# RATIONALMAPS, A PACKAGE FOR MACAULAY2

C.J. BOTT, S. 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 a number of things about rational maps between varieties. In particular, this package will compute

- 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.
- And more!

Our functions have numerous options which allow them to run much more quickly in certain examples if configured correctly. The `Verbose` option gives hints as to the best way to apply these.

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 $\mathbf{f} = \{f_0, \dots f_m\}$ of the same degree in the coordinate ring of $X$, denoted by $R$. The idea of looking at the syzygies of the forms $\mathbf{f}$ to detect the geometric properties of $\mathfrak{F}$ goes back at least to [?] in the case where $X = \mathbb{P}^n$, $Y = \mathbb{P}^m$ and $m = n$ (see also [?]). In [?] this method was developed by Russo and Simis to handle the case $X = \mathbb{P}^n$ and $m \geq n$. Simis 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 $\mathbf{f}$ [?]. Recently, Doria, Hassanzadeh, and Simis applied these Rees algebra techniques to study the birationality of $\mathfrak{F}$ [?]. Our core functions, in particular the functions related to computing inverse maps, rely heavily on this work.

*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.

## 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 $\mathfrak{F} : X \longrightarrow \mathbb{P}^m_k$ be a rational map from $X$ to projective space. Then we can choose some representative $(f_0, \cdots, f_m)$ of $\mathfrak{F}$, where each $f_i$ is the $i^{\text{th}}$ 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 this does not enlarge the base locus of

The second named author was supported by CNPq-bolsa de Produtividade.

The third named author was supported in part by the NSF FRG Grant DMS #1265261/1501115 and NSF CAREER Grant DMS #1252860/1501102.

The fourth named author was supported in part by the NSF FRG Grant DMS #1265261/1501115 and NSF CAREER Grant DMS #1252860/1501102.

$\mathfrak{F}$ since $\mathfrak{F}$ is undefined whenever the denominator of any of the $f_i$ vanishes.) Thus we can 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 $(f_0, \cdots, f_m)$. However, this can yield a base locus that's 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.** [?, 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 $I = (f_0, \cdots, f_m)$. Then the set of such representatives of $\mathfrak{F}$ corresponds bijectively to the homogeneous vectors in the rank 1 graded $R$-module* $\mathrm{Hom}_R(I, R) \cong (R :_K I)$.

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

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

be a free resolution of $I$. Then we get

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

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

The base locus of $\mathfrak{F}$ is the intersection of the sets $V(f_0^i, \cdots, f_m^i)$ as $\mathbf{f}^i = (f_0^i, \cdots, f_m^i)$ ranges over all the representatives of $\mathfrak{F}$. The above implies that this is the same as the intersection of the sets $V(w_0^i, \cdots, w_m^i)$ as $\mathbf{w}^i = (w_0^i, \cdots, w_m^i)$ 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's enough to take a generating set $\mathbf{w}^1, \cdots, \mathbf{w}^n$ 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 $\mathbf{w}^i$. Our function `baseLocusOfMap` returns this ideal.

```
i1 : loadPackage "RationalMaps";
i2 : R = QQ[x,y,z];
i3 : f = {x^2*y, x^2*z, x*y*z};
i4 : baseLocusOfMap(f);
o4 = ideal (y*z, x*z, x*y)
o4 : Ideal of R
```

If the `SaturateOutput` option is set `true`, our function will return the saturation of this ideal.

