Facial Reduction and SDP Methods for Systems of Polynomial Equations

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

Monday 5th January, 2015

Abstract

The real radical ideal of a system of polynomials with finitely many complex roots is generated by a system of real polynomials having only real roots and free of multiplicities. It is a central object in computational real algebraic geometry and important as a preconditioner for numerical solvers. Lasserre and co-workers have shown that the real radical ideal of real polynomial systems with finitely many real solutions can be determined by a combination of semi-definite programming (SDP) and geometric involution techniques. A conjectured extension of such methods to positive dimensional polynomial systems has been given recently by Ma, Wang and Zhi.

We show that regularity in the form of the Slater constraint qualification (strict feasibility) fails for the resulting SDP feasibility problems. Facial reduction is then a popular technique whereby SDP problems that fail strict feasibility can be regularized by projecting onto a face of the convex cone of semi-definite problems.

In this paper we introduce a framework for combining facial reduction with such SDP methods for analyzing 0 and positive dimensional real ideals of real polynomial systems. The SDP methods are implemented in MATLAB and our geometric involutive form is implemented in Maple. We use two approaches to find a feasible moment matrix. We

*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 cstc2013jjys0002 and West Light Foundation of the Chinese Academy of Science.

use an interior point method within the CVX package for MATLAB and also 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. Problems requiring more than one facial reduction are also presented.

1 Introduction

In breathrough work Lasserre and collaborators [25,39] have shown that the real radical ideal of real polynomial systems with finitely many real solutions can be determined by a combination of SDP and geometric involution techniques. The real radical ideal of a system of polynomials with finitely many complex roots is generated by a system of real polynomials only having real roots and free of multiplicities. It is a central object in computational real algebraic geometry and important as a preconditioner for numerical solvers. A conjectured extension of such methods to positive dimensional polynomial systems has been given recently by Ma, Wang and Zhi [27, 28].

The above approaches use the *method of moments* and the *Semi-definite Programming, SDP* formulation. In this paper we see that the Slater constraint qualification, strict feasibility, fails for the SDP formulation resulting in an ill-posed feasibility problem. Our main contribution is to use *facial reduction* to project the problem onto the *minimal face* to help regularize these computations. Our approach provides tools for working with the ideals involved, and gathering data on the open problem above.

1.1 SDP and Facial Reduction

The SDP formulation of the moment problem is equivalent to finding X for the linear feasibility system

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

where $\mathcal{S}^k_+$ denotes the convex cone of $k \times k$ real symmetric positive semi-definite 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 holds generically, e.g., [17]. 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 in the boundary of the cone, and even the slightest perturbation 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 [10, 11]. However it was only much later that the power of facial reduction was exhibited in many applications, e.g., [1, 43, 46]. 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., [12, 13, 16, 23].

A polynomial system of equations can be viewed as a linear (or coefficient matrix) function of its monomials [25, 39]. This linear function 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 Macaulay structure possessed by the polynomial systems. Remarkable advances have been recently made in this area [7, 25, 39] 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.

1.2 Prolongation projection methods for involutive bases of polynomial systems

We now look at the details in the semi-definite linear constraint $\mathcal{A}X = b$ for the polynomial systems. Polynomial systems are remarkable, in that many of their constraints are *hidden*. For example consider the degree two system

$$x^2 - x - 1 = 0, \quad xy - y - 1 = 0.$$

A single *prolongation* of this system to degree 3 is found by multiplying them by each of the variables x and y:

$$\begin{array}{rcl} x(x^2 - x - 1) & = & x^3 - x^2 - x \\ x(xy - y - 1) & = & x^2y - xy - x \\ y(x^2 - x - 1) & = & x^2y - xy - y \\ y(xy - y - 1) & = & xy^2 - y^2 - y. \end{array} \tag{1.3}$$

Projecting in our paper loosely means 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 (1.3):

$$\left\{ \begin{array}{l} x^2y - xy - x = 0 \\ x^2y - xy - y = 0 \end{array} \right\} \implies xy + x = xy + y. \tag{1.4}$$

Consequently we obtain the new projected (hidden) constraint $x = y$. This process of uncovering the hidden polynomial constraints by prolongation and projection is effected numerically through our geometric involutive form algorithm which has been implemented in Maple [34, 38].

We note that familiar methods for linear systems of equations 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* [5]; while in the approximate case we use *geometric involutive bases* [38].

1.3 Facial Reduction and SDP methods applied to real radical ideals of polynomial systems

A major motivation for our paper is the success of the work of Lasserre et al [25] which gives a new symbolic-numeric approach for computing the real radical ideal of zero dimensional polynomial systems using geometric involution and SDP techniques. Zero dimensional real polynomial systems are systems with real coefficients and finitely many complex and real roots. Another major motivation is the important work on this topic in [27, 28] which conjectures an extension of [25] to positive dimensional real radical ideals. Such ideals have associated real solution components (manifolds) of dimension ≥ 1. (See also the paper [36] for examples and many references.)

The *real radical ideal,* `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 `RRI`s and their interpretation, consider the simple case of *univariate polynomials* with real coefficients, $n = 1$. In particular, a real univariate polynomial $p(x)$ can be factored in real factors $(x - a_j)$ and conjugate complex factors $(x - \alpha_\ell)$, $(x - \bar{\alpha}_\ell)$ so that

$$p(x) = \Pi_j (x - a_j)^{d_j} \Pi_k (x - \alpha_k)^{r_k} (x - \bar{\alpha}_k)^{r_k}, \tag{1.5}$$

where d_j and r_k are the multiplicities of the roots. The *real polynomial ideal* generated by $p(x)$ is the set of polynomials of the form $g(x)p(x)$ where $g(x)$

is any real polynomial. The RRI of $p(x)$ is generated by the polynomial

$$q(x) = \Pi_j(x - a_j). \tag{1.6}$$

In many applications we are only interested in real roots, and the RRI shown here discards all the complex roots. Moreover it also discards multiplicities which is important in improving conditioning for polynomial solvers. Many general polynomial system solvers, that are capable of determining all solutions explicitly or implicitly, compute all complex and real roots first. In particular a generic system of n degree d polynomials in n variables generically has d^n roots and potentially very few roots. Thus the development of methods that avoid the calculation of the complex roots and multiplicities is important for efficiency of polynomial system solvers.

1.4 Outline

Since we use sophisticated results from diverse areas, in Section 2 we present basic ideas and objects through simple examples. We give a preliminary introduction to moment matrices and also give a preliminary simple illustration of the power of facial reduction in Section 2.3.

In Section 3 we give a condensed and more formal description of geometric involutive bases and related algorithms. In Section 4 we discuss moment matrices and related algorithms.

In Section 5 we discuss the methods we used to solve our SDP feasibility problems. Since the polynomial problems we consider fail strict feasibility, we will use facial reduction to regularize them. However standard primal-dual interior point semi-definite programming packages do not deliver the accuracy required to guarantee facial reduction. This motivates us to use *Douglas-Rachford (DR)* projection/reflection methods.

In Section 6 we will discuss our implementation of facial reduction. In Section 7 we give numerical experiments. Our concluding remarks are in Section 8.

2 Basic setup and illustrative examples

This paper uses sophisticated methods from diverse areas. To help the reader, we informally introduce the methods of the paper and illustrate them by simple examples. This helps emphasize that the operations underlying our approach are reasonably straightforward.

2.1 Real polynomial systems

For background and references to real algebraic geometry and semi-definite programming see e.g., [2, 5, 7, 39, 42].

