Matrix Polynomials and their Lower Rank Approximations

by

Joseph Haraldson

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2019

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Lihong Zhi Professor, Mathematics Mechanization Research Center Institute of Systems Science, Academy of Mathematics and System Sciences Academia Sinica
Supervisor(s):	Mark Giesbrecht Professor, School of Computer Science, University of Waterloo George Labahn Professor, School of Computer Science, University of Waterloo
Internal-External Member:	Stephen Vavasis Professor, Dept. of Combinatorics and Optimization, University of Waterloo
Other Member(s):	Eric Schost Associate Professor, School of Computer Science, University of Waterloo
Other Member(s):	Yuying Li Professor, School of Computer Science, University of Waterloo

Abstract

This thesis is a wide ranging work on computing a “lower-rank” approximation of a matrix polynomial using second-order non-linear optimization techniques. Two notions of rank are investigated. The first is the rank as the number of linearly independent rows or columns, which is the classical definition. The other notion considered is the lowest rank of a matrix polynomial when evaluated at a complex number, or the McCoy rank. Together, these two notions of rank allow one to compute a nearby matrix polynomial where the structure of both the left and right kernels is prescribed, along with the structure of both the infinite and finite eigenvalues. The computational theory of the calculus of matrix polynomial valued functions is developed and used in optimization algorithms based on second-order approximations. Special functions studied with a detailed error analysis are the determinant and adjoint of matrix polynomials.

The unstructured and structured variants of matrix polynomials are studied in a very general setting in the context of an equality constrained optimization problem. The most general instances of these optimization problems are NP hard to approximate solutions to in a global setting. In most instances we are able to prove that solutions to our optimization problems exist (possibly at infinity) and discuss techniques in conjunction with an implementation to compute local minimizers to the problem.

Most of the analysis of these problems is local and done through the Karush-Kuhn-Tucker optimality conditions for constrained optimization problems. We show that most formulations of the problems studied satisfy regularity conditions and admit Lagrange multipliers. Furthermore, we show that under some formulations that the second-order sufficient condition holds for instances of interest of the optimization problems in question. When Lagrange multipliers do not exist, we discuss why, and if it is reasonable to do so, how to regularize the problem. In several instances closed form expressions for the derivatives of matrix polynomial valued functions are derived to assist in analysis of the optimality conditions around a solution. From this analysis it is shown that variants of Newton’s method will have a local rate of convergence that is quadratic with a suitable initial guess for many problems.

The implementations are demonstrated on some examples from the literature and several examples are cross-validated with different optimization formulations of the same mathematical problem. We conclude with a special application of the theory developed in this thesis is computing a nearby pair of differential polynomials with a non-trivial greatest common divisor, a non-commutative symbolic-numeric computation problem. We formulate this problem as finding a nearby structured matrix polynomial that is rank deficient in the classical sense.

Acknowledgements

I would like to acknowledge the following for their investments in my research:

- The Natural Sciences and Engineering Research Council of Canada,
- The Government of the Province of Ontario,
- The National Science Foundation, United States of America,
- The National Security Agency, United States of America,
- David. R. Cheriton, and
- The University of Waterloo.

I would like to thank my supervisors Dr. Mark Giesbrecht and Dr. George Labahn and the other members of my examination committee, Dr. Lihong Zhi, Dr. Stephen Vavasis, Dr. Eric Schost and Dr. Yuying Li for their time and valuable feedback.

I would like to thank Dr. Guenter Krause from the University of Manitoba for providing me the opportunity to discover my interests in mathematics and for providing me with an opportunity to succeed. I would also like to thank Dr. Yang Zhang from the University of Manitoba for giving me the opportunity to work on research problems as an undergraduate student and an introduction to research in general. I would also like to thank Dr. Benqi Guo from the University of Manitoba for introducing me to numerical analysis, despite my reservations at the time where I insisted that I would "*never need to use this*".

I would like to thank Weixi and the rest of my family.

There are many other people who were supportive and inspirational along this journey who are not mentioned. I would like to mention all of them here and thank them for everything.

Dedication

To the advancement of scientific knowledge.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 A Non-Technical Overview

The general theme of this thesis will be discussing optimization problems in symbolic-numeric computation pertaining to structured matrices, matrix polynomials and their applications in symbolic-numeric computation. The goals are to apply a rigorous mathematical analysis to the problems discussed, then exploit this mathematical analysis to obtain robust numerical algorithms to approximate solutions using floating point arithmetic. The algorithms discussed in detail have a local rate of convergence that is at least quadratic under mild normalization assumptions. Other methods are discussed as well in the context of a hybrid algorithm (one that alternates between different optimization methods), but are not necessarily implemented. To understand the material presented in this thesis, the reader should be familiar with the calculus of matrix or vector valued functions and numerical linear algebra.

1.1.1 Structured Matrix Polynomials

A structured matrix polynomial is a matrix whose entries are polynomials, where those polynomials have some underlying pattern or prescribed coefficient structure. Structured scalar matrices are vacuously included in this definition. Our results are presented in the monomial basis for univariate polynomials, although they can be generalized to other bases with some effort.

An example of a matrix polynomial with some notion of structure is

$$\mathscr{A} = \begin{pmatrix} t+1 & t^3 & t^2+3 \\ t^2+3 & t+1 & t^3 \\ t^3 & t^2+3 & t+1 \end{pmatrix} \in \mathbb{R}[t]^{3\times 3},$$

in which each column is a cyclic shift of the others. What is often desirable is to find a nearby matrix $\widetilde{\mathscr{A}}$ with the same column shift and degree structure as $\mathscr{A}$ that is either rank deficient or has some prescribed spectral properties. These two problems are fundamental in determining the radius of stability in control theory or finding a nearby "interesting" system, as is commonly done in denoising applications.

The goal of my research is to pose several exact problems as a continuous optimization problem and implement efficient and robust numerical optimization algorithms to approximate solutions. In symbolic-numeric computation these tools can be applied on approximate greatest common divisor problems of univariate, multivariate, Ore and matrix polynomials, the approximate factorization of multivariate polynomials, finding a nearby lower rank matrix polynomial (approximate kernel vector or approximate kernel computation), computing a nearby matrix polynomial with an interesting Smith or Smith McMillan canonical form (approximate Smith normal form) and several related problems. All of these problems have exact arithmetic analogs that are well understood, however there is still room to improve our understanding in the context of floating point arithmetic, as these exact techniques often do not generalize to floating point computation.

Ultimately, the work presented in this thesis leads to a method to compute a nearby (structured) matrix polynomial whose companion linearization (or other suitable linearization) has a Kronecker canonical form that is prescribed to be non-trivial in some meaningful way. In other words, we seek to compute a matrix polynomial where the kernel, finite and infinite eigenvalues have some prescribed structure.

1.1.2 Optimization Problems

This work is primarily a study of some optimization problems in symbolic-numeric computation and not a study on optimization algorithms, although the theory of optimization is a key ingredient.

Broadly speaking, there are three different types of approximation problems of interest in symbolic-numeric computation and scientific computing. They are in decreasing order of specificity

1. “find a nearest”,
2. “find something within a radius $\varepsilon > 0$ or an indication that something does not exist”, and
3. “find something nearby”.

We assume “nearby” and “nearest” is taken to be with respect to a norm or distance metric that is “reasonable” in the context of the problem.

Of these three families of problems, we focus on the third formulation, which is sometimes known as a *soft approximation* problem. The first two instances are known as *hard approximation* problems, as they demand a certificate which is often difficult to produce.

The reason for the focus on the soft approximation problem, is that often soft approximations will solve the hard problem if the residual of the underlying problem is sufficiently small. In this work we will characterize several instances of problems when a solution to the soft approximation also yields a solution to one of the harder approximation problems. More importantly, the “hard approximations” are difficult in general to solve for non-convex problems in a reasonable amount of time, where as the “soft approximation” can often be computed in a polynomial amount of work. By “solve” we generally mean “approximate to some specified amount of precision” and not “compute an exact solution”. For some problems this distinction is irrelevant, such as solving a system of linear equations (which can be solved exactly or approximated to arbitrary precision in a polynomial amount of time), but for others it is essential, such as computing a solution to a semidefinite program (which can be quickly approximated to arbitrary precision but is difficult to solve exactly).

Several of the symbolic-numeric problems studied in this work are closely related to non-negative matrix factorization, structured lower rank approximation, and weighted lower rank approximation problems appearing in scientific computing, data science, computational statistics and machine learning. All of these problems are non-convex and NP-hard to approximate solutions to under reasonable models of computation. These problems generally seek a “soft approximation”, rather than a “hard approximation”. Popular existing techniques for these problems rely on descent based unconstrained optimization, first-order optimization techniques, convex relaxations (least squares, convex envelopes, semidefinite programming and several others), alternating directions, alternating optimization algorithms and other related techniques. Given the considerable similarity between these problems, we make use of several of these techniques when applicable in our problems to obtain an initial guess for a Newton method.

Most of the optimization problems encountered in this thesis are of the general form

$$\min_{x\in\Omega} \|f(x)\| \text{ subject to } h(x) = 0,$$

where $\|\cdot\|$, $f(x)$ and $h(x)$ are twice differentiable as a vector valued functions and Ω is a subset of Euclidean space. Accordingly, we make extensive use of the tools of non-linear optimization and design algorithms that exploit the specific structure of our problems at hand. All of the optimization problems studied can be considered as a constrained matrix polynomial problem. Some special problems such as approximate greatest common (right/left) divisor, approximate factorization and approximate kernel computation can be studied as an unconstrained problem (that is not related to penalty methods) of the form

$$\min_{x\in\Omega} \|f(x)\|$$

or an inequality constrained problem of the form

$$\min_{x\in\Omega} \|f(x)\| \text{ subject to } g(x) \geq 0.$$

However, the constraint that $g(x) \geq 0$ can be ignored in practice for our problems, as the inequality constraint $g(x) \geq 0$ translates into plain language as "find a solution that is at-least as good as a trivial to compute feasible point".

There is no such thing as a free lunch; the unconstrained variants often have irregular (asymptotic) solutions or other particularities that can hinder computation of (local) solutions. Approximate greatest common (right/left) divisor and approximate factorization are two such examples where irregular solutions occur in the unconstrained version but not in the matrix with equality constraints version. Another common problem is that the unconstrained formulation of the problem could be larger or require additional computational resources to compute derivatives. Such is the case in approximate kernel computation or lower rank approximations when naively using a technique called variable projection. Variable projection eliminates bilinear constraints in a bi-linearly constrained problem, so that the original optimization problem is reduced to minimizing a (non-convex) multivariate rational function (possibly subject to convex or non-convex constraints), that is possibly several orders of magnitude larger than the input if done naively. It is important to understand when an unconstrained variant is the ideal choice, when matrix based algorithms are clearly superior and when there is a comparable trade-off between the two and some hybrid technique is superior than either technique alone.

A related research area, but one we will not focus on, is optimization on matrix manifolds. Optimization on matrix manifolds relies on attempting to perform calculus on a matrix manifold. We choose not to discuss these techniques as they are not applicable to all of our problems. Additionally, all manifolds are locally equivalent to Euclidean space, which is our domain of computation. If we can prove our algorithms are "well behaved" in Euclidean space via local analysis, then the manifold version must also be "well behaved" via the local topological equivalence. So any insights derived in the version posed in Euclidean space must also be insightful into the optimization as a manifold formulation of the problem.

1.1.3 Stark Differences between Scalar and Polynomial Matrices

Naturally the methods we use for our problems are tuned specifically for the instance at hand. We include a technical analysis to justify our choice of numerical and floating point optimization algorithms. After careful consideration, most of the ideas in this thesis converge rapidly towards second-order techniques with sufficient conditions for rapid local convergence. Convex or other tractable relaxations are used extensively for initial guesses. Matrix polynomials share many linear algebra properties with their scalar counterparts, however there are some distinct computational challenges that are not analogous.

The output size of an instance of a problem can be an order of magnitude larger than the input size, which is encountered in kernel computation, matrix polynomial least square problems and determinant computations. The determinant of a matrix polynomial is not a scalar, but instead the determinant is a vector of coefficients of a polynomial that depends on the degree, dimension and "size" of the input. Indeed, if we consider

$$\mathscr{A} = \begin{pmatrix} t+1 & 3 & t-2 \\ t-1 & t & t+3 \\ t & 1 & 1 \end{pmatrix},$$

then $\det(\mathscr{A}) = -t^3 + 6t^2 + 2 \cong (2,0,6,-1)^T \in \mathbb{R}^{4\times 1}$. Furthermore, if we look at the equation

$$\mathscr{A}x = \begin{pmatrix} \det(\mathscr{A}) \\ 0 \\ 0 \end{pmatrix} \quad \text{then} \quad x = \begin{pmatrix} -3 \\ t^2+2t+1 \\ -t^2+t-1 \end{pmatrix}.$$

A matrix polynomial is rank deficient if and only if it admits a "rank factorization" analogous to the scalar matrix case, however the matrices appearing in the rank factorization may have entries that are rational functions. Additionally, the "size" of the factors can be substantially larger than the input size, even if there are no rational functions, as the product of two matrices of high degree may have very low degree.

Most problems with matrix polynomials can be transformed into a larger scalar problem with a Toeplitz-block structured matrix and several equality constraints. If the residual of the optimization problem is zero, then classical linear algebra techniques such as QR factorization or the singular value decomposition can be applied since the constraints are trivially satisfied. In the approximate formulation of the problem with a possibly non-zero residual, these equality constraints are not present in the classical, scalar analogs of our problems. These constraints present a sizable hurdle to overcome when designing an optimization algorithm with rapid local convergence.

Techniques similar to this are to approach problems in the context of evaluation and interpolation. Instead of solving one large instance of a problem, we can solve several smaller instances of a problem by evaluating a matrix polynomial at several points (such as complex roots of unity uniformly distributed on the complex unit circle). The idea is typical in both symbolic and numeric communities; evaluate, compute and interpolate to obtain an answer to the instance of our problem. In some instances interpolation is clearly the best choice. In other instances solving several small instances of a problem is more expensive than solving one large problem, especially if there is some structure that allows for fast arithmetic.

The derivatives of matrix polynomial valued functions behave in a similar way to the degree zero (scalar) instance, however there is an extra dimension in the degree that complicates computations. This extra dimension complicates estimating condition numbers of matrix valued functions and computing derivatives of quantities necessary for optimization algorithms relying on derivative information. For example, the gradient of the determinant of a scalar matrix is a vector, but in the context of matrix polynomials the first derivative of the determinant is a Jacobian matrix.

While there are many similarities between the scalar and polynomial matrix problems, there are also significant differences that hinder one from applying existing optimization or computational techniques verbatim without any special considerations. This work is not exhaustive in adapting existing techniques in the instances of matrix polynomials, nor is this intended. The intent is to study symbolic-numeric optimization problems with a focus on matrix polynomials and develop algorithms tailor made for each specific problem.

1.2 Partitioning of the Thesis

The thesis is divided into four different parts. The first part consists of this introduction and Chapter 2. The second part consists of Chapter 3. The third part consists of Chapter 4 and Chapter 5. The fourth part consists of Chapter 6. I am the primary author and main contributor of all of the papers that the chapters in this thesis are based on.

All parts depend on the first part since it establishes common notation and some basic results. The second and third parts are largely independent and can be read separately without too much loss of continuity. The fourth part depends on the previous three in some form for the ideas used.

The first part introduces some notation, terminology and basic results used throughout the thesis. Some of the results are well-known, some are folklore and others may be new but not entirely novel. The objective is to lay a solid foundation for better understanding the ideas presented later in the thesis and how to perform arithmetic on matrix polynomials in a numerically robust manner. For the most part, the notation and results here will be consistent among each chapter.

The second part discusses the problems of computing a nearby singular matrix polynomial and the more general problem of computing a nearby matrix polynomial of at most a prescribed lower rank. Some of these results appear in the conference paper papers [38] and the journal paper [37]. This work is a novel characterization of existing ideas and some new ideas specific to the problem. The focus is on matrix polynomials of non-trivial degree, however the ideas can be applied to scalar matrix polynomials with a prescribed affine structure with some minor modifications. The ideas are manifested in a family of iterative floating point local optimization algorithms with local quadratic convergence (subject to some mild normalization assumptions) and a per-iteration cost that is polynomial in the input size of the problem.

