Semidefinite Programming and facial reduction for Systems of Polynomial Equations

Greg Reid [*] Fei Wang [†] Henry Wolkowicz [‡] Wenyuan Wu [§]

Revision started July 10, 2015
Latest version as of Sunday 2nd August, 2015, at 15:55

Abstract

For a real polynomial system with finitely many complex roots, the real radical ideal, RRI, is generated by a lower degree system that has only real roots and the roots are free of multiplicities. RRI is a central object in computational real algebraic geometry. The computation of such RRI is of practical interest since multiplicities of roots yield singular Jacobians and cause problems for numerical solvers. Moreover the number of real roots can be far less than the number of complex roots and Lasserre and co-workers have shown that the RRI of a 0-dimensional real polynomial system with finitely many real solutions can be determined by a combination of techniques from a semidefinite programming (SDP) feasibility problem and geometric involution. A conjectured extension of such methods to positive dimensional polynomial systems has been given recently by Ma, Wang and Zhi.

In this paper we show that regularity in the form of the Slater constraint qualification (strict feasibility) fails for the moment matrix in the SDP feasibility problem. We use facial reduction and obtain a smaller regularized problem for which strict feasibility holds. We use this framework for analyzing RRIs of 0 and positive dimensional

[*]Dept. Appl. Math., University of Western Ontario, London, Ontario, Canada

[†]Dept. Appl. Math., University of Western Ontario, London, Ontario, Canada

[‡]Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada. Research supported in part by The Natural Sciences and Engineering Research Council of Canada (NSERC) and the U.S. Air Force Office of Scientific Research (AFOSR).

[§]Chongqing Key Lab. of Automated Reasoning and Cognition, CIGIT *Email:* wuwenyuan@cigit.ac.cn. Partly supported by NSFC 11471307 and West Light Foundation of the Chinese Academy of Science.

real polynomial systems. The SDP methods are implemented in MATLAB and our geometric involutive form is implemented in Maple. We consider two approaches to find a feasible moment matrix. We compare the SeDuMi interior point approach within the YALMIP package for MATLAB with the Douglas-Rachford (DR) projection-reflection method.

Illustrative examples show the advantages of the DR approach for some problems over standard interior point methods. We also see the advantage of facial reduction both in regularizing the problem and also in reducing the dimension of the moment matrices.

1 Introduction

The breakthrough work of Lasserre and collaborators [30, 45] shows that the *real radical ideal, RRI*, of a real polynomial system with finitely many solutions can be determined by a combination of a *semidefinite programming, SDP*, feasibility problem and the *geometric involutive form, GIF*. This RRI is generated by a system of real polynomials having only real roots that are free of multiplicities. Global numerical solvers, such as homotopy continuation solvers typically compute all real roots by first computing all complex (including real) roots. And if the roots have multiplicity, then elaborate strategies are needed to avoid difficulties that arise as the paths from the homotopy solvers approach these (singular Jacobian) roots [44]. Furthermore, random polynomial systems of k real polynomials of degree d in n variables can have d^n roots, and if the coefficients follow a certain probability distribution have only $d^{n/2}$ real roots on average, see [21] and the references therein. Therefore, consideration of only the real roots simplifies the problem. A conjectured extension of such methods to positive dimensional polynomial systems has been given recently by Ma, Wang and Zhi [33, 34]. These extensions depend on the *method of moments* within a SDP formulation.

Our SDP feasibility formulation is a moment problem equivalent to finding X for a linear system of the following type (also Problem 1.1 below)

$$\mathcal{A}X = b, \quad X \in \mathcal{S}^k_+\,, \tag{1.1}$$

where $\mathcal{S}^k_+$ denotes the convex cone of $k \times k$ real symmetric positive semidefinite matrices, and $\mathcal{A} : \mathcal{S}^k_+ \to \mathbb{R}^m$ is a linear transformation. The standard regularity assumption for (1.1) is the *Slater constraint qualification* or strict feasibility assumption:

$$\text{there exists } \hat{X} \text{ with } \mathcal{A}\hat{X} = b, \quad \hat{X} \in \operatorname{int} \mathcal{S}^k_+\,. \tag{1.2}$$

We let $X \succeq 0, \succ 0$ denote $X \in \mathcal{S}^k_+, \in \operatorname{int}\mathcal{S}^k_+$, respectively. It is well known that the Slater condition for SDP holds generically, e.g., [19]. Surprisingly, many SDP problems arising from particular applications, and in particular our polynomial system applications, are marginally infeasible, i.e., fail to satisfy strict feasibility. This means that the feasible set lies within the boundary of the cone, and even the slightest perturbation of the data can make the problem infeasible. This creates difficulties with the optimality and duality conditions as well as with numerical algorithms. To help regularize such SDP problems so that *strong duality* holds, facial reduction was introduced in 1982 by Borwein and Wolkowicz [11, 12]. However it was only much later that the power of facial reduction was exhibited in many applications, e.g., [1, 49, 52]. Developing algorithmic implementations of facial reduction that work for large classes of SDP problems and the connections with perturbation and convergence analysis has recently been achieved in e.g., [14, 17, 18, 28].

A polynomial system of maximum degree d equations in n variables can be viewed as the equation $Cx = 0$, a function of its monomials [30, 45]. Here x is a vector of the $N(n,d) = \frac{(d+n)!}{d!n!} = \begin{pmatrix} d+n \\ d \end{pmatrix}$ monomials up to the degree d of the polynomial system. This equation yields part of the system of linear constraints in the SDP formulation of polynomial systems. The convex cone for polynomials are semi-definite moment matrices encoding the real solutions of the polynomial equations and certain generalized Hankel-Macaulay structure possessed by the polynomial systems. Remarkable advances have been recently made in this area [8, 30, 45] which is an intersection between optimization and algebraic geometry. In this article we establish a framework for using facial reduction for such systems and then solving the systems using the regularized smaller SDP. We note that familiar methods for linear systems of equations when $d = 1$ are *Gaussian elimination, GE*, for exact solutions and *singular value decompositions, SVD*, for least squares solutions. For polynomial systems, the corresponding method in the exact case uses *Gröbner Bases* [4]. A major difference for Gröbner Bases to the case $d = 1$ is that generalized row operations involving multiplication by monomials and not just scalars is permitted. The operation of multiplying a polynomial by such a monomial raises its degree and is called prolongation. Eliminating between prolonged equations, is called projection. In the approximate case, as in our paper, we use *geometric involutive bases* [43] which use the SVD.

In particular a polynomial system can possess constraints resulting from this process that are *higher* than the degree of the system. So in this paper, as in [30, 45] and in Ma, Wang and Zhi [33, 34], higher degree systems can result. This continual extension of the underlying space is a significant practical and theoretical challenge in algorithm development.

The RRI of our system P is the set of all polynomials with the same zero set as P. To give the reader an informal introduction to RRIs and their interpretation, consider the simple case of *univariate polynomials* with real coefficients, $n = 1$. In this case, the factors of the coefficients are either complex or real. The RRI discards the complex factors and also the multiplicities from the polynomial, to obtain a new polynomial. This reduced polynomial is the generating polynomial for the RRI of the original polynomial, and has the same real roots, no multiplicities and no complex roots.

Combining SDP methods and applying them to a polynomial system P with coefficient matrix $C(P)$ and associated moment matrix $M(u) \in \mathbb{R}^{N(n,d)\times N(n,d)}$ yields the following problem central to our paper:

Problem 1.1 (Moment Matrix Feasibility Problem)**.** *Find* $u \in \mathbb{R}^{N(n,2d)}$ *where* $N(n,d) = \begin{pmatrix} d+n \\ d \end{pmatrix}$ *so that*

$$C(P)M(u) = 0, \quad M_{11}(u) = 1, \qquad M(u) \succeq 0.$$

Also see Problem 5.1 in Section 5.

We continue in Section 2 with material on real polynomial systems, their RRIs and the coefficient matrix representations. In Section 3 we give a condensed and more formal description of geometric involutive bases and the related algorithms. In Section 4 we combine the moment matrix and geometric involutive form algorithms to yield our fundamental Algorithm 4.1 for polynomial systems. In particular Algorithm 4.1 proceeds by putting the polynomials into GIF using Algorithm 3.1; we then solve the related moment matrix problem using Algorithm 2.1. These two steps are iterated until satisfaction of the Rank-Dim-Involutive Stopping Criterion 4.1.

In Section 5 we describe the facial reduction and projection methods for finding feasible solutions for the moment matrix feasibility problem 1.1. We also describe the *Douglas-Rachford (DR)* projection/reflection method that we use. We also present our implementation of facial reduction. Section 6 gives the numerical experiments. Our concluding remarks are in Section 7.

2 Real radical ideals and moment matrices

We now present some material on real polynomial systems, their RRIs and the coefficient matrix representation needed for our paper. For background and references to real algebraic geometry see e.g., [2, 4, 8, 45].

2.1 Real polynomial systems

We consider a (finite) system of m polynomials in n variables

$$P := \{p_1, ..., p_m\} \subset \mathbb{R}[x_1, \dots, x_n] =: \mathbb{R}[x],$$

where $\mathbb{R}[x]$ is the set of all polynomials with real coefficients in the n variables $x = \left(x_1, x_2, \dots, x_n\right)^T$. We let $d = \deg(P)$ denote the *degree of the polynomial system*, i.e., the maximum of the degrees of the polynomials p_j in P. The solution set or variety of P is

$$V_{\mathbb{K}}(p_1, ..., p_m) = \{x \in \mathbb{K}^n : p_j(x) = 0,\ \forall 1 \le j \le m\}. \tag{2.1}$$

This is the *real variety of* P if $\mathbb{K} = \mathbb{R}$ and the *complex variety of* P if $\mathbb{K} = \mathbb{C}$. The real ideal generated by $P = \{p_1, \dots, p_m\} \subset \mathbb{R}[x]$ is:

$$\langle P \rangle_{\mathbb{R}} = \langle p_1, \dots, p_m \rangle_{\mathbb{R}} = \{f_1 p_1 + \dots + f_m p_m : f_j \in \mathbb{R}[x], \forall 1 \le j \le m\}. \tag{2.2}$$

We denote a *monomial* by $x^\alpha := x_1^{\alpha_1} \cdots x_n^{\alpha_n}$, where $\alpha \in \mathbb{N}^n$, $\mathbb{N}$ is the set of nonnegative integers. The *degree of the monomial* is $|\alpha| := \|\alpha\|_1 = \alpha_1 + \cdots + \alpha_n$. It is clear that the degree of each monomial satisfies $|\alpha| \le d$, the degree of the polynomial. Throughout this paper we use *graded reverse lexicographic order, grevlex*, to order the set of monomials.[1]

We can rewrite the system of m polynomials, P, as

$$P = \left\{ \sum_{|\alpha| \le d} a_{k,\alpha}\, x^\alpha : k = 1, \dots, m \right\}. \tag{2.3}$$

This order respects the *Cartan class of variables*, which is important in our numerical determination of the geometric features of the polynomial systems such as those in Definition 3.3 below.

[1]This is often called *grevlex* in the literature. It compares the total degree first and then compares exponents of the last indeterminate but while reversing the outcome so that the monomial with smaller exponent is larger in the ordering.