We consider a (finite) system of ℓ polynomials $P = \{p_1, ..., p_\ell\} \subset \mathbb{R}[x_1, \ldots, x_n] = \mathbb{R}[x]$, where $\mathbb{R}[x]$ is the set of all polynomials with real coefficients in the n variables $x = \begin{pmatrix} x_1 & x_2 & \ldots & x_n \end{pmatrix}^T$. 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

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

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 $P = \{p_1, \ldots, p_\ell\} \subset \mathbb{R}[x]$ is:

$$\langle P \rangle_{\mathbb{R}} = \langle p_1, \ldots, p_\ell \rangle_{\mathbb{R}} = \{f_1 p_1 + \ldots + f_\ell p_\ell : f_j \in \mathbb{R}[x], \forall 1 \le j \le \ell\}. \tag{2.2}$$

Monomials are denoted 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, and the *degree of* x^α is $|\alpha| := \|\alpha\|_1 = \alpha_1 + \cdots + \alpha_n$. It is clear that the degree of each monomial $|\alpha| \le d$, the degree of the polynomial. Then for appropriate coefficients $a_{k,\alpha}$, and for each k,

$$\begin{array}{l} \text{we sort by total degree of } |\alpha| \text{ in nondecreasing order} \\ \text{with components of } \alpha \text{ sorted in lexicographic order.} \end{array} \tag{2.3}$$

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

$$P = \left\{ \sum_{|\alpha| \le d} a_{k,\alpha}\, x^\alpha : k = 1, \ldots, \ell \right\}. \tag{2.4}$$

Throughout this paper, we use graded reverse lexicographic order, which orders first by degree and then by reverse lexicographic order. This order respects the Cartan class of variables, which is important in our numerical determination geometric features of polynomial systems such as those in Definition 3.3.

Definition 2.1 (Coefficient matrix of P, $C(P)$)**.** *Let* $\boldsymbol{x}^{(\le d)}$ *be the column vector of monomials* x^α *with* $0 \le |\alpha| \le d$ *sorted as in* (2.3)*. Suppose that the coefficients* $a_{k,\alpha}$ *in* (2.4) *are similarly sorted. Then define the coefficient matrix of* P *by* $C(P) = (a_{k,\alpha})$*.*

The following lemma follows immediately.

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

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

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

The well-known presentation of polynomial systems as linear functions of their monomials and the related coefficient matrix and its kernel and rowspace has been exploited in [31–33, 40] and in the historical work by Macaulay [30].

Example 2.1. *Consider the system of two univariate polynomials*

$$P = \{x^8 - x^4 - 2, x^8 - 3x^4 + 2\} \subset \mathbb{R}[x]. \tag{2.5}$$

Here the coefficient matrix is given by $C(P)$ in the equations

$$C(P)\boldsymbol{x}^{(\leq 8)} = \begin{pmatrix} -2 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 0 & -3 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} 1 \\ x \\ \vdots \\ x^7 \\ x^8 \end{bmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{2.6}$$

A familiar computation for many readers is to eliminate the polynomials using a Gröbner basis calculation: $x^8 - x^4 - 2 - (x^8 - 3x^4 + 2) = 2x^4 - 4$ or equivalently $x^4 - 2$. The original 8 degree polynomials can be discarded since they are consequences of $x^4 - 2$. In particular $x^8 - x^4 - 2 = x^4(x^4 - 2) + (x^4 - 2) = (x^4 + 1)(x^4 - 2)$ so it lies in the ideal generated by $x^4 - 2$. Similarly $x^8 - 3x^4 + 2$ lies in the ideal generated by $x^4 - 2$ and can be discarded. All polynomials in the ideal generated by P are polynomial multiples of the single polynomial

$$x^4 - 2. \tag{2.7}$$

It is easy to see that every system of univariate polynomials is equivalent to a single univariate polynomial by applying such simple operations. For systems of multivariate polynomials, such a minimal object is called a Gröbner basis. Gröbner bases have been intensively studied [14] and usually consist of several polynomials. We use the geometric involutive form algorithm discussed in Section 3 to obtain a numerically stable cousin of Gröbner bases.

2.2 Moment matrices and polynomials

Moment matrices combined with SDP provide a method to discard the complex roots in polynomial systems with finitely many roots, such as the two complex roots of x^4-2 in Example 2.1 above. Here we focus on the construction of moment matrices. For theoretical background the reader is directed to e.g., [2, 26].

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 $M_{0,0} = 1$.

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*

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

Given a multivariate polynomial system $P \subset \mathbb{R}[x]$, with $d = \deg(P)$ and $M \in \mathbb{R}^{N(n,d)\times N(n,d)}$ be the truncated real symmetric moment matrix. The linear constraints imposed by P are, see (2.9) below,

$$C(P)\, M = 0,$$

where $C(P)$ is the coefficient matrix function given in Definition 2.1.

Example 2.2 (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:*

$$M(u) = \begin{bmatrix} u_0 & u_1 & u_2 & u_3 & u_4 & \cdots \\ u_1 & u_2 & u_3 & u_4 & u_5 & \cdots \\ u_2 & u_3 & u_4 & u_5 & u_6 & \cdots \\ u_3 & u_4 & u_5 & u_6 & u_7 & \cdots \\ u_4 & u_5 & u_6 & u_7 & u_8 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \qquad u_0 = 1. \tag{2.8}$$

Note that (2.8) is a Hankel matrix. In Example 2.1 a degree 8 input system was reduced to a degree 4 output polynomial $P = \{x^4 - 2\}$. Let us associate

$u_\alpha \leftrightarrow x^\alpha$. *Then we recover the polynomial equation using the coefficient matrix as*

$$C(P)\boldsymbol{u}_{(\leq 4)} = \begin{pmatrix} -2 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = 0.$$

This implies that in terms of the solution x:

$$C(P)\boldsymbol{x}^{(\leq 4)}(\boldsymbol{x}^{(\leq 4)})^T = \begin{pmatrix} -2 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ x \\ x^2 \\ x^3 \\ x^4 \end{pmatrix} \begin{pmatrix} 1 \\ x \\ x^2 \\ x^3 \\ x^4 \end{pmatrix}^T = 0. \quad (2.9)$$

In the SDP-moment matrix approach we impose $u_0 = 1$. We note that the association $u_\alpha \leftrightarrow x^\alpha$ extends to the formal correspondence $x^\alpha x^\beta \leftrightarrow u_{\alpha+\beta}$. This allows for the construction of the truncated moment matrix to degree $d = 4$ of the polynomial system as:

$$M(u) = \begin{pmatrix} 1 & u_1 & u_2 & u_3 & u_4 \\ u_1 & u_2 & u_3 & u_4 & u_5 \\ u_2 & u_3 & u_4 & u_5 & u_6 \\ u_3 & u_4 & u_5 & u_6 & u_7 \\ u_4 & u_5 & u_6 & u_7 & u_8 \end{pmatrix}. \quad (2.10)$$

Appending the linear constraints, we get

$$C(P)M(u) = 0\,. \quad (2.11)$$

The linear constraints (2.11) are:

$$\{u_4 - 2 = 0, u_5 - 2u_1 = 0, u_6 - 2u_2 = 0, u_7 - 2u_3 = 0, u_8 - 2u_4 = 0\} \quad (2.12)$$

which via the correspondence $u_\alpha \longleftrightarrow x^\alpha$ is equivalent to $\{x^4 - 2, x^5 - 2x, x^6 - 2x^2, x^7 - 2x^3, x^8 - 2x^4\}$. The equivalent SDP problem here is to find a maximal rank generic point $u = (u_\alpha)$ where $|\alpha| \leq 2d$ in the moment matrix with

$$M(u) \succeq 0, \qquad C(P)M(u) = 0\,. \quad (2.13)$$

By imposing these simple linear constraints we get an explicit simplified moment matrix problem in only three variables:

$$M(u) = \begin{bmatrix} 1 & u_1 & u_2 & u_3 & 2 \\ u_1 & u_2 & u_3 & 2 & 2u_1 \\ u_2 & u_3 & 2 & 2u_1 & 2u_2 \\ u_3 & 2 & 2u_1 & 2u_2 & 2u_3 \\ 2 & 2u_1 & 2u_2 & 2u_3 & 4 \end{bmatrix} \succeq 0. \tag{2.14}$$

