

ELIMINATION TEMPLATES IN MACAULAY2

MANAV BATAVIA, CHENG CHEN, ANNA NATALIE CHLOPECKI, TIMOTHY DUFF, WILLIAM HUANG,
AOLONG LI, AND WANCHUN SHEN

Abstract: We introduce the package `EliminationTemplates` for the Macaulay2 computer algebra system, which provides tools for constructing automatic solvers for families of zero-dimensional radical ideals depending on algebraically independent parameters. This article provides a self-contained description of how elimination templates are constructed for such families and their specialization properties. Additionally, we describe the main functionality and datatypes provided by our package, and illustrate its usage on several examples, including applications from computer vision from which elimination templates originated.

1. INTRODUCTION

Finding the solutions to a system of polynomial equations is a common problem throughout mathematics and its various applications. This problem may be very difficult in general. However, it is quite common in practice that these systems have a particular structure—typically, they depend on some number of input parameters, such that generic parameter values produce polynomials generating a zero-dimensional ideal.

In Section 2.1, we recall the well-known fact that solving zero-dimensional polynomial systems can be reduced to solving a (non-symmetric) eigenvalue problem. However, implementing this reduction in practice demands significant care, as it generally involves exact Gröbner basis computation, whereas eigenvalue computation is inherently inexact. Naively combining these two disparate computational paradigms may fail, for instance, due to (1) the discontinuity of Gröbner bases to perturbations in polynomials’ coefficients [16], or by producing eigenvalue problems which may be (2) poorly-conditioned [1, 6, 17] or (3) simply too large to be solved at all. Failure (1), however, may be addressed by seeking reductions that are specific to the parametric family of interest; these may be stable to perturbations in the parameters of interest, rather than the coefficients they determine. Failures of types (2)/(3) are generally unavoidable, but may be mitigated in practice by experimenting with different reduction strategies and identifying structural features of parametric families that enable solving more stably/efficiently than black-box approaches.

In this paper, we provide a self-contained introduction to the *elimination template* approach to solving 0-dimensional systems, and describe an implementation of this approach in the `EliminationTemplates` package for the computer algebra system `Macaulay2` [7]. Elimination templates are an incarnation of the Gröbner trace method algorithm [19], with extensive developments tailored towards applications in computer vision. In these applications, elimination templates are used to construct *minimal solvers* which repeatedly solve systems from a fixed parametric family for many different parameter values depending on data. The outputs of minimal solvers are later used to remove outlier data and initialize least-squares estimation.

To set the stage, we pose a simple minimal problem which can be solved using elimination templates.

Example 1.1. Consider two lines in complex projective space, $L_1, L_2 \subset \mathbb{P}^3$, which are known to intersect. Our goal will be to recover a linear map $P : \mathbb{P}^3 \rightarrow \mathbb{P}^2$ from L_1, L_2 , and the images of these lines, $\ell_1 = P(L_1), \ell_2 = P(L_2) \subset \mathbb{P}^2$. The map P will model a camera with variable focal length rotating around the origin. Letting f be the focal length of our camera lens, we may represent P by a matrix of the form

$$\begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} (R \mid \mathbf{0}),$$

where R is a 3×3 rotation matrix such that $RR^T = I_3$ and $\det(R) = 1$, i.e. $R \in \text{SO}(3)$. Notice that the fourth column of the 3×4 augmented matrix $(R \mid \mathbf{0})$ is made up of 0’s because the center of the camera is

at the origin.

Alternatively, P may be described using a *scaled quaternion parametrization*, which is the map

$$(1) \quad Q : \mathbb{P}^3 \rightarrow \mathbb{P}(\mathbb{C}^{3 \times 3})$$

$$[w : x : y : z] \mapsto \begin{pmatrix} w^2 + x^2 - y^2 - z^2 & 2xy - 2wz & 2wy + 2xz \\ 2xy + 2wz & w^2 - x^2 + y^2 - z^2 & -2wx + 2yz \\ -2wy + 2xz & 2wx + 2yz & w^2 - x^2 - y^2 + z^2 \end{pmatrix}$$

Note that for a real-valued point in the domain $[w : x : y : z] \in \mathbb{P}_{\mathbb{R}}^3$, we may always assume homogeneous coordinates are normalized so that they satisfy

$$(2) \quad w^2 + x^2 + y^2 + z^2 = 1.$$

With this normalization, the matrix of (1) lies in $\mathrm{SO}_{\mathbb{R}}(3)$. Conversely, for any real rotation $R \in \mathrm{SO}_{\mathbb{R}}(3)$, there exists a point in $[w : x : y : z] \in \mathbb{P}_{\mathbb{R}}^3$ with $Q([w : x : y : z]) = R$.

In Example 4.2, we illustrate how to construct synthetic parameters $(L_1, L_2, \ell_1, \ell_2)$ and generate the elimination template of the corresponding zero-dimensional ideal, generated by 5 equations (including (2)) in the 5 unknowns (f, w, x, y, z) . We also show how the resulting template can be reused when presented with new parameters.

To date, software for generating elimination templates has been implemented in `Matlab` [12, 13] (using `Macaulay2` as a back-end) and `Maple` [14, 15]. Once an elimination template is determined, it is straightforward to derive a minimal solver in the implementer's language of choice. Finally, we note the extensive literature [1, 9, 10, 11, 12, 14, 15] that might be used for future improvements of our package.

Section 2 of this paper describes our approach to constructing elimination templates. Our exposition largely follows the approach given in [15], but is self-contained, relying only on basic algebraic geometry at the level of [3]. Two novel aspects of our treatment are the construction of templates which are universal for generic linear forms and an emphasis on specialization properties for parametric systems. In Section 3, we describe the main datatypes and functionality provided by our package. Finally, in Section 4, we illustrate the application of our package to Section 1.1 and related problems.