## 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 $\mathbf{f} = \{f_0, \ldots f_m\}$ 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 over a field $k$ and $\dim R \geq 1$. The idea to find a ring theoretic criterion for birationality and on top of that to find the inverse of a rational map, is to study the Rees algebra of the ideal $I = (\mathbf{f})$ in $R$. So that let $R \simeq k[x_0, \ldots, x_n] = k[\mathbf{X}]/\mathfrak{a}$ with $k[\mathbf{X}] = k[X_0, \ldots, X_n]$ and $\mathfrak{a}$ a homogeneous ideal. The Rees algebra is defined by the polynomial relations among $\{f_0, \ldots f_m\}$ in $R$. To this end, we consider the polynomial extension $R[\mathbf{Y}] = R[Y_0, \ldots, Y_m]$. 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 $R[\mathbf{Y}]/\mathfrak{J} \simeq \mathcal{R}_R((\mathbf{f}))$, with $\mathfrak{J}$ a bihomogeneous *presentation ideal.* $\mathfrak{J}$ is a bigraded ideal depends only on the rational map defined by $\mathbf{f}$ and not on this particular representative.

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

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

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

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

One then goes to construct a matrix which can measure the birationality of the map. The first step is to lift the polynomials $Q_i(\mathbf{Y}) \in R[\mathbf{Y}]$ into $k[\mathbf{X}, \mathbf{Y}]$. Since the $\{y_0, \cdots, y_m\}$ are indeterminates over $R$, each pair of such representations of the same form gives a syzygy of $\{x_0, \ldots, x_n\}$ with coefficients in $k$. This is where one must take into attention whether $X \subseteq \mathbb{P}^n$ in 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 non-degenerated.

Next, one can pick a minimal set of generators of the ideal $(\mathfrak{J}_{1,*})$ consisting of a finite number of forms of bidegree $(1, q)$, for various $q$'s. Let's assume $X \subseteq \mathbb{P}^n$ is non-degenerated. Let $\{P_1, \ldots, P_s\} \subset k[\mathbf{X}, \mathbf{Y}]$ denote liftings of these biforms, consider the Jacobian matrix of the polynomials $\{P_1, \ldots, P_s\}$ with respect to $\{x_0, \cdots, x_n\}$. This is a matrix with entries in $k[\mathbf{Y}]$. Write $\psi$ for the corresponding matrix over $S = k[\mathbf{Y}]/\mathfrak{b}$, the coordinate ring of $Y$. This matrix is called the *weak Jacobian dual matrix* associated to the given set of generators of $(\mathfrak{J}_{1,*})$. Note that a weak Jacobian matrix $\psi$ 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 [**?**, Lemma 2.13] that if the weak Jacobian matrix associated to one set of bihomogeneous minimal generators of $(\mathfrak{J}_{1,*})$ has rank over $S$ then the weak Jacobian matrix associated to any other set of bihomogeneous minimal generators of $(\mathfrak{J}_{1,*})$ has rank over $S$ and the two ranks coincide.

The following criterion is [**?**, Theorem 2.18 ]. In the package we consider only the cases where $\mathbf{X}$ is irreducible i.e. $R$ is a domain.

**Theorem 3.1.** *Let $X \subseteq \mathbb{P}^n$ be non-degenerate. Then $\mathfrak{F}$ is birational onto $\mathbf{Y}$ 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 one*) *null space of $\psi$ 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 $\psi$ of rank* $\operatorname{edim}(R) - 1$.

As expected, the most expensive part of applying this theorem is computing the Rees ideal $\mathfrak{J}$. 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` 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 non-zero element in $R$ (we assume $R$ to be a domain). This strategy appears to be slower in some examples, though one might be able improve this option in the future by stopping the computation of the saturation at a certain step.

Computing inverse maps is the most important functionality of this package, and is done by the function `inverseOfMap`. 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 sub-matrix of $\psi$ of rank $\operatorname{edim}(R) - 1$. Each way has its own benefits. Method (1) is quite fast in many cases, however method (2) is very useful if the rank of the Jacobian dual matrix $\psi$ is relatively small compared to the degrees of the entries of $\psi$. Our function `inverseOfMap` starts by using the second method and later switches to the first method if the second method didn't work. The timing of this transition from the first method to the second method is controlled by the option `MinorsCount`. Setting `MinorsCount` to zero will mean that no minors are checked and the inverse map is computed just by looking at the syzygies of $\psi$. If `MinorsCount` is left as null (the default value), the program will try to make an educated guess as to how big to set this option, depending on varieties the user is working with.