The third part discusses the problem of computing a nearby matrix polynomial with a prescribed spectral structure. The work is based on the conference paper [39] and manuscripts of [40, 52]. The minors of a matrix polynomial reveal the spectral structure which leads to a largely theoretical analysis of the problem as a non-linearly structured approximate greatest common divisor problem in the coefficients of the minors. The approximate greatest common divisor formulation is used to derive new insights into the problem, in particular, the behavior of irregular and regular solutions (and how to detect them). The ideas are expanded upon using linearization theory to prescribe the spectral structure further and develop local iterative floating point optimization techniques with a per-iteration cost that is polynomial in the input size of the problem and a rate of convergence that is locally quadratic (subject to some mild normalization assumptions). When combined, the second and third parts provide a hybrid algorithm that can compute a nearby matrix pencil with a prescribed Kronecker canonical form, although this is not discussed explicitly.

The fourth and final part of the thesis discusses the approximate greatest common (right) divisor of differential operators and is based on the journal paper [36] and consists of some theoretical improvements to the theory presented in my Master's thesis [51]. In the instance of approximate greatest common (right) divisor we provide an equality constrained matrix polynomial algorithm that is an analog of an unconstrained algorithm with local quadratic convergence. The matrix version does not suffer from solutions occurring in projective space, that are otherwise irregular, asymptotic or infinite.

1.2.1 Overview of Chapters

The following content is found in each chapter:

1. A basic review of optimization, matrix polynomials and numerical linear algebra is covered in Chapter 2.

2. Computing a nearby reduced rank matrix polynomial in Chapter 3. This chapter is a combination of the conference paper [38] and the journal paper [37]. The notions of rank and kernel of a matrix polynomial are reviewed in the context of a floating point setting and some techniques to compute "nearby" matrix polynomials of reduced rank are discussed. Some selected examples are provided that demonstrate our implementation of an optimization algorithm based on Newton's method to compute a nearby matrix polynomial of reduced rank.

3. Studying the matrix polynomial determinant and the matrix polynomial (classical) adjoint operator and their first two derivatives in Chapter 4. This chapter is based on an unpublished manuscript of [52] and the introduction in the manuscript [40]. Optimization algorithms that use the spectral structure or prescribe the spectral structure either directly or indirectly involve the determinant. Closely related to the determinant is the adjoint operator which consists of all $(n-1) \times (n-1)$ minors scaled by a factor of ± 1. Understanding how to compute their derivatives efficiently as well as the structure of the underlying problems is important for computing a nearby matrix polynomial with a prescribed spectral structure.

4. Computing a nearby matrix polynomial with an interesting spectral structure is discussed in Chapter 5. This chapter is based on the conference paper [39] and a manuscript of [40]. The Smith normal form reveals the finite spectral structure of a matrix polynomial and the Smith McMillan form reveals the structure of eigenvalues

at infinity. We discuss how to compute a nearby matrix polynomial with an interesting (i.e. non-trivial) Smith normal form. The ideas are generalized to a matrix polynomial with a prescribed finite and infinite spectral structure and rely heavily of the ideas of Chapter 4.

6. We revisit some symbolic-numeric problems relying on structured lower-rank approximation in Chapter 6. We look at applying algorithms for lower-rank approximations of matrix polynomials on the approximate greatest common (right) divisor of differential operators. The new results presented in this section are based on the paper [36] and are new with respect to my previous work in my Master's thesis [51]. The focus is on a hybrid algorithm to more robustly compute a nearby rank deficient differential Sylvester matrix which draws extensively on Chapter 3.

7. The thesis is concluded in Chapter 7 with a brief summary of what was accomplished and what related open problems remain.

Chapter 2

Preliminaries

This chapter covers some basic concepts from numerical linear algebra, vector calculus, continuous optimization and other areas of mathematics and computer science that are useful throughout the thesis. The notation introduced in this chapter will for the most part be consistent and unchanged throughout the thesis. The reader is not expected to possess significant expertise in the areas discussed, but they should be familiar with the concepts mentioned here, as they will be frequently used without explicit reference.

2.1 Domain of Computation and Basic Notions

Vectors will generally be lower-case letters and matrices will be upper-case letters. Scalar matrices, i.e. degree zero matrix polynomials, will be written in italicized letters and matrix polynomials will be written in script letters. Matrices will be indexed in a standard notation of A_{ij} is the entry of A in row i and column j. For matrix polynomials $\mathscr{A}_{ijk}$ is the k^{th} coefficient of the entry in row i and column j. Vectors are indexed in the same standard notation where b_i is the i^{th} entry of the vector b and vectors of polynomials are indexed as $\mathscr{b}_{ij}$ is the j^{th} coefficient of the i^{th} entry of $\mathscr{b}$. All vectors are implicitly assumed to be column vectors unless explicitly stated.

A $m \times n$ matrix (polynomial) with entries over a ring R ($\mathsf{R}[t]$) will be denoted as $\mathsf{R}^{m\times n}$ ($\mathsf{R}[t]^{m\times n}$). We will generally take $\mathsf{R} = \mathbb{R}$ in the case of scalar matrices. Another popular choice of R is $\mathbb{C}$, however we can handle these as a special case of $\mathbb{R}$ in our optimization problems and so we only extend to $\mathbb{C}$ if necessary. We will generally assume that $m = n$, i.e. the matrices are square unless stated otherwise. The theory we will discuss generalizes in most instances to non-square matrices by padding a matrix with rows or columns of zeros. The degree of a polynomial is the highest-order non-zero term and the degree of a matrix polynomial is the largest degree of all the entries.

A matrix polynomial $\mathcal{A} \in \mathsf{R}[t]^{m\times n}$ can be written as a t scaled sum of $d+1$ scalar matrices as

$$\mathcal{A} = \sum_{j=0}^{d} t^j A_j, \quad \text{where} \quad A_j \in \mathsf{R}^{m\times n},$$

although the choice of basis is largely irrelevant in our computations. The degree of a polynomial or matrix polynomial is the highest-order non-zero term in t appearing.

Our notion of "cost" is in floating point operations or FLOPs over the ground field R. We will not make use of "fast" algorithms unless mentioned, as we are primarily concerned with stability rather than speed. To measure the number of FLOPs used we employ standard asymptotic notation such as $\Omega(\cdot), O(\cdot)$ and $\Theta(\cdot)$. We also sometimes employ "soft Oh" notation, $\widetilde{O}(\cdot)$, which is the usual "big Oh" notation with poly-logarithmic factors suppressed, i.e. for some finite $\ell > 0$, $g(n) \in O(\log^{\ell}(n) f(n))$ implies that $g(n) \in \widetilde{O}(f(n))$.

2.2 Numerical Linear Algebra

In this section we will consider $\mathsf{R} = \mathbb{R}$ and $\mathsf{R} = \mathbb{C}$ exclusively.

Definition 2.2.1 (Identity Matrix)**.** *The identity matrix of dimension $n \times n$ is denoted as I_n is a diagonal matrix whose diagonal entries are all 1. We will write $I_n = I$ when the dimension is implicitly clear.*

Definition 2.2.2 (Vec Operator)**.** *We define the operator* vec : $\mathsf{R}[t] \to \mathsf{R}^{(d+1)\times 1}$ *as follows:*

$$p = \sum_{j=0}^{d} p_j t^t \in \mathsf{R}[t] \quad \mapsto \quad \mathrm{vec}(p) = (p_0, p_1, \ldots, p_d)^T \in \mathsf{R}^{(d+1)\times 1}$$

The vec *operator* $\mathrm{vec}(\cdot)$ *is extended to map* $\mathsf{R}[t]^{m\times n}$ *to a single vector in* $\mathsf{R}^{mn(d+1)\times 1}$ *by stacking columns of (padded) coefficient vectors on top of each other as follows:*

$$\mathcal{A} \in \mathsf{R}[t]^{m\times n} \mapsto \mathrm{vec}(\mathcal{A}) = \begin{pmatrix} \mathrm{vec}(\mathcal{A}_{11}) \\ \vdots \\ \mathrm{vec}(\mathcal{A}_{mn}) \end{pmatrix} \in \mathsf{R}^{mn(d+1)\times 1}.$$

The process of devectorizing a vector into a vector of polynomials is implicitly defined by this definition.

Definition 2.2.3 (Polynomial Vec Operator)**.** *The* pvec *operator maps* $\mathsf{R}[t]^{m\times n}$ *to a vector* $\mathsf{R}[t]^{mn\times 1}$ *as*

$$\mathcal{A} \in \mathsf{R}[t]^{m\times n} \mapsto \operatorname{pvec}(\mathcal{A}) = \begin{pmatrix} \mathcal{A}_{11} \\ \vdots \\ \mathcal{A}_{mn} \end{pmatrix} \in \mathsf{R}[t]^{mn\times 1}.$$

The pvec$(\cdot)$ operator acts like the usual vectorization operator over $\mathsf{R}[t]$. We define the vectorization of matrix polynomials in this somewhat non-standard way to facilitate the computation of derivatives of matrix polynomial valued functions.

Definition 2.2.4 (Kronecker Product)**.** *The Kronecker product of* $\mathcal{A} \in \mathsf{R}[t]^{m\times n}$ *and* $\mathcal{B} \in \mathsf{R}[t]^{k\times \ell}$ *denoted as* $\mathcal{A} \otimes \mathcal{B}$ *is the* $mk \times n\ell$ *matrix over* $\mathsf{R}[t]$ *defined as*

$$\mathcal{A} \otimes \mathcal{B} = \begin{pmatrix} \mathcal{A}_{11}\mathcal{B} & \cdots & \mathcal{A}_{1n}\mathcal{B} \\ \vdots & & \vdots \\ \mathcal{A}_{m1}\mathcal{B} & \cdots & \mathcal{A}_{mn}\mathcal{B} \end{pmatrix} \in \mathsf{R}[t]^{mk\times n\ell}.$$

This definition of Kronecker product, sometimes referred to as the "outer product", also holds for scalar matrices (and vectors).

Lemma 2.2.5. *For scalar matrices of compatible dimension* A, X *and* B *over* R*, we have*

$$\operatorname{vec}(AXB) = (B^T \otimes A)\operatorname{vec}(X).$$

Likewise, for matrix polynomials $\mathcal{A}, \mathcal{X}$ *and* $\mathcal{B}$ *of compatible dimension over* $\mathsf{R}[t]$*, we have*

$$\operatorname{pvec}(\mathcal{A}\mathcal{X}\mathcal{B}) = (\mathcal{B}^T \otimes \mathcal{A})\operatorname{pvec}(\mathcal{X}).$$

The Kronecker product can also be used to re-write matrix equations of the form $AX = B$, for matrices A, B and X of compatible dimensions, to

$$\operatorname{vec}(AX) = (X^T \otimes I)\operatorname{vec}(A) = (I \otimes A)\operatorname{vec}(X) = \operatorname{vec}(B).$$

Lemma 2.2.6 (see [83])**.** *There exists a permutation matrix* $\boldsymbol{K}_{m,\ell} \in \mathbb{Z}^{m\ell\times m\ell}$ *called the* commutation matrix *that satisfies for* $A \in \mathsf{R}^{m\times n}$ *and* $B \in \mathsf{R}^{k\times \ell}$

$$\operatorname{vec}(A \otimes B) = (I_n \otimes \boldsymbol{K}_{m,\ell} \otimes I_k)(\operatorname{vec}(A) \otimes \operatorname{vec}(B)).$$

Of course the relationship holds for $\mathcal{A} \in \mathsf{R}[t]^{m\times n}$ and $\mathcal{B} \in \mathsf{R}[t]^{k\times \ell}$ with

$$\operatorname{pvec}(A \otimes B) = (I_n \otimes \boldsymbol{K}_{m,\ell} \otimes I_k)(\operatorname{pvec}(A) \otimes \operatorname{pvec}(B)).$$

Importantly, we can compute $\boldsymbol{K}_{m,\ell}$ as

$$\boldsymbol{K}_{m,\ell} = \sum_{j=1}^{\ell} (e_j^T \otimes I_m \otimes e_j),$$

where $e_j \in \mathbb{Z}^{n\times 1}$ is a column vector with j^{th} component 1 and zero elsewhere. The commutation matrix satisfies other useful properties, however we only need it to transform the vec of Kronecker products into the Kronecker products of their vecs to perform differentiation of matrix functions.

Definition 2.2.7 (Rank). *The rank of $\mathcal{A} \in \mathsf{R}[t]^{m\times n}$ with $n \le m$ is the number of linearly independent (over $\mathsf{R}[t]$) rows or columns of $\mathcal{A}$. The rank of $A \in \mathsf{R}^{m\times n}$ is the number of linearly independent (over R) rows or columns of A.*

Definition 2.2.8 (SVD [47]). *The Singular Value Decomposition (SVD) of $A \in \mathbb{C}^{m\times n}$ with $n \le m$ is given by $U^*\Sigma V$, where $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ satisfy $UU^* = I$, $VV^* = I$ where $*$ denotes the Hermitian (conjugate) transpose and $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_n) \in \mathbb{R}^{m\times n}$ is a diagonal matrix whose diagonal consists of non-negative entries that are the singular values of A, ordered in descending order.*

The distance to the nearest (unstructured) matrix of rank $s < m$ is $\sigma_{s+1}(A)$. If A has only real entries, then U and V are orthogonal (contain only real entries) instead of unitary, i.e. $UU^T = I$ and $VV^T = I$.

Sometimes we will refer to a "thin" SVD, in which $A \in \mathsf{R}^{m\times n}$ has rank s and we can assume $U \in \mathsf{R}^{s\times m}$, $V \in \mathsf{R}^{s\times n}$ and $\Sigma \in \mathbb{R}^{s\times s}$. The diagonal matrix Σ consists of the non-zero singular values, where the matrices U and V have some rows removed corresponding to singular values that are zero. Sometimes we will say that we "compress" A if we write A as a factorization arising from a thin SVD.

Definition 2.2.9. *The Moore-Penrose pseudo inverse of $A \in \mathsf{R}^{m\times n}$ of rank s is denoted as A^+ and is defined as $A^+ = V^*\Sigma^+U$, where $\Sigma^+ = \operatorname{diag}(\sigma_1^{-1}, \ldots, \sigma_s^{-1}, 0, \ldots, 0) \in \mathbb{R}^{n\times m}$. If $s = n$ and $m = n$ then $A^+ = A^{-1}$, which is the usual matrix inverse.*

While working with matrix polynomials the inverse matrix also exists over $\mathsf{R}(t)^{n\times n}$, however the entries are rational functions. A pseudo-inverse can also be defined, but this is not directly useful for us to do.

Definition 2.2.10 (QR Decomposition)**.** *Every matrix $A \in \mathbb{C}^{m\times n}$ can be written as $A = QR$ where $R \in \mathbb{C}^{m\times n}$ is upper triangular and $Q \in \mathbb{C}^{m\times m}$ satisfies $QQ^* = I$. To ensure that the QR decomposition is unique*[1] *we typically assume that the diagonal entries of R are real and non-negative.*

One can also define an analogous QL decomposition where L is lower triangular.

The QR decomposition and variations thereof are useful as a constructive and numerically robust method to obtain reduced basis elements for certain problems.

Definition 2.2.11 (Definite Matrix)**.** *A matrix $A \in \mathbb{R}^{n\times n}$ is positive (semi) definite if A is symmetric and has all positive (non-negative) eigenvalues. Negative definiteness is defined analogously.*

We write $A \succ 0$ if A is positive definite and $A \succeq 0$ if A is positive semidefinite.

The notion of definiteness also holds for complex valued matrices that are complex-conjugate symmetric (Hermitian).

Definition 2.2.12 (Matrix Norms)**.** *For scalar matrices we frequently write $\|\cdot\|_2$ for the largest singular value, and $\sigma_{\min}(\cdot)$ for the smallest singular value. Our use of $\|\cdot\|$ for degree zero matrices will be $\|\cdot\|_2$ unless otherwise stated. The $\|\cdot\|_2$ condition number of $A \in \mathsf{R}^{m\times n}$ with $\mathsf{R} \in \{\mathbb{R}, \mathbb{C}\}$ is denoted as $\kappa_2(A) = \frac{\|A\|}{\sigma_{\min}(A)}$.*

Definition 2.2.13 (Matrix Polynomial Norms)**.** *We define the norm of a polynomial as the coefficient 2-norm, so for $a \in \mathbb{C}[t]$ of degree at most d, we define*

$$\|a\|_2 = \|(a_0, a_1, \ldots, a_d)\|_2 = \sqrt{\sum_{0\le j\le d} |a_j|^2}.$$

Matrix polynomial norms are defined as a distributed norm over the coefficients, some times referred to as the Frobenius norm. For $\mathcal{A} \in \mathbb{C}[t]^{m\times n}$ we define $\|\mathcal{A}\|_F = \|\operatorname{vec}(\mathcal{A})\|_2$.