2. BACKGROUND ON ELIMINATION TEMPLATES

2.1. Constructing Elimination Templates. Let $I \subset R := \mathbb{K}[x_1, \dots, x_n]$ be a zero-dimensional ideal over a field \mathbb{K} , so that the quotient ring R/I is a finite-dimensional vector space over \mathbb{K} . When I is radical, the vector space dimension $\dim_{\mathbb{K}}(R/I)$ is equal to the number of points in the variety $V_{\overline{\mathbb{K}}}(I)$, where $\overline{\mathbb{K}} \supset \mathbb{K}$ is an algebraic closure of \mathbb{K} . We will be largely interested in ideals defined over the field $\mathbb{K} = \mathbb{Q}$ of rational numbers, as well as points in the associated variety $V(I)$, whose coordinates belong to the fields $\overline{\mathbb{Q}} \subset \mathbb{C}$. In practice, these coordinates are approximated in floating-point. Two other important cases arise when \mathbb{K} is a function field generated by algebraically independent indeterminates (see Section 2.2) and when \mathbb{K} is a finite field of prime characteristic (as illustrated in Example 4.4.)

Fix a monomial order $<$ on R , so that R/I has an ordered \mathbb{K} -basis \mathcal{B} of standard monomials, namely

$$(3) \quad \mathcal{B} = \{x^{\alpha_1}, \dots, x^{\alpha_d}\}, \quad x^{\alpha_i} \notin \mathrm{in}_{<}(I) \quad \forall i = 1, \dots, d, \quad \mathbf{0} = \alpha_d < \dots < \alpha_2 < \alpha_1.$$

We denote the sorted vector of standard monomials by

$$(4) \quad v(\mathcal{B}) = (x^{\alpha_1} \quad \dots \quad x^{\alpha_d})^T \in R^{d \times 1}.$$

Multiplication by any element $f \in R$ determines a \mathbb{K} -linear transformation

$$(5) \quad m_f : R/I \rightarrow R/I$$

$$(6) \quad [g] \mapsto [fg].$$

Fixing such an f , we may write

$$(7) \quad \begin{aligned} m_f([x^{\alpha_1}]) &= a_{11}[x^{\alpha_1}] + \dots + a_{1d}[x^{\alpha_d}], \\ &\vdots \\ m_f([x^{\alpha_d}]) &= a_{d1}[x^{\alpha_1}] + \dots + a_{dd}[x^{\alpha_d}], \end{aligned}$$

where $A = (a_{ij}) \in \mathbb{K}^{d \times d}$ is the *action matrix* associated to the triple $(I, <, f)$.

The following result is the basis of reducing 0-dimensional solving to solving the eigenvalue problem.

Theorem 2.1. (See e.g. [3, §2.4]) For any zero-dimensional radical ideal I , the eigenvalues of the action matrix A are precisely the values of f on the points of $V(I)$. If f takes $d = \dim_{\mathbb{K}}(R/I)$ distinct values on these points, then every eigenvalue of A has algebraic multiplicity 1, and every eigenvector of A is a scalar multiple of $v(\mathcal{B})|_{x=P}$, the $d \times 1$ vector obtained by evaluating standard monomials at some point $P \in V(I)$.

Gröbner basis algorithms are typically needed to determine the quotient ring basis \mathcal{B} and action matrix A . The idea behind *elimination templates* is this: for fixed $<$ and $f \in R$ (typically a general linear form), and two ideals $I, J \subset R$ of a “similar structure” (see Section 2.2 for a precise statement), the full cost of computing a Gröbner basis need only be incurred once for the ideal I , after which the action matrix of J may be determined using just linear algebra over the field k .

Let us assume that $f \in R$ is a general linear form so that, in particular, f will have the point separation property required by Theorem 2.6. Our construction of an elimination template adjoins a new variable s to R . By abuse of notation, we identify elements of R with their images under the inclusion map $\iota : R \hookrightarrow R[s]$. We equip $R[s]$ with the *elimination order* $<_s$ defined by

$$(8) \quad s^a \mathbf{x}^\alpha <_s s^b \mathbf{x}^\beta \quad \text{if} \quad \begin{cases} a < b, & \text{or} \\ a = b & \text{and} \quad \mathbf{x}^\alpha < \mathbf{x}^\beta. \end{cases}$$

Finally, we define the *graph ideal* associated to I and f ,

$$(9) \quad I_{f \rightarrow s} := I + \langle s - f \rangle \subset R[s].$$

Passing from I to the graph ideal $I_{f \rightarrow s}$ preserves the action matrix in a very precise sense.

Proposition 2.2. The triples $(I, <, f)$ and $(I_{f \rightarrow s}, <_s, s)$ have the same action matrix.

Proof. By construction, the basis of standard monomials (3) of R/I is also a basis of standard monomials of $R[s]/I_{f \rightarrow s}$, and the linear relations (7) lift to express the action of m_s on this basis. \square

Now, suppose the ideal $I = \langle f_1, \dots, f_r \rangle$ is given in terms of a fixed generating set $F = \{f_1, \dots, f_r\}$, and that I has a reduced Gröbner basis $G = \{g_1, \dots, g_t\}$ with respect to $<$. The vectors

$$(10) \quad v(F) := (f_1 \ \cdots \ f_r)^T \in R^{r \times 1}, \quad v(G) := (g_1 \ \cdots \ g_t)^T \in R^{t \times 1}$$

may be related by a *change matrix*,

$$(11) \quad H_{F \rightarrow G} \in R^{t \times r} \quad \text{s.t.} \quad v(G) = H_{F \rightarrow G} \cdot v(F).$$

From the basis of standard monomials (3) for I with respect to $<$, we may consider the following *vector of action relations* in the ring R ,

$$(12) \quad v(A, R) := f \cdot v(\mathcal{B}) - A \cdot v(\mathcal{B}).$$

By construction, each entry of $v(A, R)$ belongs to the ideal I . Thus, using multivariate polynomial division, there exists another polynomial matrix

$$(13) \quad H_{G \rightarrow A} \in R^{d \times t} \quad \text{s.t.} \quad v(A, R) = H_{G \rightarrow A} \cdot v(G).$$

Now, we set

$$(14) \quad H_0 = H_{G \rightarrow A} \cdot H_{F \rightarrow G} \in R^{d \times r}.$$

Working in the extended polynomial ring $R[s]$, we may write the action relations as follows:

$$(15) \quad s \cdot v(\mathcal{B}) - A \cdot v(\mathcal{B}) = (s - f)v(\mathcal{B}) + v(A, R) = (s - f)v(\mathcal{B}) + H_0 \cdot v(F).$$

Equation (15) shows that the action of A on standard monomials can be recovered by multivariate polynomial arithmetic, provided that the Gröbner basis G has already been computed. The construction of an elimination template recasts this multivariate polynomial computation as row-reduction of an associated Macaulay-type matrix whose entries lie in the ground field \mathbb{K} .

