0|a2*b2*a1*b1*e1*a0*b0*c0*e0|
      +-------------------------+-------------------------+
      |c2*d2*e2*c1*d1*e1*c0*d0*e0|a1*b1*c1*a0*b0*c0*d0*e0   |
      +-------------------------+-------------------------+
      |a1*b1*e1*a0*b0*c0*d0*e0   |c1*d1*e1*a0*b0*c0*d0*e0   |
      +-------------------------+-------------------------+
```

With the exception of the second row, many vertex covers arise as indicated in [Galetto et al. 2021, Propositions 5.2 and 5.3].

4. JETS OF DETERMINANTAL VARIETIES. Determinantal varieties are classical geometric objects whose jets have been studied with a certain degree of success [Košir and Sethuraman 2005a; 2005b; Yuen 2007a; Ghorpade et al. 2014; Docampo 2013; Mallory 2021]. For our example, we consider the determinantal varieties X_r of 3×3 matrices of rank at most r, which are defined by the vanishing of minors of size $r+1$. We illustrate computationally some of the known results about jets.

```
i1 : needsPackage "Jets";

i2 : R=QQ[x_(1,1)..x_(3,3)];

i3 : G=genericMatrix(R,3,3)

o3 = | x_(1,1) x_(2,1) x_(3,1) |
     | x_(1,2) x_(2,2) x_(3,2) |
     | x_(1,3) x_(2,3) x_(3,3) |

              3       3
o3 : Matrix R  <--- R
```

Since X_0 is a single point, its first jet scheme consists of a single (smooth) point.

```
i4 : I1=minors(1,G);

o4 : Ideal of R

i5 : JI1=jets(1,I1);

o5 : Ideal of QQ[x0     ..x0     ][x1     ..x1     ]
                 1,1      3,3      1,1      3,3

i6 : dim JI1, isPrime JI1

o6 = (0, true)

o6 : Sequence
```

The jets of X_2 (the determinantal hypersurface) are known to be irreducible (see [Košir and Sethuraman 2005a, Theorem 3.1] or [Docampo 2013, Corollary 4.13]). Since X_2 is a complete intersection and has rational singularities [Weyman 2003, Corollary 6.1.5(b)], this also follows from a more general result of M. Mustaţă [2001, Theorem 3.3].

```
i7 : I3=minors(3,G);

o7 : Ideal of R

i8 : JI3=jets(1,I3);

o8 : Ideal of QQ[x0     ..x0     ][x1     ..x1     ]
                 1,1      3,3      1,1      3,3

i9 : isPrime JI3

o9 = true
```

For the case of 2×2 minors, [Košir and Sethuraman 2005a, Theorem 5.1], [Yuen 2007a, Theorem 5.1], and [Docampo 2013, Corollary 4.13] all count the number of components; the first two of these references describe the components further. As expected, the first jet scheme of X_1 has two components, one of them an affine space.

```
i10 : I2=minors(2,G);

o10 : Ideal of R
```

```
i11 : JI2=jets(1,I2);
o11 : Ideal of QQ[x0     ..x0     ][x1     ..x1     ]
                   1,1      3,3      1,1      3,3
i12 : P=primaryDecomposition JI2; #P
o13 = 2
i14 : P_1
o14 = ideal (x0     , x0     , x0     , x0     , x0     , x0     , x0     , x0     , x0     )
              3,3      3,2      3,1      2,3      2,2      2,1      1,3      1,2      1,1
o14 : Ideal of QQ[x0     ..x0     ][x1     ..x1     ]
                   1,1      3,3      1,1      3,3
```

The other component is the so-called principal component of the jet scheme, i.e., the Zariski closure of the first jets of the smooth locus of X_1. To check this, we first establish that the first jet scheme is reduced (i.e., its ideal is radical), then use the `principalComponent` method with the option `Saturate=>false` to speed up computations. (We invite the reader to consult the package documentation for more details.)

```
i15 : radical JI2==JI2
o15 = true
i16 : P_0 == principalComponent(1,I2,Saturate=>false)
o16 = true
```

Finally, as observed in [Ghorpade et al. 2014, Theorem 18], the Hilbert series of the principal component of the first jet scheme of X_1 is the square of the Hilbert series of X_1.