Definition 2.1 (Coefficient matrix of P, $C(P)$)**.** *Let $x^{(\leq d)} = (x^\alpha)$ be the column vector of monomials x^α with $0 \leq |\alpha| \leq d$ ordered as in grevlex above. Suppose that the coefficients $a_{k,\alpha}$ in* (2.3) *are similarly ordered. Then define the coefficient matrix of P by $C(P) = (a_{k,\alpha})$.*

The following lemma follows immediately.

Lemma 2.1. *With $C(P), x^{(\leq d)}$ defined in Definition 2.1, we have*

$$P = C(P)x^{(\leq d)}, \tag{2.4}$$

with $C(P) \in \mathbb{R}^{m \times N(n,d)}$ and $N(n,d) := \begin{pmatrix} d+n \\ d \end{pmatrix}$ is the number of monomials in $x^{(\leq d)}$.

The well known presentation of polynomial systems as linear functions of their monomials along with the related coefficient matrix and its kernel and rowspace has been exploited in [37–39, 46] and in the historical work by Macaulay [36]. For an introductory example see [41].

2.2 Moment matrices

Moment matrices $M(\mu)$ arise as a means of representing real polynomial systems. We outline the procedure for finding $M(\mu)$ in Algorithm 2.1. For theoretical background the reader is directed to e.g., [2, 31].

A moment matrix is an infinite real symmetric matrix $M = (M_{\alpha,\beta})$ with indices corresponding to the indices of the monomials $\alpha, \beta \in \mathbb{N}^n$. Here α is the index for rows and β is the index for columns. Without loss of generality, we assume that $M_{0,0} = 1$. The matrix arises from considering the product of monomials $x^\alpha x^\beta = x^{\alpha+\beta}$ and then the correspondence $u_\alpha \leftrightarrow x^\alpha$ extends to the formal correspondence $x^\alpha x^\beta \leftrightarrow u_{\alpha+\beta}$.

Definition 2.2 (Moment matrix)**.** *Let $u = \{u_\alpha : \alpha \in \mathbb{N}^n, |\alpha| \leq d\} \in \mathbb{R}^{N(n,d)}$ be a vector of indeterminates where the entries are indexed corresponding to the exponent vectors of the monomials in n variables of degree at most d. The degree d moment matrix of u is a $N(n,d) \times N(n,d)$ symmetric matrix with rows and columns corresponding to monomials in n variables of degree at most d, and defined as*

$$M(u) = \left[u_{\alpha+\beta}\right]_{|\alpha|,|\beta| \leq d}.$$

Given a multivariate polynomial system $P \subset \mathbb{R}[x]$ with $d = \deg(P)$ we let M denote the *truncated moment matrix*.

Lemma 2.2. *The truncated moment matrix $M \in \mathcal{S}_+^{N(n,d)}$. The linear constraints imposed by P from* (2.4) *are $C(P)M = 0$, where $C(P)$ is the coefficient matrix function given in Definition 2.1.* □

Example 2.1 (Moment matrix for univariate example $x = (x_1)$)**.** *The moment matrix in the univariate ($n = 1$) case is the infinite matrix whose (α, β) entry is $u_{\alpha+\beta}$ and $\alpha, \beta \in \mathbb{N}$ given by:*

$$M(u) = \begin{bmatrix} u_0 & u_1 & u_2 & u_3 & u_4 & \cdots \\ u_1 & u_2 & u_3 & u_4 & u_5 & \cdots \\ u_2 & u_3 & u_4 & u_5 & u_6 & \cdots \\ u_3 & u_4 & u_5 & u_6 & u_7 & \cdots \\ u_4 & u_5 & u_6 & u_7 & u_8 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \qquad u_0 = 1. \tag{2.5}$$

Note that (2.5) is a Hankel matrix. Let us associate $u_\alpha \leftrightarrow x^\alpha$. Then we recover the polynomial equation using the coefficient matrix as $C(P)x^{(\leq d)}$. This implies that in terms of the moment matrix, we get $C(P)M(u) = 0$.

Algorithm 2.1: `M` - Moment Matrix
1 Input($P \subset \mathbb{R}[x_1, \ldots, x_n]$. Set $d := \deg(P)$);
2 Use an SDP method to find a maximum rank moment matrix $M(\mu^*)$ with the additional coefficient constraint $C(P)\, M(u^*) = 0$;
3 Output(`M`$(u^*) \succeq 0$, the maximum rank moment matrix)

3 Geometric involutive bases

In this section we introduce the basic objects for geometric involutive bases. Algorithm 3.1 finds the GIF. For more details and examples see [9, 41].

Involutivity originates in the geometry of differential equations. See Kuranishi [29] for a famous proof of termination of Cartan's prolongation algorithm for nonlinear partial differential equations. A by-product of these methods has been their implementation for linear homogeneous partial differential equations with constant coefficients, and consequently for polynomial algebraic systems. See [24] for applications and symbolic algorithms for polynomial systems. The symbolic-numeric version of a *geometric involutive form, GIF*, was first described and implemented in Wittkopf and Reid [47].

It was applied to approximate symmetries of differential equations in [9] and to polynomial solving in [40, 42, 43]. See [51] where it is applied to the deflation of multiplicities in multivariate polynomial solving.

Definition 3.1. *Let P be a finite subset of $\mathbb{R}[x]$ of degree d. The k-th prolongation of system P is*

$$\widehat{\mathrm{D}}^k(P) := \{x^\alpha p : 0 \le \deg(x^\alpha p) \le d + k, \alpha \in \mathbb{N}^n,\, p \in P\}.$$

For example $\widehat{\mathrm{D}}^k(P)$ for $P = \{x^2 - x - 1, xy - y - 1\}$ consists of P together with the 4 polynomials in

$$\begin{array}{rcl} x(x^2 - x - 1) & = & x^3 - x^2 - x \\ x(xy - y - 1) & = & x^2 y - xy - x \\ y(x^2 - x - 1) & = & x^2 y - xy - y \\ y(xy - y - 1) & = & xy^2 - y^2 - y. \end{array} \tag{3.1}$$

We can *project* by eliminating higher degree monomials in favour of lower degree ones. In the prolonged system we can project the system from degree 3 to degree 2 by eliminating the highest degree term x^2y that occurs in the second and third equations of (3.1) to obtain the new projected equation $y - x = 0$.

Definition 3.2. *Given a subspace V of $J^d := \mathbb{R}^{N(n,d)}$ and $m \le d$, define $\pi^m(V)$ as the vectors of V with the components of degree $\ge d - m$ discarded. Given $P \subset \mathbb{R}[x]$ of degree d define $\pi^m(P) := \pi^m \ker C(P)$. The k-th prolongation of the kernel is $D^k(P) := \ker C(\widehat{D}^k P)$.*

See for example [43] and the references in [41] for the stable numerical implementations of this paper's operations using SVD methods. In Remark 3.5 of [41] we discuss how prolongation and projection can equivalently be computed in the kernel or rowspace, and how polynomial generators can always be extracted. Underlying this is a $1-1$ correspondence between the relevant vector spaces (not elements).

Definition 3.3 (**Symbol, class and Cartan involution test**)**.** *Suppose $P \subset \mathbb{R}[x]$ of degree d. The symbol matrix $\mathcal{S}(P)$ of P is the submatrix of $C(P)$ corresponding to its degree d monomials. Then the class of a monomial x^α is the least j such that $\alpha_j \neq 0$.*

Suppose that the columns of $\mathcal{S}(P)$ are sorted in descending order by class and that it is reduced to Gauss echelon form. For $k = 1, 2, ..., n$ define the quantities $\beta_d^{(k)}$ as the number of pivots in this reduced matrix of class k. In a generic system of coordinates the symbol is involutive if

$$\sum_{k=1}^{k=n} k\beta_d^{(k)} = \text{rank } \mathcal{S}(\widehat{\mathrm{D}}P) \tag{3.2}$$

Suppose $Q \subset \mathbb{R}[x]$ has degree d' and a basis for $\ker C(Q)$ is given by the rows of the matrix B. To extract the $\beta_q^{(k)}$ in (3.2) at projected degree $d \leq d'$ we first numerically project $\ker C(Q)$ onto the subspace J^d by deleting the coordinates in B of degree $> d$ to give a spanning set $\tilde{B}$ for $\pi^{d'-d}Q$. Then delete the columns in $\widetilde{B}$ corresponding to variables of degree $< d$ to obtain a matrix A_d corresponding to the orthogonal complement of the degree d symbol. Let $A_d^{(k)}$ be the submatrix of $\widetilde{B}$ with columns corresponding to variables of class $\leq k$. In generic coordinates for $k = 1 \ldots n$:

$$\beta_d^{(k)} = \begin{pmatrix} n+d-k-1 \\ d-1 \end{pmatrix} - \left(\text{rank } A_d^{(k-1)} - \text{rank } A_d^{(k)}\right).$$

Then the SVD can approximate the ranks in this equation for carrying out the Cartan Test (3.2).

Definition 3.4 (Involutive System)**.** *A system of polynomials $P \subset \mathbb{R}[x]$ is involutive if* $\dim\, \pi DP = \dim\, P$ *and the symbol of P is involutive.*

Definition 3.5. *Let $P \in \mathbb{R}[x]$ with $d = \deg P$ and k, m be integers with $k \geq 0$ and $0 \leq m \leq k+d$. Then $\pi^m D^k P$ is projectively involutive if* $\dim \pi^m D^k P = \dim \pi^{m+1} D^{k+1} P$ *and the symbol of $\pi^m D^k P$ is involutive.*

In [9] we proved that a system is projectively involutive if and only if it is involutive. In Algorithm 3.1 we seek the smallest k such that there exists an m with $\pi^m \mathrm{D}^k P$ approximately involutive, and generates the same ideal as the input system. We choose the system corresponding to the largest such $m \leq k$ if there are several such values for the given k.

Algorithm 3.1: GIF: Geometric involutive form

Input($P \subset \mathbb{R}[x_1, \ldots, x_n]$*; tolerance* ϵ*.*);
Set $k := 0$, $d := \deg(P)$ and B for $\ker C(P)$, $J = \{\}$;
while $J \neq \emptyset$ **do**
    Compute $\mathrm{D}^k(P)$; initialize set of involutive systems $I := \{\}$;
    for j *from* 0 *to* $(d+k)$ **do**
        Compute $R := \pi^j \mathrm{D}^k(P)$;
        if R *involutive* **then**
            $I := I \cup \{R\}$
        end if
    end for
    Select all $\bar{R}$ from I: $\mathrm{D}^{d+k-\bar{d}}\bar{R} \subseteq \mathrm{D}^k(P)$ where $\bar{d} = \deg(\bar{R})$;
    Place the selected involutive $\bar{R}$ from I in the set J ;
    $k := k + 1$
end while
Output(*Return* $R =$ `GIF`(P) *the polynomial generators of the involutive system in* J *of lowest degree.*)

The degree of the geometric involutive basis in our method can be lower than that given in [33, 34] since Algorithm 3.1 updates the generators with projections. However, in the absence of a proof of determination of the real radical, we conclude that the larger moment matrices of [34] can capture new members of the real radical in situations where our method has already terminated.

Additional discussion and examples are given in the long version of our work [41].

4 Combining the moment matrix and geometric involutive form algorithms

The complete method that combines the moment matrix and geometric involution techniques is given in Algorithm 4.1.