Remark 2.3. For an ideal J “similar” to I , given the Macaulay-type matrix H_0 for I , the action matrix for J can be computed without any Gröbner basis computation for J . See Section 2.2 for details.

Remark 2.4. Observe that equation (15) still holds if we replace H_0 with any matrix $H \in R^{d \times r}$ of the form $H = H_0 + \Theta \cdot H_1$, where $\Theta \in R^{d \times p}$ and the rows of $H_1 \in R^{p \times r}$ generate the first syzygy module of F :

$$(16) \quad s \cdot v(\mathcal{B}) - A \cdot v(\mathcal{B}) = (s - f)v(\mathcal{B}) + H \cdot v(F).$$

This observation is the basis of strategies for reducing the size of an elimination template—see Section 3.2.

For any matrix $H \in R^{d \times r}$ and each $i = 1, \dots, r$ we let $\mathcal{S}_i(H)$ denote the i -th *shift set*, consisting of all monomials appearing in some entry of the i -th column of H . We denote the union of all shift monomials by

$$(17) \quad \mathcal{S}(H) := \mathcal{S}_1(H) \cup \dots \cup \mathcal{S}_r(H).$$

Now, let $\mathcal{M}(H)$ denote the set of all monomials appearing in either \mathcal{B} , $s \cdot \mathcal{B}$, or in the support of some shifted generator of the form $x^{\beta_{ij}} f_i$ with $x^{\beta_{ij}} \in \mathcal{S}_i(H)$. Let us partition $\mathcal{M}(H)$ into three sets,

$$\begin{aligned} (\text{the } \textit{basic} \text{ monomials}) & \quad \mathcal{B}(H) := \mathcal{B}, \\ (\text{the } \textit{reducible} \text{ monomials}) & \quad \mathcal{R}(H) := s \cdot \mathcal{B}, \\ (\text{the } \textit{excessive} \text{ monomials}) & \quad \mathcal{E}(H) := \mathcal{M}(H) \setminus (\mathcal{B}(H) \cup \mathcal{R}(H)). \end{aligned}$$

We may form the following Macaulay-type matrix

$$(18) \quad \begin{array}{c} x^{\beta_{11}} f_1 \\ \vdots \\ x^{\beta_{r1}} f_r \\ \vdots \\ (s - f)v(\mathcal{B}) \end{array} \begin{array}{c} \mathcal{E}(H) \\ \mathcal{R}(H) \\ \mathcal{B}(H) \end{array} \in \mathbb{K}^{(d + \#\mathcal{S}(H)) \times (2d + \#\mathcal{E}(H))}.$$

Now, using the polynomial identity (16), we may express the action relations as linear combinations of the rows of this matrix, which thus has a row-echelon form given block-wise by

$$(19) \quad \left(\begin{array}{c|c|c} * & \mathbf{0} & * \\ \mathbf{0} & \mathbf{I}_d & -A \end{array} \right).$$

We see that, in order to recover the action matrix A with row operations over \mathbb{K} , the columns corresponding to the reducible monomials $\mathcal{R}(H)$ may harmlessly be dropped.

Definition 2.5. Let $I = \langle f_1, \dots, f_s \rangle$ be a zero-dimensional ideal in the polynomial ring R , with $d \times d$ action matrix A for some monomial order $<$, and $H \in R^{s \times d}$ any matrix satisfying (16). We define the $(\#\mathcal{S}(H) + d) \times (\#\mathcal{E}(H) + d)$ *template matrix* M whose first $\#\mathcal{S}(H)$ rows are the coefficient vectors of each $x^{\beta_{ij}} f_i$ for each $x^{\beta_{ij}} \in \mathcal{S}_i$, and whose last d rows are the the coefficient vectors of $(s - f)v(\mathcal{B})$. The columns of M are divided into consecutive blocks $\mathcal{E}(H)$ and $\mathcal{B}(H)$, each in decreasing order with respect to $<$. Pictorially,

$$(20) \quad M = \begin{array}{c} x^{\beta_{11}} f_1 \\ \vdots \\ x^{\beta_{s1}} f_s \\ \vdots \\ (s - f)v(\mathcal{B}) \end{array} \begin{array}{c} \mathcal{E}(H) \\ \mathcal{B}(H) \end{array} \in \mathbb{K}^{(d + \#\mathcal{S}(H)) \times (d + \#\mathcal{E}(H))}.$$

Let us emphasize that both the action matrix A and the template matrix M associated to an ideal I are determined by several choices: namely, the input generators F , the monomial order $<$, and the matrix H in (16). These choices may have significant influence on a solver in practice, as they effect both the runtime of row-reducing the template matrix M and the conditioning of the eigenvalue problem for A .

2.2. Specializing Elimination Templates. Suppose now that our ground field is the complex function field $\mathbb{K} = \mathbb{C}(p_1, \dots, p_m)$. We think of $p = (p_1, \dots, p_m)$ as parameters which specify family of ideals. We would like to formalize the intuition that elimination templates for zero-dimensional ideals $I \subset \mathbb{K}[x_1, \dots, x_n]$ are preserved under specializations $\Psi_P : \mathbb{K}[x_1, \dots, x_n] \rightarrow \mathbb{C}[x_1, \dots, x_n]$ for generic $P \in \mathbb{C}^m$. To do this, suppose that I is the extension of an ideal $\tilde{I} \subset S := \mathbb{C}[x_1, \dots, x_n, p_1, \dots, p_m]$ under the inclusion map

$$(21) \quad \iota : S \hookrightarrow \mathbb{K}[x_1, \dots, x_n].$$

Let us furthermore assume that the parameters p are independent in the sense that

$$(22) \quad \tilde{I} \cap \mathbb{C}[p_1, \dots, p_m] = \{0\}.$$

Under these conditions, Gröbner bases for a particular class of monomial orders have well-known specialization properties. We quote the following result from [2, Ch. 6, §3, Proposition 1].