In addition, to improve the speed of the function `inverseOfMap`, we 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. Similarly, if `CheckBirational` set `false`, `inverseOfMap` will not check birationality although it still computes the Jacobian dual matrix.

In general, as long as `Verbose` is `true`, 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 : 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}});
o3 : RingMap Q <--- Q
i4 : time inverseOfMap(phi, AssumeDominant=>true,CheckBirational=>false, MinorsCount=>50000)
Starting inverseOfMapSimis(SimisStrategy or HybridStrategy)
inverseOfMapSimis:  About to compute partial Groebner basis of rees ideal up to degree {1, 1}.
inverseOfMapSimis:  About to compute partial Groebner basis of rees ideal up to degree {1, 2}.
inverseOfMapSimis:  About to compute partial Groebner basis of rees ideal up to degree {1, 4}.
inverseOfMapSimis:  About to compute partial Groebner basis of rees ideal up to degree {1, 7}.
inverseOfMapSimis:  About to compute partial Groebner basis of rees ideal up to degree {1, 11}.
inverseOfMapSimis:  About to compute partial Groebner basis of rees ideal up to degree {1, 16}.
inverseOfMapSimis:  We give up. Using all of the previous computations,
                                 we  compute the whole Groebner basis of the rees ideal.
                                 Increase HybridLimit and rerun to avoid this.
inverseOfMapSimis: Found Jacobian dual matrix (or a weak form of it), it has  5 columns
                                 and about  20 rows.
inverseOfMapSimis: Looking for a nonzero minor
Starting nonZeroMinor, looking for rank: 4, we will run it 50000 times.
                                        If this is slow, rerun with MinorsCount=>0.
```

```
nonZeroMinor: Found a nonzero minor
inverseOfMapSimis: We found a nonzero minor.
     -- used 3.51586 seconds
```

## 4. Embeddings

Our package also checks whether a rational map $\mathfrak{F} : X \longrightarrow Y$ 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 if 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.

```
i1 : needspackage "divisor"; --used to quickly define a map
i2 : c = QQ[x,y,z]/(x^4+x^2*y*z+y^4+z^3*x);
i3 : q = ideal(y,x+z); --a point on our curve
i6 : loadpackage "rationalmaps";
i8 : f2 = maptoprojectivespace(12*divisor(q));
o8 : RingMap C <--- QQ[YY , YY , YY , YY , YY , YY , YY , YY , YY , YY  ]
                         1    2    3    4    5    6    7    8    9    10
i9 : time isEmbedding(f2)
isEmbedding: About to find the image of the map.
If you know the image, you may want to specify that and set AssumeDominant=>true option if
        this is slow.
isEmbedding: Checking to see if the map is a regular map
isEmbedding: computing the inverse  map
Starting inverseOfMapSimis(SimisStrategy or HybridStrategy)
inverseOfMapSimis:  About to compute partial Groebner basis of rees ideal up to degree {1, 1}.
inverseOfMapSimis: We computed enough of the Groebner basis.
inverseOfMapSimis: Found Jacobian dual matrix (or a weak form of it), it has  3 columns  and
        about  17 rows.
inverseOfMapSimis: Failed to find a nonzero minor.  We now compute syzygies instead.
        If this doesn't terminate quickly, you may want to try increasing the option MinorsCount.
isEmbedding: checking if the inverse map is a regular map
     -- used 3.01948 seconds