Recall that $M = M(u) = (M_{\alpha,\beta})$ denotes the moment matrix indexed by α, β for rows and columns, respectively. And, $d = \deg(P)$, $M \in \mathcal{S}^{N(n,d)}$, and the linear constraints imposed by our system of polynomials $P \subset \mathbb{R}[x]$ are given using the coefficient matrix $C(P)M = 0$. We let $\langle P \rangle_{\mathbb{R}}$ denote the *associated polynomial ideal* and let

$$\sqrt[\mathbb{R}]{\langle P\rangle_{\mathbb{R}}} = \{f \in \mathbb{R}[x] : f^{2m} + \sum_{j=1}^{s} q_j^2 \in \langle P\rangle_{\mathbb{R}}\,, q_j \in \mathbb{R}[x], m \in \mathbb{N}_+\}$$

denote the RRI generated by polynomials P over $\mathbb{R}$. A fundamental result [4] that is a consequence of the real nullstellensatz is

$$\sqrt[\mathbb{R}]{\langle P\rangle_{\mathbb{R}}} = \{f(x) \in \mathbb{R}[x] : f(x) = 0, \forall x \in V_{\mathbb{R}}(P)\}.$$

Algorithm 4.1 proceeds by putting the polynomials into GIF using Algorithm 3.1; we then solve the related moment matrix problem using Algorithm 2.1. These two steps are iterated until satisfaction of the Rank-Dim-Involutive Stopping Criterion 4.1, that is $r = d$. If the ideal generated by the input system is zero dimensional then the output is a GIF for the real radical. If the input system is positive dimensional, then the output is a GIF for an intermediate idea between the input ideal and the real radical.

Algorithm 4.1: $\texttt{GIF} - \texttt{SDP}$ Method

Input($P = \{p_1, ..., p_k\} \subset \mathbb{R}[x_1, \ldots, x_n]$);
Set $P_0 := P, \quad j := 0$;
while $r = d$ **do**
  $d := \dim\ker \texttt{GIF}(P_j), \quad P_{j+1} := \texttt{GIF}(P_j)$;
  Find $u^* \in \mathbb{R}^{N(n,2d)}$: $M(u^*) \succeq 0, C(P_{j+1})M(u^*) = 0$ (Described in Algorithm 2.1);
  $r := \mathrm{rank}(\texttt{M}(u^*)), \quad P_{j+2} := \texttt{gen}(\ker \texttt{M}(u^*))$;
  $j := j + 2$
end while
Output($P_{j+1} \subset \mathbb{R}[x_1, \ldots, x_n]$; P_{j+1} is in geometric involutive form ; $\sqrt[\mathbb{R}]{\langle P\rangle_{\mathbb{R}}} \supseteq \langle P_{j+1}\rangle_{\mathbb{R}} \supseteq \langle P\rangle_{\mathbb{R}}$.)

The Algorithms 2.1, 3.1, and 4.2 are *subroutines* for our principal Algorithm 4.1.

Algorithm 4.2: $\texttt{gen}$

Input($\ker \texttt{M}(u^*)$ *where* $\texttt{M}(u^*)$ *is the optimal max-rank moment matrix.*);
Output(Polynomial generators corresponding to $\ker \texttt{M}(u^*)$)

Remark 4.1 (**Rank-Dim-Involutive Stopping Criterion**). *A natural termination criterion used in Algorithm 4.1 is that the generators stabilize at some iteration and the system is involutive:*

$$\mathtt{gen}(\mathtt{GIF}(P)) = \mathtt{gen}(\ker \mathtt{M}(u^*)) \text{ and } P \text{ involutive}, \tag{4.1}$$

where u^* corresponds to the optimal moment matrix $M(\mu^*)$. From results in [30], $\langle \mathtt{gen}(\ker \mathtt{M}(P_{j+1})) \rangle$ is a sequence of ideals containing $\sqrt[\mathbb{R}]{\langle P \rangle}$. We get an ascending chain of ideals in a Noetherian ring $\mathbb{R}[x_1, ..., x_n]$. Hence, together with the finiteness of the Cartan-Kuranishi geometric involutive form algorithm, Algorithm 4.1 terminates in a finite number of steps.

5 Facial reduction and projection methods

In this section we describe the facial reduction and projection methods for finding feasible solutions for the moment matrix feasibility problem. Our moment problem is given in Problem 5.1, where $M(u)$ implicitly denotes the moment matrix constraints, i.e., the intersection of the space of generalized Hankel matrices with the semidefinite cone.

Problem 5.1 (Moment Matrix Feasibility Problem). *Let $C = C(P)$ be a given $N(n, d) \times m$ (coefficient) matrix of full column rank. Find $u \in \mathbb{R}^{N(n,2d)}$ so that*

$$C^T M(u) = 0, \quad M(u)_{11} = 1, \quad M(u) \succeq 0.$$

5.1 Representations for linear constraints for moment problems

An important initial step for our methods is building an efficient (onto) matrix representation for the linear constraints on the moment matrices resulting from the polynomial systems. Recall that we introduced moment matrices informally by a simple example in Section 2.2; see also Definition 2.2. Let $u_\alpha := u_{(\alpha_1,\ldots,\alpha_n)}$ where $\alpha \in \mathbb{N}^n$ and the degree of u_α is $|\alpha| = \alpha_1 + \ldots + \alpha_n$. Let $\left(u_{(\alpha \leq d)}\right)$ be an array of the u_α's with $0 \leq |\alpha| \leq d$ and sorted in grevlex order as described above.

Consider a truncated moment matrix $M(u) = (u_{\alpha+\beta})_{\alpha,\beta \in \mathbb{N}^n, |\alpha|,|\beta| \leq d}$. The generalized truncated moment matrix can be represented as follows, where

$$\langle f_i(u), f_j(u) \rangle_* = u(i) + u(j).$$

We assume the length of $\langle u_{(\alpha \le d)} \rangle$ is $k+1$. (We provide a formula for k in Algorithm 5.1 below.)

$$M(u) = \begin{bmatrix} \langle f_0(u), f_0(u) \rangle_* & \langle f_0(u), f_1(u) \rangle_* & \langle f_0(u), f_2(u) \rangle_* & \dots & \langle f_0(u), f_k(u) \rangle_* \\ \langle f_1(u), f_0(u) \rangle_* & \langle f_1(u), f_1(u) \rangle_* & \langle f_1(u), f_2(u) \rangle_* & \dots & \langle f_1(u), f_k(u) \rangle_* \\ \langle f_2(u), f_0(u) \rangle_* & \langle f_2(u), f_1(u) \rangle_* & \langle f_2(u), f_2(u) \rangle_* & \dots & \langle f_2(u), f_k(u) \rangle_* \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \langle f_k(u), f_0(u) \rangle_* & \langle f_k(u), f_1(u) \rangle_* & \langle f_k(u), f_2(u) \rangle_* & \dots & \langle f_k(u), f_k(u) \rangle_* \end{bmatrix}$$

In the univariate case the moment matrices have Hankel structure as shown in (2.5). In Table 5.1 we display a truncated bivariate moment matrix partitioned into block submatrices having the same degree. Notice that the

$$M(u) = \left[\begin{array}{c|cc|ccc|cccc} u_{00} & u_{10} & u_{01} & u_{20} & u_{11} & u_{02} & u_{30} & u_{21} & u_{12} & u_{03} \\ \hline u_{10} & u_{20} & u_{11} & u_{30} & u_{21} & u_{12} & u_{40} & u_{31} & u_{22} & u_{13} \\ u_{01} & u_{11} & u_{02} & u_{21} & u_{12} & u_{03} & u_{31} & u_{22} & u_{13} & u_{04} \\ \hline u_{20} & u_{30} & u_{21} & u_{40} & u_{31} & u_{22} & u_{50} & u_{41} & u_{32} & u_{23} \\ u_{11} & u_{21} & u_{12} & u_{31} & u_{22} & u_{13} & u_{41} & u_{32} & u_{23} & u_{14} \\ u_{02} & u_{12} & u_{03} & u_{22} & u_{13} & u_{04} & u_{32} & u_{23} & u_{14} & u_{05} \\ \hline u_{30} & u_{40} & u_{31} & u_{50} & u_{41} & u_{32} & u_{60} & u_{51} & u_{42} & u_{33} \\ u_{21} & u_{31} & u_{22} & u_{41} & u_{32} & u_{23} & u_{51} & u_{42} & u_{33} & u_{24} \\ u_{12} & u_{22} & u_{13} & u_{32} & u_{23} & u_{14} & u_{42} & u_{33} & u_{24} & u_{15} \\ u_{03} & u_{13} & u_{04} & u_{23} & u_{14} & u_{05} & u_{33} & u_{24} & u_{15} & u_{06} \end{array}\right]$$

Table 5.1: block partitioned bivariate moment matrix; submatrices have same degree

matrix in Table 5.1 is not Hankel. However each of its block matrices is rectangular Hankel; though even this feature is lost for multivariate moment matrices in more than two variables. As mentioned above, without loss of generality we assume that $u_{00} = 1$.

Besides being a symmetric matrix, the moment matrix also has other linear constraints among its entries. One can easily see these constraints in the truncated univariate matrix (2.5) and bivariate matrix in Table 5.1. An important requirement of our projection methods is to maintain these constraints. For example, in the bivariate case above, the matrix elements $M(u)_{14} = M(u)_{22}$ are both equal to u_{20}. We now outline a simple algorithm to find a non-redundant matrix representation of these constraints in the general n variable case. To list these constraints we start from the first row and traverse the matrix from left to right across the rows and then traverse the rows from top to bottom. Note also that we only need to examine entries above the main diagonal since the matrix is symmetric.

For $M(u)$ in Table 5.1 the first linear constraint traversing from the first row is $M(u)_{14} = M(u)_{22}$. We denote e_i as the *i-th unit vector* and $E_{ij} = \frac{1}{2}(e_i^T e_j + e_j^T e_i)$ as the *ij-th unit matrix*. To impose this first constraint on a matrix $M \in \mathcal{S}_+^{k+1}$, we construct matrix $A_2 = E_{22} - E_{14}$. The constraint is then given by

$$\langle A_2, M\rangle = \text{trace}((E_{22} - E_{14})M) = 0.$$

Since we always assume $M(u)_{1,1} = 1$, we need to set $A_1 = E_{11}$. We can similarly construct $A_3, A_4, \cdots, A_r$, where r is the number of the total linear constraints. We denote A_t the *matrix representative* of the *t-th* linear constraint.

Algorithm 5.1: Matrix representation of moment matrix constraints

Input(d, n) (d is the degree, n is the number of the variables) ;
Compute $k := N(n, d) - 1 = \binom{d+n}{d} - 1$.
Initialize an array $T = \langle \alpha_{(\leq d)} \rangle$ of length $k+1$, $T(i)$ is the *i-th* element of T.
Initialize an array $S = \langle s \rangle$ of length $k+1$ with the *i-th* element $S(i) = [(1, i); T(i)]$.
Let $t = 2$ and $A_1 = E_{11}$. **for** *i from 2 to $k+1$*, **do**
  for *j from i to $k+1$*, **do**
    if *$\exists g, h, \alpha$ with $s = [(g, h); \alpha] \in S$ such that $T(i) + T(j) = \alpha$* **then**
      $A_t = E_{ij} - E_{gh}$, $t = t + 1$
    else
      Adjoin a new element $s = [(i, j); \alpha]$ to S where $\alpha = T(i) + T(j)$
    end if
  end for
end for
Output(Return an array of $(k+1) \times (k+1)$ matrix representatives $\{A_t\}$ where $t \in \mathcal{E}$, $\mathcal{E} = \{1, 2, \ldots, r\}$ and r is the total number of the linear constraints.);

Algorithm 5.1 determines all the (non-redundant) matrix representatives of the linear constraints of the multivariate moment matrix. For example, if the input is $(d,n) = (2,2)$, then $T = [(0,0),(1,0),(0,1),(2,0),(1,1),(0,2)]$ and

$$S = [[(1,1);(0,0)],[(1,2);(1,0)],\cdots,[(1,6);(0,2)]]$$

There are no redundant constraints produced by this algorithm. This avoids having an overdetermined linear system.