The Frobenius norm is useful since it is easy to compute, translates from scalar to polynomial matrices in the obvious way, and is twice differentiable. By equivalence of

[1]This will make the QR factorization unique when A has full rank. If A is rank deficient we can define a "thin" QR decomposition that will be unique. In this work we are generally not concerned about the uniqueness of QR factorizations, since we use the decomposition as an intermediate step, then re-scale the resulting quantity.

norms in finite dimensional vector spaces, the theory developed will hold under minor modifications for other twice continuously differentiable norms, and some metrics that are twice differentiable almost everywhere. The choice to treat a matrix polynomial like a vector instead of a collection of $d+1$ scalar matrices is useful when studying matrix polynomial valued functions as a vector valued mapping from $\mathsf{R}^{n^2(d+1)} \to \mathsf{R}^N$ for some $N \geq 1$. We will later observe that $\|\cdot\|_F$, naturally appears when looking at first and second-order Taylor series approximations.

We emphasize that in the instances of matrix polynomials we treat them like a vector, rather than a collection of $d+1$ scalar matrices of dimension $m \times n$. In a finite dimensional vector space, all norms are equivalent up to a scaling factor (possibly depending on the dimension of the space), so other choices are certainly valid but will generally not be considered further. Additionally, all norms are convex, hence differentiable almost everywhere with respect to the Lebesgue measure. Accordingly, most results can be generalized with the use of sub-gradients, a generalization of the gradient to non-smooth continuous functions.

Other norms are possible, and some popular norms in the literature are the matrix polynomial 2-norm defined as $\|\mathscr{A}\|_2 = \left\|\begin{pmatrix} A_0 & A_1 & \cdots & A_d \end{pmatrix}\right\|_2$ and the vector 1-norm $\|\mathscr{A}\|_1 = \|\operatorname{vec}(\mathscr{A})\|_1$. We do not study other norms explicitly, but do make use of them when it facilitates the presentation of results or in the proofs of technical results.

2.3 The Calculus of Vector and Matrix Valued Functions

In this thesis we need to employ several results from the calculus of vector and matrix valued functions. We generally treat matrix valued functions as a vector valued function (where we apply the $\operatorname{vec}(\cdot)$ operator to the matrix), thus the matrix analogs to these results are defined implicitly through vectorization.

Definition 2.3.1 (Gradient)**.** *The gradient of $f(x_1, \ldots, x_m) = y$ with respect to $(x_1, \ldots, x_m)$ where $f : \mathbb{R}^m \to \mathbb{R}$, is the row vector*

$$\nabla f = \begin{pmatrix} \frac{\partial y}{\partial x_1} & \frac{\partial y}{\partial x_2} & \cdots & \frac{\partial y}{\partial x_m} \end{pmatrix} \in \mathbb{R}^{1 \times m}.$$

Definition 2.3.2 (Jacobian)**.** *The Jacobian matrix of $f(x_1, \ldots, x_m) = (y_1, \ldots, y_n)$ with* *respect to* $(x_1, \ldots, x_m)$ *where* $f : \mathbb{R}^m \to \mathbb{R}^n$*, is the* $n \times m$ *matrix*

$$\nabla f = J_f = \begin{pmatrix} \nabla y_1 \\ \nabla y_2 \\ \vdots \\ \nabla y_n \end{pmatrix} \in \mathbb{R}^{n \times m}.$$

Some authors define the gradient and Jacobian as the transpose of what is presented here, however the distinction is largely irrelevant so long as one is consistent. One notes that the Jacobian is the generalization of the gradient to a vector valued function. It is irrelevant under our mathematical view whether f and g are represented as row or column vectors.

Definition 2.3.3 (Hessian Matrix)**.** *The Hessian of* $f(x_1, \ldots, x_m) = y$ *with respect to* $(x_1, \ldots, x_m)$ *where* $f : \mathbb{R}^m \to \mathbb{R}$*, is the real symmetric* $m \times m$ *matrix*

$$\nabla^2 f = \nabla (\nabla f) \in \mathbb{R}^{m \times m},$$

which is simply the Jacobian matrix of the gradient.

The Hessian can also be generalized to vector-valued functions, however we do not need to make explicit use of this generalization and omit it from our discussion. The Hessian matrix describes the local curvature of a function.

The Jacobian is the "best" first-order approximation[2] of a vector-valued function $f : \mathbb{R}^m \to \mathbb{R}^n$, that is

$$f(x + x_0) \approx f(x_0) + J_f(x_0)x.$$

The Jacobian matrix describes the instantaneous rate of change around the point x_0 of f. In the analysis of algorithms, the Jacobian can be used to define the condition number of a matrix (polynomial) valued function when it has full rank. We can write (ignoring higher-order terms)

$$J_f^+(x)(f(x + \Delta x) - f(x)) \approx \Delta x,$$

where $J_f^+(x)$ is the Moore-Penrose pseudo inverse of the Jacobian matrix of f evaluated at x.

For a function $f : \mathbb{R}^m \to \mathbb{R}$, the Hessian matrix is the best second-order approximation of f, that is

$$f(x + x_0) \approx f(x_0) + \nabla f(x_0)x + \frac{1}{2}x^T \nabla^2 f(x_0)x.$$

[2]Note that the approximation is usually given as $f(x) \approx f(x_0) + J_f(x_0)(x - x_0)$, although the shifted variant is the one we use in this thesis.

2.4 Smooth Continuous Optimization

In this thesis we will need to study several optimization problems of the constrained and unconstrained varieties. We first review some basic results about unconstrained optimization, then generalize them to the constrained case. Most of these well known statements can be found in [10, 89] or another book on optimization.

An unconstrained problem is simply a problem of the form

$$\min_{x \in \Omega} f(x),$$

where $f(x) : \mathbb{R}^m \to \mathbb{R}$ is a function that is bounded below over a domain $\Omega \subseteq \mathbb{R}^m$. A constrained problem is of the form

$$\min_{x \in \Omega} f(x) \text{ subject to } \begin{cases} h(x) = 0, \\ g(x) \geq 0. \end{cases}$$

We assume that all functions f, g and h are at least twice differentiable, hence they are "smooth" with local Lipschitz continuity. Most of the problems we deal with involve multivariate polynomials which are locally Lipschitz.

Definition 2.4.1 (Open Neighborhood). *We denote the open neighborhood around x of radius ε as $B(x; \varepsilon) = \{z \in X : d(x, z) < \varepsilon\}$. Here $d(\cdot, \cdot)$ is a distance function defined over a set X, typically a norm.*

Definition 2.4.2 (Local Minimizer). *A local minimizer of $f(x)$ is a point $x^\star$ where $f(x) \geq f(x^\star)$ for all x in some non-trivial open neighborhood around $x^\star$.*

Definition 2.4.3 (Convexity). *The set X is convex if for all $x_1, x_2 \in X$ and $\gamma \in [0, 1]$ we have that $\gamma x_1 + (1 - \gamma)x_2 \in X$.*

A function $f : X \to \mathbb{R}$ is convex if

$$f(\gamma x_1 + (1 - \gamma)x_2) \leq \gamma f(x_1) + (1 - \gamma)f(x_2).$$

We say that f is strictly convex if the above inequalities are strict whenever $\gamma \in (0, 1)$ and $x_1 \neq x_2$.

For example, norms are convex functions since the triangle inequality implies convexity. Open or closed neighborhoods are also convex sets.

Definition 2.4.4 (Rates of Convergence)**.** *We say that a sequence* $\{x^{(k)}\}_{k=0}^{\infty}$ *for* $x^{(k)} \in X$*, converges to* $x^\star$ *at a rate of convergence that is linear if there exists* $c > 0$ *and* $\beta \in (0, 1)$ *such that for all* k *we have that*

$$\|x^{(k)} - x^\star\| \leq c\beta^k.$$

This implies that

$$\limsup_{k\to\infty} \frac{\|x^{(k+1)} - x^\star\|}{\|x^{(k)} - x^\star\|} \leq \beta,$$

which is the usual quotient rate definition.

We say that the rate of convergence is p *super linear (super linear of order* p*) if there exists* $p > 1$ *such that*

$$\|x^{(k)} - x^\star\|_2 \leq c\beta^{p^k}.$$

We are particularly interested in quadratic convergence, which occurs when $p = 2$*.*

Likewise, this implies that p *super linear convergence occurs when*

$$\limsup_{k\to\infty} \frac{\|x^{(k+1)} - x^\star\|}{\|x^{(k)} - x^\star\|^p} < \infty.$$

In general, optimization methods that use only first-order information, that is information from the first-order derivatives, generally obtain linear convergence. Methods that use second-order information, i.e. second-order derIn ivative or curvature information, are generally able to obtain super linear rates of convergence. A well-known method that frequently has quadratic convergence is Newton's method.

Definition 2.4.5 (Newton's Method)**.** *A straightforward version of Newton's method is defined as solving* $F(x) = 0$ *for* $F : \mathbb{R}^m \to \mathbb{R}^n$ *by iteratively computing* $x^{(k+1)} = x^{(k)} + \Delta x$ *where* Δx *is defined from the linear system of equations*

$$\nabla F(x^{(k)})(\Delta x) = -F(x^{(k)}).$$

In our application of Newton's method we will generally assume that $m = n$, since it is mostly used to solve for second-order optimality conditions, which satisfy this condition. It should be noted that when $m \neq n$, then the iteration may not be well-defined.

Newton's method generally has super-linear convergence under mild assumptions and quadratic convergence when $\nabla F(x)$ has full rank and $F(x)$ satisfies some local Lipschitz continuity assumptions. One of the main topics of this thesis is how to obtain methods with a local rate of convergence that is quadratic for several symbolic-numeric optimization problems when the Jacobian matrix is rank deficient, as this condition is sufficient for fast convergence, but certainly not necessary. Newton's method is also a local technique, in that the convergence rate is local and it may not converge to a solution if the initial point $x^{(0)}$ is too far away from the solution.

Informally, "Newton's method for optimization" relies on using "Newton's method" to solve for the gradient of some function vanishing. It is perhaps unsurprising, that essentially any method that has quadratic convergence is asymptotically equivalent to Newton's method [87]. Indeed, there are many variants of "Newton's method" and we will consider several in this work. When we refer to a "*Newton-like method*" we mean a method that is asymptotically "*Newton's method*".

A technique similar to Newton's method is the Gauss-Newton method that attempts to solve $F(x) = 0$ (in a least-squares sense) for $F : \mathbb{R}^m \to \mathbb{R}^n$ by iteratively computing $x^{(k+1)} = x^{(k)} + \Delta x$ where Δx is defined from the linear system of equations

$$\left(\nabla F(x^{(k)})\right)^T \nabla F(x^{(k)})(\Delta x) = -\left(\nabla F(x^{(k)})\right)^T F(x^{(k)}).$$

A regularized variant of Gauss-Newton (which is essentially Gauss-Newton with a trust-region, sometimes referred to as Levenberg-Marquardt) solves the same problem with the modified iteration

$$\left[\left(\nabla F(x^{(k)})\right)^T \nabla F(x^{(k)}) + \nu_k I\right](\Delta x) = -\left(\nabla F(x^{(k)})\right)^T F(x^{(k)}),$$

where $\nu_k \geq 0$ is a regularization parameter. The choice of ν_k is generally specific for each problem. Our implementation uses a heuristic method to pick ν_k when $F(x^{(k)})$ is far away from zero and $\nu_k \approx \|F(x^{(k)})\|_2$ when $F(x^{(k)}) \approx 0$ (see [32, 110]).

The advantage of Gauss-Newton and related variants is that they often have local quadratic convergence (when $F(x) = 0$) under reasonable assumptions and can be "globalized" to converge to a stationary point of the merit function $\|F(x)\|_2$. In several instances Newton's method may not converge, where a Gauss-Newton method will. When the Jacobian is rank deficient, the Gauss-Newton method and it's variants may converge to a stationary point of the merit function instead of $F(x) = 0$. The condition number of the Gauss-Newton iteration is approximately $\kappa_2^2(\nabla F(x^{(k)}))$ in the worst-case, which may limit some of the instances in which Gauss-Newton like techniques can be applied using floating point computation.

2.4.1 Unconstrained Optimization

We now discuss some basic results about unconstrained optimization problems of the form $\min_{x\in\Omega} f(x)$ for some $f(x)$ that is bounded below.

Lemma 2.4.6 (Necessary Conditions)**.** *Suppose that $x^\star \in \Omega$ is an interior point and a local minimizer of $f(x)$. Then*

1. $\nabla f(x^\star) = 0$ *and*
2. $\nabla^2 f(x^\star) \succeq 0$.

These conditions are *necessary* for $x^\star$ to be a local minimizer of $f(x)$, but they are not *sufficient*. We refer to them as the first-order and second-order necessary conditions. The first-order condition says is that in an infinitesimal neighborhood around f that f is no longer changing and the second-order condition says that the curvature is "non-increasing", or that $f(x^\star)$ is convex. When $x^\star$ is not in the interior of Ω, then a solution occurs on the boundary and these conditions do not necessarily describe such solutions. Fortunately, we generally do not concern ourselves with such solutions, thus we will generally ignore them without much loss of generality.

Lemma 2.4.7 (Sufficient Conditions)**.** *Suppose that $x^\star \in \Omega$ satisfies*

1. $\nabla f(x^\star) = 0$ *and*
2. $\nabla^2 f(x^\star) \succ 0$,

then $x^\star$ is a strict local minimizer of $f(x)$.

What this condition says is that if the gradient of f vanishes and f is strictly convex in some neighborhood of $x^\star$, then $x^\star$ is a local minimizer. Note that the second-order sufficient condition implies that solutions are locally isolated.

2.4.2 Constrained Optimization

We now discuss some basic results from constrained optimization. An inequality constraint $g(x) \geq 0$ is *active* if $g(x) = 0$ and *inactive* if $g(x) \neq 0$. In our work, our view is mostly local and the inequality constraints that could be added to our problems are almost always inactive. The problems we study in this work are almost all equality constrained, so we ignore the conditions for inequality constrained problems, and instead focus on equality constrained problems.

One notes that an inequality constrained problem is essentially an unconstrained problem (in a local sense) if the constraints are all inactive, or an equality constrained problem if some constraints are active, and some are not.

One technique known as the penalty method transforms an equality constrained problem into an unconstrained problem. If $h(x) = 0$ then it is necessary that $\nu\|h(x)\| = 0$ as well, for any $\nu > 0$. One can penalize violating the constraint by writing

$$\min_{x\in\Omega} f(x) \text{ subject to } h(x) = 0 \text{ as } \min_{x\in\Omega} f(x) + \nu\|h(x)\|_2^2,$$

where $\nu > 0$ is a "penalty term" that is sufficiently large. The penalty term forces the constraints to be satisfied, and as $\nu \to \infty$, then the minima of both problems will agree. Penalty functions are useful in several instances and can be used to augment other techniques, but individually they suffer from several draw backs. As $\nu \to \infty$, the condition number of the Hessian matrix tends to ∞, so high accuracy is not achievable unless ν can be bounded in advance. Without any special information, this is difficult. We will reference some methods in this thesis that make use of penalty techniques either directly or indirectly.

Necessary conditions for $x^\star$ to be a local minimizer to the constrained optimization problem are known as the Karush-Kuhn-Tucker (KKT) conditions. The KKT conditions without inequality constraints simplify into the method of Lagrange multipliers.

We define the Lagrangian as

$$L = f(x) + \lambda^T h(x),$$

where λ is a vector of scalars with the same number of components as $h(x)$. Lagrange multipliers classify *regular* solutions, which for the purpose of this thesis are *finite* (non-asymptotic) solutions, but in general include several other instances. The Lagrange multipliers, when unique, describe the sensitivity of the problem and characterize how much the objective function can be improved by violating constraints. There will either be zero, one or infinitely many Lagrange multipliers for a particular local solution.