o9 = true
```

## 5. Functionality overlap with other packages

We note that our package has some overlaps in functionality with other packages.

While the `Parametrization` package [?] focuses mostly on curves, it also includes a function called `invertBirationalMap` which has the same functionality as `inverseOfMap`. On the other hand, these two functions were implemented somewhat differently and so sometimes one function can be substantially faster than the other.

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

- `isBirational` gives a probabilisitc answer to the question of whether a map between varieties is birational. Furthermore, if the source is projective space, then `degreeOfRationalMap` with `MathMode=>true` can give a deterministic answer that is frequently faster than what our package can provide with `isBirationalMap`.
- `invertBirMap` 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 you pass this function a map not from projective space, then it calls a modified, faster version of `invertBirationalMap`

originally from `Parametrization`. Even in some cases with maximal linear rank, our `inverseOfMap` function appears to be quite competitive however.

5.1. **Speed comparisons.** First we do comparisons using examples with maximal linear rank where `Cremona` excels. These examples were run using version 2.0 of `Cremona` and version 0.21 of `RationalMaps`.

Indeed, in some of those examples with maximal linear rank, `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});
i5 : time inverseOfMap(phi, AssumeDominant=>true, Verbose=>false)-- Function from "RationalMaps"
     -- used 1.12522 seconds
i6 : time invertBirMap phi -- Function from "Cremona"
     -- used 0.0789913 seconds
i7 : isSameMap(o5, o6) -- Function from "RationalMaps"
o7 = true
```

However, sometimes our function is substantially faster even in examples with maximal linear rank.

```
i1 : ZZ/33331[t_0..t_6];
i2 : loadPackage "Cremona"; loadPackage "RationalMaps";
i4 : phi=toMap minors(3,matrix{{t_0..t_4},{t_1..t_5},{t_2..t_6}});
i5 : J=kernelComponent(phi,2);
i6 : phi=toMap(phi,Dominant=>J);
i7 : time inverseOfMap(phi, AssumeDominant=>true, Verbose=>false) -- Function from "RationalMaps"
     -- used 0.126314 seconds
i8 : time invertBirMap phi;  -- Function from "Cremona"
     -- used 0.375856 seconds
i9 : isSameMap(o7, o8)
o9 = true
```

Note both these examples were taken directly from the documentation of `Cremona`. On the other hand, when the source and target are not projective space, or the map does not have maximal linear rank, in our experience, our functions appear to be faster. Here is an example where the source and target are not projective space.

```
i1 : needsPackage "Divisor"; loadPackage "RationalMaps"; loadPackage("Cremona", Reload=>true);
i5 : C = QQ[x,y,z]/(x^4+x^2*y*z+y^4+z^3*x);
i6 : Q = ideal(y,x+z); --a point on our curve
i7 : (f3 = mapOntoImage(mapToProjectiveSpace(12*divisor(Q)));
i8 : (time g1 = inverseOfMap(f3, Verbose=>false);); -- Function from "RationalMaps"
     -- used 1.1996 seconds
i9 : (time g2 = invertBirMap(f3);); -- Function from "Cremona"
     -- used 13.7981 seconds
i10 : (g3 = map(target g1, source g2, sub(matrix g2, target g1));
i11 : isSameMap(g3, g1)
o11 = true
```

We also include an example where the map does not have maximal linear rank.

```
i1 : loadPackage "RationalMaps"; loadPackage "Cremona";
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 inverseOfMap(phi, AssumeDominant=>true,CheckBirational=>false, MinorsCount=>50000));
     -- Function from "RationalMaps"
     -- used 5.80638 seconds
i9 : (time h = invertBirMap(phi)); -- Function from "Cremona"
     -- used 71.8998 seconds
i10 : isSameMap(o5, h)
```

```
o10 = true
```

Department of Mathematics, Department of Mathematics, 275 TMCB Brigham Young University, Provo, UT 84602
*E-mail address*: cjamesbott@gmail.com

Department of Mathematics, Federal University of Rio de Janeiro, Brazil
*E-mail address*: hamid@im.ufrj.br

Department of Mathematics, University of Utah, Salt Lake City, UT 84112
*E-mail address*: schwede@math.utah.edu

Department of Mathematics, University of Utah, Salt Lake City, UT 84112
*E-mail address*: smolkin@math.utah.edu