Theorem 2.6. Let \tilde{I} be an ideal in S such that (22) holds. Fix an elimination order $<_p$ such that $x^\alpha <_p x^\beta$ implies $x^\alpha p^\gamma <_p x^\beta$ for all γ . Let $\tilde{G} = \{g_1, \dots, g_t\}$ be a reduced Gröbner basis of \tilde{I} .

For $i = 1, \dots, t$, write $g_i \in \tilde{G}$ in the form

$$g_i = h_i(p_1, \dots, p_m)x^{\alpha_i} + (\text{terms } <_p x^{\alpha_i}),$$

where $h_i \in \mathbb{C}[p_1, \dots, p_m]$ is nonzero. Then, for all $P = (a_1, \dots, a_m) \in \mathbb{C}^m \setminus V(h_1, \dots, h_t)$, the set

$$G = \{g_1(x_1, \dots, x_n, a_1, \dots, a_m), \dots, g_t(x_1, \dots, x_n, a_1, \dots, a_m)\}$$

is a Gröbner basis of $\Psi_P(I)$ with respect to the order $<$ on $\mathbb{C}[x_1, \dots, x_n]$ obtained by restricting $<_p$.

Now, let M be a template matrix for $I \subset \mathbb{K}[x_1, \dots, x_n]$, and A the corresponding action matrix. Let us assume a fixed row-echelon form for the template matrix M of the form

$$(23) \quad E_1(p) \cdots E_\ell(p) \cdot M = \left(\begin{array}{c|c} * & * \\ \mathbf{0} & -A \end{array} \right),$$

where $E_1(p), \dots, E_\ell(p) \in \text{GL}(d + \#\mathcal{S}(H), \mathbb{K})$ are elementary matrices. We make the following observations:

- (1) For $1 \leq i \leq \ell$, each of the conditions $\det(E_i(p)) \neq 0$ defines a nonempty Zariski-open subset of \mathbb{C}^m . Let $\mathcal{U}_1 \subset \mathbb{C}^m$ denote the intersection of all such sets, so that specialization at generic $P \in \mathcal{U}_1$ preserves the row echelon form (23).
- (2) Theorem 2.6 defines a nonempty Zariski-open set $\mathcal{U}_2 \subset \mathbb{C}^m$ for which Gröbner bases specialize; in particular, this shows that the specialization of the action matrix for I remains an action matrix for the specialized ideal $\Psi_P(I)$.

Thus, setting $\mathcal{U} = \mathcal{U}_1 \cap \mathcal{U}_2$, we obtain the following conclusion.

Theorem 2.7. For $\mathbb{K} = \mathbb{C}(p_1, \dots, p_m)$ and $I \subset \mathbb{K}[x_1, \dots, x_n]$ any zero-dimensional ideal obtained as the extension of $\tilde{I} \subset S$ such that (22) holds, and for any monomial order $<$ there exists a dense Zariski-open subset $\mathcal{U} \subset \mathbb{C}^m$ such that the elimination template for I constructed in Definition 2.5 specializes to an elimination template for the specialized ideal $\Psi_P(I)$ for all $P \in \mathcal{U}$.

Example 2.8. To demonstrate Theorem 2.7, we consider the parametric ideal

$$I = (x^2 + ay^2 - 1, xy - b) \in \mathbb{C}(a, b)[x, y].$$

The code below illustrates an echelon form (23) involving parameters a and b .

```
FF = frac(QQ[a,b,c,d]);
R = FF[x, y, MonomialOrder => Lex];
l = x - 2*y;
I = ideal(x^2 + a*y^2 - 1, x*y - b);
needsPackage "EliminationTemplates";
ET = eliminationTemplate(l, I);
M = getTemplateMatrix ET
(P, L, U) = LUdecomposition M;
```

We may inspect the template matrix and its LU factors as follows:

i7 : M

```
o7 = {-4} | 1 0 0 0 0 0 a 0 0 0 0 0 -1 0 0 |
      {-3} | 0 1 0 0 0 0 0 0 0 0 0 a 0 -1 0 |
      {-4} | 1 0 0 0 -b 0 0 0 0 0 0 0 0 0 0 |
      {-3} | 0 1 0 0 0 -b 0 0 0 0 0 0 0 0 0 |
      {-4} | 0 0 1 0 0 0 0 0 0 0 0 0 -b 0 0 |
      {-3} | 0 0 0 1 0 0 0 0 0 0 0 0 -b 0 0 |
      {-2} | 0 0 0 0 1 0 0 0 0 0 0 0 0 0 -b |
      {-3} | 0 0 -1 0 0 0 2 1 0 0 0 0 0 0 0 |
      {-2} | 0 0 0 -1 0 0 0 0 1 0 0 2 0 0 0 |
      {-1} | 0 0 0 0 -1 0 0 0 0 1 0 0 2 0 0 |
      {0}  | 0 0 0 0 0 -1 0 0 0 0 1 0 0 2 0 |
```

```
11      15
o7 : Matrix FF <-- FF
```

i8 : L

```
o8 = | 1 0 0 0 0 0 0 0 0 0 0 0 0 |
      | 0 1 0 0 0 0 0 0 0 0 0 0 0 |
      | 0 0 1 0 0 0 0 0 0 0 0 0 0 |
      | 0 0 0 1 0 0 0 0 0 0 0 0 0 |
      | 1 0 0 0 1 0 0 0 0 0 0 0 0 |
      | 0 1 0 0 0 1 0 0 0 0 0 0 0 |
      | 0 0 0 0 (-1)/b 0 1 0 0 0 0 0 |
      | 0 0 -1 0 0 0 (-2b)/a 1 0 0 0 0 |
      | 0 0 0 -1 0 0 0 0 0 1 0 0 0 |
      | 0 0 0 0 1/b 0 -1 0 0 1 0 0 0 |
      | 0 0 0 0 0 1/b 0 0 0 0 1 0 0 |
```

```
11      11
o8 : Matrix FF <-- FF
```