In many of our problems the Lagrange multipliers are generally not unique, so the usual Linearly Independent Constraint Qualification (LICQ) or *regularity* condition is usually not satisfied. Despite this, one can still prove Lagrange multipliers exist. If the LICQ is satisfied, then this ensures the existence of Lagrange multipliers. In our problems the LICQ may not be satisfied because constraints are redundant at or around a solution, that is $h_j(x^\star) = 0 \iff h_k(x^\star) = 0$ in some situations. Accordingly, several Lagrange multipliers are *inactive* (i.e. $\lambda_j = 0$ for some j). The lack of uniqueness of Lagrange multipliers and redundancy of constraints is typically caused by using auxiliary variables to encode non-convex rank constraints.

There are problems where Lagrange multipliers do not exist, and these solutions are classified as *irregular*. In some problems a similar condition to LICQ known as the Constant Rank Constraint Qualification (CRCQ) will hold, which is simply that for a $\varepsilon > 0$, we have that $\operatorname{rank}(\nabla h(x))$ does not change for all $x \in B(x^\star, \varepsilon)$. In general, constraint qualifications ensure the existence of Lagrange multipliers. We will study some optimization problems where solutions are irregular, and approach these problems by studying a dual problem (not the Lagrangian dual) that satisfies regularity conditions.

Lemma 2.4.8 (KKT (Necessary) Conditions)**.** *Suppose that $x^\star$ is a* regular *local minimizer of $f(x)$ subject to the constraint that $h(x) = 0$. Then there exists a vector of Lagrange multipliers $\lambda^\star$ such that*

1. $\nabla L(x^\star, \lambda^\star) = 0$ *and*
2. $\ker(\nabla h(x^\star))^T \ \nabla_{xx} L(x^\star, \lambda^\star) \ \ker(\nabla(h(x^\star))) \succeq 0.$

These conditions are necessary, but not sufficient for a point $(x^\star, \lambda^\star)$ to be a local minimizer with associated Lagrange multiplier. In practice though, they are often sufficient, and if a constraint qualification is satisfied, then they will be sufficient and necessary for convex problems. The problems we discuss in this thesis are non-convex with many local extrema, so we are content to obtain local solutions when possible.

Lemma 2.4.9 (Sufficient Conditions)**.** *Suppose that there exists a point $(x^\star, \lambda^\star)$ where*

1. $\nabla L(x^\star, \lambda^\star) = 0$ *and*
2. $\ker(\nabla h(x^\star))^T \ \nabla_{xx} L(x^\star, \lambda^\star) \ \ker(\nabla(h(x^\star))) \succ 0.$

Then $x^\star$ is a regular *local minimizer of $f(x)$ subject to the constraint that $h(x) = 0$ with Lagrange multiplier $\lambda^\star$.*

The second-order sufficient condition assumes the existence of Lagrange multipliers, but it is sufficient to obtain an algorithm with *local rapid convergence* despite the LICQ failing to hold. In our problems Lagrange multipliers are not unique because some constraints are redundant or the problem is over-padded with zeros, so the CRCQ holds or else the problem can be modified so that the CRQC holds. Alternatively, in several instances one can force the LICQ to hold by eliminating constraints that are repeated. We will discuss this in more detail for each specific problem.

The matrix $\nabla^2 L$ has a special structure and is sometimes called the "KKT matrix", which appears frequently in Newton-like methods. We note that

$$\nabla^2 L = \begin{pmatrix} \nabla^2_{xx} L & \nabla_x h(x)^T \\ \nabla_x h(x) & 0 \end{pmatrix},$$

so the behavior of the Jacobian matrix of the constraints is very important when studying convergence properties of Newton-like methods. In particular, if the Jacobian of the constraints is rank deficient, then $\nabla^2 L$ will be rank deficient.

The following well-known result will be used several times without explicit reference in this thesis.

Lemma 2.4.10 (Weierstrass' Theorem). *Let $\Omega \subseteq \mathbb{R}^m$ be a compact set with respect to the Euclidean norm. Suppose that $f : \Omega \to \mathbb{R}$ is a continuous function with respect to a reasonable metric, then $\max_{x\in\Omega} f(x)$ and $\min_{x\in\Omega} f(x)$ exist.*

What this says is that if we minimize (or maximize) over a compact set, we are guaranteed that a solution to the optimization problem exists. In Euclidean space, the notion of compactness is equivalent to closed and bounded. Unfortunately, this result provides us with no means to actually compute a solution.

Several results in this work follow the general form of

1. Prove that solutions exist,
2. Argue that there exist "KKT points" (i.e. regular solutions to the optimization problem exist),

3a. Show that some KKT points (i.e. a global minimizer if the residual is small) satisfy second-order sufficient conditions, and

3b. Use a numerical algorithm to compute a KKT point (that probably satisfies second-order conditions).

We could also try to enforce that the second-order necessary condition holds if we viewed our results in a "global" scope instead of a "local" one. Our view is almost entirely local given that the geometry of solutions are preserved locally, and not globally. In several instances, large perturbations will radically change the geometry of solutions to an optimization problem. Accordingly, our view is local, which is entirely reasonable given that we prove that under mild conditions, the stronger second-order sufficient condition will hold for several of our problems. If the computed solution does not satisfy second-order sufficient conditions, then the instance of the problem is likely ill-posed or the initial guess was poorly chosen, as the second-order sufficient conditions imply the existence of a non-trivial radius of stability for a Newton-like method.

2.5 Basic Results About Matrix Polynomials

Matrix polynomials frequently occur in the linear form $tA_1 - A_0 \in \mathsf{R}[t]^{n\times n}$, which is sometimes referred to as a matrix pencil. Matrix pencils appear when solving polynomial eigenvalue problems, which is seeking a solution to

$$A_0 x = tA_1 x \text{ for } x \in \mathsf{R}^{n\times 1}.$$

Most of the literature focuses on matrix pencils because every $n \times n$ matrix polynomial of degree d can be linearized into the form $\mathscr{P} = tP_1 - P_0$, where $\mathscr{P}$ has the same spectral structure as $\mathscr{A}$ and $P_0, P_1 \in \mathsf{R}^{nd\times nd}$. In general, polynomial eigenvalue problems of the form $\mathscr{A}x = 0$ can be solved by linearization, using the QZ decomposition or modifying the problem so that the QZ decomposition can be applied.

Definition 2.5.1 (Companion Linearization)**.** *Many linearizations exist [44], although we generally restrict ourselves to* strong linearizations *that preserve the finite and infinite spectral structure, such as the well-known companion linearization that takes*

$$P_0 = \begin{pmatrix} & I & & & \\ & & \ddots & & \\ & & & I & \\ -A_0 & -A_1 & \cdots & \cdots & -A_{d-1} \end{pmatrix} \text{ and } P_1 = \begin{pmatrix} I & & & \\ & I & & \\ & & \ddots & \\ & & & A_d \end{pmatrix}.$$

Note that $\mathscr{P}$ is a vector with n^2d^2 entries, where as $\mathscr{A}$ is a vector with $n^2(d+1)$ entries. Of course other linearizations are possible, such as those encountered in [81, 82], which we do not explicitly consider.

Definition 2.5.2 (QZ Decomposition)**.** *Let $\mathscr{P} \in \mathbb{C}^{nd\times nd}$ be a matrix pencil. Then there exist unitary matrices $Q \in \mathbb{C}^{nd\times nd}$ and $Z \in \mathbb{C}^{nd\times nd}$ such that*

$$QP_0Z = \begin{pmatrix} r_1 & * & \cdots & * \\ & r_2 & \cdots & * \\ & & \ddots & \vdots \\ & & & r_{nd} \end{pmatrix} \quad \textit{and} \quad QP_1Z = \begin{pmatrix} s_1 & * & \cdots & * \\ & s_2 & \cdots & * \\ & & \ddots & \vdots \\ & & & s_{nd} \end{pmatrix}.$$

Accordingly, $\mathscr{P}$ can be triangularized by two-sided unitary transformations, and the eigenvalues can be easily extracted as a ratio of the diagonal entries. Note that when the input is real valued, the arising decomposition will usually be complex valued.

The QZ decomposition reveals the finite and infinite eigenvalues of $\mathscr{P}$, but not their multiplicity structure. The finite eigenvalues are the zeros of $\det(\mathscr{P})$. The infinite eigenvalues appear when P_1 is rank deficient, so $\deg(\det(\mathscr{P})) \neq nd$. The infinite eigenvalues are characterized by $-tP_0 + P_1$. We note that the eigenvalues at infinity are always "known" so a reasonable implementation of the QZ decomposition will compute the infinite eigenvalues first then compute the remaining eigenvalues with this knowledge.

Definition 2.5.3 (Smith Normal Form)**.** *The Smith Normal Form (SNF) of a matrix polynomial $\mathscr{A} \in \mathsf{R}[t]^{n\times n}$ of degree at most d is defined as*

$$\mathscr{S} = \begin{pmatrix} s_1 & & & & & & \\ & s_2 & & & & & \\ & & \ddots & & & & \\ & & & s_{\operatorname{rank}(\mathscr{A})} & & & \\ & & & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 0 \end{pmatrix} \in \mathsf{R}[t]^{n\times n},$$

where $s_1, \ldots, s_{\operatorname{rank}(\mathscr{A}))}$ are monic (leading coefficient is 1) and $s_i | s_{i+1}$ for $1 \leq i < \operatorname{rank}(\mathscr{A})$, such that there exist unimodular $\mathscr{U}, \mathscr{V} \in \mathsf{R}[t]^{n\times n}$ (i.e., with determinants in $\mathsf{R}\backslash\{0\}$) with $\mathscr{S} = \mathscr{U}\mathscr{A}\mathscr{V}$. The Smith form always exists and is unique, although the matrices $\mathscr{U}$, $\mathscr{V}$ are not unique [65]. There always exist $\mathscr{U}$ and $\mathscr{V}$ with $\deg(\mathscr{U}) \leq nd$ and $\deg(\mathscr{V}) \leq nd$, but there also exists $\mathscr{U}$ and $\mathscr{V}$ with degrees that are arbitrarily high as well. The diagonal entries $s_1, \ldots, s_n$ are referred to as the invariant factors *of $\mathscr{A}$.*

We can write $s_j = p_{1,j}(t)^{\alpha_{1,j}} \cdots p_{\ell,j}(t)^{\alpha_{\ell,j}}$. The exponents $\{\alpha_{\gamma,j}\}_{\gamma=1}^{\ell}$ are known as the structural supports of s_j.

There is also a canonical form analogous to the Smith Normal Form applied to the rational matrix $\mathcal{A}(t^{-1})$ that has a similar divisibility property (involving rational functions) that characterizes the infinite spectral structure, known as the Smith-McMillan form. For matrix polynomials, studying the spectral structure of $t = 0$ of $t^d\mathcal{A}(t^{-1})$ reveals the spectral structure of eigenvalues at infinity.

The companion linearization, and several others also satisfy the property that $\mathrm{SNF}(\mathscr{P}) = \mathrm{diag}(I, I, \ldots, I, \mathrm{SNF}(A))$.

There is a more general form due to Kronecker that reveals the finite and infinite spectral structure, as well as classifying the left and right kernels.

Definition 2.5.4 (Kronecker Canonical Form)**.** *The Kronecker Canonical Form (KCF) of a matrix polynomial [65] $\mathcal{A} \in \mathbb{C}[t]^{n\times n}$ of degree 1 is defined as the block-diagonal matrix*

$$K = \mathrm{diag}(L_{\mu_1}, L_{\mu_2}, \ldots, L_{\mu_\alpha}, \widetilde{L}_{\nu_1}, \widetilde{L}_{\nu_2}, \ldots, \widetilde{L}_{\nu_\alpha}, tJ - I, tI - F)$$

where

1. *F is in Jordan form,*
2. *J is a nilpotent Jordan matrix (a matrix in Jordan normal form with all zero eigenvalues),*
3. *L_μ is a $\mu \times (\mu + 1)$ matrix of the form* $\begin{pmatrix} t & -1 & & & \\ & t & -1 & & \\ & & \ddots & \ddots & \\ & & & t & -1 \end{pmatrix}$ *and*
4. *$\widetilde{L}_\nu$ is a $(\nu + 1) \times \nu$ matrix of the form* $\begin{pmatrix} t & & & \\ -1 & t & & \\ & \ddots & \ddots & \\ & & -1 & t \end{pmatrix}$.

The blocks L and $\widetilde{L}$ do not appear if $\mathcal{A}$ has full rank (sometimes we say $\mathcal{A}$ is regular), and these blocks are known as the right and left Kronecker indices. The block $tI - F$ consists of the eigenvalues of $\mathcal{A}$ (zeros of $\det(\mathcal{A})$) and each Jordan block contains the information about the multiplicity or structural support information of the eigenvalues. We are especially interested in this block because it reveals the SNF of $\mathcal{A}$. The block $tJ - I$

consists of the eigenvalues at infinity, which reveals the Smith-McMillan form of a matrix of rational functions (a generalization of the SNF to matrices whose entries are rational functions). The KCF is a generalization of the Jordan canonical form of scalar matrices.

The Kronecker indices that occur when $\mathscr{A}$ is rank deficient correspond to a left or right kernel basis of $\mathscr{A}$ that is "minimal" or has kernel vectors of minimal degree. The minimal kernel is useful, but it is not always required in our problems.

Definition 2.5.5. *The adjoint matrix of $\mathscr{A} \in \mathsf{R}[t]^{n\times n}$ is denoted as $\operatorname{Adj}(\mathscr{A})$ and is simply the transpose of the co-factor matrix,*

$$\operatorname{Adj}(\mathscr{A}) = \begin{pmatrix} \det(\mathscr{A}_{11}) & -\det(\mathscr{A}_{12}) & \cdots & (-1)^{n+1}\det(\mathscr{A}_{1n}) \\ -\det(\mathscr{A}_{21}) & \det(\mathscr{A}_{22}) & \cdots & (-1)^{n+2}\det(\mathscr{A}_{2n}) \\ \vdots & \vdots & \cdots & \vdots \\ (-1)^{n+1}\det(\mathscr{A}_{n1}) & (-1)^{n+2}\det(\mathscr{A}_{n2}) & \cdots & (-1)^{2n}\det(\mathscr{A}_{nn}) \end{pmatrix}^T,$$

where $\det(\mathscr{A}_{ij})$ is the determinant of a sub-matrix with the i^{th} row and j^{th} column removed. The adjoint satisfies $\operatorname{Adj}(\mathscr{A})\mathscr{A} = \mathscr{A}\operatorname{Adj}(\mathscr{A}) = \det(\mathscr{A})I$. This relationship implies that $\operatorname{Adj}(\mathscr{A}) = \det(\mathscr{A})\mathscr{A}^{-1}$ when $\mathscr{A}$ has full rank.

Note that this definition applies verbatim to scalar matrices by taking $d = 0$ (and in general holds over arbitrary commutative rings and their fraction fields). If $\mathscr{A} \in \mathsf{R}[t]^{n\times n}$ has full rank, then unless $\det(\mathscr{A})$ is a scalar, we have that $\mathscr{A}^{-1} \in \mathsf{R}(t)^{n\times n}$.

Definition 2.5.6 (Affine/Linear Structure)**.** *A non-zero matrix polynomial $\mathscr{A} \in \mathsf{R}[t]^{n\times n}$ of degree at most d has a* linear structure *from a set $\mathscr{K}$ if $\mathscr{A} \in \operatorname{span}(\mathscr{K})$ as a vector space over R, where*

$$\mathscr{K} = \left\{C_{0,0}, \ldots, C_{0,k}, tC_{1,0}, \ldots, tC_{1,k}, \ldots, t^d C_{d,0}, \ldots, t^d C_{d,k}\right\},$$

where $C_{l,j} \in \mathsf{R}^{n\times n}$ for $0 \le j \le k$, where $k > 0$ is a finite index variable. If $\mathscr{A} = \mathcal{C}_0 + \mathcal{C}_1$, where $\mathcal{C}_0 \in \mathsf{R}[t]^{n\times n}$ is fixed and $\mathcal{C}_1 \in \operatorname{span}(\mathscr{K})$, then $\mathscr{A}$ is said to have an affine *structure from the set $\mathscr{K}$.*

Affinely and linearly structured matrices are best thought of imposing linear equality constraints on the entries. Examples of matrices with a linear structure include matrices with prescribed zero entries/coefficients, Toeplitz/Hankel matrices, Sylvester matrices, resultant-like matrices, Ruppert matrices and several other matrices appearing in symbolic-numeric computation. Matrices with an affine structure include all matrices with a linear structure and other matrices, such as ones with prescribed non-zero constant entries/coefficients, such as monic matrix polynomials.