5.2 First step of facial reduction

Semidefinite programming has become an important tool in many areas of optimization and algebraic geometry, e.g., [2, 8, 48]. The semidefinite cone $\mathcal{S}^t_+$ has been extensively studied and the facial structure is well understood. If $X \in \mathcal{S}^t_+$, then we let face $(X, \mathcal{S}^t_+)$ denote the smallest face of $\mathcal{S}^t_+$ containing X. And if f is a face of $\mathcal{S}^t_+$, denoted $f \trianglelefteq \mathcal{S}^t_+$, then the *conjugate face* is $f^c := f^\perp \cap \mathcal{S}^t_+$. Let $X = \begin{bmatrix} U & V \end{bmatrix} \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} U & V \end{bmatrix}^T$ be the spectral decomposition of X with $\begin{bmatrix} U & V \end{bmatrix}$ orthogonal and both $D \in \mathcal{S}^r_{++}$ and diagonal. Then

$$\begin{array}{rcl} \text{face}\,(X, \mathcal{S}^t_+) & = & U\mathcal{S}^r_+ U^T \\ & = & \{Y \in \mathcal{S}^t_+ : V^T Y = 0\} \\ & = & \{Y \in \mathcal{S}^t_+ : \text{trace}(VV^T)Y = 0\}. \end{array}$$

Similarly,

$$\begin{array}{rcl} \text{face}\,(X, \mathcal{S}^t_+)^c & = & V\mathcal{S}^{t-r}_+ V^T \\ & = & \{Z \in \mathcal{S}^t_+ : U^T Z = 0\} \\ & = & \{Z \in \mathcal{S}^t_+ : \text{trace}(UU^T)Z = 0\}. \end{array}$$

Problem 5.2 (Moment Matrix Feasibility Problem)**.** *Our main problem is the following feasibility problem for the moment matrix M:*

$$\mathcal{A}(M) = b = e_1, \quad B^T M = 0, \quad M \in \mathcal{S}^{k+1}_+, \tag{5.1}$$

Here k and the linear transformation $\mathcal{A}$ is obtained from Algorithm 5.1. $\mathcal{A}(M) = \left(\langle A_t, M\rangle\right)_{\forall t \in \mathcal{E}} \in \mathbb{R}^{r\times 1}$. The full column rank matrix B is obtained from the coefficient matrix in Definition 2.1 and equation (2.4)*.*

The following Theorem 5.1 provides the details of the system after 1 step of facial reduction obtained by applying the coefficient matrix constraint to the moment matrix, i.e., $B^T M = 0$. Recall from Algorithm 5.1, we get an array of representing matrix A_t's where $t \in \mathcal{E}$, $\mathcal{E} = \{1, 2, \ldots, r\}$.

Theorem 5.1 (First step facial reduction)**.** *Let $B \in \mathbb{R}^{N(n,d)\times m}$ be as above and full column rank. Let $V \in \mathbb{R}^{N(n,d)\times(N(n,d)-m)}$ satisfy $V^TB = 0$ and $\begin{bmatrix} B & V \end{bmatrix}$ nonsingular. Let*

$$\bar{A}_t := V^T A_t V, \quad \forall t \in \mathcal{E} = \{1, 2, \ldots, r\}$$

and define the linear transformation $\bar{\mathcal{A}} : \mathcal{S}^{N(n,d)-m} \to \mathbb{R}^{r\times 1}$ by

$$\bar{\mathcal{A}}(P) := \left(\langle \bar{A}_t, P\rangle\right)_{t\in\mathcal{E}}. \tag{5.2}$$

Then Problem 5.1 is equivalent to

$$\bar{\mathcal{A}}(P) = b, \qquad P \in \mathcal{S}_+^{N(n,d)-m}, \tag{5.3}$$

where we can recover the moment matrix using $M = VPV^T$. □

Note that for stability, we need to process the linear constraint (5.2) further to obtain an equivalent linear system $\hat{\mathcal{A}}(\hat{P}) = \hat{b}$ where $\hat{A}$ is an onto map.

5.2.1 Potential second facial reduction

Our initial semidefinite moment problem is a feasibility problem of the form

$$B^T M(u) = 0, \quad M(u) \succeq 0, \tag{5.4}$$

where B is a given coefficient matrix and the moment matrix $M(u)$ is a linear function of the variables u. Constraints on $M(u)$ are described in Section 5.1. In Section 5.3 the problem is changed to equality form and then uses facial reduction to get the form

$$\bar{\mathcal{A}}(P) = b, \qquad P \succeq 0. \tag{5.5}$$

This form includes the first step of facial reduction using the matrix B, see Theorem 5.1 and (5.2).

The projection methods behave poorly, converge slowly, when the Slater condition fails, e.g., [18]. We therefore attempt to apply further steps of facial reduction and reduce system (5.5) until a strictly feasible point exists. We use the following theorem of the alternative or characterization of a strictly feasible point; see e.g., [13]:

$$\begin{array}{c} \exists P, \bar{\mathcal{A}}(P) = b, P \succ 0 \\ \Longleftrightarrow \\ Z = \bar{\mathcal{A}}^* y \succeq 0, b^T y = 0 \implies Z = 0. \end{array} \tag{5.6}$$

Note that if a $Z \neq 0$ can be found satisfying the left part of the bottom half of (5.6) and for the top half $P \succeq 0, \bar{\mathcal{A}}(P) = b$, then

$$0 = b^T y = \langle \bar{\mathcal{A}}(P), y \rangle = \langle P, Z \rangle \implies PZ = 0 \implies \operatorname{range} P \subseteq \operatorname{null} Z.$$

Therefore, if the full column rank matrix W satisfies $\operatorname{range} W = \operatorname{null} Z$, then we can facially reduce the problem to a lower matrix $\bar{P}$ using the substitution $P = W\bar{P}W^T$, i.e., we can restrict the feasibility problem in (5.5) to the face $W\bar{\mathcal{S}}_+W^T$.

We can implement the test in (5.6) in several ways. One way is to solve the following minimization problem [2]

$$\begin{array}{rll} p^* := & \min & \frac{1}{2}(\bar{b}^T y)^2 \\ & \text{s.t.} & Z = \bar{\mathcal{A}}^* y \succeq 0 \\ & & \operatorname{trace} \bar{\mathcal{A}}^* y = 1 \end{array}$$

where

$$\bar{\mathcal{A}}^* y = \sum_{t=1}^{r} (\bar{A}_t y).$$

If the objective p^* is 0, then it implies we may need a second facial reduction. A *stable* approach, in the sense that strict feasibility holds, to solving this auxiliary problem is given in [13] as

$$\begin{array}{ll} \max & \delta \\ \text{s.t.} & Z = \bar{\mathcal{A}}^* y \succeq \delta I \\ & \operatorname{trace} Z = 1 \\ & \bar{b}^T y = 0 \end{array} \tag{5.7}$$

5.2.2 Backward stability for facial reduction steps

We now see that we can find the equivalent facial reduced problem efficiently and accurately. We start with the Moment Matrix Feasibility Problem in (5.1).

$$\mathcal{A}(M) = b = e_1, \quad B^T M = 0, \quad M \in \mathcal{S}_+^{N(n,d)}.$$

As above, $B \in \mathbb{R}^{(k+1)\times m}$ and is full column rank. We apply the QR factorization and numerically obtain the output $B \approx \tilde{Q}\tilde{R}$, where $Q = \begin{bmatrix} \tilde{U} & \tilde{V} \end{bmatrix}$ is orthogonal, and $\tilde{R}$ upper triangular with the last m rows being zero, see

[2] This can be implemented in e.g., CVX using the *norm* function or absolute value function for the objective, i.e., we minimize $|\bar{b}^T y|$ rather than using the squared term.

e.g., [25]. The QR factorization is backwards stable, i.e., we get the exact equation

$$\tilde{Q}\tilde{R} = B + \delta B, \qquad \frac{\|\delta B\|}{\|B\|} = O(\epsilon_{machine}), \tag{5.8}$$

Thus we have *exactly* found the QR factorization of a nearby matrix. We then use Theorem 5.1 to obtain the facially reduced problem in (5.3) i.e., we form the matrices $\tilde{A}_t$. The matrix V has orthonormal columns. Therefore the congruence is a backward stable operation and we have

$$\tilde{A}_t = \tilde{V}^T(A_t + \delta A_t)\tilde{V}, \quad \frac{\|\delta A_t\|}{\|A_t\|} = O(\epsilon_{machine}), \forall t \in \mathcal{E} = \{1, 2, \ldots, r\}. \tag{5.9}$$

Therefore, we can combine the above two steps and conclude that the first step of facial reduction is a stable operation, i.e.,

$$\tilde{\mathcal{A}}(P) = b, \qquad P \in \mathcal{S}_+^{N(n,d)-m}, \tag{5.10}$$

is obtained efficiently and accurately; we have found the *exact* facial reduction of a nearby problem.

Note that we then use a singular value decomposition to remove the redundant linear constraints so that the linear map $\tilde{A}$ in the resulting linear constraints can be assumed to be onto. This can be done using the SVD factorization, again a backwards stable algorithm. We have shown the following.

Theorem 5.2 (Backward stability of first FR)**.** *The first step of facial reduction is backward stable. More precisely, we find a linear system* (5.10) *with $\tilde{A}$ onto and equivalent to a nearby system to the original moment matrix feasibility problem in the sense of* (5.8) *and* (5.9)*.* □

We do not include the analysis for a second step of facial reduction. This is more difficult as we need to include the accuracy in solving the auxiliary problem for the theorem of the alternative discussed in Section 5.2.1. Such an analysis can be found in [13, Theorem 1.38].

5.3 Projection methods

We now consider two projection methods. We first consider the method of *alternating projection, MAP* and use the defined projections to introduce the *Douglas-Rachford reflection-projection* method. It is the latter method that we implement as it displayed better convergence properties in our tests.

5.3.1 Method of alternating projections, MAP

The method of alternating projections, MAP, is particularly simple, see e.g., the recent book [22]. Let s2vec denote the mapping (isometry) from a matrix to a column vector taken columnwise with the off-diagonal elements multiplied by $\sqrt{2}$. Let $\mathrm{s2Mat} = \mathrm{s2vec}^* = \mathrm{s2vec}^{-1}$ be the inverse mapping from a column vector to a matrix. The inverse here is identical to the *adjoint map*. Let $L = (\mathrm{s2vec}(\bar{A}_t)^T)_{t\in\mathcal{E}}$ denote the matrix representation for $\bar{\mathcal{A}}$ in Theorem 5.1 ($\mathrm{s2vec}(\bar{A}_t)^T$ is the *t-th* row of L).