i9 : U

```
o9 = | 1 0 0 0 0 0 a 0 0 0 0 0 -1 0 0 |
      | 0 1 0 0 0 0 0 0 0 0 0 a 0 -1 0 |
      | 0 0 1 0 0 0 0 0 0 0 0 0 -b 0 0 |
      | 0 0 0 1 0 0 0 0 0 0 0 0 0 -b 0 |
      | 0 0 0 0 -b 0 -a 0 0 0 0 0 1 0 0 |
      | 0 0 0 0 0 -b 0 0 0 0 0 -a 0 1 0 |
      | 0 0 0 0 0 0 (-a)/b 0 0 0 0 0 1/b 0 -b |
      | 0 0 0 0 0 0 0 1 0 0 0 0 (-ab+2)/a 0 (-2b2)/a |
      | 0 0 0 0 0 0 0 0 1 0 0 2 0 -b 0 |
      | 0 0 0 0 0 0 0 0 0 1 0 0 2 0 -b |
      | 0 0 0 0 0 0 0 0 0 0 1 a/b 0 (2b-1)/b 0 |
```

```
11      15
o9 : Matrix FF <-- FF
```

Inspecting the entries of U , we should enforce the condition $ab \neq 0$ in order for the elimination template to specialize. Furthermore, computing a Gröbner basis with respect to the given lexicographic order shows that the same conditions are sufficient for specialization, giving us the open set

$$\mathcal{U} = \{(a, b) \in \mathbb{C}^2 \mid ab \neq 0\}$$

promised by Theorem 2.7. It may also be of interest to consider a smaller open subset of specializations $P = (a, b) \in \mathcal{U}$ for which the parametric action matrix A has four distinct eigenvalues. We find A below:

```
i10 : getActionMatrix ET
o10 = {-3} | 0      (ab-2)/a 0      2b2/a |
      {-2} | -2    0      b      0      |
      {-1} | 0     -2     0      b      |
      {0}  | -a/b  0      (-2b+1)/b 0      |

              4      4
o10 : Matrix FF  <-- FF
```

By computing the discriminant of its characteristic polynomial, we see that this parametric action matrix has four distinct eigenvalues for all parameters in the open set

$$\{(a, b) \in \mathcal{U} \mid a - 4 \neq 0, 4ab^2 - 1 \neq 0, ab + 4b - 2 \neq 0\}.$$

2.3. Using Elimination Templates to Solve 0-Dimensional Systems. For completeness, we briefly describe how Theorem 2.1 can be used to find the points in $V(I)$. We retain the notation established in Theorem 2.1. Assume that $f \in R$ is chosen such that all eigenvalues of the action matrix have algebraic multiplicity 1. If λ is one such eigenvalue, then Theorem 2.1 gives a corresponding eigenvector

$$v = v(\mathcal{B})_{x=a} = (a^{\alpha_1}, \dots, a^{\alpha_{d-1}}, 1)$$

for some $a = (a_1, \dots, a_n) \in V(I)$. When f is a general linear form, we may assume its support contains every variable in the ring R . Thus, using the template construction of Definition 2.5, we see that every variable appears either in the set of basic variables $\mathcal{B}(H)$ or in the set of excessive monomials $\mathcal{E}(H)$. In the former case, $x_i \in \mathcal{B}(H)$, and the coordinate a_i can be located as a coordinate of the eigenvector v .

In the latter case, taking $H = H_0$, any variable $x_i \in \mathcal{E}(H_0)$ must still appear as a pivot column in M . In other words, the template matrix encodes an equation linear in the variable x_i . Let us list all such variables in order, x_{i_1}, \dots, x_{i_l} with $i_1 < \dots < i_l$. For $j = 1, \dots, l$, there are elements $g_j \in G$ such that

$$g_j = x_{i_j} + \text{terms involving } x_i \text{ for } i > i_j.$$

On evaluating at $P = (a_1, \dots, a_n)$, we obtain

$$0 = a_{i_j} + \text{terms involving } a_i \text{ for } i > i_j.$$

We can first solve for a_{i_l} as the values of a_i for $i > i_l$ are known to us from previous computations. Once we know a_{i_l} , we can solve for $a_{i_{l-1}}$, and so on.

3. PACKAGE DESIGN AND FUNCTIONALITY

In this section, we will describe the functionality provided by the `EliminationTemplates` package and various strategies which implement the constructions of the previous section.

Thus, in general, all coordinates of all points in $V(I)$ may be recovered by row-reducing M and computing the eigenvalues of A . For numerical stability, we row-reduce M using LU decomposition with partial pivoting.

3.1. Data Structures and Functions. Our package consists of one main data type, `EliminationTemplate`. One way to call the corresponding `eliminationTemplate` constructor is to pass arguments of class `RingElement`, representing the action variable f defining the action matrix, and a zero-dimensional ideal I , of class `Ideal`. Objects in this class store the necessary data required for an elimination template computation. Objects of class `EliminationTemplate` are designed to cache information such as the monomial basis \mathcal{B} , the template matrix M , and the action matrix A , so that they only need to be computed once.

We illustrate our package data structures and functionality with a running example. First, we create a new template object, as shown in the code below:

```

i1: needsPackage "EliminationTemplates";
i2: R = QQ[x,y];
i3: I = ideal(x^4+x*y+y^2-3, x^2*y+y^3-2);
i4: E = eliminationTemplate(x,I)
o4: action variable: x

```

The method `getTemplate` accepts an `EliminationTemplate` object and outputs a tuple of objects of type `List` containing a `ShiftSet`, as in (17), and `MonomialPartition` associated to the matrix H in (14). Alternatively, we can call `getTemplate` using the inputs `RingElement`, `Matrix`, which contains a basis for R/I , and `Ideal` I . The user is not expected to interact with the internal data types `ShiftSet` and `MonomialPartition`, which are cached inside of the `EliminationTemplate` object. The `getTemplateMatrix` method takes in an object of class `EliminationTemplate` and returns the associated template matrix.

```

i5: getTemplateMatrix(E)
o5: | 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 -3 0 0 0 |
    | 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 -3 0 |
    | 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 -3 |
    | 1 0 1 0 0 0 -2 0 0 0 0 0 0 0 0 0 0 0 0 0 |
    | 0 1 0 1 0 0 0 0 0 0 0 0 0 0 -2 0 0 0 0 0 |
    | 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 -2 0 |
    | 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 -2 |