The companion linearization is a matrix polynomial with an affine structure. The theory of matrix pencils (degree one matrix polynomials) with an affine coefficient structure is sufficient to study most matrix polynomial problems in theory.

Definition 2.5.7 (Convolution Matrix)**.** *Polynomial multiplication between polynomials $a, b \in \mathsf{R}[t]$, of degrees d_1 and d_2, respectively may be expressed as a Toeplitz-matrix-vector product. We define*

$$\phi_{d_2}(a) = \begin{pmatrix} a_0 & & \\ \vdots & \ddots & \\ a_{d_1} & & a_0 \\ & \ddots & \vdots \\ & & a_{d_1} \end{pmatrix} \in \mathsf{R}^{(d_1+d_2+1)\times(d_2+1)}. \qquad \textit{It follows that} \quad \operatorname{vec}(ab) = \phi_{d_2}(a)\operatorname{vec}(b).$$

When a is non-zero, we can also define division through pseudo-inversion of the convolution matrix since $\phi_{d_2}(a)^{+}\operatorname{vec}(ab) = \operatorname{vec}(b)$.

Definition 2.5.8 (Block Convolution Matrix)**.** *We can express multiplication of a matrix and vector of polynomials, $\mathscr{A} \in \mathsf{R}[t]^{m\times n}$ and $\mathscr{b} \in \mathsf{R}[t]^{n\times 1}$, of degrees at most d_1 and d_2 respectively, as a scalar linear system*

$$\operatorname{vec}(\mathscr{A}\mathscr{b}) = \Phi_{d_2}(\mathscr{A})\operatorname{vec}(\mathscr{b}),$$

where

$$\Phi_{d_2}(\mathscr{A}) = \begin{pmatrix} \phi_{d_2}(\mathscr{A}_{11}) & \cdots & \phi_{d_2}(\mathscr{A}_{1n}) \\ \vdots & & \vdots \\ \phi_{d_2}(\mathscr{A}_{m1}) & \cdots & \phi_{d_2}(\mathscr{A}_{mn}) \end{pmatrix} \in \mathsf{R}^{m(d_1+d_2+1)\times n(d_2+1)}.$$

The block convolution matrix is sometimes referred to as a "Sylvester matrix" or "resultant matrix" associated with $\mathscr{A}$. We differ in terminology when we say "Sylvester matrix" as we mean a Sylvester matrix appearing in the GCD of two (or more) polynomials (which is sometimes referred to as a resultant matrix as well). We will sometimes use "embedding" to refer to block convolution matrices with several rows and columns deleted as well.

The block convolution matrix is a scalar matrix with a linear structure, and it is important when performing arithmetic on matrix polynomials and analyzing derivatives of matrix polynomial valued functions.

We briefly describe some properties about the kernel of matrix polynomials.

Lemma 2.5.9 (Useful Degree Bounds)**.** *Let $\mathcal{A} \in \mathsf{R}[t]^{n\times n}$ be a matrix polynomial of degree at most d. Then* $\deg(\det(\mathcal{A})) \leq nd$ *and if* $\operatorname{rank}(\mathcal{A}) = s < n$ *then there exists* $\mathcal{b} \in \mathsf{R}[t]^{n\times 1}$ *such that* $\deg(\mathcal{b}) \leq sd$ *and* $\mathcal{A}\mathcal{b} = 0$*.*

Proof. The fact that $\deg(\det(\mathcal{A})) \leq nd$ is well known and follows immediately by an application of Cramer's rule.

When $\mathcal{A}$ is rank deficient we can assume without loss of generality that the leading $(s \times s)$ sub matrix of $\mathcal{A}$ has full rank. There is a unique vector of the form

$$c = (\mathcal{b}_1/\gamma, \ldots, \mathcal{b}_{n-r}/\gamma, -1, 0, \ldots, 0),$$

from Cramer's rule such that $\mathcal{A}c = 0$, where $\gamma \in \mathsf{R}[t]$ is the determinant of the leading $s \times s$ minor of $\mathcal{A}$ and all of $b_1, \ldots, b_s, \gamma \in \mathsf{R}[t]$ have degree at most $sd \leq nd$. If we multiply through by γ we have $\mathcal{b} = \gamma c$ satisfies the requirements. □

So, in the case of $\mathcal{A}$ having full rank, we note that $\operatorname{Adj}(\mathcal{A})$ has degree at most $(n-1)d$. Thus we can look at $\det(\cdot)$ as a mapping from $\mathsf{R}^{n^2(d+1)} \to \mathsf{R}^{nd+1}$ and $\operatorname{Adj}(\cdot)$ as mapping from $\mathsf{R}^{n^2(d+1)} \to \mathsf{R}^{n^2((n-1)d+1)}$.

Lemma 2.5.10. *$\mathcal{A} \in \mathsf{R}[t]^{m\times n}$ with $n \leq m$ is rank deficient if and only if $\Phi_{nd}(\mathcal{A})$ is rank deficient.*

Proof. Suppose that $\mathcal{b} \in \mathsf{R}[t]^{n\times 1}$ is a kernel vector of $\mathcal{A}$ of degree at most nd. Then $\mathcal{A}\mathcal{b} = 0$ and this occurs if and only if $\Phi_{nd}(\mathcal{A}) \operatorname{vec}(\mathcal{b}) = 0$. A kernel vector of degree at most nd exists if and only if $\mathcal{A}$ is rank deficient, thus the proof is complete. □

What the previous two lemmas reveal is a limitation of the block convolution matrix, which is that if $\mathcal{b} \in \ker(\mathcal{A})$ satisfies $\deg(\mathcal{b}) \leq nd$ then $\Phi_{nd}(\mathcal{A})\mathcal{b} = 0$. However, if $\deg(\mathcal{b}) < nd$, then $p(t)\mathcal{b} \in \ker(\mathcal{A})$ for any $p(t) \in \mathsf{R}[t]\backslash\{0\}$ with $\deg(p(t)) \leq nd - \deg(\mathcal{b})$. This means that there is generally not a one-to-one correspondence between $\ker(\mathcal{A})$ and $\ker(\Phi_{nd}(\mathcal{A}))$. In particular, if $\mathcal{A}$ is rank deficient, then $\dim(\ker(\Phi_{nd}(\mathcal{A}))) \geq 2$. We can normalize $\Phi_{nd}(\mathcal{A})$ and $B \subseteq \ker(\Phi_{nd}(\mathcal{A}))$ by deleting some rows and columns in a special way, which we will discuss in detail when needed.

To emphasize why the block-convolution matrices are useful, we note that together with the Kronecker product, we can often transform a matrix polynomial equation into a linear algebra problem over R. For example if we assume $\mathcal{A}\mathcal{X} = \mathcal{B}$ has polynomial solutions in $\mathcal{X}$ of degree at most μ, then we can write

$$(I \otimes \mathcal{A}) \operatorname{pvec}(\mathcal{X}) = \operatorname{pvec}(\mathcal{B}) \implies \Phi_\mu\left([I \otimes \mathcal{A}]\right) \operatorname{vec}(\mathcal{X}) = \operatorname{vec}(\mathcal{B}),$$

which is a linear algebra problem over R. The problem of determining if $\mathcal{X}$ has coefficients from $\mathsf{R}[t]$ and not $\mathsf{R}(t)$ without solving the actual system in advance is another problem which we do not discuss. A useful application of this fact is that when $\mathcal{A}$ has full rank, $\mathcal{A}\operatorname{Adj}(\mathcal{A}) = \det(\mathcal{A})I$, so the adjoint matrix is characterized by the system of equations $\mathcal{A}\mathcal{X} = \det(\mathcal{A})I$, i.e. $\mathcal{X} = \operatorname{Adj}(\mathcal{A})$. Thus, we can easily analyze the Jacobian matrices $\nabla \det(\cdot)$ and $\nabla \operatorname{Adj}(\cdot)$ using most of the theory from matrix calculus.

Lemma 2.5.11. *The block-convolution matrix is quasi-distance preserving in that for $\mu \geq 0$ we have*

$$\|\mathcal{A}\|_F = \frac{\|\Phi_\mu(\mathcal{A})\|_F}{\mu+1}.$$

Proof. This follows by observing columns are cyclic shifts of each other. □

This means that several matrix polynomial optimization problems are equivalent to working on a manifold of linearly structured matrices. This fact will be useful to establish equivalence between several problems. In particular, we can use the SVD on $\Phi_\mu(\mathcal{A})$ but not on $\mathcal{A}$, thus we can use $\Phi_\mu(\cdot)$ to obtain semi-tight lower-bounds on several problems.

Lemma 2.5.12. *For $\mathcal{A} \in \mathsf{R}[t]^{m\times n}$ and $\mathcal{B} \in \mathsf{R}[t]^{n\times k}$ of degree at most d, that*

$$\|\mathcal{A}\mathcal{B}\|_F \leq (d+1)\|\mathcal{A}\|_F\|\mathcal{B}\|_F.$$

In other words, $\|\cdot\|_F$ is $d+1$ sub-multiplicative on matrix polynomials.

Proof. For two polynomials $a, b \in \mathsf{R}[t]$ of degree at most d we have that $\|ab\|_2 = \|\phi_d(a)\operatorname{vec}(b)\|_2 \leq (d+1)\|a\|_2\|b\|_2$. This implies that

$$\begin{aligned}\|\mathcal{A}\mathcal{B}\|_F &\leq (d+1)\left\|\begin{pmatrix}\|\mathcal{A}_{11}\|_2 & \cdots & \|\mathcal{A}_{1n}\|_2\\ \vdots & & \vdots\\ \|\mathcal{A}_{m1}\|_2 & \cdots & \|\mathcal{A}_{mn}\|_2\end{pmatrix}\right\|_F\left\|\begin{pmatrix}\|\mathcal{B}_{11}\|_2 & \cdots & \|\mathcal{B}_{1k}\|_2\\ \vdots & & \vdots\\ \|\mathcal{B}_{n1}\|_2 & \cdots & \|\mathcal{B}_{nk}\|_2\end{pmatrix}\right\|_F\\ &= (d+1)\|\mathcal{A}\|_F\|\mathcal{B}\|_F.\end{aligned}$$ □

It is important to note that this is generally an over-estimate, because

$$\|\operatorname{vec}(ab)\|_2 \leq \min\{\|\phi_d(a)\|_2\|b\|_2, \|\phi_d(b)\|_2\|a\|_2\} \leq (d+1)\|a\|_2\|b\|_2.$$

2.6 Polynomial Approximate Greatest Common Divisor

In this section we briefly review some key results from the theory of polynomial approximate Greatest Common Divisor (GCD). The theory of approximate GCD is essential to normalize solutions to $\mathscr{A}\mathscr{X} = 0$ to ensure that they are *primitive* (the GCD of all entries is 1). The ideas of polynomial approximate GCD are useful to understand how to obtain a lower-rank approximation of a matrix polynomial. Polynomial approximate GCD also appears again in computing the spectral structure of matrix polynomials in a floating point enviroment.

2.6.1 Exact Polynomial Greatest Common Divisor

The Greatest Common Divisor (GCD) of $f, g \in \mathsf{R}[t]$ is a polynomial $h \in \mathsf{R}[t]$ of maximal degree such that h is a factor of both f and g. We typically assume h is monic (leading coefficient is 1), so that h is unique. The GCD of f and g is denoted as $\gcd(f, g) = h$, and if $\deg(h) = 0$ then we define $h = 1$ and say that f and g are relatively prime. This implies that there exists f^*, g^* such that $f = f^*h$ and $g = g^*h$. The quantities f^* and g^* are the co-factors of f and g. The co-factors need not belong to the domain of computation, however in the case of polynomials they will when $\mathsf{R} = \mathbb{R}$ or $\mathsf{R} = \mathbb{C}$, i.e. $f^*, g^* \in \mathsf{R}[t]$, but this does not always need to hold.

Lemma 2.6.1. *The polynomials $f, g \in \mathsf{R}[t]$ have a non-trivial GCD if and only if there exist $u, v \in \mathsf{R}[t]$ satisfying $\deg(u) < \deg(g)$ and $\deg(v) < \deg(f)$ such that $uf + vg = h$. The polynomials u and v are known as the Bézout coefficients.*

Proof. The proof follows by applying the Euclidean algorithm. See [34, Chapter 3] for details. □

The Bézout coefficients can be written in matrix form (padding with zeros as appropriate) as $\begin{pmatrix} \phi_{d_2-1}(f) & \phi_{d_1-1}(g) \end{pmatrix} \begin{pmatrix} \operatorname{vec}(u) \\ \operatorname{vec}(v) \end{pmatrix} = \operatorname{vec}(h)$. If the degree of h is known, then one can solve a linear system of equations to compute h.

Definition 2.6.2 (Sylvester Matrix)**.** *Given $f, g \in \mathsf{R}[t]$ of degrees d_1 and d_2 respectively, we define the Sylvester matrix of f and g as*

$$\operatorname{Syl}(f, g) = \begin{pmatrix} \phi_{d_2-1}(f)^T \\ \phi_{d_1-1}(g)^T \end{pmatrix} \in \mathsf{R}^{(d_1+d_2)\times(d_1+d_2)}.$$

The Sylvester matrix is a matrix with a linear structure and is important due to the fact that it encodes the existence of a GCD between two polynomials. We follow convention by defining the Sylvester matrix with the blocks transposed, as this allows a very straight forward generalization to more than two polynomials. The Sylvester matrix is also sometimes referred to as a resultant matrix by some authors.

Definition 2.6.3 (Generalized Sylvester Matrix)**.** *Given* $\mathbf{f} = (f_1, \ldots, f_k) \in \mathsf{R}[t]^k$ *with sequence of degrees* $\deg(f_1) \geq \deg(f_2) \geq \cdots \geq \deg(f_k)$*, we define the generalized Sylvester matrix of* $\mathbf{f}$ *as*

$$\mathrm{Syl}(\mathbf{f}) = \begin{pmatrix} \phi_{\deg(f_2)-1}(f_1)^T \\ \phi_{\deg(f_1)-1}(f_2)^T \\ \vdots \\ \phi_{\deg(f_1)-1}(f_k)^T \end{pmatrix} \in \mathsf{R}^{(\deg(f_2)+(k-1)(\deg(f_1)))\times(\deg(f_1)+\deg(f_2))}.$$

This definition of generalized Sylvester matrix reduces to the usual Sylvester matrix when $k = 2$. It is possible to define a smaller matrix that encodes the same information, however this is not useful for our purposes since these matrices do not allow higher degree coefficients to be perturbed. This definition is much more amenable to optimization problems and mathematical analysis. The results from the usual Sylvester matrix hold for the generalized one with some minor modifications.

Lemma 2.6.4. *Let* $f, g \in \mathsf{R}[t]$*, then*

$$\dim(\ker(\mathrm{Syl}(f, g))) = \deg(\gcd(f, g)).$$

Proof. The proof follows by applying the Euclidean algorithm. See [34, Chapter 3] for details. □

This result also holds for the generalized Sylvester matrix when the entries of $\mathbf{f}$ have non-zero leading coefficients.

Lemma 2.6.5 ([75])**.** *If the Sylvester matrix is triangularized using only row operations, then the last (highest-index) non-zero row of* Syl *contains the coefficients of the polynomial GCD.*

The Sylvester matrix allows one to use techniques from numerical linear algebra to approach a polynomial problem, and generalized versions of this matrix for several polynomials will be discussed later. In several instances QR factoring the Sylvester matrix will provide a robust approximation of the GCD of two or more polynomials [22].

Definition 2.6.6 (Primitive Matrix Polynomial)**.** *We say that* $\mathcal{A} \in \mathsf{R}[t]^{n\times m}$ *is primitive if the GCD of all entries are relatively prime.*

We say that $\mathcal{A}$ *is approximately primitive if* $\mathcal{A}$ *if the distance to a matrix polynomial that is not primitive is reasonably large.*

When we seek "primitive" solutions to problems in a floating point enviroment, we almost always mean "approximately primitive", because these solutions (or approximations to solutions) are much better behaved.

2.6.2 Approximate Greatest Common Divisor Problems

Computing the GCD of two (or more) polynomials is known to be an ill-posed problem numerically, in the sense that if $\gcd(f, g)$ is non-trivial (i.e. $\deg(\gcd(f, g)) \geq 1$) then the perturbed problem $\gcd(f + \Delta f, g + \Delta g)$ will produce a trivial answer with probability 1.