We begin with an initial estimate, e.g., $P_c = \alpha I \in \mathcal{S}_+^{N(n,d)-m}$ for a *large* $\alpha > 0$. There are two projections we use to update the current point P_c. First, we look at $\mathcal{P}_{\mathcal{L}}$*, the linear manifold projection*. We map P_c to a column vector $p_c = \mathrm{s2vec}(P_c)$, then for the linear system $Lp = b = e_1$ where L has full row rank, we solve the nearest point problem $\min\left\{\frac{1}{2}\|p - p_c\|_2^2 : Lp = b\right\}$, i.e., we find the projection onto the linear manifold for the linear constraints. We use $L^\dagger$*, the Moore-Penrose generalized inverse* of L. The residual and the p_l satisfying the minimization problem are then

$$r_c = b - Lp_c; \qquad p_l = p_c + L^\dagger r_c. \tag{5.11}$$

Second, we project the updated symmetric matrix $P_L = \mathcal{P}_{\mathcal{L}}(P_c) = \mathrm{s2Mat}(p_l)$ onto the semidefinite cone using the Eckart-Young Theorem [20], i.e., we diagonalize and zero out the negative eigenvalues. We denote $\mathcal{P}_{\mathcal{S}_+}$*, the positive semidefinite projection* and get the new positive semidefinite approximation $\mathcal{P}_{\mathcal{S}_+}(P_L)$.

We repeat the projection steps in Items 1, 2, 3 described above till a sufficiently small desired tolerance is obtained in the norm of the residual.

1. Evaluate the residual $r_c = b - Lp_c$. Use the residual to evaluate the linear projection and obtain the update
$$P_L = \mathcal{P}_{\mathcal{L}}(P_c).$$
2. Evaluate the positive semidefinite projection using the Eckart-Young Theorem and update the current approximation
$$P_{PSD} = \mathcal{P}_{\mathcal{S}_+}(P_L).$$
3. Update the cosine value in (5.12). Then update $P_c = P_{PSD}$.

The (linear) convergence rate is measured using cosines of angles from three consecutive iterates

$$\cos(\theta) = \left(\frac{\operatorname{trace}\left((P_L - P_c)^*(P_{PSD} - P_L)\right)}{\|P_L - P_c\|\,\|P_{PSD} - P_L)\|}\right). \tag{5.12}$$

5.3.2 Douglas-Rachford reflection method

Recall the projections defined above $\mathcal{P}_{\mathcal{L}}, \mathcal{P}_{\mathcal{S}_+}, P_{PSD}$. We want to find, see (5.3),

$$P \in \mathcal{G} \cap \mathcal{S}_+^{N(n,d)-m}, \quad \text{where } \mathcal{G} := \left\{ P : \bar{\mathcal{A}}(P) = b = R \right\}.$$

We now apply the Douglas-Rachford (DR) projection/reflection method [16]. (See also e.g., [3, 10].)

Using the QR algorithm applied to B to find V and $\bar{\mathcal{A}}$, we start with an initial estimate

$$P_0 = \alpha I \in \mathcal{S}_+^{N(n,d)-m} \text{ for some } \alpha\,. \tag{5.13}$$

Define the *reflections* $\mathcal{R}_{\mathcal{L}}, \mathcal{R}_{PSD} : \mathcal{S}_+^{N(n,d)-m} \to \mathcal{S}_+^{N(n,d)-m}$ using the corresponding projections, i.e.,

$$\mathcal{R}_{\mathcal{L}}(P) := 2\mathcal{P}_{\mathcal{L}}(P) - P, \quad \mathcal{R}_{PSD}(P) := 2\mathcal{P}_{\mathcal{S}_+}(P) - P.$$

- **Initialization:** We set our current estimate $P_c = P_0$. We calculate the residual $Res_{\mathcal{L}} = R - \bar{\mathcal{A}}(P_c)$, set $normres = \|Res_{\mathcal{L}}\|$, denote the reflected residual $Resrefl_{\mathcal{L}} = Res_{\mathcal{L}}$ and reflected point $\mathcal{R}_{PSD} = P_c$.

- **Iterate:** We continue iterating from this point while $normres > toler$, our desired tolerance.

- 1. We use $Resrefl_{\mathcal{L}}$ to project the current reflected PSD point $\mathcal{R}_{PSD}$ onto the linear manifold to get the projected point $P_{\mathcal{L}} = \mathcal{R}_{PSD} + \text{s2Mat}(L^\dagger Resrefl_{\mathcal{L}})$. Then we reflect to get our second reflection point $\mathcal{R}_{\mathcal{L}} = 2P_{\mathcal{L}} - \mathcal{R}_{PSD}$.
  2. At this time we set our new/current estimate for convergence to be $P_c = P_{new} = (P_c + \mathcal{R}_{\mathcal{L}})/2$.
  3. We now project P_c to get $P_{PSD} = \mathcal{P}_{\mathcal{S}_+}(P_c)$. We check the residual here for the stopping criteria $normres = \|Res_{\mathcal{L}}\| = \|R - \bar{\mathcal{A}}(P_{PSD})\|$.
  4. We now calculate the first reflection point $\mathcal{R}_{PSD} = 2P_{PSD} - P_c$ and update the reflected residual $Resrefl_{\mathcal{L}} = R - \bar{\mathcal{A}}(\mathcal{R}_{PSD})$.

Also according to the basic theorem on the convergence of the sequence $\Pi_G(X_k)_k$, [10, Thm 3.3, Page 11], the residuals of the projections of the iterates on one of the sets have to be used for the stopping criteria. We use the residual after the projection onto the SDP cone since we want our final matrix to be semidefinite.

Algorithm 5.2 summarizes our Facial reduction & Douglas-Rachford method.

Algorithm 5.2: FDR method

1. Input(Degree of system d, number of variables n, a $N(n,d) \times m$ coefficient matrix B) ;
2. Compute the matrix representation A using Algorithm 5.1.;
3. Use QR to find V s.t. $V^T B = 0$ and $\begin{bmatrix} B & V \end{bmatrix}$ nonsingular; compute the matrix representation L of the linear transformation $\bar{\mathcal{A}}$ described in Theorem 5.1.;
4. Start at an initial point P_0 satisfying (5.13).;
5. Iterate: $P_{j+1} = \frac{1}{2}(P_j + \mathcal{R}_{PSD}(\mathcal{R}_{\mathcal{L}}(P_j)))$, for all $j = 0, 1, \ldots$.;
6. Stop if $normres \leq toler$.;
7. Output(*A PSD* $N(n,d) \times N(n,d)$ *moment matrix* $M = VP_{j+1}V^T$.)

Our empirical studies showed that the Douglas-Rachford approach outperformed MAP and also outperformed the SeDuMi interior point method within the YALMIP toolbox. Though the Douglas-Rachford iteration has only a linear convergence rate, the method converged robustly to the intersection of the linear constraints and the semidefinite cone. We not that for two subspaces, the linear rate for the method is given by the cosine of the Friedrichs angle between them, see e.g., [5,6]. Details on the numerical tests follow.

6 Numerical experiments

In this section we present the numerical tests for the GIF-Moment Matrix Algorithm 4.1 that combines the Geometric Involutive Form with an SDP solver. We consider the two SDP feasibility solving algorithms: the FDR Algorithm 5.2 with facial reduction and the standard interior point solver SeDuMi but without facial reduction. GIF is combined with the two SDP approaches to yield GIF-FDR and GIF-SeDuMi, respectively.

In Section 6.1 we consider a class of random univariate polynomials with varying degree d. The results are displayed in Figure 6.1 on page 23, and Figure 6.2 on page 23. Results for the examples given in Sections 6.2 and 6.3 are summarized in Table 6.1 page 28.

We used MATLAB version 2014a and Maple version 18. The computations were carried out on a desktop with ubuntu 12.04 LTS, Intel Core$^{\text{TM}}$2 Quad CPU Q9550 @ 2.83 GHz $\times$ 4, 8GB RAM, 64-bit OS, x64-based processor.

6.1 A class of random univariate polynomials

We first consider root finding for polynomials of the form

$$p_d(x) = a_{d,0} + a_{d,1}\, x + a_{d,2}\, x^2 + \cdots + a_{d,d}\, x^d, \;\; d = 1, 3, 5, \cdots \qquad (6.1)$$

where $a_{d,j} \sim N(0,1)$. A famous early work on random polynomials such as (6.1) is given by Kac in [27] who derived an integral formula for the average number of real roots of $p_d(x)$:

$$E_d = \frac{4}{\pi} \int_0^1 \sqrt{\frac{1}{(1-t^2)^2} - \frac{(d+1)^2\, t^{2\,d}}{(1-t^{2\,d+2})^2}}\, \mathrm{d}t. \qquad (6.2)$$

An asymptotic form for large d was determined to be $E_d \approx \frac{2}{\pi} \log(d) + 0.6257358072... + \frac{2}{\pi d} + O\left(\frac{1}{d^2}\right)$, e.g., [21] and the references therein.

We applied GIF-FDR and GIF-SeDuMi to the random polynomials $p_d(x)$ for odd degrees d with $3 \leq d \leq 51$. For each odd degree j, 10 sample random polynomials were generated by selecting their coefficients as independent samples from $N(0,1)$. Algorithms GIF-FDR and GIF-SeDuMi were then applied to approximate the minimal polynomial generating their real radical. The residual error for each polynomial at odd degree j was computed by substituting that roots of the minimal polynomial into the original input polynomial $|p_j|$. The average of the $\log_{10}$ of all these 10 residual errors was computed for each degree j. We also checked that the mean number of the real roots of these samples was approximately given by (6.2).

We report on the comparison of the average residual errors versus degree in Figure 6.1. It is clear that GIF-FDR consistently obtains significantly better accuracy than GIF-SeDuMi. Figure 6.1 also contains comparison for cpu-time. Each instance was solved by GIF-SeDuMi first and the residual error recorded. This error was then used for the desired residual error when applying GIF-FDR. The average cpu-times per degree are plotted. Again we see that GIF-FDR performed consistently better even though it has a theoretical linear convergence time whereas interior point methods have a theoretical superlinear convergence time. In Figure 6.2 we used the popular performance profile approach [15] with the following performance profile function

$$\rho_s(\tau) = \frac{\mathrm{size}\{p \in \mathcal{P} : r_{p,s} \leq \tau\}}{\mathrm{size}(\mathcal{P})}, \quad s = 1,\ 2 \qquad (6.3)$$

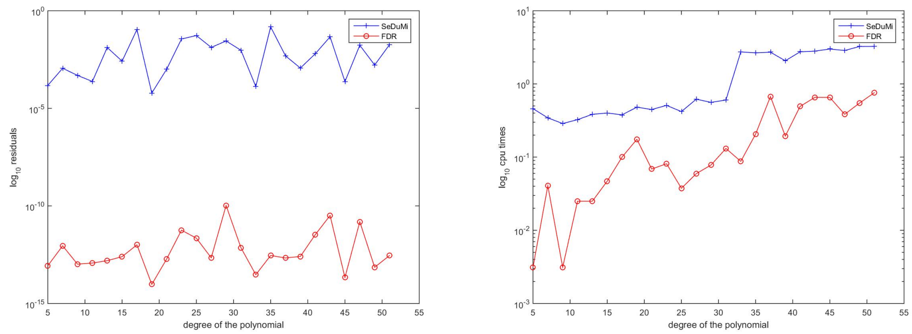

Figure 6.1: Comparison in residual and cputime of GIF-FDR vs GIF-SeDuMi for random polynomials $p_d(x) = \Sigma_1^d a_{d,j} x^j$ at odd degrees $3 \leq d \leq 51$ with $a_{d,j} \sim N(0, 1)$.

where $\mathcal{P}$ is the set of problems and $r_{p,s}$ is the ratio of the performance of solver s to the best performance by any solver on this problem p. These figures show FDR ($s = 2$) has outperformed SeDuMi ($s = 1$) in residual and cputime.