We note that the substitution of the linear constraints to simplify the problem and reduce the number of variables is equivalent to facial reduction; see Section 6 below. This moment matrix problem in (2.14) *is then sent to an SDP solver to approximately find a vector* (u_1, u_2, u_3) *if possible such that* M *is a positive semi-definite matrix with maximum rank. This solver returns an approximation which can be recognized for illustrative convenience as* $(u_1, u_2, u_3) = (0, \sqrt{2}, 0), u_0 = 1, u_4 = 2$. *Its associated moment matrix and moment matrix kernel are:*

$$M = \begin{bmatrix} 1 & 0 & \sqrt{2} & 0 & 2 \\ 0 & \sqrt{2} & 0 & 2 & 0 \\ \sqrt{2} & 0 & 2 & 0 & 2\sqrt{2} \\ 0 & 2 & 0 & 2\sqrt{2} & 0 \\ 2 & 0 & 2\sqrt{2} & 0 & 4 \end{bmatrix},$$

$$\ker M = \operatorname{span}_{\mathbb{R}} \left\{ \begin{pmatrix} -2 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} -\sqrt{2} \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ -\sqrt{2} \\ 0 \\ 1 \\ 0 \end{pmatrix} \right\}.$$

The kernel yields the generating set of three polynomials

$$\begin{aligned} \mathcal{S} &= \{-2 + x^4, -\sqrt{2} + x^2, -\sqrt{2}x + x^3\} \\ &= \{(\sqrt{2} + x^2)(-\sqrt{2} + x^2), -\sqrt{2} + x^2, x(-\sqrt{2} + x^2)\}. \end{aligned} \tag{2.15}$$

The factorization in (2.15) *allows a trivial Application of the geometric involutive form algorithm that yields a geometric involutive basis*

$$\{-\sqrt{2} + x^2\}. \tag{2.16}$$

The first and third polynomials in (2.15) *are a consequence of* $-\sqrt{2} + x^2$ *by our inclusion test, so are discarded, e.g., [26]. Thus we have a basis of the* `RRI` *in* (2.16). *There are efficient eigenvalue methods that can exploit this*

geometric form to efficiently numerically compute the roots as eigenvalues [33, 35, 37, 40]. For such solving methods tailored to the real radical and its advantages see [25]. The degree 8 *system trivially has two real roots given by the polynomial in* (2.16)*, i.e.,* $\pm 2^{1/4}$.

2.3 A class of univariate geometric polynomials

In this section we experimentally explore the behavior of our facial reduction approach (Facial Douglas-Rachford, or abbreviated as FDR) compared to a standard SDP solver (Yalmip SDP, abbreviated as YSDP) which does not use facial reduction. In particular we consider the class of univariate geometric polynomials which are the partial sums to odd degree d of the geometric series:

$$p_d(x) = 1 + x + x^2 + \cdots + x^{d-1} + x^d$$

where $d = 1, 3, 5, \ldots$. Then for odd degree d we have

$$p_d(x) = (x+1)(1 + x^2 + \cdots + x^{d-3} + x^{d-1})$$

where the even degree factor $1 + x^2 + \cdots + x^{d-3} + x^{d-1}$ has only complex roots. The d roots are $x = \exp\left(\frac{2j\pi i}{d+1}\right)$, $j = 1, \cdots, d$, and the non-real roots appear in complex conjugate pairs. Consequently a generator for the `RRI` is $x + 1$.[1]

We solved this class of problems for odd degrees d using both the FDR[2] method with MATLAB R2013b and the YSDP (Yalmip SDP, R20140605) method. We used a laptop (Windows 8.1, Intel Core(TM) i7-4600U CPU @2.10GHz 2.70 GHz, 8GB RAM, 64-bit OS, x64-based processor).

The running times (in cpu secs) for both methods are given in Figure 2.3; the range of values for the FDR method is clearly better.

3 Geometric involutive bases

In this section we introduce the basic objects for geometric involutive bases. For details and examples see [8, 36].

Involutivity originates in the geometry of differential equations. See Kuranishi [24] for a famous proof of termination of Cartan's prolongation algorithm for nonlinear partial differential equations. A by-product of these

[1] We denote the generator of the `RRI` by $\sqrt[\mathbb{R}]{\langle p_d(x)\rangle_{\mathbb{R}}} = \langle x+1\rangle_{\mathbb{R}}$.

[2] The Facial reduction Douglas Rachford method is presented in Section 5.2.2 below.

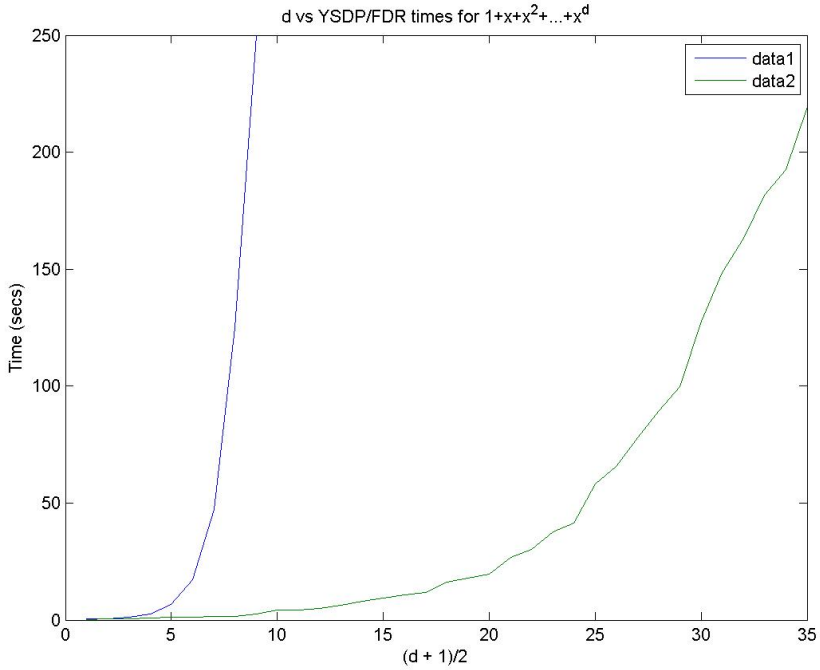

Figure 2.1: Times (cpu secs) for the FDR method versus the YSDP methods applied to $p_d(x) = 1 + x + \ldots + x^d$ for odd degrees $1 \leq d \leq 69$. The blue curve (data1 on the left) shows YSDP times and the green curve (data2 on the right) shows the significantly better FDR times.

methods has been their implementation for linear homogeneous partial differential equations with constant coefficients, and consequently for polynomial algebraic systems. See [21] for applications and symbolic algorithms for polynomial systems. The symbolic-numeric version of a geometric involutive form was first described and implemented in Wittkopf and Reid [41]. It was applied to approximate symmetries of differential equations in [8] and to polynomial solving in [35,37,38]. See [45] where it is applied to the deflation of multiplicities in multivariate polynomial solving.

Definition 3.1. *Let P be (as usual) a finite subset of $\mathbb{R}[x]$ of degree d. The k-th prolongation of system P is $\widehat{\boldsymbol{D}}^k(P) = \{x^\alpha p : 0 \leq \deg(x^\alpha p) \leq d+k, \alpha \in \mathbb{N}^n, p \in P\}$.*

For example $\widehat{\mathbf{D}}^k(P)$ for $P = \{x^2-x-1, xy-y-1\}$ consists of P together

with the 4 polynomials in (1.3).

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

See for example [38] and the published references in [36] for the stable numerical implementations of this paper's operations using SVD methods. In Remark 3.5 of [36] 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 to 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

$$\sum_{k=1}^{k=n} k\beta_d^{(k)} = \text{rank } \mathcal{S}(\widehat{\mathbf{D}}P) \tag{3.1}$$

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 (3.1) 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 $\boldsymbol{\pi}^{d'-d}Q$. 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 (3.1).