The first attempts at polynomial GCD with floating point arithmetic [96] generally tried to use the Euclidean algorithm in an intelligent way, which unfortunately is highly unstable. Our view of approximate GCD is more modern, in that we look at the problem of given f and g, compute a "nearby" $\widetilde{f}$ and $\widetilde{g}$ such that $\gcd(\widetilde{f}, \widetilde{g})$ is non-trivial and $\|f - \widetilde{f}\| + \|g - \widetilde{g}\|$ is sufficiently small or minimized for a reasonable $\|\cdot\|$. These type of problems are known as "approximate GCD" problems, and several different views and perspectives exist. Some authors view them as a root perturbation problem, and others view the problem strictly in the coefficients (both are equivalent if the residual to the problem is zero, i.e. $\gcd(f, g) \neq 1$ they both approximate the exact GCD). The root perturbation view does not make sense in our domain of computation since we work on coefficients, and we will ignore it henceforth.

For the most part, the theory of polynomial approximate GCD is similar if we take $\mathsf{R} = \mathbb{C}$ or $\mathsf{R} = \mathbb{R}$. If an explicit difference occurs, then we will emphasize this. Several results with matrix polynomials rely either directly or indirectly on GCD computations.

There are several variations of the approximate GCD problem, but we focus on the following versions.

Problem 2.6.7 (Weak Approximate GCD Problem)**.** *Given* $f, g \in \mathsf{R}[t]$ *and* $d_h > 0$*, find* $\widetilde{f}, \widetilde{g} \in \mathsf{R}[t]$ *with* $\deg(\widetilde{f}) \leq \deg(f)$ *and* $\deg(\widetilde{g}) \leq \deg(g)$ *such that*

$$\|f - \widetilde{f}\|_2^2 + \|g - \widetilde{g}\|_2^2$$

is "adequately small" and $\deg(\gcd(\widetilde{f}, \widetilde{g})) \geq d_h$*.*

In plain language, given a pair of polynomials, find a nearby pair of polynomials of the same or lower degree that have a non-trivial GCD of prescribed degree. This is relaxed from the more strict versions of the problem, since we need:

1. The distance to be "adequately small" and not "minimal", and
2. We prescribe the degree of the divisor.

In practice d_h will be 1 or 2 for $\mathsf{R} = \mathbb{R}$ since there will always be a linear or irreducible quadratic divisor. For the case of $\mathsf{R} = \mathbb{C}$, we can generally take $d_h = 1$. For practical purposes, QR factorization [22] and the SVD [21] are adequate to solve this problem if the residual is close to zero.

This version of approximate GCD is well suited-towards optimization algorithms where we need an initial guess or just need an upper bound on a problem to be adequately small. It is useful to infer the degree of a primitive vector of polynomials, or to normalize a vector of polynomials so that it is primitive.

Problem 2.6.8 (Strong Approximate GCD Problem). *Given $f, g \in \mathsf{R}[t]$ find $\widetilde{f}, \widetilde{g} \in \mathsf{R}[t]$ with $\deg(\widetilde{f}) \leq \deg(f)$ and $\deg(\widetilde{g}) \leq \deg(g)$ such that*

$$\|f - \widetilde{f}\|_2^2 + \|g - \widetilde{g}\|_2^2$$

is minimized and $\deg(\gcd(\widetilde{f}, \widetilde{g})) \geq 1$.

There is a polynomial-time solution to this problem due to [72] that relies on a modified version of variable projection and bivariate polynomial root finding. For $\mathsf{R} = \mathbb{C}$ there will always be a degree one divisor and for $\mathsf{R} = \mathbb{R}$ there will be a degree one or two divisor. The degrees of the divisor can be forced to be higher, but the computation time is no longer polynomial.

Related versions to the strong Approximate GCD problem [112] is to seek a local solution to the unconstrained optimization problem

$$\min_{f^*, g^*, h} \|f - f^*h\|_2^2 + \|g - g^*h\|_2^2, \tag{2.1}$$

where h is normalized to be monic (leading coefficient is 1) and proceed to use a Gauss-Newton method to obtain a local minimizer to the problem, using the SVD to seed an initial guess. Note that by making h monic, the degree of the divisor is prescribed. This is a practical compromise between the Strong and Weak Approximate GCD problems.

A modified version I presented in [35, 36] originally developed for Ore Polynomials, that uses Newton's method for unconstrained optimization obtains a rate of convergence that is quadratic when the residual is sufficiently small. The work [36] also provides a sufficient condition for the minimization problem (2.1) to have a regular solution, as the problem in question is known to have irregular solutions. We postpone the discussion to Chapter 5 on how to deal with instances with irregular solutions to the approximate GCD problems.

Chapter 3

Structured Lower Rank Approximations of Matrix Polynomials

This chapter discusses some problems in Structured Lower Rank Approximations (SLRA) of matrix polynomials. This chapter is heavily based on the conference paper [38] and a manuscript of the journal paper [37].

The goal of this chapter is to derive an algorithm that converges with a quadratic rate of convergence to a rank $n - r$ matrix polynomial with a suitable initial guess under some normalization assumptions. We will also discuss numerical algorithms for matrix polynomial kernel computation. Computing and normalizing the kernel is a key step in the algorithms in this chapter.

3.1 Introduction

Matrix polynomials appear in many areas of computational algebra, control systems theory, differential equations, and mechanics. The algebra of matrix polynomials is typically described assuming that the individual polynomial coefficients come from an exact arithmetic domain. However, in the case of applications these coefficients typically have numeric coefficients, usually real or complex numbers. As such, arithmetic can have numerical errors and algorithms are prone to numerical instability.

Numerical errors have an impact, for example, in determining the rank of a matrix polynomial with floating point coefficients. In an exact setting determining the rank or determinant of a matrix polynomial is straightforward, and efficient procedures are available, for example from [102]. However, in a numeric environment, a matrix polynomial may appear to have full or high rank while at the same time being close to one having lower rank. Here "close" is defined naturally under the Frobenius norm on the underlying coefficient matrices of the matrix polynomial. Rather than computing the rank of the given matrix polynomial exactly, one can ask how far away it is from one that is rank deficient, and then to find one at that distance. In the case of matrices with scalar entries this is a problem solved via the SVD. However, in the case of matrix polynomials no equivalent rank revealing factorization has thus far been available.

In this chapter we consider the problem of computing the nearest matrix polynomial to an input matrix polynomial in $\mathbb{R}[t]^{m\times n}$, $n \leq m$ having a kernel with rank at most a specified value r. More precisely, given an integer r and an $\mathcal{A} \in \mathbb{R}[t]^{m\times n}$ of full rank, we want to compute $\Delta\mathcal{A} = \mathcal{A}^{\Delta} \subseteq \mathbb{R}^{mn(d+1)}$, such that under "reasonable" structured perturbation function $\boldsymbol{\Delta}(\cdot) : \mathbb{R}^{mn(d+1)} \to \mathbb{R}[t]^{m\times n}$ we have that $\mathcal{A} + \boldsymbol{\Delta}(\mathcal{A}^{\Delta})$ has rank at most $n - r$ and $\|\mathcal{A}^{\Delta}\|_F = \|\Delta\mathcal{A}\|_F$ is (locally) minimized. Although we primarily consider real valued problems in this thesis, $\boldsymbol{\Delta}(\cdot)$ generalizes in the obvious way for complex valued problems.

Definition 3.1.1 (Structured Perturbation)**.** *For ease of readability, we will slightly abuse notation. We will write $\boldsymbol{\Delta}(\mathcal{A}^{\Delta}) = \widetilde{\Delta}\mathcal{A}$ to denote the structured perturbation of $\mathcal{A}$ and use $\Delta\mathcal{A}$ to denote unstructured perturbations to $\mathcal{A}$.*

For the purpose of performing calculus on matrix-polynomial valued functions, when we write $\operatorname{vec}(\Delta\mathcal{A})$ we are treating $\Delta\mathcal{A}$ as a vector of variables with at most $mn(d + 1)$ components. We can always relate the derivatives of structured perturbations to the unstructured case by applying the chain rule, so it is usually adequate to study the calculus of unstructured perturbations. Under the linear and affine structure assumptions, the required structured derivatives can be computed once, as the transformation Jacobian matrices will be constant.

Note that in the case of unstructured perturbations it is largely irrelevant if we treat $\Delta\mathcal{A} \in \mathbb{R}[t]^{m\times n}$ as a matrix polynomial of degree d or as a vector of $mn(d+1)$ variables, since the two are isomorphic. Furthermore, vectorizing the matrix polynomial yields a vector of the variables, further justifying this decision and making the vector space isomorphism explicit.

In other words, the problem is compute a local minimizer to

$$\min \|\Delta\mathcal{A}\|_F \quad \text{subject to} \quad \begin{cases} (\mathcal{A} + \widetilde{\Delta}\mathcal{A})\mathcal{B} = 0, \\ \operatorname{rank}(\mathcal{B}) = r. \end{cases} .$$

In the case where $n - r$ is one less than the row or column size then this is the problem of finding the nearest matrix polynomial which is *singular*.

In the problems we consider, we generally assume that $\widetilde{\Delta}\mathcal{A} \in \mathbb{R}[t]^{m\times n}$ has independent entries, satisfies $\|\Delta\mathcal{A}\|_F = \|\widetilde{\Delta}\mathcal{A}\|_F$ and is such that one of the following holds:

1. $\deg(\widetilde{\Delta}\mathcal{A}) \leq \deg(\mathcal{A})$, that is the degree of the matrix polynomial does not increase,
2. $\deg(\widetilde{\Delta}\mathcal{A}_{ij}) \leq \deg \mathcal{A}_{ij}$, that is the degrees of individual entries do not increase, and
3. $\widetilde{\Delta}\mathcal{A}_{ij}$ has the same support as $\mathcal{A}_{ij}$, that is lower-order zero terms of $\mathcal{A}$ are not perturbed.

These are all linear perturbation structures that are unstructured except that they do not allow zero entries to be perturbed. Each perturbation structure is suitable in different contexts of computation. The first perturbation structure where degrees do not increase is the most generic and natural to consider. However, if $\mathcal{A}$ has many coefficients that are zero, then perturbing them to be small non-zero entries may not make sense in the context of the problem. Furthermore, if $\mathcal{A}$ is reasonably sparse or consists of sparse polynomials, it may be desirable to preserve this sparsity in some way, in which the other two coefficient perturbation structures may be more desirable. Of course other structures may be defined where $\|\Delta\mathcal{A}\|_F \neq \|\widetilde{\Delta}\mathcal{A}\|_F$, which would correspond to some entries being directly or indirectly weighted. This is done later in Chapter 6 with a special type of Sylvester matrix.

Example 3.1.2. *Consider the matrix polynomial*

$$\mathcal{A} = \begin{pmatrix} t+1 & 1 \\ t & 0 \end{pmatrix} \in \mathbb{R}[t]^{2\times 2}.$$

If we choose $\mathbf{\Delta}(\cdot)$ to preserve the support of $\mathcal{A}$ then

$$\mathbf{\Delta}_{support}(\Delta\mathcal{A}) = \begin{pmatrix} \Delta\mathcal{A}_{111}t + \Delta\mathcal{A}_{110} & \Delta\mathcal{A}_{120} \\ \Delta\mathcal{A}_{211}t & 0 \end{pmatrix}.$$

Similarly, if we choose $\mathbf{\Delta}(\cdot)$ to preserve the degree of each entry of $\mathscr{A}$ then

$$\mathbf{\Delta}_{entry}(\Delta\mathscr{A}) = \begin{pmatrix} \Delta\mathscr{A}_{111}t + \Delta\mathscr{A}_{110} & \Delta\mathscr{A}_{120} \\ \Delta\mathscr{A}_{211}t + \Delta\mathscr{A}_{210} & 0 \end{pmatrix}.$$

Finally, if we choose $\mathbf{\Delta}(\cdot)$ to preserve the degree of $\mathscr{A}$ then

$$\mathbf{\Delta}_{degree}(\Delta\mathscr{A}) = \begin{pmatrix} \Delta\mathscr{A}_{111}t + \Delta\mathscr{A}_{110} & \Delta\mathscr{A}_{221}t + \Delta\mathscr{A}_{120} \\ \Delta\mathscr{A}_{211}t + \Delta\mathscr{A}_{210} & \Delta\mathscr{A}_{221}t + \Delta\mathscr{A}_{220} \end{pmatrix}.$$

The main results in this chapter center on the characterization of the geometry of minimal solutions. We show that minimal solutions exist under linear perturbation structures, that is, for a given r there exists a $\Delta\mathscr{A}$ of minimal norm such that $\mathscr{A} + \widetilde{\Delta\mathscr{A}}$ has rank at most $n - r$ and meets the required constraints on perturbed coefficients. In addition, we show that minimal solutions are isolated and are surrounded by a non-trivial open neighborhood of non-minimal solutions. Regularity and second-order sufficiency conditions are generically satisfied. A restricted version of the problem always satisfies these conditions under unstructured perturbation structures that preserve zero-coefficients, the support of entries or the degree of the matrix polynomial. Finally we show that we can also generalize several of our results to the lower rank approximation instance of matrix polynomials generated by an affine structure. Thus, these results generalize to low-rank approximations of structured matrices by taking the degree to be zero.

We demonstrate efficient algorithms for computing our minimal lower rank approximations. That is, for an input matrix polynomial $\mathscr{A} \in \mathbb{R}[t]^{m\times n}$ (with prescribed affine structure) sufficiently close to a singular matrix polynomial, we give iterative schemes which converges to a rank a deficient matrix polynomial at minimal distance, at a provably quadratic rate of convergence. We further generalize the iterative schemes so that they converge (with a suitable initial guess) to a matrix polynomial with a kernel of dimension at least r, at a (local) minimal distance and a provable local quadratic rate of convergence with a suitable initial guess. Finally, we also discuss a Maple implementation which demonstrates the convergence and numerical robustness of our iterative schemes.

3.1.1 Outline

This chapter has the following objectives:

1. Review existing algorithms for computing the kernel of a matrix polynomial and how to modify them to obtain an initial guess to an optimization problem.

2. Show that lower rank approximations of matrix polynomials exist and the problem is generally well-posed.

3. Analyze some geometric properties of solutions to the problem by examining a constrained rank factorization formulation via Lagrange multipliers. A closed-form expression for the intermediate quantities appearing in the rank factorization are derived and used to show that Lagrange multipliers exist and how to derive an iterative algorithm with rapid local convergence.

4. Use the existence and isolation of solutions to build an iterative algorithm based on solving the KKT conditions. Special attention is paid to the generic and non-generic instances of the problem and how to ensure that the problem is posed so that Lagrange multipliers exist. The method is shown to converge with a rate of convergence that is super linear of order at least two under normalization assumptions via a *minimal embedding* of the problem.

5. Discuss an implementation in Maple with examples comparing our techniques to examples the existing literature with an affine structure.

6. The chapter concludes with a discussion about the problem and some directions for future research.

3.1.2 Previous research

Much of the work in this area has often been done under the heading of *matrix pencils*. See [44] for an excellent overview. Non-singular (full rank) square matrix polynomials are sometimes referred to as *regular matrix polynomials*.

In the case of finding the nearest singular matrix pencil this problem was studied in [38]. Previous to that this problem was posed for linear matrix pencils in [18] and followed up in [17]. The nearest singular matrix polynomial relates to the stability of polynomial eigenvalue problems, linear time invariant systems and differential-algebraic equations studied subsequently in [50, 74]. For non-linear matrix polynomials/pencils, previous works rely on embedding a non-linear (degree greater than one) matrix polynomial into a linear matrix polynomial of much higher order. Theorem 1.1 in Section 7.2 of [44] shows that any regular $\mathcal{A} \in \mathbb{R}[t]^{n\times n}$ of degree d, is equivalent (in terms of the spectrum and minimal kernel invariants) to a linear matrix polynomial $\mathscr{P} = tP_1 - P_0$, for $P_0, P_1 \in \mathbb{R}^{nd\times nd}$. However, this equivalence is (obviously) not an isomorphism, nor is it distance preserving[1].

[1]The equivalence mapping is not surjective.