```
i17 : apply({P_0,I2}, X -> hilbertSeries(X,Reduce=>true))
                      2    3    4          2
       1 + 8T + 18T  + 8T  + T    1 + 4T + T
o17 = {-----------------------, -----------}
                        10              5
               (1 - T)         (1 - T)
o17 : List
i18 : numerator (first oo) == (numerator last oo)^2
o18 = true
```

SUPPLEMENT. The online supplement contains version 1.1 of `Jets`.

ACKNOWLEDGEMENTS. We are grateful to Greg Smith for valuable feedback on an early version of the package. We also thank the anonymous referees for suggesting several improvements to the original version of this article and package.

REFERENCES.

[Denef and Loeser 2001] J. Denef and F. Loeser, "Geometry on arc spaces of algebraic varieties", pp. 327–348 in *European Congress of Mathematics*, *Vol. I* (Barcelona, 2000), edited by C. Casacuberta et al., Progr. Math. **201**, Birkhäuser, Basel, 2001. MR

[Docampo 2013] R. Docampo, "Arcs on determinantal varieties", *Trans. Amer. Math. Soc.* **365**:5 (2013), 2241–2269. MR Zbl

[Ein and Mustaţă 2009] L. Ein and M. Mustaţă, "Jet schemes and singularities", pp. 505–546 in *Algebraic geometry* (Seattle, 2005), edited by D. Abramovich et al., Proc. Sympos. Pure Math. **80**, Amer. Math. Soc., Providence, RI, 2009. MR Zbl

[Francisco et al. 2009] C. A. Francisco, A. Hoefel, and A. Van Tuyl, "EdgeIdeals: a package for (hyper)graphs", *J. Softw. Algebra Geom.* **1** (2009), 1–4. MR Zbl

[Fröberg 1990] R. Fröberg, "On Stanley–Reisner rings", pp. 57–70 in *Topics in algebra* (Warsaw, 1988), edited by S. Balcerzyk et al., Banach Center Publ. **26**, PWN, Warsaw, 1990. MR Zbl

[Galetto et al. 2021] F. Galetto, E. Helmick, and M. Walsh, "Jet graphs", *Involve* **14**:5 (2021), 793–812. MR Zbl

[Ghorpade et al. 2014] S. R. Ghorpade, B. Jonov, and B. A. Sethuraman, "Hilbert series of certain jet schemes of determinantal varieties", *Pacific J. Math.* **272**:1 (2014), 147–175. MR Zbl

[Goward and Smith 2006] R. A. Goward, Jr. and K. E. Smith, "The jet scheme of a monomial scheme", *Comm. Algebra* **34**:5 (2006), 1591–1598. MR Zbl

[Hartshorne 1977] R. Hartshorne, *Algebraic geometry*, Graduate Texts in Mathematics **52**, Springer, 1977. MR Zbl

[Košir and Sethuraman 2005a] T. Košir and B. A. Sethuraman, "Determinantal varieties over truncated polynomial rings", *J. Pure Appl. Algebra* **195**:1 (2005), 75–95. MR Zbl

[Košir and Sethuraman 2005b] T. Košir and B. A. Sethuraman, "A Groebner basis for the 2×2 determinantal ideal mod t^2", *J. Algebra* **292**:1 (2005), 138–153. MR

[Macaulay2] D. R. Grayson and M. E. Stillman, "Macaulay2: a software system for research in algebraic geometry", available at http://www.math.uiuc.edu/Macaulay2.

[Mallory 2021] D. Mallory, "Minimal log discrepancies of determinantal varieties via jet schemes", *J. Pure Appl. Algebra* **225**:2 (2021), art. id. 106497. MR Zbl

[Mustaţă 2002] M. Mustaţă, "Singularities of pairs via jet schemes", *J. Amer. Math. Soc.* **15**:3 (2002), 599–615. MR Zbl

[Mustaţă 2001] M. Mustaţă, "Jet schemes of locally complete intersection canonical singularities", *Invent. Math.* **145**:3 (2001), 397–424. MR Zbl

[Nash 1995] J. F. Nash, Jr., "Arc structure of singularities", *Duke Math. J.* **81**:1 (1995), 31–38. MR Zbl

[Van Tuyl 2013] A. Van Tuyl, "A beginner's guide to edge and cover ideals", pp. 63–94 in *Monomial ideals, computations and applications*, edited by P. G. Anna M. Bigatti and E. S. de Cabezón, Lecture Notes in Math. **2083**, Springer, 2013. MR Zbl

[Weyman 2003] J. Weyman, *Cohomology of vector bundles and syzygies*, Cambridge Tracts in Mathematics **149**, Cambridge University Press, 2003. MR Zbl

[Yuen 2006] C. Yuen, "Multiplicity of jet schemes of monomial schemes", 2006. arXiv math/0607638

[Yuen 2007a] C. Yuen, "Jet schemes of determinantal varieties", pp. 261–270 in *Algebra, geometry and their interactions*, edited by J. M. Alberto Corso and C. Polini, Contemp. Math. **448**, Amer. Math. Soc., Providence, RI, 2007. MR Zbl

[Yuen 2007b] C. Yuen, "The multiplicity of jet schemes of a simple normal crossing divisor", *Comm. Algebra* **35**:12 (2007), 3909–3911. MR Zbl

RECEIVED: 12 Aug 2021 REVISED: 13 Jun 2022 ACCEPTED: 20 Oct 2022

FEDERICO GALETTO:

f.galetto@csuohio.edu
Department of Mathematics and Statistics, Cleveland State University, Cleveland, OH, United States

NICHOLAS IAMMARINO:

nickiammarino@gmail.com
Department of Mathematics and Statistics, Cleveland State University, Cleveland, OH, United States

Journal of Software for Algebra and Geometry vol 12, no 1, 2022