```

The method `templateSolve` takes as input either an object of class `EliminationTemplate` or a `RingElement` f and an `Ideal` I with common polynomial ring R , and outputs a list of solutions. Each solution is a list of complex coordinates, one per ring variable. We exemplify the use of this function below.

```

i6: sols1 = templateSolve(E);
i7: sols2 = templateSolve(x,I);
i8: assert(all(sols1, x -> 1e-6 > norm sub(sub(gens I, QQ[gens R]), matrix{x})))
o8: true

```

Using `copyTemplate`, we can create a copy of an `EliminationTemplate`, resulting in an `EliminationTemplate` instance defined using the same action variable but a different defining ideal.

```

i9: J = ideal(x^4 + 2*x*y + y^2-1, 3*x^2*y + y^3 - 5);
i10: F = copyTemplate(E, J);
i11: sols3 = templateSolve(F);
i12: assert(all(sols3, x -> 1e-6 > norm sub(sub(gens J, QQ[gens R]), matrix{x})))
o12: true

```

3.2. Additional Strategies. There are three strategies which we allow the user to manually input, each which determine the matrix H in equation 14, namely

1. the default strategy
2. the strategy "Larsson", implementing the syzygy-based reduction of H [12]
3. the strategy "Greedy", implementing the linear algebra-based reduction of H introduced in [15].

As in Remark 2.4, equation (16) remains valid if H is replaced by any matrix of the form

$$H = H_0 + \Theta \cdot H_1,$$

where the rows of H_1 generate the first syzygy module of F .

In the default strategy, we simply set $H = H_0$ as in Equation (14). No reduction is performed, and the resulting template is typically far from optimal.

In the Larsson strategy, we exploit the fact that H_0 can be replaced by $H_0 + \Theta \cdot H_1$, where the rows of H_1 generate the first syzygy module of F . We compute a Gröbner basis of the syzygy module with respect to a degree-first term order and reduce each row of H_0 to its normal form. This minimizes the maximum degree among equivalent representations [12], which often reduces template size in practice.

In the greedy strategy, the matrix Θ is selected greedily, with the goal of reducing the number of polynomial shifts and excessive monomials, and hence the size of the template at each step. This is more expensive than the Larsson strategy, but directly targets template size. We refer to [15] for more details. Example 4.4 below illustrates how different strategies may be used to reduce template size.

4. EXAMPLES

Example 4.1. As a basic demonstration, suppose we wanted to find all complex solutions to the system

$$(24) \quad \begin{cases} x^2 + y^2 - 1 & = 0 \\ x^2 + y^3 + xy - 2 & = 0 \end{cases}$$

To do this, we use the linear form $f = x + 4y$ to construct our elimination template.

```
i1: R = QQ[x,y];
i2: I = ideal(x^2+y^2-1,x^2+y^3+x*y-2);
i3: E = eliminationTemplate(x+4*y, I);
```

The elimination template yields a set of solutions, which we can then check

```
i4: sols = templateSolve(E);
i5: all(sols, x -> 1e-6 > norm sub(sub(gens I, QQ[gens R]), matrix{x}))
o5: true
```

We now illustrate how `copyTemplate` can be used to exploit the specialization property described in Theorem 2.7, by using the previously-constructed template to solve the new system given as follows:

$$(25) \quad \begin{cases} x^2 + y^2 - 2 & = 0 \\ x^2 + y^3 + 3xy - 5 & = 0 \end{cases}$$

To solve this new system, we simply run `copyTemplate` and then `templateSolve`.

```
i6: J = ideal(x^2+y^2-2,x^2+y^3+3*x*y-5);
i7: F = copyTemplate(E, J);
i8: sols = templateSolve(F);
i9: all(sols, x -> 1e-6 > norm sub(sub(gens J, QQ[gens R]), matrix{x}))
o9: true
```

Example 4.2. We now explain in detail how our package may be used to solve the camera pose problem introduced in Example 1.1. First we generate synthetic data representing two general lines L_1, L_2 in \mathbb{P}^3 :

```
i1: FF = QQ;
i2: a = random(FF^3, FF^1) || matrix{{1}};
i3: b1 = random(FF^3, FF^1) || matrix{{1}};
i4: b2 = random(FF^3, FF^1) || matrix{{1}};
```

We implement a function `Q2R` which implements the quaternion map (1)

```
i5: Q2R = (w,x,y,z) -> matrix{
  {w^2+x^2-y^2-z^2, 2*x*y-2*w*z, 2*w*y+2*x*z},
  {2*x*y+2*w*z, w^2-x^2+y^2-z^2, -2*w*x+2*y*z},
  {-2*w*y+2*x*z, 2*w*x+2*y*z, w^2-x^2-y^2+z^2}
};
```

Next, we need to give two lines $\ell_1, \ell_2 \subset \mathbb{P}^2$. In practice these data would come from experiments using a physical camera which would ensure the existence of a solution. For the purpose of generating a template, however, we may generate a ground-truth camera and use it to generate the line images:

```

i6: (w0, x0, y0, z0, f0) := (random FF, random FF, random FF, random FF, random FF);
i7: R0 := Q2R(w0, x0, y0, z0);
i8: P0 := diagonalMatrix{f0, f0, 1} * (R0 | matrix{{0},{0},{0}});
i9: l1 = gens ker transpose(P0 * (a | b1));
i10: l2 = gens ker transpose(P0 * (a | b2));

```

Here w_0, x_0, y_0, z_0, f_0 are the camera parameters we wish to recover which in practice would come from a physical camera. In the above code, the lines ℓ_i are represented using implicit equations. Using this, we generate the camera matrix P and the ideal which represents the constraints on P .

```

i11: S = FF[w..z, f];
i12: R = Q2R(w,x,y,z);
i13: P = diagonalMatrix{f,f,1} * (R | matrix{{0},{0},{0}});
i14: I = ideal(w^2+x^2+y^2+z^2-1, transpose l1 * P * a, transpose l1 * P * b1,
             transpose l2 * P * a, transpose l2 * P * b2);

```

Now we can use our package to solve for the camera parameters w_0, x_0, y_0, z_0, f_0 .

```

i15: l = random(1, ring I);
i16: ET = eliminationTemplate(l, I);
i17: sols = templateSolve ET;

```

Finally, we verify that the ground truth camera is recovered as one of the solutions.

```

i18: groundTruthSolution = matrix{(1/sqrt(w0^2+x0^2+y0^2+z0^2))*{w0,x0,y0,z0} | {f0}};
i19: position(sols, x -> norm(matrix{x} - groundTruthSolution) < 1e-10)
o19: {0}

```

In this formulation, the elimination template solver is efficient with the default settings, and the action matrix is 16×16 . Inspecting the solutions, we see that there are a number of symmetries; these arise (for example) due to the 2-1 nature of the quaternion map (1). Thus, symbolic symmetry-reduction techniques [10] and analysis of this minimal problem's Galois group [4, 5] would likely lead to further improvements in efficiency. We leave the task of integrating these tools into the `EliminationTemplates` package as future work.