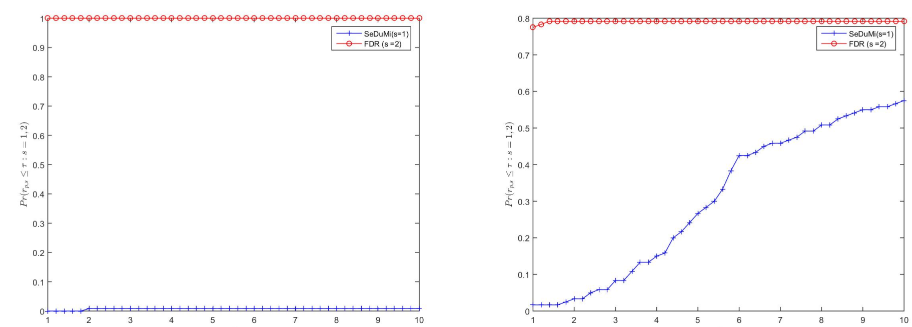

Figure 6.2: Performance profile of GIF-FDR vs GIF-SeDuMi for random polynomials $p_d(x) = \Sigma_1^d a_{d,j}\, x^j$ at each odd degrees $3 \leq d \leq 51$ with $a_{d,j} \sim N(0, 1)$. The profile function used is (6.3).

6.2 Examples of Ma, Wang and Zhi [34]

Ma, Wang and Zhi [33, 34] present an approach using Pommaret Bases coupled with moment matrix completion to approximate the real radical ideal of a polynomial variety. We applied our approach to [34, Examples 4.1-4.6], with the results shown in Table 6.1. In each of the examples we first applied GIF-FDR and then GIF-SeDuMi (i.e., FDR replaced with SeDuMi SDP solver). In each case we obtained a geometric involutive basis which can be independently verified as a geometric involutive basis for the real radical. In [34] Pommaret bases are successfully obtained for the real radical for these examples.

Here are the 6 systems of polynomials corresponding to the examples in [34]:

$$\{x_1^2 + x_1x_2 - x_1x_3 - x_1 - x_2 + x_3,\ \ x_1x_2 + x_2^2 - x_2x_3 - x_1 - x_2 + x_3,\\ x_1x_3 + x_2x_3 - x_3^2 - x_1 - x_2 + x_3\} \tag{6.4a}$$

$$\{x_1^2 - x_2,\ \ x_1x_2 - x_3\} \tag{6.4b}$$

$$\{x_1^2 + x_2^2 + x_3^2 - 2,\ \ x_1^2 + x_2^2 - x_3\} \tag{6.4c}$$

$$\{x_3^2 + x_2x_3 - x_1^2,\ \ x_1x_3 + x_1x_2 - x_3,\ \ x_2x_3 + x_2^2 + x_1^2 - x_1\} \tag{6.4d}$$

$$\{(x_1 - x_2)(x_1 + x_2)^2(x_1 + x_2^2 + x_2),\ \ (x_1 - x_2)(x_1 + x_2)^2(x_1^2 + x_2^2)\} \tag{6.4e}$$

$$\{(x_1 - x_2)(x_1 + x_2)(x_1 + x_2^2 + x_2),\ (x_1 - x_2)(x_1 + x_2)(x_1^2 + x_2^2),\\ x_1 \geq 1,\ x_2 \geq 1\} \tag{6.4f}$$

System (6.4a) **for [34, Example 4.1]**: The first step of applying Algorithm 4.1 is to use Maple and apply the GIF Algorithm 3.1, page 10, with input tolerance 10^{-10} to (6.4a). This shows that the system is already in geometric involutive form. The corresponding Pommaret basis is given in [34, Example 4.1]. The Pommaret basis looks different from the system, but is just a linear combination of the system's polynomials to accomplish the Gröbner-like requirement for its highest terms under the term ordering prescribed in the problem. The resulting coefficient matrix of this GIF form, is a full rank $m = 3$, 3×10 matrix which is input to the FDR algorithm. The dimension of the kernel for GIF form is $d = 7$. Since the coefficient matrix has rank $m = 3$, one facial reduction yields a reduced $(10 - m) \times (10 - m) = 7 \times 7$ moment matrix. Application of the FDR algorithm yields convergence in 2 iterations and 0.02 secs, with a projected residual error of 10^{-15}. These statistics are shown in Table 6.1. The output of FDR is a full 10×10 moment matrix of rank $r = 7$. Since $d = 7 = r$, Algorithm 4.1 terminates with the input system as its output. It can be checked that the ideal generated by this system is real radical.

For comparison, application of GIF-SeDuMi to (6.4a) using a tolerance of 10^{-10} in Maple resulted in a residual error of 10^{-10}, as listed in the last column of Table 6.1, and an approximation of the generators of the real radical.

System (6.4d) **for [34, Example 4.4]**: This is very similar to the previous system (6.4a). As [34] notes the coordinates for this example are not delta-regular, which they and we remedy by a linear change of coordinates. We show that the original system is geometrically involutive, which is equivalent to the determination of a Pommaret basis by [34]. Just as in the previous example, we form a 10×10 moment matrix from the GIF form, which is transformed by one facial reduction to a 7×7 matrix. There are no additional facial reductions, and the full moment matrix and its rank r are determined. We find that dimension of the kernel for GIF form is $d = 7 = r$, so Algorithm 4.1 terminates with the input system as its output. It can be verified the the output is a GIF form for the real radical of the ideal.

Application of GIF-SeDuMi to (6.4d) using a tolerance of 10^{-8} in Maple resulted in a residual error of 10^{-8} and an approximation of the generators of the real radical.

System (6.4b) **for [34, Example 4.2]**: This is quite similar to the systems (6.4b) and (6.4d). Our methods are similarly efficiently applied to this system. Our GIF algorithm first applied one prolongation to the second system (6.4b) to yield a degree 3 system. After projecting from this degree 3 system it shows that the resulting degree 2 system is involutive and consists of 3 polynomials. This degree 2 system is geometrically equivalent to the Pommaret basis found by [34]. This system is simply the original 2 polynomials, together with their compatibility condition or S-polynomial $x_2(x_1^2 - x_2) - x_1(x_1x_2 - x_3) = x_1x_3 - x_2^2$. Thus the input system R is replaced with $\pi \mathrm{D} R$ represented by its 3×10 coefficient matrix. The resulting 10×10 moment matrix is facially reduced to a 7×7 moment matrix. As in the previous examples, no new relations are detected in the kernel of the output matrix of the FDR method, $d = r = 7$ and the algorithm terminates. It can be verified that the GIF form is a basis for the real radical ideal of the input system.

Application of GIF-SeDuMi to (6.4b) using a tolerance of 10^{-9} in Maple resulted in a residual error of 10^{-9} and an approximation of the generators of the real radical.

Unlike the systems (6.4a),(6.4b),(6.4d), the remaining three systems (6.4c),(6.4e),(6.4f) of [34] lead to new members in the kernel of their moment matrices.

System (6.4c) **for [34, Example 4.3]**: Our initial application of FDR showed slow convergence. However a random linear change of coordinates applied to the input system R dramatically improved the convergence. Applying the GIF algorithm we found that $\widehat{\mathrm{D}}R$ is involutive and has a 8×20 coefficient matrix. The dimension of its kernel is $d = 12$. Applying the FDR algorithm, we obtain a PSD moment matrix with rank $r = 7 \neq d$ so the algorithm has not terminated. The new member of the real radical arising in the moment matrix kernel can be alternatively derived by hand by elimination of two of the systems polynomials: $x_1^2+x_2^2+x_3^2-2-(x_1^2+x_2^2-x_3) = x_3^2+x_3-2 = (x_3+2)(x_3-1)$. Then noting, as explained in [34], that only the root $x_3 = 1$ leads to real solutions. The GIF form of the new system from the kernel of the moment matrix is computed which has degree 2. Its coefficient matrix is 5×10 and has kernel of dimension $d = 5$. After applying FDR algorithm, the second PSD moment matrix then was computed quickly and accurately as a 10×10 matrix. The rank of the second moment matrix is $r = 5 = d$, so our algorithm has terminated. It can be checked that the output is equivalent to that found by [34] and that the resulting GIF form is a basis for the real radical.

Application of GIF-SeDuMi to (6.4c) using a tolerance of 10^{-8} in Maple resulted in a residual error of 10^{-9} and an approximation of the generators of the real radical.

System (6.4e) **for [34, Example 4.5]**: Direct application of Algorithm 4.1 to (6.4e) is relatively inefficient. Instead of this approach we consider an alternative subsystem approach which has the potential to be applied to larger systems. Exploiting subsystem structure is a long established approach in system solving.

We apply Algorithm 4.1 to the subsystem consisting of the first polynomial of $P_1 = (x_1-x_2)(x_1+x_2)^2(x_1+x_2^2+x_2)$ of (6.4e). The GIF form of P_1 is just P_1, and its coefficient matrix is 1×21 matrix with a kernel of dimension $d = 20$. The corresponding moment matrix is 21×21, which is reduced to a 20×20 matrix after one facial reduction. It has rank $r = 18 \neq d$. So the algorithm has not terminated, and new members of the real radical are identified from the kernel of the moment matrix. The new system is degree 5 and has 3 polynomials. Algorithm GIF shows that the first projection of this system is involutive and is a single fourth degree polynomial. Its coefficient matrix is 1×15 and its kernel has dimension $d = 14$. The FDR algorithm produces a 15×15 positive semidefinite moment matrix with the rank being $r = 14 = d$. The algorithm terminates to coefficient errors within 10^{-10} with output as a single polynomial which is approximately:

$$(x_1 - x_2)(x_1 + x_2)(x_1 + x_2^2 + x_2) \tag{6.5}$$

It can be checked that (6.5) is a geometric involutive basis for the real radical for the ideal generated by P_1.

Similarly we apply Algorithm 4.1 to the second polynomial of (6.4e) which is given by $P_2 = (x_1 - x_2)(x_1 + x_2)^2(x_1^2 + x_2^2)$. The algorithm now terminates with output as a single polynomial which is approximately:

$$(x_1 - x_2)(x_1 + x_2) \tag{6.6}$$

This can be verified to be a geometric involutive basis for the real radical of the ideal generated by P_2.

Then we consider the system

$$(x_1 - x_2)(x_1 + x_2)(x_1 + x_2^2 + x_2), \;\; (x_1 - x_2)(x_1 + x_2) \tag{6.7}$$

Application of GIF to (6.7) reduces it to a geometric involutive basis which is approximately

$$(x_1^2 - x_2^2) \tag{6.8}$$

A further application of FDR reveals that (6.8) is a GIF form for the real radical of the ideal of (6.4e).

Application of GIF-SeDuMi to (6.4e) also yields an approximation of the generators of the real radical. The most notable feature of this calculation was the its requirement of fairly large tolerances (10^{-4} and 10^{-5}). Reference [34, Example 4.5] also notes a similarly large tolerance in their calculations, to correctly compute the real radical for this example.