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

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

In [8] we prove that a system is projectively involutive if and only if it is involutive. In the following algorithm we seek the smallest k such that there exists an ℓ with $\boldsymbol{\pi}^\ell \mathbf{D}^k P$ approximately involutive, and generates the same ideal as the input system. We choose the system corresponding to the largest such $\ell \leq k$ if there are several such values for the given k.

Algorithm 3.1: `GIF`: Geometric involutive form

Input($Q \subset \mathbb{R}[x_1, \ldots, x_n]$*; tolerance* ϵ.);
Set $k := 0$, $d := \deg(Q)$ and $P := \ker C(Q)$;
while $I \neq \emptyset$ **do**
    Compute $\mathbf{D}^k(P)$; initialize set of involutive systems $I := \{\}$;
    for ℓ *from* 0 *to* $(d + k)$ **do**
        Compute $R := \boldsymbol{\pi}^\ell \mathbf{D}^k(P)$;
        if R *involutive* **then**
            $I := I \cup \{R\}$
        end if
    end for
    Remove systems $\bar{R}$ from I: $\mathbf{D}^{d+k-\bar{d}} \bar{R} \not\subseteq \mathbf{D}^k(P)$;
    $k := k + 1$
end while
Output(*Return the polynomial generators of the* `GIF` $(\bar{R})$ *in I of lowest degree* $\bar{d} = \deg \bar{R}$.)

The degree of the geometric involutive basis in our method can be lower than that given in [27, 28] since Algorithm 3.1 updates the generators with projections. However in the absence of a proof of determination of the real radical the larger moment matrices of [28] 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 [36].

4 Moment matrices & algorithms

In this section we outline algorithms for combining geometric involutive form and moment matrix methods; see Definition 2.2. Recall that $M = M(u) = (M_{\alpha,\beta})$ denotes the moment matrix indexed by α, β for rows and columns, respectively. And, $d = \deg(P)$, $M \in \mathbb{R}^{N(n,d)\times N(n,d)}$, and the linear constraints imposed by our system of polynomials $P \subset \mathbb{R}[x]$ are given by the coefficient times moment matrix multiplication $C(P)M = 0$. 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 *real radical ideal generated by polynomials P over $\mathbb{R}$*. A fundamental result [5] 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: `GIF` – `M` Method

Input($P = \{p_1, ..., p_k\} \subset \mathbb{R}[x_1, \ldots, x_n]$);
Set $Q_0 := P, \quad j := 0$;
while $r = d$ **do**
  $d := \dim\ker \texttt{GIF}(Q_j), \quad Q_{j+1} := \texttt{gen}(\texttt{GIF}(Q_j))$;
  Find $u^* = u(Q_{j+1}) \in \mathbb{R}^{N(n,2d)}$: $M(u^*) \succeq 0, C(Q_{j+1})M(u^*) = 0$;
  $r := \operatorname{rank}(\texttt{M}(u^*)), \quad Q_{j+2} := \texttt{gen}(\ker \texttt{M}(u^*))$;
  $j := j + 2$
end while
Output($Q_{j+1} \subset \mathbb{R}[x_1, \ldots, x_n]$; Q_{j+1} is in geometric involutive form ; $\sqrt[\mathbb{R}]{\langle P\rangle_{\mathbb{R}}} \supseteq \langle Q_{j+1}\rangle_{\mathbb{R}} \supseteq \langle P\rangle_{\mathbb{R}}$.)

Algorithm 4.1 uses the following subroutines described as Algorithms 4.2 and 4.3.

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:*

$$\texttt{gen}(\texttt{GIF}(Q)) = \texttt{gen}(\ker \texttt{M}(u^*)) \text{ and } Q \text{ involutive where } u^* = u(Q) \quad (4.1)$$

By [25] $\langle \texttt{gen}(\ker \texttt{M}(Q_{j+1}))\rangle$ 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,

Algorithm 4.2: `M` - Moment Matrix

1 Input($Q \subset \mathbb{R}[x_1, \ldots, x_n]$. Set $d := \deg(Q)$.);
2 Construct the moment matrix to degree $2d$.;
3 Use SDP methods to numerically solve for a generic point $u^* = u(Q)$ that maximizes the rank of the moment matrix subject to the constraints $C(Q)\, M(u^*) = 0$.;
4 Output(Return $\mathtt{M}(u^*) \succeq 0$ the moment matrix evaluated at this generic point.)

Algorithm 4.3: `gen`

1 Input(*GIF*(Q) *or* $\ker \mathtt{M}(u^*)$ *where* $u^* = u(Q)$.);
2 Output(Polynomial generators corresponding to $\mathtt{GIF}(Q)$ or $\ker \mathtt{M}(u^*)$)

together with the finiteness of the Cartan-Kuranishi geometric involutive form algorithm, Algorithm 4.1 terminates.

5 Mathematical background for the projection methods

In this section we describe the background for the projection methods for finding feasible solutions for the moment problems. An important part of these methods is building an efficient matrix representation for the linear constraints on the moment matrices resulting from the polynomial systems.

5.1 Linear constraints for multivariate polynomial moment matrices

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 $\langle \alpha_{(\leq d)} \rangle$ be an array of the subscripts α of $\langle u_\alpha \rangle$ with $0 \leq |\alpha| \leq d$ and sorted as in (2.3).

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

$\langle\cdot\rangle$ yields the addition of the subscripts for the f_j:

$$M(u) = \begin{bmatrix} \langle f_0(u), f_0(u)\rangle & \langle f_0(u), f_1(u)\rangle & \langle f_0(u), f_2(u)\rangle & \dots & \langle f_0(u), f_l(u)\rangle \\ \langle f_1(u), f_0(u)\rangle & \langle f_1(u), f_1(u)\rangle & \langle f_1(u), f_2(u)\rangle & \dots & \langle f_1(u), f_l(u)\rangle \\ \langle f_2(u), f_0(u)\rangle & \langle f_2(u), f_1(u)\rangle & \langle f_2(u), f_2(u)\rangle & \dots & \langle f_2(u), f_l(u)\rangle \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \langle f_l(u), f_0(u)\rangle & \langle f_l(u), f_1(u)\rangle & \langle f_l(u), f_2(u)\rangle & \dots & \langle f_l(u), f_l(u)\rangle \end{bmatrix}.$$

Here, $\langle f_0, f_1, ..., f_l\rangle$ corresponds to the array $\langle u_\alpha\rangle$ with $0 \le |\alpha| \le d$ sorted as in (2.3). We denote the *i-th* element in $\langle u_\alpha\rangle$ by u_α^i. Then $f_i(u)$ is u_α^i.

In the univariate case the moment matrices have Hankel structure as shown in (2.10). In Table 5.1 we display a truncated bivariate moment matrix partitioned into block submatrices having the same degree. Notice that

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

Table 5.1: A truncated bivariate moment matrix partitioned into block submatrices having the same degree.

the 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 $u_{00} = 1$. As an abbreviation, we may denote $M = M(u) = M_d(u)$.

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 (2.10) 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 $M(u)_{14} = M(u)_{22} = u_{20}$ are equal.

We now outline a simple algorithm to find a non-redundant matrix representation of these constraints. 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 examine entries above the main diagonal since the matrix is symmetric.

For (2.10) the first linear constraint traversing from the first row downwards is $M(u)_{14} = M(u)_{22}$. We denote e_i as the *i-th unit vector* and $E_{ij} = \frac{1}{2}(e_i^T e_j + e_j^T e_i)$. To impose this constraint, we construct matrix $A_t = E_{22} - E_{14}$, where t represents the index of the linear constraints and $t = 2$ in this case. The constraint is then given by

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

Since we always assume $M(u)_{1,1} = 1$, we need to set $A_1 = E_{11}$. Here A_t is called the *matrix representative* of the *t-th* linear constraint. The collection of all such matrix representatives for a given moment matrix is called the *matrix representation* of the moment matrix structure.

Algorithm 5.1 below determines all the (non-redundant) matrix representatives of the linear constraints defining the matrix representation of the multivariate moment matrix structure.