Hence a nearby singular matrix polynomial to $\mathscr{P} \in \mathbb{R}[t]^{nd \times nd}$ (even when constrained to a degree one perturbation) almost certainly does not correspond to a nearby singular matrix polynomial to $\mathscr{A} \in \mathbb{R}[t]^{n \times n}$. To overcome this issue, one would need to look at perturbations that preserve the affine structure of the linearization (i.e. same companion or other structure), which is computing a nearby matrix polynomial with an affine structure.

In the context of computer algebra the notion of symbolic-numeric algorithms for polynomials has been an active area of research for a number of years, and the general framework of finding nearby instances with a desired algebraic property is being thoroughly explored. Closest to our work here is work on approximate Greatest Common Divisors (GCD) [5, 6, 21], multivariate polynomial factorizations [66], and especially the optimization-based approaches employing the Structured Total Least Norm algorithm [68, 69, 77, 113] and Riemannian SVD [13]. More recently, we have explored computing the approximate GCRD of (non-commutative) differential polynomials [35, 36] and resolve similar issues.

The computer algebra community has made impressive progress on fast, exact algorithms for matrix polynomials, including nearly optimal algorithms for computing ranks, factorizations and various normal forms; see [67] and references therein for a recent overview. Part of our goal in this current chapter is establish a basis for extending the reach of these symbolic techniques to matrices of polynomials with floating point coefficients.

In a more general setting our problem can be formulated as a Structured Low Rank Approximation (SLRA) problem. A popular method to solve SLRA problems is the Structured Total Least Norm (STLN) approach [92, 93]. These are iterative methods and in general their convergence to stationary points is linear (first-order), rather than quadratic, unless additional assumptions are made. In the event STLN converges to a solution, there may be other solutions arbitrarily nearby, as second-order sufficient conditions may not hold (STLN does not account for the curvature of the entire problem). The SLRA problem is a non-linear least squares problem and accordingly other techniques such as the Restricted and Riemannian SVD [23, 24, 25] provide general tools for solving such problems. Other heuristic tools applicable to our problem include variable projection [45, 46] and Newton's method [1]. We would expect these methods to perform very poorly in our case, as one can expect problems with large residuals to perform poorly and the rational function arising from variable projection can be too costly to deal with for modestly sized problems. The problem may also be considered as optimization on a manifold [3], however we do not explicitly consider this approach. For a survey of affinely structured low-rank approximation, see [85, 86].

Other methods for structured low-rank approximation involve the family of lift and project algorithms, with the best known being Cadzow's algorithm [19]. More recently

[97] gives a sequence of alternating projections that provably converge quadratically to a fixed point. However, lift and project algorithms do not generally satisfy necessary first-order optimality conditions. While they may converge (quickly) to a fixed point, there is no guarantee that the fixed point is an optimal solution, though it is usually quite good. Several alternating optimization problems use a block Gauss-Seidel-like method, which can obtain suitable initial guesses for other optimization algorithms after a few iterations [49]. In any case, for specific problems such as ours, understanding the geometry of the minimal solutions (and hence the well-posedness of the problem) is key to effective algorithms for their computation.

SLRA problems are in general NP-hard to solve (i.e. approximate a solution over $\mathbb{Q}$ to arbitrarily high precision), see for example [14, 91]. In general the hardness stems from determining if a bi-linear system of equations admits a non-trivial solution. In the instance of classical matrix polynomials it is trivial to construct feasible points since the underlying scalar matrix problem is linearly structured. This presents a gap from the theory and practice, which we do not attempt to answer.

In the same mindset, the scalar matrix problems of weighted lower rank approximations and non-negative approximate matrix factorizations are similar in spirit to the problems that we approach.

Weighted lower rank approximations seek a lower rank approximation of a matrix where some entries are weighted differently when perturbed by the objective function [98]. In our problem the affine structure can implicitly encode weights of various entries, namely entries that do not change have a weight of “infinity”, i.e. the penalty for perturbing them is infinitely large. Conversely, it is also possible to leave some entries zero-weighted or unweighted, meaning their value has no impact on the objective function of the minimization problem. We generally do not consider zero-weighted instances of the problem, however the ideas presented here can be modified so that they are applicable. Instances with zero-weighted terms are closely related to matrix completion problems (see [20]), which we do not consider (we are not interested in minimizing the rank, we are interested in minimizing some perturbation, backwards error or other suitable loss function).

The non-negative matrix factorization problem in the most straightforward form [108] seeks a lower-rank approximation of a matrix, subject to the *inequality* constraints that the matrix factors into a product of two factors such that the entries of the factors are non-negative, i.e. $A + \Delta A = UV$ where $U_{ij} \geq 0$ and $V_{ij} \geq 0$. In general this problem is NP hard to solve under reasonable models of computation [107]. We deal exclusively with equality constraints, and in a local sense, non-negative matrix factorization is an equality constrained problem thus there is some similarity between the two problems.

Another similarity between these problems is that they are amenable to block coordinate descent techniques if a matrix-factorization (possibly with inequality or inequality constraints) approach is used. We briefly study matrix factorization techniques for computing lower-rank approximations of matrix polynomials to prove some theoretical results. The drawback of equality constrained factorization problems is that they do not immediately guarantee a rank at most approximation, whereas they do for their scalar cousins of weighted and non-negative matrix factorization. Another issue with rank factorizations, is that the dimension of the problem is output sensitive. The output size of several matrix polynomial problems (such as kernel basis computation) can be an order of magnitude larger than the input. This translates into algorithms with poor complexity in terms of the number of FLOPs.

Likewise, one could use the results of [43] on a modified formulation of our problems in lieu of [14, 91] to argue that computing the nearest affinely structured matrix polynomial is an NP hard problem under reasonable models of computation. As noted earlier, we generally stay away from weighted lower-rank approximation as this is not the primary concern for us. To see the equivalence, one notes that if an entry of $\mathscr{A}$ is not permitted to change then the entry has a weight of “infinity”. Our problem can be slightly modified to accommodate an entry that has a weight of zero by ignoring variables pertaining to it (or by assigning an adequately low weight to other entries that may be perturbed, analogous to a penalty method).

Most of our contributions apply to matrix polynomials with an affine structure provided that feasible points exist, that is, singular matrix polynomials with a prescribed structure exist, which is NP-hard in general. In particular, in the degree zero case our algorithms and techniques apply to affine SLRA problems. Thus, computing the nearest (affinely structured) matrix polynomial is equivalent to SLRA problems with an affine structure.

While the contributions in this chapter focus on local properties of SLRA, the local properties also imply global results. The Sum of Squares (SOS) hierarchy is a global framework for studying polynomial optimization problems subject to polynomial constraints [76]. The SOS optimization tools have found experimental success in computing structured distances to singularity and extracting minimizers when the solutions are locally unique, see for example [54]. In general the SOS hierarchy converges for an infinite order of relaxations, but for several problems the relaxations converge after a finite order. The finite convergence is in polynomial time with respect to the input and the number of relaxations. In particular, this finite convergence was observed for affine SLRA problems (notably, computing the nearest singular structured matrix) in [54] but little theory was provided to indicate the reason why. The later work of [88] shows that, under regularity and second-order sufficiency conditions, finite convergence always occurs and that it is possible to extract a minimal solution. In our contributions we prove that second-order sufficiency and regularity conditions hold generically (and if they do not, then they will hold on a restricted subset of the problem) for computing the nearest singular matrix polynomial. The corollary to this is that the SOS hierarchy will have finite convergence for several affine SLRA problems if a solution exists, such as computing the distance of the nearest rank deficient matrix polynomial. If the embedding is minimal then a minimizer may be extracted as well. Another useful feature of the SOS hierarchy is even if convergence cannot be certified, a *structured lower-bound* is obtained.

3.2 Approximate Kernel Computation

A fundamental sub-problem in this thesis is computing a basis for the kernel of a matrix polynomial using floating point arithmetic. We will provide a brief survey of some existing results and discuss some variations, which may or may not be new. In general, the techniques described in a floating point setting will compute the kernel of a nearby matrix polynomial (that is, they are backwards stable). The emphasis on this section is not being fast; it is being numerically stable and remaining polynomial-time.

Most symbolic polynomial linear algebra algorithms rely on GCD computations either directly or indirectly at some level. In a floating point enviroment, round-off and other floating point errors prevent one from accurately computing the exact GCD of two or more polynomials. The inability to compute GCDs correctly results in large degree swell in underlying expressions when performing variants of Gaussian elimination. Replacing exact GCD computations with an approximate GCD computation is generally not advised, as during intermediate steps of various algorithms this introduces errors that will accumulate throughout subsequent computations and can lead to unreliable results. The computed solution could have a large residual in a least squares sense, or worse, have the wrong degree due to the mishandling of a GCD at or near infinity. The fundamental issue is that Gaussian elimination with polynomials is essentially performing the Euclidean algorithm at some level, so any technique based on Gaussian elimination will not be robust to noise or errors in the data. Other symbolic methods that do not rely on Gaussian elimination may not suffer from these drawbacks, but it is not clear that they provide a solution to the underlying least squares problem. Furthermore, to ensure that the computed kernel vectors are minimal, some type of approximate GCD needs to be performed. We will generally not consider exact symbolic methods, but in some instances they may be applicable.

The notion of approximate primitivity is important to normalize solutions to optimization problems to ensure that they are unique. To obtain solutions to $\mathscr{A}x = 0$ that are approximately primitive one can compute $\ker(\Phi_{nd}(\mathcal{A}))$ using the SVD. One can use polynomial approximate GCD and least squares division to extract an approximate kernel vector that is relatively prime. The procedure is backwards stable. Alternatively, one can reduce the degree of the entries of kernel vectors until there is only a single one left, i.e. compute $\ker(\Phi_\mu(\mathcal{A}))$ for different values of μ, by basically performing a binary search on the degree of the entries of a kernel vector of minimal degree.

3.2.1 Rank Computation

A problem related to computing the kernel of a matrix polynomial is determining if a matrix polynomial is rank deficient. Our emphasis is on numerical methods for this problem, as we are more interested in the two related problems:

1. Is the matrix polynomial $\mathcal{A}$ nearby a matrix polynomial that is rank deficient?
2. Can we infer the rank of a nearby matrix polynomial that is rank deficient?

The issue with exact algorithms is that generically all matrix polynomials have full rank, so a rank deficient matrix polynomial that is perturbed by an infinitesimal amount will have full rank. Despite exact quasi-optimal algorithms existing, they are generally not numerically robust. Most algorithms that compute the rank of a matrix polynomial can also be used to extract a kernel basis as well or can be modified accordingly to do this.

Formally, there are several definitions of the rank of a matrix polynomial $\mathcal{A} \in \mathsf{R}[t]^{m\times n}$, all of which are equivalent. The two most useful characterizations to us are to define the rank as the number of linearly independent rows or columns over $\mathsf{R}[t]$, or as $\max_{\omega\in\mathbb{C}}\{\operatorname{rank}(\mathcal{A}(\omega))\}$, which is the maximum rank of $\mathcal{A}$ when evaluated at a generic point.

Exact Algorithms for Rank Computation

In an exact setting [102], we can compute the rank of a matrix polynomial in $\widetilde{O}(n^3 d)$ operations over R. In the bit complexity model of computation, the bit complexity is polynomial in the input size (the bit complexity of the input is constant for our purposes, since it is always a “word”). As discussed previously, these methods are not generally suitable for numerical use.

Rank via Interpolation

Computing the rank of a matrix polynomial numerically is straight forward and can be done efficiently using interpolation and the SVD. $\mathcal{A} \in \mathbb{R}[t]^{n\times n}$ is rank deficient if at $nd+1$ evaluation points $\{\omega_j\}_{j=0}^{nd}$, $\mathcal{A}(\omega_j)$ has reduced rank. In practice, a single evaluation point is adequate, so long as ω_j is not a zero of $\det(\mathcal{A})$. If $\mathcal{A}$ has full rank, then this is verified with high probability by using a single random evaluation point.

We can perform the evaluation quickly by choosing the ω_j to be uniformly distributed on the unit circle then using the Fast Fourier Transform (FFT). Thus computing the rank of a matrix polynomial can be done in $\widetilde{O}(n^3+n^2d)$ FLOPs with high probability (the probability of success is 1 since there are finitely many eigenvalues and uncountably infinitely many possible evaluation points) and $\widetilde{O}(n^4d)$ FLOPs deterministically in the worst case. It is straightforward to compute a kernel vector from the images, as there is always some $\mathcal{b} \in \ker(\mathcal{A})$ such that $\deg(\mathcal{b}) \leq (n-1)d$, although we do not investigate this.

Rank via Block Convolution Matrices

Another technique to compute the rank is via block convolution matrices.

Lemma 3.2.1 ([55])**.** *Let $\mathcal{A} \in \mathsf{R}[t]^{m\times n}$. Let us suppose that $\mathcal{A} = \begin{pmatrix}\mathcal{A}_1 & \cdots \mathcal{A}_n\end{pmatrix}$, where $\mathcal{A}_j \in \mathsf{R}[t]^{m\times 1}$. Let $\mu_L = \sum_{j=1}^{n}\deg(\mathcal{A}_j) - \min_{1\leq j\leq n}\deg(\mathcal{A}_j)$.*

Additionally, let us assume that $\mathcal{A}^T = \begin{pmatrix}\mathcal{A}_1' & \cdots & \mathcal{A}_m'\end{pmatrix}$ where $\mathcal{A}_j' \in \mathsf{R}[t]^{n\times 1}$. Let $\mu_R = \sum_{j=1}^{m}\deg(\mathcal{A}_j') - \min_{1\leq j\leq m}\deg(\mathcal{A}_j')$.

Now suppose that $\mu = \min\{\mu_L, \mu_R\}$. Then the rank of $\mathcal{A}$ is given by

$$\operatorname{rank}(\mathcal{A}) = \min\{\operatorname{rank}(\Phi_\mu(\mathcal{A})) - \operatorname{rank}(\Phi_{\mu-1}(\mathcal{A})), \operatorname{rank}(\Phi_\mu(\mathcal{A}^T) - \operatorname{rank}(\Phi_{\mu-1}(\mathcal{A}^T))\}.$$

Informally, the difference in rank between two block convolution matrices yields the rank because there exists a kernel basis where vectors are a cyclic shift of each other, and increasing the dimension of the block convolution matrix results in r additional kernel vectors, where $\mathcal{A}$ has rank $n-r$. The degree bounds are needed to account for the highest possible degree of the determinant. From this result it is straightforward to compute a kernel basis in $O(n^6d^3)$ FLOPs.

The degree bound on the kernel of $(n-r)d$ is tight in some settings, but in general, it is not the tightest possible. Regardless, the cost of computing $\ker(\Phi_\mu(\mathcal{A}))$ is $O(n^6d^3)$ FLOPs in the worst case. The advantage of this method over interpolation, is that interpolation methods also rely on the conditioning of a Vandermonde matrix or the computational cost of complex arithmetic to stabilize the Vandermonde matrices.

If $\mathcal{A} \in \mathsf{R}[t]^{n\times n}$ has degree d, then the matrix $\Phi_\mu(\mathcal{A})$ is similar to the block-Toeplitz matrix [114]

$$\Phi_\mu^{Top}(\mathcal{A}) = \begin{pmatrix} A_d & & & \\ A_{d-1} & A_d & & \\ \vdots & A_{d-1} & \ddots & \\ A_0 & \vdots & \ddots & A_d \\ & A_0 & & A_{d-1} \\ & & \ddots & \vdots \\ & & & A_0 \end{pmatrix} \in \mathsf{R}^{n(\mu+d+1)\times n(\mu+1)},$$

and satisfies the same spectral properties as $\Phi_\mu(\mathcal{A})$ since it differs only by row and column permutations. In fact, if we write $\boldsymbol{b} = \boldsymbol{b}_1 + t\boldsymbol{b}_j + \cdots + t^\mu \boldsymbol{b}_\mu$, where $\boldsymbol{b}_j \in \mathsf{R}^{n\times 1}$ then we have

$$\Phi_\mu^{Top}(\mathcal{A}) \begin{pmatrix} \boldsymbol{b}_\mu \\ \boldsymbol{b}_{\mu-1} \\ \vdots \\ \boldsymbol{b}_0 \end{pmatrix} = 0 \iff \mathcal{A}\boldsymbol{b} = 0 \iff \Phi_\mu(\mathcal{A})\,\mathrm{vec}(\boldsymbol{b}) = 0.$$

Now it is evident that there exist permutation matrices P_{row} and P_{col} such that

$$P_{row}\Phi_\mu^{Top}(\mathcal{A})P_{col} = \Phi_\mu(\mathcal{A}).$$

Thus for matrix analysis purposes the two matrices are equivalent, as they have the same rank and spectral information. Accordingly, most of the other properties hold as well.