System (6.4f) **for [34, Example 4.6]**: Let $Q_1 = \{(x_1 - x_2)(x_1 + x_2)(x_1 + x_2^2 + x_2), \;\; (x_1 - x_2)(x_1 + x_2)(x_1^2 + x_2^2)\}$ then (6.4f) is Q_1 subject to the constraints $x_1 \geq 1$, $x_2 \geq 1$.

Applying Algorithm 4.1 to Q_1 yields a geometric involutive basis which is approximately $x_1^2 - x_2^2$. This can be independently verified to be a geometric basis for the real radical of Q_1. The statistics of this reduction are given in Table 6.1 in the row labeled as Ex 4.6 Q_1.

To impose $x_1 \geq 1$, $x_2 \geq 1$ we substitute $x_1 = x_3^2 + 1, x_2 = x_4^2 + 1$ into the geometric involutive basis of the real radical of Q_1, that is into $x_1^2 - x_2^2$, and reduce the resulting polynomial $Q_2 = (x_3^2 + 1)^2 - (x_4^2 + 1)^2 = (x_3^2 - x_4^2)(x_3^2 + x_4^2 + 2)$ with Algorithm 4.1 to yield a basis for its real radical which is $x_3^2 - x_4^2$ or equivalently $x_1 - x_2$ in agreement with [34, Example 4.6]. The statistics of this reduction are given in Table 6.1 in the row labeled as Ex 4.6 Q_2.

Application of GIF-SeDuMi to (6.4f) also yields an approximation of the real radical. The most notable feature of this calculation was the large tolerance 10^{-6} and residual error for the reduction of Q_1.

Polyn. System	Input data (n,d,m)	FDR # its (1,2)	FDR cpu-sec (1,2)	FDR res-err max(1,2)	Mom Mtx redn factor $s(M)/s(\hat{M})$	GIF-SeDuMi Int Pt tol, res err
Ex 4.1	(3,2,3)	2	0.02	10^{-15}	$\frac{10}{7}$	$10^{-10}, 10^{-10}$
Ex 4.2	(3,2,2)	156	0.23	10^{-14}	$\frac{10}{7}$	$10^{-9}, 10^{-9}$
Ex 4.3	(3,2,2)	256, 2	2.4, 0.08	10^{-13}	$\frac{20}{12}, \frac{10}{5}$	$10^{-8}, 10^{-9}$
Ex 4.4	(3,2,3)	106	0.06	10^{-15}	$\frac{10}{7}$	$10^{-8}, 10^{-8}$
Ex 4.5 P_1	(2,5,1)	9582, 29	7.0, 0.17	10^{-13}	$\frac{21}{20}, \frac{15}{14}$	$10^{-4}, 10^{-8}$
Ex 4.5 P_2	(2,5,1)	148, 1	0.3,0.06	10^{-14}	$\frac{21}{20}, \frac{6}{5}$	$10^{-5}, 10^{-8}$
Ex 4.6 Q_1	(2,4,2)	34, 2	0.11,0.08	10^{-13}	$\frac{21}{15}, \frac{6}{5}$	$10^{-6}, 10^{-8}$
Ex 4.6 Q_2	(2,4,1)	86, 1	0.28, 0.03	10^{-14}	$\frac{15}{14}, \frac{6}{5}$	$10^{-8}, 10^{-9}$
Cyl2d	(2,2,1)	1	0.06	10^{-15}	$\frac{6}{5}$	$10^{-10}, 10^{-13}$
Cyl3d	(3,2,2)	2	0.09	10^{-15}	$\frac{20}{12}$	$10^{-8}, 10^{-9}$
Cyl4d	(4,2,3)	7	0.31	10^{-14}	$\frac{70}{28}$	$10^{-7}, 10^{-8}$
Cyl5d	(5,2,4)	10	0.52	10^{-14}	$\frac{252}{64}$	DNC

Table 6.1: **Statistics for the application of GIF-FDR and GIF-SeDuMi:** Ex 4.1-4.6 are 6 examples in MWZ [34]; Cyl2d-Cyl5d are cylinder examples; n number of variables; d maximum polynomial degree; m number of polynomials; two entries (1,2) are included for the number of iterations and cpu-time if FDR is used twice in the example; $(s(M), s(\hat{M}))$ sizes of moment matrix M and facially reduced matrix $\hat{M}$, resp. Rightmost two columns are SVD tolerance and moment matrix residual error for the Interior Point calculation using SeDuMi combined with GIF. DNC - Did Not Converge. The Maple SVD computations in GIF-FDR were executed with tolerance $:= 10^{-10}$ and $Digits := 15$.

6.3 Intersecting higher dimensional cylinders

Consider the systems of polynomials defining the intersection of $n-1$ cylinders in $\mathbb{R}^n$

$$Cyl_{nd} := x_1^2 + x_2^2 - 1, x_1^2 + x_3^2 - 1, \cdots, x_1^2 + x_n^2 - 1. \tag{6.9}$$

Application of the GIF algorithm to the systems Cyl_{nd} for $n = 2, 3, 4, 5$ show that the systems become geometrically involutive after $0, 1, 2, 3$ prolongations respectively. The GIF-FDR algorithm converges quickly and accurately (see Table 6.1). It can be independently determined that in each case it yields an geometric involutive basis for the real radical. However SeDuMi-GIF crashes after several hours on the largest system Cyl_{5d} .

Further it can be determined that the cylinders form a complete intersection and the length of the prolongation to make them involutive, can be determined from the symbol of the initial system [37]. The lower degree input systems (6.9) are geometrically formally integrable, and it would be interesting to develop methods based on such lower degree systems, to determine, whether one can rule out new members in the kernel of the moment matrix of the prolonged involutive system from such lower degree systems.

Recently certain critical point methods have been developed for determining witness points [26,50] on real components of real polynomial systems. Indeed the method developed in [50] is successful in finding a point on every component, if the ideal is both real radical, and forms a regular sequence. Consequently for systems such as those above, the real radical is an important property for such solvers. The regular sequence requirement can be checked by dimension computation and can exploit a formally integrable system which has lower degree than the involutive system. Interesting related results are given in [35]. By experiment we found that the 0 dimensional systems for the critical points of (6.9) are also real radical and remarkably have no non-real roots. The number of real critical points corresponding to $n = 2, 3, 4, 5$ can be determined to be $2, 4, 8, 16$.

7 Conclusion

SDP feasibility problems typically involve the intersection of the convex cone of semidefinite matrices with a linear manifold. Their importance in applications has led to the development of many specific algorithms. However these feasibility problems are often marginally infeasible, i.e., they do not satisfy strict feasibility as is the case for our polynomial applications. Such problems are *ill-posed* and *ill-conditioned*.

The main contribution of this paper is to introduce facial reduction, for the class of SDP problems arising from analysis and solution of systems of real polynomial equations for real solutions. Facial reduction yields an equivalent problem for which there are strictly feasible points and which, in addition, are smaller. Facial reduction also reduces the size of the moment matrices occurring in the application of SDP methods. For example the determination of a $k \times k$ moment matrix for a problem with m linearly independent constraints is reduced to a $(k-m) \times (k-m)$ moment matrix by one facial reduction. We use facial reduction with our MATLAB implementation of Douglas-Rachford iteration (our FDR method). In the case of only one constraint, say as in the case of univariate polynomials, one might expect that the improvement in convergence due to that facial reduction would be minor. However we present a class of random univariate polynomials, where one such facial reduction combined with DR iteration, yields the real radical much more efficiently than the standard interior point method in SeDuMi. The high accuracy required by facial reduction and also the ill-conditioning commonly encountered in numerical polynomial algebra [46] motivated us to implement Douglas-Rachford iteration.

A fundamental open problem is to generalize the work of [30, 45] to positive dimensional ideals. The algorithm of [33, 34] for a given input real polynomial system P, modulo the successful application of SDP methods at each of its steps, computes a Pommaret basis Q:

$$\sqrt[\mathbb{R}]{\langle P \rangle_{\mathbb{R}}} \;\supseteq\; \langle Q \rangle_{\mathbb{R}} \;\supseteq\; \langle P \rangle_{\mathbb{R}} \tag{7.1}$$

and would provide a solution to this open problem if it is proved that $\langle Q \rangle_{\mathbb{R}} = \sqrt[\mathbb{R}]{\langle P \rangle_{\mathbb{R}}}$. We believe that the work [33, 34] establishes an important feature – involutivity – that will necessarily be a main condition of any theorem and algorithm characterizing the real radical. Involutivity is a natural condition, since any solution of the above open problem using SDP, if it establishes radical ideal membership, will necessarily need (at least implicitly) a real radical Gröbner basis. Our algorithm, uses geometric involutivity, and similarly gives an intermediate ideal, which constitutes another variation on this family of conjectures.

In addition to implementing an algorithm to determine a first facial reduction. We also implemented a test for the existence of additional facial reductions beyond the first (e.g., in the cases of Examples 4.3 and 4.5 of [34]). By using the CVX package or Douglas-Rachford iteration to solve for the auxiliary problem (5.7), we can determine if we need a second facial reduction by checking whether the optimal value of the auxiliary problem is close to 0. Our implementation of auxiliary facial reductions, as still preliminary and needs improvement. So a more detailed study of this aspect is worthwhile.

Numerical polynomial algebra has been a rapidly expanding and popular area [46]. Its problems are typically very demanding, motivating the implementation of methods to improve accuracy. For example Bertini, the homotopy package developed for numerical polynomial algebra, uses variable precision arithmetic, with particularly demanding problems requiring thousands of digits of precision. Consequently this is also a motivation to develop higher accuracy methods, such as the FDR method of this paper. Manipulations with radical ideals would be a by-product from such work. An important open problem is the following: *Give an numerical algorithm, capable in principle of determining an approximate real witness point on each component of a real variety.* We note that the methods of Wu and Reid [50] and Hauenstein [26] only answer this question under certain conditions, say that the ideal is real radical and defined by a regular sequence. Also see [32], which gives an alternative extension of complex numerical algebraic geometry to the reals, in the complex curve case.

We provided a small set of examples, that illustrate some aspects of our algorithms. In Maple all of our examples were executed with Maple's $Digits := 15$ and the input tolerance $:= 10^{-10}$ for the `GIF` algorithm which intensively uses LAPack's SVD. Accuracy in the projected residual error for our tests were between 10^{-14} and 10^{-12}. The normalized generators obtained for our experiments had coefficients differing less than 10^{-10} from the exact coefficients.

In addition we prove that our facial reduction steps are backwards stable. See Theorem 5.2 and Section 5.2.2. The advantage for the use of Douglas-Rachford iterations in our SDP solution techniques and its linear convergence is discussed at the end of Section 5.3.2. We note that the simplest structured matrices from polynomial systems are Hankel matrices and are notoriously ill-conditioned, see e.g., [7, 23]. In particular such matrices all lie *close* to the boundary of the semidefinite cone. Therefore, even after successful facial reduction guarantees a strictly feasible solution, the set of Hankel matrices are all *nearly singular*. This makes the related feasibility problems particularly difficult. Despite this we were successful in finding feasible solutions. Such conditioning issues warrant further study. Indeed consider $p(x, y) = x^2 + y^2 + \epsilon = 0$. Even though $(x, y) = (0, 0)$ is the unique solution for $\epsilon = 0$, with associated real radical ideal $\langle x, y\rangle_{\mathbb{R}}$, the solution is not a real continuous function of ϵ as ϵ passes through 0. So the problem in terms of the variety is not well-posed. An interesting challenge is to formulate appropriate well-posed nearby problems in an appropriate space. The backwards stable tools, of facial reduction and auxiliary reduction, and associated spaces are interesting possibilities for such approaches.