Example 4.3. The problem of 5-point essential matrix estimation is an important and well-known minimal problem, which is widely used for estimating the relative orientation of two calibrated cameras. The first practical solution to this problem, due to Nistér [18], involved a careful derivation reducing this problem to computing the roots of a degree-10 polynomial. The code below provides one means of automating this reduction. Our goal is to estimate three variables x, y, z representing the matrix

$$(26) \quad E = xE_1 + yE_2 + zE_3 + E_4,$$

where we are given the 3×3 E_1, E_2, E_3, E_4 . This matrix subject to 10 cubic Demazure constraints:

$$(27) \quad EE^T E - \frac{1}{2} \text{Tr}(E^T E)E = 0, \quad \det E = 0.$$

Given rational-valued data $E_1, \dots, E_3 \in \mathbb{Q}^{3 \times 3}$, these constraints give us 10 polynomials in $\mathbb{Q}[x, y, z]$. As in the previous example, we generate our template using random rational data.

```

i1: setRandomSeed 0;
i2: R = QQ[x, y, z];
i3: Es = apply(4, i -> random(QQ^3, QQ^3));
i4: E = x * Es#0 + y * Es#1 + z * Es#2 + Es#3;
i5: I = ideal(E * transpose E * E - (1/2) * trace(E * transpose E) * E, det E);
i6: ET = eliminationTemplate(l, I);
i7: sols = templateSolve(ET);
i8: all(sols, x -> 1e-6 > norm sub(sub(gens I, CC[gens R]), matrix{x}))
i10: true

```

The optimal template size for this example is simply the 10×20 matrix with rows indexed by the Demazure constraints; in other words, the rows depending on s in Definition 2.5 are omitted.

```
i11 : M = getTemplateMatrix ET;
```

```

          10      20
o11 : Matrix QQ  <-- QQ
```

Writing $M = (M_1 \mid M_2)$ with $M_1, M_2 \in \mathbb{Q}^{10 \times 10}$, and assuming the parameters are sufficiently generic, we see that the associated action matrix is simply $M_1^{-1}M_2$. In practice, the parameters E_0, \dots, E_3 determining this matrix are obtained by from matched pairs of 2D image points. When solving for new parameter values, we may use `copyTemplate` just as we did in Example 4.1.

```

i12: Es = apply(4, i -> random(QQ^3, QQ^3));
i13: E = x * Es#0 + y * Es#1 + z * Es#2 + Es#3;
i14: J = ideal(E*transpose E * E - (1/2) * trace(E * transpose E) * E);
i15: ET' = copyTemplate(ET, J);
i16: sols = templateSolve(ET');
i17: all(sols, x -> 1e-6 > norm sub(sub(gens J, CC[gens R]), matrix{x}))
o17: true
```

Example 4.4. Our last example illustrates the flexibility of the `EliminationTemplates` package for solving larger minimal problems and the different strategies that may be used. Like the previous example, this problem comes from estimating the relative pose of two cameras. In this case, however, the two cameras share an unknown focal length. This leads to the following generalized Demazure constraints

$$(28) \quad \det F = 0,$$

$$(29) \quad FQF^TQF - \frac{1}{2} \operatorname{Tr}(FQF^TQ)F = 0,$$

for the 3×3 polynomial matrices

$$(30) \quad F = F_1 + yF_2 + zF_3, \quad Q = \operatorname{diag}(1, 1, x),$$

where F_1, F_2, F_3 are 3×3 matrices of parameters and x, y, z are unknown as in Example 4.3. For more background on this problem, we refer the reader to sources [8] and [15] which study similar formulations.

For this larger minimal problem, it is more effective to solve the *offline problem* of constructing the template matrix over a finite field $\mathbb{K} = \mathbb{Z}/p\mathbb{Z}$ of large, prime characteristic. This is a standard trick which allows the offline phase to run faster by reducing coefficient swell in Gröbner basis calculations; using rational reconstruction techniques, it is possible to recover a template matrix over \mathbb{Q} from several carefully chosen primes. Below, we simply fix $p = 32749$ for illustrative purposes.

```

i1: setRandomSeed 0
i2: FF = ZZ/32749;
i3: R = FF[x,y,z];
i4: Fs = apply(4, i -> random(QQ^3, QQ^3));
i5: F = Fs#0 + y*Fs#1 + z*Fs#2;
i6: Q = diagonalMatrix{1_R, 1_R, x};
i7: I = ideal(2*F*Q*transpose(F)*Q*F -trace(F*Q*transpose(F)*Q)*F) + ideal(det F);
```

For this example, we compare the template sizes produced by the default strategy and optional strategies "Larsson" and "Greedy" below. As the monomial order in our code above is unspecified, all three templates use the `GRevLex` order with $x > y > z$.