Algorithm 5.1: Matrix representation of moment matrix structure

Input(d, n);
Initialize array $T = \langle \alpha_{(\leq d)} \rangle$ and $T(i)$ is the *i-th* element of T.
Initialize n array $S = \langle s \rangle$ with the same length as $\langle \alpha_{(\leq d)} \rangle$ and $S(i) = [(1, i); \alpha_{(\leq d)}(i)]$ where $S(i)$, $\alpha_{(\leq d)}(i)$ is the *i-th* element of S, $\langle \alpha_{(\leq d)} \rangle$.
Let m be the length of T, $t = 2$ and $A_1 = E_{11}$.
for *i from* 2 *to* m, **do**
    for *j from i to* m, **do**
        if *there exists an* $s = [(g, h); \alpha] \in S$ *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 matrix representatives $\{A_t\}$ where $t \in \mathcal{E}$, $\mathcal{E} = \{1, 2, \ldots, \eta\}$ and η is the total number of the linear constraints.);

There are no redundant relations produced by this algorithm so we can avoid an overdetermined system.

In what follows for applications to multivariate polynomial systems of degree d in n variables we have

$$k := N(n,d) = \begin{pmatrix} d+n \\ d \end{pmatrix} \tag{5.1}$$

Our main problem is the following.

Problem 5.1 (Main Problem)**.** *Let B be a given $(k+1)\times m$ matrix of full column rank. Find $u \in \mathbb{R}^{2k+1}$ so that*

$$B^T M(u) = 0, \quad \operatorname{trace} E_{11} M(u) = 1, \qquad M(u) \succeq 0.$$

We denote $\mathcal{H}^{k+1}$, *space of generalized Hankel matrices*. That is these matrices have the multivariate structure whose matrix representation is computed by Algorithm 5.1. It is well known that the special case of Hankel matrices are notoriously ill-conditioned. This means that the cone $\mathcal{S}_+^{k+1} \cap \mathcal{H}^{k+1}$ is *thin*, i.e., it is close to the boundary of $\mathcal{S}_+^{k+1}$, e.g., [4, 6, 20]. Therefore, solving Problem 5.1 using semi-definite programming techniques results in numerical difficulties.

5.2 Methods of alternating projection and Douglas-Rachford projection-reflection

To apply the methods of *alternating projection, MAP* or *Douglas-Rachford reflection-projection*, we want to express the main Problem 5.1 as an equivalent problem with moment matrix $M = M(u)$:

$$\mathcal{A}(M) = b, \quad B^T M = 0, \quad M \in \mathcal{S}_+^{k+1}. \tag{5.2}$$

Here the linear transformation $\mathcal{A}$ is obtained from Algorithm 5.1. The following Corollary 5.1 provides the details of the system that we want to solve. We first apply facial reduction and get a smaller system. Recall from Algorithm 5.1, we get an array of representing matrix A_ts where $t \in \mathcal{E}$, $\mathcal{E} = \{1, 2, \ldots, \eta\}$.

Corollary 5.1. *Let V be $(k+1)\times(k+1-m)$ and satisfy $V^TV = I, V^TB = 0$. Let $\bar{A}_t \leftarrow V^T A_t V, \forall t \in \mathcal{E}$. Let $\bar{\mathcal{A}} : \mathcal{S}^{k+1-m} \to \mathbb{R}^{\mathcal{E}}$ be defined by*

$$\bar{\mathcal{A}}(\bar{M}) := \left(\left(\operatorname{trace} \bar{A}_t \bar{M}\right)_t\right)_{\forall t \in \mathcal{E}} \tag{5.3}$$

Then the main Problem 5.1 with $V\bar{M}V^T = M(u)$ *is equivalent to* (5.2)*, i.e., to*

$$\bar{\mathcal{A}}(\bar{M}) = e_1, \qquad \bar{M} \in \mathcal{S}^{k+1-m},$$

and we get $M(u) = V\bar{M}V^T$. □

Let L denote the matrix representation for $\bar{\mathcal{A}}$ in the linear constraints in Corollary 5.1. 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*. 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 update p_+ are then

$$r_c = b - Lp_c; \qquad p_+ = p_c + L^\dagger r_c. \tag{5.4}$$

Second, we project the updated symmetric matrix $P_+ = \mathcal{P}_{\mathcal{L}}(P_c) = \text{sHMat}(p_+)$ onto the semi-definite cone using the Eckart-Young Theorem [18], i.e., we diagonalize and zero out the negative eigenvalues. Here $\text{sHMat} = \text{sHvec}^* = \text{sHvec}^{-1}$ is both the adjoint and the inverse mapping. We denote $\mathcal{P}_{\mathcal{S}_+^k}$*, the positive semi-definite projection* and get the new positive semi-definite approximation $\mathcal{P}_{\mathcal{S}_+^k}(P_+)$.

5.2.1 Method of alternating projections

The MAP method is particularly simple, see e.g., the recent book [19]. We begin with an initial estimate, e.g., $P_c = \alpha I \in \mathcal{M}^{mk}$ for a large $\alpha > 0$. We then repeat the projection steps in Items 1, 2, 3 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 semi-definite projection using the Eckart-Young Theorem and update the current approximation

$$P_{SDP} = \mathcal{P}_{\mathcal{S}_+^k}(P_L).$$

3. Update the cosine value in (5.5). Then update $P_c = P_{SDP}$.

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

$$\cos(\theta) = \left(\frac{\text{trace}\left((P_L - P_c)^*(P_{SDP} - P_L)\right)}{\|P_L - P_c\| \, \|P_{SDP} - P_L)\|} \right). \tag{5.5}$$

5.2.2 Douglas-Rachford reflection method

Recall the projections defined above $\mathcal{P}_{\mathcal{L}}, \mathcal{P}_{\mathcal{S}_+^k}$. We want to find, see (5.2),

$$P \in \mathcal{G} \cap \mathcal{S}_+^{k+1}, \quad \text{where } \mathcal{G} := \left\{ P \in \mathcal{S}_+^{k+1} : \mathcal{A}(P) = b \right\}.$$

We now apply the Douglas-Rachford (DR) projection/reflection method [15]. (See also e.g., [3, 9].)

Using the QR algorithm applied to B and A, we start with an initial estimate

$$P_0 \succeq 0 \text{ with } B'P_0 = 0 \text{ and } (1,1) \text{ component } = 1. \tag{5.6}$$

Define the *reflections* $\mathcal{R}_{\mathcal{L}}, \mathcal{R}_{PSD} : \mathcal{S}_+^{k+1} \to \mathcal{S}_+^{k+1}$ using the corresponding projections, i.e.,

$$\mathcal{R}_{\mathcal{L}}(P) := 2\mathcal{P}_{\mathcal{L}}(P) - P, \quad \mathcal{R}_{PSD}(P) := 2\mathcal{P}_{PSD}(P) - P, \quad \forall\, P \in \mathbb{H}^{mk}.$$

- **Initialization:** We set our current estimate $P_c = P_0$ to satisfy (5.6). We calculate the residual $Res_{\mathcal{L}} = R - A * \text{s2Mat}(P_c)$, 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.

- We use Resrefl to project the current reflected PSD point $\mathcal{R}_{PSD}$ onto the linear manifold to get the projected point $P_{\mathcal{L}} = \mathcal{R}_{PSD} + A^{\dagger} Resrefl$. Then we reflect to get our second reflection point $\mathcal{R}_{\mathcal{L}} = 2 * P_{\mathcal{L}} - \mathcal{R}_{PSD}$

- At this time we set our new/current estimate for convergence to be $P_c = P_{new} = (P_c + \mathcal{R}_{\mathcal{L}})/2$.

- We now project P_c to get P_{PSD}. We check the residual here for the stopping criteria $normres = \|Res_{\mathcal{L}}\| = \|R - \mathcal{A}P_{PSD}\|$.

- We now calculate the first reflection point $\mathcal{R}_{PSD} = 2 * P_{PSD} - P_c$ and update the reflected residual $Resrefl = R - A\,\text{s2vec}(\mathcal{R}_{PSD})$.