References

[1] A. Alfakih and H. Wolkowicz. Matrix completion problems. In *Handbook of semidefinite programming*, volume 27 of *Internat. Ser. Oper. Res. Management Sci.*, pages 533–545. Kluwer Acad. Publ., Boston, MA, 2000. 3

[2] A.F. Anjos and J.B. Lasserre, editors. *Handbook on Semidefinite, Conic and Polynomial Optimization.* International Series in Operations Research & Management Science. Springer-Verlag, 2011. 5, 6, 15

[3] F.J.A. Artacho, J.M. Borwein, and M.K. Tam. Recent results on Douglas-Rachford methods. *Serdica Mathematical Journal*, 39:313–330, 2013. 20

[4] S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in Real Algebraic Geometry*, volume 10 of *Algorithms and Computation in Math.* Springer-Verlag, 2 edition, 2006. 3, 5, 11

[5] H. H. Bauschke and D. Noll. On the local convergence of the Douglas-Rachford algorithm. *Arch. Math. (Basel)*, 102(6):589–600, 2014. 21

[6] H.H. Bauschke, J.Y. Bello Cruz, T.T.A. Nghia, H.M. Phan, and X. Wang. The rate of linear convergence of the Douglas-Rachford algorithm for subspaces is the cosine of the Friedrichs angle. *J. Approx. Theory*, 185:63–79, 2014. 21

[7] B. Beckermann. The condition number of real Vandermonde, Krylov and positive definite Hankel matrices. *Numer. Math.*, 85(4):553–577, 2000. 31

[8] G. Blekherman, P.A. Parrilo, and R.R. Thomas, editors. *Semidefinite optimization and convex algebraic geometry*, volume 13 of *MOS-SIAM Series on Optimization.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA; Mathematical Optimization Society, Philadelphia, PA, 2013. 3, 5, 15

[9] J. Bonasia, F. Lemaire, G.J. Reid, and L. Zhi. Determination of approximate symmetries of differential equations. *Group Theory and Numerical Analysis*, 39:249, 2005. 7, 8, 9

[10] J.M. Borwein and M.K. Tam. A Cyclic Douglas–Rachford Iteration Scheme. *J. Optim. Theory Appl.*, 160(1):1–29, 2014. 20

[11] J.M. Borwein and H. Wolkowicz. Facial reduction for a cone-convex programming problem. *J. Austral. Math. Soc. Ser. A*, 30(3):369–380, 1980/81. 3

[12] J.M. Borwein and H. Wolkowicz. Regularizing the abstract convex program. *J. Math. Anal. Appl.*, 83(2):495–530, 1981. 3

[13] Y-L. Cheung, S. Schurr, and H. Wolkowicz. Preprocessing and regularization for degenerate semidefinite programs. In D.H. Bailey, H.H. Bauschke, P. Borwein, F. Garvan, M. Thera, J. Vanderwerff, and H. Wolkowicz, editors, *Computational and Analytical Mathematics, In Honor of Jonathan Borwein's 60th Birthday*, volume 50 of *Springer Proceedings in Mathematics & Statistics*, pages 225–276. Springer, 2013. 16, 17, 18

[14] Y.-L. Cheung and H. Wolkowicz. Sensitivity analysis of semidefinite programs without strong duality. Technical report, University of Waterloo, Waterloo, Ontario, 2014. submitted June 2014, 37 pages. 3

[15] E.D. Dolan and J.J. Moré. Benchmarking optimization software with performance profiles. *Math. Program.*, 91(2, Ser. A):201–213, 2002. 22

[16] Jr.J. Douglas and Jr.H.H. Rachford. On the numerical solution of heat conduction problems in two and three space variables. *Trans. Amer. Math. Soc.*, 82:421–439, 1956. 20

[17] D. Drusvyatskiy, N. Krislock, Y-L. Cheung Voronin, and H. Wolkowicz. Noisy sensor network localization: robust facial reduction and the Pareto frontier. Technical report, University of Waterloo, Waterloo, Ontario, 2014. arXiv:1410.6852, 20 pages. 3

[18] D. Drusvyatskiy, G. Li, and H. Wolkowicz. Alternating projections for ill-posed semidenite feasibility problems. Technical report, University of Waterloo, Waterloo, Ontario, 2014. submitted Sept. 2014, 12 pages. 3, 16

[19] M. Dür, B. Jargalsaikhan, and G. Still. The Slater condition is generic in linear conic programming. Technical report, University of Trier, Trier, Germany, 2012. 3

[20] C. Eckart and G. Young. A principal axis transformation for non-Hermitian matrices. *Bull. Amer. Math. Soc.*, 45:118–121, 1939. 19

[21] A. Edelman and E. Kostlan. How many zeros of a random polynomial are real? *Bull. Amer. Math. Soc. (N.S.)*, 32(1):1–37, 1995. 2, 22

[22] R. Escalante and M. Raydan. *Alternating projection methods*, volume 8 of *Fundamentals of Algorithms*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2011. 19

[23] W. Gautschi and G. Inglese. Lower bounds for the condition number of Vandermonde matrices. *Numer. Math.*, 52(3):241–250, 1988. 31

[24] V.P. Gerdt and Y.A. Blinkov. Involutive bases of polynomial ideals. *Mathematics and Computers in Simulation*, 45(5):519–541, 1998. 7

[25] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, Maryland, 3^{nd} edition, 1996. 18

[26] Jonathan D Hauenstein. Numerically computing real points on algebraic sets. *Acta applicandae mathematicae*, 125(1):105–119, 2013. 29, 31

[27] M. Kac. On the average number of real roots of a random algebraic equation. *Bull. Amer. Math. Soc.*, 49:314–320, 1943. 22

[28] N. Krislock and H. Wolkowicz. Explicit sensor network localization using semidefinite representations and facial reductions. *SIAM Journal on Optimization*, 20(5):2679–2708, 2010. 3

[29] M. Kuranishi. On e. cartan's prolongation theorem of exterior differential systems. *American Journal of Mathematics*, pages 1–47, 1957. 7

[30] J.B. Lasserre, M. Laurent, and P. Rostalski. A prolongation–projection algorithm for computing the finite real variety of an ideal. *Theoretical Computer Science*, 410(27):2685–2700, 2009. 2, 3, 4, 12, 30

[31] M. Laurent and P. Rostalski. The approach of moments for polynomial equations. In Miguel F. Anjos and Jean B. Lasserre, editors, *Handbook*

on semidefinite, conic and polynomial optimization, International Series in Operations Research & Management Science, 166, pages 25–60. Springer, New York, 2012. 6

[32] Y. Lu, D.J. Bates, A.J. Sommese, and C.W. Wampler. Finding all real points of a complex curve. In *Algebra, geometry and their interactions*, volume 448 of *Contemp. Math.*, pages 183–205. Amer. Math. Soc., Providence, RI, 2007. 31

[33] Y. Ma. *Polynomial Optimization via Low-rank Matrix Completion and Semidefinite Programming.* PhD thesis, 2012. 2, 4, 10, 23, 30

[34] Y. Ma, C. Wang, and L. Zhi. A certificate for semidefinite relaxations in computing positive dimensional real varieties. *Journal of Symbolic Computation*, 72:1 – 20, 2016. 2, 4, 10, 23, 24, 25, 26, 27, 28, 30, 40, 41

[35] Y. Ma and L. Zhi. Computing real solutions of polynomial systems via low-rank moment matrix completion. In *Proceedings of the 37th International Symposium on Symbolic and Algebraic Computation*, pages 249–256. ACM, 2012. 29

[36] F.S. Macaulay and P. Roberts. *The algebraic theory of modular systems.* Number 19. University press Cambridge, 1916. 6

[37] H.M. Möller and T. Sauer. H-bases for polynomial interpolation and system solving. *Advances in Computational Mathematics*, 12(4):335–362, 2000. 6, 29

[38] B. Mourrain. Isolated points, duality and residues. *Journal of Pure and Applied Algebra*, 117:469–493, 1997. 6

[39] B. Mourrain. A new criterion for normal form algorithms. In *Applied algebra, algebraic algorithms and error-correcting codes*, pages 430–442. Springer, 1999. 6

[40] G.J. Reid, J. Tang, and L. Zhi. A complete symbolic-numeric linear method for camera pose determination. In *Proceedings of the 2003 international symposium on Symbolic and algebraic computation*, pages 215–223. ACM, 2003. 8

[41] G.J. Reid, F. Wang, and W. Wu. Geometric involutive bases for positive dimensional polynomial ideals and sdp methods. Technical report, Department of Appl. Math., University of Western Ontario, 2014. 6, 7, 8, 10

[42] G.J. Reid and L. Zhi. Solving polynomial systems via symbolic-numeric reduction to geometric involutive form. *Journal of Symbolic Computation*, 44(3):280–291, 2009. 8

[43] R. Scott, G.J. Reid, W. Wu, and L. Zhi. Geometric involutive bases and applications to approximate commutative algebra. In Lorenzo Robbiano and John Abbott, editors, *Approximate Commutative Algebra*, pages 99–124. Springer, 2010. 3, 8

[44] A.J. Sommese and C.W. Wampler. *The Numerical solution of systems of polynomials arising in engineering and science*, volume 99. World Scientific, 2005. 2

[45] F. Sottile. *Real solutions to equations from geometry*, volume 57 of *University Lecture Series*. American Mathematical Society, Providence, RI, 2011. 2, 3, 4, 5, 30

[46] Hans J. Stetter. *Numerical polynomial algebra*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2004. 6, 30, 31

[47] A.D. Wittkopf and G.J. Reid. Fast differential elimination in c: The cdiffelim environment. *Computer Physics Communications*, 139(2):192–217, 2001. 7

[48] H. Wolkowicz, R. Saigal, and L. Vandenberghe, editors. *Handbook of semidefinite programming*. International Series in Operations Research & Management Science, 27. Kluwer Academic Publishers, Boston, MA, 2000. Theory, algorithms, and applications. 15

[49] H. Wolkowicz and Q. Zhao. Semidefinite programming relaxations for the graph partitioning problem. *Discrete Appl. Math.*, 96/97:461–479, 1999. Selected for the special Editors' Choice, Edition 1999. 3

[50] W. Wu and G.J. Reid. Finding points on real solution components and applications to differential polynomial systems. In *Proceedings of the 38th international symposium on International symposium on symbolic and algebraic computation*, pages 339–346. ACM, 2013. 29, 31

[51] X. Wu and L. Zhi. Determining singular solutions of polynomial systems via symbolic–numeric reduction to geometric involutive forms. *Journal of Symbolic Computation*, 47(3):227–238, 2012. 8

[52] Q. Zhao, S.E. Karisch, F. Rendl, and H. Wolkowicz. Semidefinite programming relaxations for the quadratic assignment problem. *J. Comb. Optim.*, 2(1):71–109, 1998. Semidefinite programming and interior-point approaches for combinatorial optimization problems (Fields Institute, Toronto, ON, 1996). 3

Index

Contents

List of Tables

List of Algorithms

List of Figures