As determined by the template size alone, the "Greedy" strategy is the top-performer, followed by "Larsson", then finally the default strategy. This is shown below.

```

i8: ET = eliminationTemplate(x, I);
i9: getTemplateMatrix(ET);
      53      73
o9 : Matrix FF  <-- FF
i10:
getTemplateMatrix(ET, Strategy => "Larsson");  -- Larsson : 53 x 73
      53      73
o10 : Matrix FF  <-- FF
i11:
getTemplateMatrix(ET, Strategy => "Greedy")    -- Greedy : 31 x 50
      31      50
o11 : Matrix FF  <-- FF

```

The user running these lines of code will surely observe that, in terms of the time needed to generate the template, the order of these methods is reversed; the default strategy is faster than "Larsson", and both are significantly faster than "Greedy". However, template generation is an offline task which, in principle, only needs to be performed once. To properly evaluate the *online* minimal solvers produced by these templates, one must copy each template into a ring of characteristic 0 and measure both the runtime and residuals produced by `templateSolve`, as we have demonstrated in previous examples. The `EliminationTemplates` package enables users, for the first time, to perform the complete workflow of constructing, copying, and evaluating the performance of template-based solvers entirely within `Macaulay2`.

ACKNOWLEDGEMENTS

The Macaulay2 workshop at University of Wisconsin-Madison in June-July 2025 served as a first introduction for many of the authors. We thank the organizers, the Department of Mathematics at University of Wisconsin-Madison, and the National Science Foundation for supporting the workshop through NSF Grant DMS-2508868. We also thank Ikenna Nometa for helpful conversations early on. MB was supported by NSF Grants DMS-2302430 and DMS-2100288, and by Simons Foundation Grant SFI-MPS-TSM-00012928.

REFERENCES

1. Martin Byröd, Klas Josephson, and Kalle Åström, *Fast and stable polynomial equation solving and its application to computer vision*, International Journal of Computer Vision **84** (2009), no. 3, 237–256.
2. David Cox, John Little, and Donal O’Shea, *Ideals, varieties, and algorithms*, second ed., Undergraduate Texts in Mathematics, Springer-Verlag, New York, 1997, An introduction to computational algebraic geometry and commutative algebra. MR 1417938
3. _____, *Using algebraic geometry*, Graduate Texts in Mathematics, vol. 185, Springer-Verlag, New York, 1998.
4. Timothy Duff, *A Galois-Theoretic Complexity Measure for Solving Systems of Algebraic Equations*, arXiv preprint, 2503.17884 (2025).
5. Timothy Duff, Viktor Korotynskiy, Tomas Pajdla, and Margaret H Regan, *Galois/monodromy groups for decomposing minimal problems in 3D reconstruction*, SIAM Journal on Applied Algebra and Geometry **6** (2022), no. 4, 740–772.
6. Emil Graf and Alex Townsend, *Numerical instability of algebraic rootfinders*, Linear Algebra and its Applications **729** (2026), 308–336.
7. Daniel R. Grayson and Michael E. Stillman, *Macaulay2, a software system for research in algebraic geometry*, Available at <http://www2.macaulay2.com>.
8. Joe Kileel, Zuzana Kukelova, Tomas Pajdla, and Bernd Sturmfels, *Distortion varieties*, Foundations of Computational Mathematics **18** (2018), no. 4, 1043–1071.
9. Zuzana Kukelova, Martin Bujnak, and Tomás Pajdla, *Automatic generator of minimal problem solvers*, European Conference on Computer Vision, Springer, 2008, pp. 302–315.
10. Viktor Larsson and Kalle Åström, *Uncovering symmetries in polynomial systems*, European Conference on Computer Vision 2016, Lecture Notes in Computer Science, Springer, 2016, pp. 252–267.

11. Viktor Larsson, Kalle Åström, and Magnus Oskarsson, *Polynomial solvers for saturated ideals*, International Conference on Computer Vision, IEEE Computer Society, 2017, pp. 2307–2316.
12. Viktor Larsson, Kalle Åström, and Magnus Oskarsson, *Efficient solvers for minimal problems by syzygy-based reduction*, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2383–2392.
13. Bo Li and Viktor Larsson, *GAPS: generator for automatic polynomial solvers*, arXiv preprint, 2004.11765 (2020).
14. Evgeniy Martynushev, Snehal Bhayani, and Tomas Pajdla, *Automatic solver generator for systems of Laurent polynomial equations*, International Journal of Computer Vision **134** (2026), no. 2, 59.
15. Evgeniy Martynushev, Jana Vrábliková, and Tomas Pajdla, *Optimizing elimination templates by greedy parameter search*, 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 15733–15743.
16. Bernard Mourrain, *A new criterion for normal form algorithms*, Applied Algebra, Algebraic Algorithms and Error-Correcting Codes 1999 (Marc P. C. Fossorier, Hideki Imai, Shu Lin, and Alain Poli, eds.), Lecture Notes in Computer Science, Springer, 1999, pp. 430–443.
17. Bernard Mourrain, Simon Telen, and Marc Van Barel, *Truncated normal forms for solving polynomial systems: Generalized and efficient algorithms*, Journal of Symbolic Computation **102** (2021), 63–85.
18. David Nistér, *An efficient solution to the five-point relative pose problem*, IEEE Transactions on Pattern Analysis and Machine Intelligence **26** (2004), no. 6, 756–770.
19. Carlo Traverso, *Gröbner trace algorithms*, Symbolic and algebraic computation (Rome, 1988), Lecture Notes in Comput. Sci., vol. 358, Springer, Berlin, 1989, pp. 125–138.

DEPARTMENT OF MATHEMATICS, PURDUE UNIVERSITY, 150 N UNIVERSITY ST., WEST LAFAYETTE, IN 47907, USA
Email address: `mbatavia@purdue.edu`

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF WISCONSIN, 480 LINCOLN DR., MADISON, WI 53706, USA
Email address: `chengchen@math.wisc.edu`

DEPARTMENT OF MATHEMATICS, PURDUE UNIVERSITY, 150 N UNIVERSITY ST., WEST LAFAYETTE, IN 47907, USA
Email address: `achlopec@purdue.edu`

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF MISSOURI, 810 ROLLINS ST, COLUMBIA MO 65203, USA
Email address: `tduff@missouri.edu`

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF WISCONSIN, 480 LINCOLN DR., MADISON, WI 53706, USA
Email address: `whuang259@wisc.edu`

META PLATFORMS, INC., 1 META WAY, MENLO PARK, CALIFORNIA 94025, USA
Email address: `aolongli@meta.com`

DEPARTMENT OF MATHEMATICS, HARVARD UNIVERSITY, 1 OXFORD ST., CAMBRIDGE, MA 02138, USA
Email address: `wshen@math.harvard.edu`