The Douglas-Rachford projection/reflection method is simply:

1. Start at an initial point $P_0 \in \mathcal{S}_+^{k+1}$ satisfying (5.6)

2. Iterate: $P_{j+1} = \frac{1}{2}(P_j + \mathcal{R}_{PSD}(\mathcal{R}_{\mathcal{L}}(P_j))$, for all $j = 0, 1, \ldots$.

Also the basic theorem on the convergence of the sequence $\Pi_G(X_k)_k$, [9, Thm 3.3, Page 11], carma.newcastle.edu.au/jon/cycDRinfeas.pdf. so 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 finding the residual with respect to the linear manifold is inexpensive.

To check the linear convergence rates we use the cosine of the angles for the vectors of successive iterates, i.e., for three successive iterates $P_c, \mathcal{R}_{PSD}, \mathcal{R}_{\mathcal{L}}$, and

$$\cos(\theta) = \left| \frac{\text{trace}\left((\mathcal{R}_{PSD} - \mathcal{R}_{\mathcal{L}})^*(\mathcal{R}_{PSD} - P_c)\right)}{\|(\mathcal{R}_{PSD} - \mathcal{R}_{\mathcal{L}})\| \, \|\mathcal{R}_{PSD} - P_c\|} \right|.$$

6 Facial reduction implementation

Our moment problem is a feasibility problem of the form

$$B^T M(u) = 0, \quad M(u) \succeq 0, \tag{6.1}$$

where B is a given matrix and $M(u)$ is a linear function of the variables u. Constraints on $M(u)$ are described in Section 5.2, where the problem is changed to equality form and then uses facial reduction to get the form

$$\bar{\mathcal{A}}(P) = \bar{b}, \qquad P \succeq 0. \tag{6.2}$$

This form includes the first step of facial reduction using the matrix B, see Corollary 5.1 and (5.3). Here $\bar{\mathcal{A}}(P) = (\text{trace}\, \bar{A}_i P) \in \mathbb{R}^m$, for specific symmetric matrices $\bar{A}_i$.

The projection methods behave poorly when Slater condition fails. We therefore attempt to apply further steps of facial reduction and reduce system (6.2) 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].

$$\begin{array}{c} \exists \hat{P}, \bar{\mathcal{A}}(\hat{P}) = \bar{b}, \hat{P} \succ 0 \\ \iff \\ Z = \bar{\mathcal{A}}^* y \succeq 0, \bar{b}^T y = 0 \implies Z = 0. \end{array} \tag{6.3}$$

Note that if a $Z \neq 0$ can be found satisfying the left part of the bottom half of (6.3) and for the top half $\hat{P} \succeq 0, \bar{(}\hat{P}) = \bar{b}$, then

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

Therefore, if the full column rank matrix W satisfies $\operatorname{range} W = Z$, then we can facially reduce the problem using the substitution $\hat{P} = W\bar{P}W^T$, i.e., we can restrict the feasibility problem in (6.2) to the face $W \cdot W^T$.

We can implement the test in (6.3) in several ways. We suppose that $\bar{A}$ is the matrix representation of $\bar{\mathcal{A}}$, i.e., we let $p = \text{s2vec}(P)$ and then we have

$$\bar{A}p = (\bar{\mathcal{A}}\,\text{s2Mat})(\text{s2vec}(P)) = \bar{\mathcal{A}}P, \quad \bar{\mathcal{A}}^* y = \text{s2Mat}(\bar{A}^T y).$$

One way would be to first evaluate the orthogonal matrix $\begin{bmatrix} \frac{1}{\|b\|} b & U \end{bmatrix}$ and find v so that

$$\text{s2Mat}(\bar{A}^T(Uv)) \succeq 0, \quad \operatorname{trace} \bar{\mathcal{A}}^*(Uv) = (\bar{\mathcal{A}}(I)^T U)v = 1.$$

Alternatively, we solve [3]

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

7 Numerical experiments

7.1 Examples of Ma, Wang and Zhi [28]

Ma, Wang and Zhi [27,28] 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 [28, Examples 4.1-4.6]. with the results shown in Table 7.1. In each case we obtained a geometric involutive basis which can be independently verified as a geometric involutive basis for the real radical. In [28] Pommaret bases are successfully obtained for the real radical for these examples.

Here are the 6 systems of polynomials corresponding to the examples

[3] 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.

in [28]:

$$\{x_1^2 + x_1x_2 - x_1x_3 - x_1 - x_2 + x_3,\ \ x_1x_2 + x_2^2 - x_2x_3 - x_1 - x_2 + x_3,$$
$$x_1x_3 + x_2x_3 - x_3^2 - x_1 - x_2 + x_3\} \tag{7.1a}$$

$$\{x_1^2 - x_2,\ \ x_1x_2 - x_3\} \tag{7.1b}$$

$$\{x_1^2 + x_2^2 + x_3^2 - 2,\ \ x_1^2 + x_2^2 - x_3\} \tag{7.1c}$$

$$\{x_3^2 + x_2x_3 - x_1^2,\ \ x_1x_3 + x_1x_2 - x_3,\ \ x_2x_3 + x_2^2 + x_1^2 - x_1\} \tag{7.1d}$$

$$\{(x_1 - x_2)(x_1 + x_2)^2(x_1 + x_2^2 + x_2),\ \ (x_1 - x_2)(x_1 + x_2)^2(x_1^2 + x_2^2)\} \tag{7.1e}$$

$$\{(x_1 - x_2)(x_1 + x_2)(x_1 + x_2^2 + x_2),\ \ (x_1 - x_2)(x_1 + x_2)(x_1^2 + x_2^2)\} \tag{7.1f}$$

System (7.1a) **for [28, Example 4.1]**: Our `GIF` algorithm 3.1 with input tolerance 10^{-10} shows that the system is already in geometric involutive form. The corresponding Pommaret basis is given in [28, 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. Since it 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 using the reduced moment matrix, yields convergence in 13 iterations and 0.09 secs, with a projected residual error of 10^{-14}. These statistics are shown in Table 7.1. The reduction in moment matrix size from 10×10 to a 7×7 matrix is recorded in the rightmost column of the Table by the fraction $\frac{10}{7}$. Determination of this reduced moment matrix then yields the full 10×10 moment matrix of rank $r = 7$. Since the dimension of the kernel for `GIF` form is $d = 7 = r$ 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. Our facial reduction algorithms in Section 6 provide checks for the existence of additional facial reductions. They show that there are no additional facial reductions for this problem.

System (7.1d) **for [28, Example 4.4]**: This is very similar to the previous system (7.1a). As [28] 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 [28]. 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.

System (7.1b) **for [28, Example 4.2]**: This is quite similar to the systems (7.1b) and (7.1d). Our methods are similarly efficiently applied to this system. Our GIF algorithm first applied one prolongation to the second system (7.1b) to yield a degree 3 system. After projectiing 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 [28]. 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 $\boldsymbol{\pi}\mathbf{D}R$ with corresponding 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 next moment matrix, $d = r = 7$ and the algorithm terminates. It can be verified that the GIF form is a basis for the real radical ideal of the input system.

Unlike the systems (7.1a),(7.1b),(7.1d), the remaining three systems (7.1c),(7.1e),(7.1f) of [28] lead to new members in the kernel of their moment matrices.

System (7.1c) **for [28, 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 $\widehat{\mathbf{D}}R$ is involutive and has a 8×20 coefficient matrix. The dimension of its kernel is $d = 12$. Facial reduction then reduces the 20×20 moment matrix to a 12×12 moment matrix which has 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 [28], that only the root $x_3 = 1$ leads to real solutions. The GIF form of degree 2 of the new system is computed. Its coefficient matrix is 5×10 and has kernel of dimension $d = 5$. We note that even with the change of coordinates the FDR iteration of this second moment matrix did not initially converge until we reduced the required projected residual error for production of the first moment matrix to 10^{-14}. The second moment matrix then was computed quickly and accurately as a 10×10 matrix which is reduced by one facial reduction to a 5×5 matrix. Since the rank of the moment matrix is $r = 5 = d$ our algorithm has terminated. It can be

checked that the output is equivalent to that found by [28] and that the resulting `GIF` form is a basis for the real radical.

System (7.1e) **for [28, Example 4.5]**: Direct application of Algorithm 4.1 to (7.1e) 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 (7.1e). 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 moment matrix which facially reduced to a 14×14 moment matrix. The rank of the moment matrix is $r = 14 = d$. The algorithm terminates to coefficient errors within 10^{-10} with output as a single polynomial which is approximately:

$$(x_1 - x_2)(x_1 + x_2)(x_1 + x_2^2 + x_2) \tag{7.2}$$

It can be checked that (7.2) is a geometric involutive basis for the real radical for the ideal generated by P_1.

Similarly we apply Algorithm 4.1 to the first polynomial of (7.1e) 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:

$$(x_1 - x_2)(x_1 + x_2) \tag{7.3}$$

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

Then we consider the system

$$(x_1 - x_2)(x_1 + x_2)(x_1 + x_2^2 + x_2),\ \ (x_1 - x_2)(x_1 + x_2) \tag{7.4}$$

The calculation for (7.1f)for Example 4.6 below yields a geometric involutive basis which is approximately

$$(x_1^2 - x_2^2) \tag{7.5}$$

Syst.	(n,d,p)	FDR # its	FDR secs	FDR proj res err	GIF-FDR its (# FR)	GIF tol	Mom Mtx redn factors $s(M)/s(\hat{M})$
Ex4.1	(3,2,3)	13	0.09	10^{-14}	1(1)	10^{-10}	$\frac{10}{7}$
Ex4.2	(3,2,2)	28	0.01	10^{-14}	1(1)	10^{-10}	$\frac{10}{7}$
Ex4.3	(3,2,2)	888, 238	2.3, 0.6	$10^{-14}, 10^{-13}$	2(2,1)	10^{-10}	$\frac{20}{12}, \frac{10}{5}$
Ex4.4	(3,2,3)	346	0.53	10^{-14}	1(1)	10^{-10}	$\frac{10}{7}$
Ex4.5 P_1	(2,5,1)	22314, 50	37.6, 0.3	$10^{-12}, 10^{-14}$	2 (2, 1)	10^{-10}	$\frac{21}{20}, \frac{15}{14}$
Ex4.5 P_2	(2,5,1)	957, 1	4.4, 0.1	$10^{-12}, 10^{-14}$	2 (2, 1)	10^{-10}	$\frac{21}{20}, \frac{6}{5}$
Ex4.6 Q_1	(2,4,1)	170, 1	1.0, 0.09	$10^{-12}, 10^{-14}$	2(2,1)	10^{-10}	$\frac{21}{15}, \frac{6}{5}$
Ex4.6 Q_2	(1,4,1)	484, 1	1.4, 0.08	$10^{-12}, 10^{-14}$	2(2,1)	10^{-10}	$\frac{15}{14}, \frac{6}{5}$
Cyl2d	(2,2,1)	10	0.19	10^{-15}	1(1)	10^{-10}	$\frac{6}{5}$
Cyl3d	(3,2,2)	33	0.77	10^{-14}	1(1)	10^{-10}	$\frac{20}{12}$
Cyl4d	(4,2,3)	142	8.45	10^{-14}	1(1)	10^{-10}	$\frac{70}{28}$

Table 7.1: **Statistics for the application of GIF and FDR to polynomial systems**: n = number of variables, d = maximum polynomial degree, p = the number of polynomials; $s(M)$, $s(\hat{M})$ sizes of moment matrix M and the faciallly reduced matrix $\hat{M}$, resp. Ex 4.1-4.6 are the 6 examples in MWZ [28]; Cyl2d-Cyl4d are the intersecting cylinder examples.

It can be independently checked that this is a `GIF` form for the real radical of the ideal of (7.1e).

System (7.1f) **for [28, Example 4.6]**: This concerns the real solution of $Q_1 = (7.1f) = (7.1f)$ 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 indepdently verified to be a geometric basis for the real radical of Q_1. The statistics of this reduction are given in the table in the row labeled as Ex 4.6 Q_1.

To impose $x_1 \geq 1$, $x_2 \geq 1$ we substitute $x_1 = x_3^2 + 1, x_2 = x_4^2 + 1$ and reduce the resulting polynomial Q_2 with Algorithm 4.1. We obtain $x_1 - x_2$ in agreement with [28, Example 4.6]. The statistics of this reduction are given in Table 7.1 in the row labeled as Ex 4.6 Q_2.

7.2 Intersecting higher dimensional cylinders

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

$$Cyl_{nd} := x_1^2 + x_2^2 - 1, x_1^2 + x_3^2 - 1, \cdots, x_1^2 + x_n^2 - 1. \tag{7.6}$$

Application of the `GIF` algorithm to the systems Cyl_{nd} for $n = 2, 3, 4$ show that the systems become geometrically involutive after $0, 2, 3$ prolongations

respectively. Table 7.1 shows the statistics for the subsequent application of Algorithm 4.1 to these systems. The algorithm converges quickly and accurately. Indeed it can be independently determined that the it yields an geometric involutive basis for the real radical.

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 [31]. The lower degree system, is 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.

Finally we mention that recently certain so-called critical point methods have been developed for determining witness points [22, 44] on real components of real polynomial systems. Indeed the method developed in [44] is successful in finding a point on every component, if the ideal is both real radical, and forms a regular sequence. Consequently the systems above, the real radical is an important property for such solvers. Such a regular sequence can be checked by dimension computation, we only need a formally integrable system which has lower degree than the involutive system, this leads to a smaller size of moment matrix. Other interesting related results are given in [29].

7.3 Example of Matlab routine FDR

Example 7.1. *We first use the matrix from* (7.7)

$$B_1{}^T = \begin{bmatrix} 2 & 0 & 0 & 0 & -1 \end{bmatrix}. \tag{7.7}$$

The moment matrix we get is the exactly the same as that in [36, Equation (37)]:

$$P = \begin{bmatrix} 1.0000 & -0.0000 & 1.4142 & -0.0000 & 2.0000 \\ -0.0000 & 1.4142 & -0.0000 & 2.0000 & -0.0000 \\ 1.4142 & -0.0000 & 2.0000 & -0.0000 & 2.8284 \\ -0.0000 & 2.0000 & -0.0000 & 2.8284 & -0.0000 \\ 2.0000 & -0.0000 & 2.8284 & -0.0000 & 4.0000 \end{bmatrix}$$

The nullity/kernel matrix of P is the same as in [36, Equation (37)] as well:

$$\begin{bmatrix} 2 & 0.026491 & -0.23757 \\ 0 & -0.81147 & -0.090484 \\ 0 & -0.09366 & 0.83995 \\ 0 & 0.57379 & 0.063982 \\ -1 & 0.052982 & -0.47515 \end{bmatrix}$$

though it is difficult to see from the last two columns.

To check whether the matrix B_1 in (7.7) *provides the same nullity as the nullity of the matrix P, one can look at the following short MATLAB code and see that it is so, i.e., the rank is correct and the spans do not change.*

```
B1=[ B'
  sqrt2 0 -1 0 0
0 sqrt2 0 -1 0]
B1 =
    2.0000         0         0         0   -1.0000
    1.4142         0   -1.0000         0         0
         0    1.4142         0   -1.0000         0

>> B1=B1'
B1 =

    2.0000    1.4142         0
         0         0    1.4142
         0   -1.0000         0
         0         0   -1.0000
   -1.0000         0         0
>> K=[null(P) B1]
K =
    0.8099    0.4053    0.1922    2.0000    1.4142         0
   -0.2574    0.1542    0.7593         0         0    1.4142
   -0.4913    0.6222   -0.2930         0   -1.0000         0
    0.1820   -0.1091   -0.5369         0         0   -1.0000
   -0.0575   -0.6426    0.1110   -1.0000         0         0
>> svd(K)
K>> svd(K)
    2.8284
    2.0000
    1.4142
```

```
    0.0000
    0.0000
```

Following is the output during the MATLAB program. Note the quick and accurate convergence; though we have to remember this is a tiny problem. It took 118 iterations to get 15 *decimals accuracy. The moment matrix* P *has the correct rank.*

```
Starting with new B value
using [no*VV'] as initial starting point for P
time for matrix repres.  0.0468003
Starting while loop for Douglas-Rachford algorithm
iter       cos-vecs     norm-proj.-resid.   PSD-proj-per.iter.time
 10           0.9938         0.04919         6.23e-05
 20                1        0.005256        6.377e-05
 30                1       0.0004443        6.188e-05
 40                1       3.282e-05         6.23e-05
 50                1       2.167e-06         0.000109
 60                1       1.271e-07        6.467e-05
 70                1        6.36e-09        6.551e-05
 80                1       2.341e-10        6.251e-05
 90                1       3.539e-12        6.349e-05
 100               1        1.037e-12        6.439e-05
 110               1        1.324e-13        6.572e-05
 118               1        7.531e-15        6.404e-05
time for iterations/while loop is  0.0780005
max cosine value is  1
checking feas error in DRalg.m using ***projected*** last iterate Rpsd
error for norm(B'*P)  is  0
```

8 Conclusion

SDP feasibility problems typically involve the intersection of the convex cone of semi-definite 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 geometric univariate polynomials of odd degree, where one such facial reduction combined with DR iteration, yields the real radical much more efficiently than the standard interior point method Yalmip. The high accuracy required by facial reduction and also the ill-conditioning commonly encountered in numerical polynomial algebra [40] motivated us to implement Douglas-Rachford iteration.

A fundamental open problem is to generalize the work of [25, 39] to positive dimensional ideals. The algorithm of [27, 28] 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:

$$\sqrt[\mathbb{R}]{\langle P \rangle_{\mathbb{R}}} \;\supseteq\; \langle Q \rangle_{\mathbb{R}} \;\supseteq\; \langle P \rangle_{\mathbb{R}} \tag{8.1}$$

and would provided 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 [27, 28] establishes an important feature – involutivity – that will necessarily be a 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 [28]). By using the CVX package or Douglas-Rachford iteration to solve for the auxiliary problem, we can determine that if we need a second facial reduction by checking whether the optimal value of the auxiliary problem is close to 0. So far only moderate improvements in convergence have been obtained by our preliminary implementation for construction of additional

facial reductions.

Numerical polynomial algebra has been a rapidly expanding and popular area [40]. It's 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.

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 whch 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.

Our implementation of auxiliary facial reductions, as still preliminary and needs improvement. Even if the real radical is theoretically accessible, the conditioning of the polynomial system, as measured by the sensitivity of changes in the solutions to changes in the coefficients, is a significant computational affect. So a more detailed study of this aspect is worthwhile.

Index

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. 6, 8

[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. 21

[4] S.G. Bartels and D.J. Higham. The structured sensitivity of Vandermonde-like systems. *Numer. Math.*, 62(1):17–33, 1992. 19

[5] 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. 4, 6, 15

[6] B. Beckermann. The condition number of real Vandermonde, Krylov and positive definite Hankel matrices. *Numer. Math.*, 85(4):553–577, 2000. 19

[7] G. Blekherman, P.A. Parrilo, and R.R. Thomas, editors. *Semidefinite Optimization and Convex Algebraic Geometry*. Number 13 in MOS-SIAM Series on Optimization. 2012. 3, 6

[8] 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. 11, 12, 14

[9] J.M. Borwein and M.K. Tam. A Cyclic Douglas–Rachford Iteration Scheme. *J. Optim. Theory Appl.*, 160(1):1–29, 2014. 21, 22

[10] 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

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

[12] Y-L. Cheung, D. Drusvyatskiy, N. Krislock, and H. Wolkowicz. Noisy sensor network localization: robust facial reduction and the Pareto frontier. Technical report, University of Waterloo, Waterloo, Ontario, 2014. in progress. 3

[13] 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. 3, 22

[14] David A. Cox, John B. Little, and Don O'Shea. *Ideals, Varieties, and Algorithms*. Springer-Verlag, NY, 2nd edition, 1996. 536 pages. 7

[15] 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. 21

[16] 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. 3

[17] 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

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

[19] R. Escalante and M. Raydan. *Alternating projection methods*, volume 8 of *Fundamentals of Algorithms*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2011. 20

[20] W. Gautschi and G. Inglese. Lower bounds for the condition number of Vandermonde matrices. *Numer. Math.*, 52(3):241–250, 1988. 19

[21] V.P. Gerdt and Y.A. Blinkov. Involutive bases of polynomial ideals. *Mathematics and Computers in Simulation*, 45(5):519–541, 1998. 12

[22] Jonathan D Hauenstein. Numerically computing real points on algebraic sets. *Acta applicandae mathematicae*, 125(1):105–119, 2013. 28

[23] 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

Kuranishi:1957 [24] M. Kuranishi. On e. cartan's prolongation theorem of exterior differential systems. *American Journal of Mathematics*, pages 1–47, 1957. 11

LasserreLaurentRostalski:2009 [25] 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, 11, 15, 31

LaurentRostalski:2012 [26] 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. 8, 10

Ma:2012 [27] Y. Ma. *Polynomial Optimization via Low-rank Matrix Completion and Semidefinite Programming*. PhD thesis, 2012. 2, 4, 14, 23, 31

MWZ:2012 [28] Y. Ma, C. Wang, and L. Zhi. A certificate for semidefinite relaxations in computing positive dimensional real varieties. Technical Report arXiv:1212.4924, KLMM, Academy of Mathematics and Systems Science, CAS, 2012. 2, 4, 14, 23, 24, 25, 26, 27, 31

MaZhi:2012 [29] 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. 28

Macaulay:1916 [30] F.S. Macaulay and P. Roberts. *The algebraic theory of modular systems*. Number 19. University press Cambridge, 1916. 7

MollerSauer:2000 [31] H.M. Möller and T. Sauer. H-bases for polynomial interpolation and system solving. *Advances in Computational Mathematics*, 12(4):335–362, 2000. 7, 28

Mourrain:1996 [32] B. Mourrain. Isolated points, duality and residues. *Journal of Pure and Applied Algebra*, 117:469–493, 1997. 7

Mourrain:1999 [33] B. Mourrain. A new criterion for normal form algorithms. In *Applied algebra, algebraic algorithms and error-correcting codes*, pages 430–442. Springer, 1999. 7, 11

ReidLinWittkopf:2001 [34] G.J. Reid, P. Lin, and A.D. Wittkopf. Differential elimination–completion algorithms for dae and pdae. *Studies in Applied Mathematics*, 106(1):1–45, 2001. 4

ReidTangZhi:2003 [35] 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. 11, 12

ReidWangWu:14 [36] 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. 4, 11, 13, 14, 28, 29

ReidZhi2009 [37] 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. 11, 12

SRWZ2010 [38] 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. 4, 12, 13

MR2830310 [39] F. Sottile. *Real solutions to equations from geometry*, volume 57 of *University Lecture Series*. American Mathematical Society, Providence, RI, 2011. 2, 3, 6, 31

Stetter:2004 [40] Hans J. Stetter. *Numerical polynomial algebra*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2004. 7, 11, 31, 32

ReidWittkopf:2001 [41] A.D. Wittkopf and G.J. Reid. Fast differential elimination in c: The cdiffelim environment. *Computer Physics Communications*, 139(2):192–217, 2001. 12

SaVaWo:97 [42] 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. 6

WoZh:96 [43] 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

WuReid:13 [44] 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. 28

WuZhi12 [45] 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. 12

KaReWoZh94 [46] 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