By performing the LQ decomposition (without pivoting) on the columns of Φ_μ^{Top}, it is possible to obtain a kernel basis in $O(n^3 d \deg(\ker(\mathcal{A}))^3)$ FLOPs, which is $O(n^6 d^4)$ FLOPs in the worst case. It is argued [114] that the worst case is not typical in practical problems, and the performance is closer to $O(n^3 d)$ FLOPs in practice. Matrices involving a subset of the rows and columns of $\Phi_\mu^{Top}(\mathcal{A})$ appear in determining the Smith-McMillan form of rational functions [105] and accordingly can be used to obtain both finite and infinite spectral information from an associated matrix pencil in the instance of matrix polynomials. A modified LQ decomposition in this instance yields a kernel vector of minimal degree.

Matrix Pencil Methods

Another technique to compute the rank of a matrix polynomial is to linearize it into a pencil. This can be accomplished using the companion linearization or any other linearization. Rank-revealing factorizations, such as the QZ decomposition, can give us rank information in $O(n^3d^3)$ FLOPs or the well known "pencil method" can be used [8].

3.2.2 Kernel Basis via Block Convolution

A family of techniques to compute a kernel basis involve block convolution matrices. Block convolution matrices have the drawback that extracting rank information is not immediate, however it is straightforward to obtain a basis for the kernel of $\mathscr{A}$ from $\Phi_{nd}(\mathscr{A})$. Computing $\ker(\Phi_{nd}(\mathscr{A}))$ is done easily by the SVD or QR factorization (in $O(n^6d^3)$ FLOPs). Thus, the problem is reduced to given some kernel $K = \ker(\Phi_{nd}(\mathscr{A}))$, compute $\mathscr{K} = \ker(\mathscr{A})$ from the information in K. Note that any technique that triangularizes $\mathscr{A}$ can be used to give us a kernel basis, although these techniques may not be stable (due to a lack of pivoting). Additionally, using the smallest singular vectors from the SVD is a numerically robust way to compute an initial guess for our optimization problems.

Extracting a Primitive Kernel Vector

Computing a single kernel vector that is primitive is straightforward using block convolution matrices. One can triangularize K using orthogonal (or unitary if $\mathsf{R} = \mathbb{C}$) transformations (i.e. via Householder reflection on the columns of K, which is essentially a modified RQ or LQ factorization) into a quasi-triangular form that resembles

$$KQ^T = \underbrace{\left(\begin{array}{cccc} & & & * \\ & & \iddots & \vdots \\ & * & \cdots & * \\ * & * & \cdots & * \\ \hline * & * & \cdots & * \\ \vdots & \vdots & \cdots & * \\ * & * & \cdots & * \end{array}\right)}_{K_L} \quad \text{or} \quad \underbrace{\left(\begin{array}{cccc} * & * & \cdots & * \\ \vdots & \vdots & \cdots & \vdots \\ * & * & \cdots & * \\ \hline * & * & \cdots & * \\ \vdots & \vdots & \iddots & \\ * & * & & \\ * & & & \end{array}\right)}_{K_R}.$$

The first (left-most) column of K_L when de-vectorized will be $t^k \mathscr{b}_L$ where $\mathscr{b}_L$ is a primitive kernel vector coming from a Column Reduced Echelon Form (CREF) basis of K and $k =$

$\mu - \deg(\mathscr{b}_L)$, where μ is an upper-bound for the degree of the kernel (i.e. $\mu = nd$ generically). It is straightforward to divide by t^k to obtain the desired kernel vector. The last (right-most) column of K_R when de-vectorized will be $\mathscr{b}_R$ where $\mathscr{b}_R$ is a primitive kernel vector from a CREF basis of $\mathscr{K}$. Note that one could triangularize K using column operations to obtain similar vectors (i.e. Gaussian elimination with some form of pivoting), but in our implementation we use orthogonal (unitary) transformations for stability purposes.

Computing a Minimal Embedding

The kernel vectors $\mathscr{b}_L$ and $\mathscr{b}_R$ are special because they can be represented with the minimum number of (polynomial) equations to ensure that $\mathscr{A}\mathscr{b}_R = 0$ or $\mathscr{A}\mathscr{b}_L = 0$, since the high or low index entries are forced to be zero (thus one assumes several polynomial entries of $\mathscr{b}$ are known to be zero). For the purpose of optimization algorithms, kernel vectors of the form $\mathscr{b}_R$ or $\mathscr{b}_L$ satisfy regularity conditions and are locally unique (once higher index or higher order degree coefficients are constrained to be zero), thus making them desirable.

We note that

$$\mathscr{b}_L = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \mathscr{b}_{pivot} \\ \mathscr{b}_{pivot+1} \\ \vdots \\ \mathscr{b}_n \end{pmatrix} \quad \text{and} \quad \mathscr{b}_R = \begin{pmatrix} \mathscr{b}_1 \\ \vdots \\ \mathscr{b}_{pivot-1} \\ \mathscr{b}_{pivot} \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

The factorization reveals the index of $\mathscr{b}_{pivot}$ (i.e. what entry is the pivot) and the degree of $\mathscr{b}_{pivot}$. Accounting for the lower/higher index terms that are zero, this is the minimal number of polynomial equations needed to represent $\mathscr{A}\mathscr{b} = 0$ (which is important if $\operatorname{rank}(\mathscr{A}) \leq n-2$). If we force $\mathscr{b}_{pivot}$ to not increase in degree, then the vector will remain primitive if perturbed by a reasonable amount, and the degrees of the other entries are implicitly upper bounded as, the degree of the pivot reveals the degree of each entry. This is a constructive argument for writing $\mathscr{A}\mathscr{b} = 0$ using the minimum number of polynomial equations, thus useful for obtaining a normalization. Note that $\mathscr{b}_L$ and $\mathscr{b}_R$ need not agree, and in practice they will be different if there are at least two kernel vectors. If we delete columns of $\Phi_\mu(\mathscr{A})$ (to obtain a matrix $\Phi_\mu^{min}(\mathscr{A})$) that correspond to the zero rows of $\mathscr{b}_L$ or $\mathscr{b}_R$, then these will be the only possible kernel vectors for $\ker(\Phi_\mu^{min}(\mathscr{A}))$. Thus the vector $\mathscr{b}$ is embedded into a ground field linear algebra problem with the minimum number of equations, which we will sometimes call a *minimal embedding*.

This computation will cost roughly $O(n^6d^3)$ FLOPs in the worst case. Using approximate GCD techniques, one can verify that the computed kernel vector is numerically primitive or enforce this.

The triangularization is sufficient to obtain a kernel vector that is "locally unique" in a sense that there are no other kernel vectors with the same support (coefficient structure). If another kernel vector was chosen, call it $\mathscr{b}$, then we could (without loss of generality) have that $\mathscr{b} + \mathscr{b}_R\varepsilon$ is another kernel vector that is arbitrarily close with the same degree structure. In fact, we can always find a kernel vector $\mathscr{b}$ of the form $(1+\varepsilon t)\mathscr{b}_R$ where this occurs. This lack of "local uniqueness" is a hindrance to applying optimization methods on this formulation of the problem.

Another notable advantage of using block convolution matrices is that $\sigma_{\min}(\Phi_{nd}(\mathscr{A}))$ gives us a lower bound on the distance to a matrix polynomial with a non-trivial kernel, whereas the interpolation technique does not readily yield a nice lower bound. Using the smallest singular vector of $\Phi_{nd}(\mathscr{A})$ or $\Phi_{(n-1)d}(\mathscr{A})$ is a way to obtain an initial guess for an optimization routine to compute a nearby matrix polynomial that is singular.

Extracting an Entire Primitive Kernel

While computing a single kernel vector that is primitive (and locally unique) is straight forward, the techniques do not immediately generalize to computing the kernel in an efficient manner without some care.

Let $\mathscr{K}_1, \dots, \mathscr{K}_r \in \mathsf{R}[t]^{n\times 1}$ be primitive column vectors forming a basis for $\mathscr{K} = \ker(\mathscr{A})$ that are of minimal degree. Furthermore, let $d_j = \deg(\mathscr{K}_j) \leq nd$. The matrix

$$K_{lin} = \begin{pmatrix} \mathrm{vec}(\mathscr{K}_1) & \cdots & \mathrm{vec}(t^{nd-d_1}\mathscr{K}_1) & \cdots & \mathrm{vec}(\mathscr{K}_r) & \mathrm{vec}(t\mathscr{K}_r) & \cdots & \mathrm{vec}(t^{nd-d_r}\mathscr{K}_r) \end{pmatrix}$$

has a Toeplitz-block structure and $K_{lin} = \ker(\Phi_{nd}(\mathscr{A}))$. Using a variant of Gaussian elimination, one can transform K into K_{lin} and extract a kernel basis in this manner. If one knew that $\mathscr{A}$ has rank $n-r$, or more generally $\deg(\ker(\mathscr{A})) = \mu$, via some other means, then an analogous routine could be performed on $\Phi_\mu(\mathscr{A})$. Typically $\Phi_{nd}(\mathscr{A})$ is computed, but the ideas generalize in the obvious way for other degree bounds.

Computing a Minimal Kernel Embedding

For example it is useful for normalization purposes if $\mathcal{K}$ has primitive columns and is in a CREF, i.e. we can write when $\mathcal{K}$ has a generic rank profile,

$$\mathcal{K}_{CREF} = \begin{pmatrix} * & & \\ & \ddots & \\ & & * \\ \hline * & \cdots & * \\ \vdots & \cdots & \vdots \\ * & \cdots & * \end{pmatrix} \quad \text{or} \quad \begin{pmatrix} * & \cdots & * \\ \vdots & \cdots & \vdots \\ * & \cdots & * \\ \hline * & & \\ & \ddots & \\ & & * \end{pmatrix},$$

or in the case of a less generic form the CREF pivots would be in a different location. Note that the entries of $\mathcal{K}_{CREF}$ are polynomial. The CREF is unique up to scaling from a unit over $\mathsf{R}[t]$, so ensuring each column is primitive makes these (or similar) forms unique (up to a non-zero constant from R). Such a normalization can lead to coefficient or degree growth, so instead we can look at ensuring that the vectorized form of $\mathcal{K}$ is in a CREF (over R). If we assume kernel vectors have this form, then we can can delete columns of $\Phi_\mu(\mathcal{A})$ that correspond to entries forced to be zero and proceed in an analogous way to *minimally embed each column in a matrix*.

One could reduce K into CREF (or some analogous form where pivots are not 1, but normalized reasonably) to overcome this issue, however this is not as stable as RQ/LQ factoring. The important aspect is that the kernel vectors are linearly independent, and by adding a constraint on the kernel that it is (locally) in CREF, this ensures uniqueness of the kernel (up to scaling factors). This is done by forcing some coefficients of the kernel to be zero.

Regardless of the technique used to compute a primitive kernel basis (note that a kernel basis of minimal degree must be primitive), assuming one can compute K_{lin} efficiently and robustly, we can normalize

$$K_{min} = \begin{pmatrix} \mathrm{vec}(K_1) & \cdots & \mathrm{vec}(K_r) \end{pmatrix} \in \mathsf{R}^{n(\mu+1)\times r}$$

into a CREF over R (i.e. pivots have unit norm, columns have unit norm or something similar) that each column, when de-vectorized, remains primitive. The CREF constraints imply that the de-vectorized kernel $\mathcal{K}_{min}$ remains locally unique when coefficients are perturbed. In a local sense the constraints are not "deformed" under this normalization, and linear independence of kernel vectors is maintained. Now, we can minimally embed each vector of K_{min} into $\Phi_{\mu}(\mathscr{A})$ by deleting columns corresponding to zero entries. If we do this r times, then we will have r matrices with a different kernel vector that is minimally embedded.

3.2.3 Initial Guesses for Optimization Algorithms

The ideas for computing a kernel basis generalize to obtaining an initial guess for a kernel basis. If $\mathscr{A}$ is sufficiently close to a matrix polynomial of rank $n-r$ then there will be at least r singular vectors of $\Phi_{nd}(\mathscr{A})$ that are suitable candidates for an initial guess for a kernel. If $\mathscr{B}$ is an approximate kernel for $\mathscr{A}$ built from these singular vectors, then $\|\mathscr{A}\mathscr{B}\|_F$ will be reasonably small.

One popular technique for SLRA problems, like this one, is to find the nearest rank deficient matrix to $\Phi_{nd}(\mathscr{A})$ and project this to the nearest block convolution matrix. These type of techniques are known as lift and project or alternating projections, and they will generally converge (with a linear rate of convergence in the best case typically) to a fixed point, that need not always be a local minimizer. The kernel basis techniques are required here to actually verify that the computed initial guess for the kernel satisfies the rank criteria, so that it can be normalized, or that the initial guess has reasonable prospects to converge to a kernel with the desired properties.

A technique to obtaining a (nearby) rank deficient matrix polynomial is through linearization and the QZ decomposition. Linearizing a matrix polynomial into a matrix pencil and zeroing diagonal elements arising from the QZ decomposition (or a Schur decomposition) will produce a rank deficient matrix polynomial. The issue with the QZ and related decompositions is that one is almost sure to obtain a matrix with complex coefficients. Additionally, the nearest (or any nearby) rank deficient matrix pencil (or a lower rank approximation) is unlikely to preserve the structure of the linearization, thus extracting a rank deficient matrix polynomial from the pencil may not be possible (or the polynomial extracted may have no meaning in the context of the problem).

Another technique is to use evaluation and interpolation in conjunction with the SVD. The idea is to choose $nd+1$ distinct evaluation points $\{\omega_j\}_{j=0}^{nd}$ and perturb $\mathscr{A}(\omega_j)$ to the nearest singular (or rank $n-r$ matrix) via the SVD. Once the singular images are computed, one computes $\Delta\mathscr{A}(\omega_j)$ and then attempts to solve for $\Delta\mathscr{A}$ in a least squares sense. The issue here is if complex evaluation points are chosen, then one needs to project to a real matrix polynomial which will probably have full rank. If real evaluation points are used, then the images may be ill-conditioned due to multiplication by a Vandermonde matrix. The procedure can be iterated to improve the quality of the guess, as it's not expected that $\mathscr{A} + \Delta\mathscr{A}$ will be singular (but it will be relatively close to a singular matrix polynomial).

3.2.4 Summary of Rank Computing Techniques

We now briefly summarize the discussed techniques for computing the rank of a matrix polynomial. In Figure 3.1 we note the FLOPs required for computing the rank (but not the kernel) and whether

- they are numerically stable,
- if a kernel basis can be computed in a numerically stable manner,
- if the kernel basis routine is suitable for generating an initial guess to an optimization routine and
- if the initial guess is structure preserving.

Figure 3.1: Overview of Rank Computation Techniques

	FLOPs	Stable	Kernel Basis	Initial Guess	Structure Preserving
Exact	$\widetilde{O}(n^3d)$	No	No	No	
Pencil	$\widetilde{O}(n^3d^3)$	Yes	Yes	Sometimes	No
Interpolation	$\widetilde{O}(n^4d)$	Yes	Sometimes	Sometimes	No
Convolution	$\widetilde{O}(n^6d^3)$	Yes	Yes	Yes	Yes

The interpolation approach is fast but if used for an initial guess it is not structure preserving with respect to the original problem, and the kernel basis extracted may not be primitive or be normalized appropriately, requiring additional computational resources. Additionally, to build an optimization problem using interpolation there are $O(rn^2d)$ constraints arising from the $O(nd)$ images required.

The block convolution approach is the slowest in terms of the FLOPs required for rank and kernel operations, but it can be used to obtain a kernel basis in a stable manner. Furthermore, the kernel algorithms are easily adjusted to obtain an initial guess for an optimization problem that is structure preserving (alternating least squares or projections for example). Like interpolation, there are $O(rn^2d)$ constraints arising from the equation $(\mathcal{A} + \widetilde{\Delta}\mathcal{A})\mathcal{B} = 0$. Given that we prefer to work over $\mathsf{R} = \mathbb{R}$ instead of $\mathsf{R} = \mathbb{C}$, complex numbers are generally troublesome, but we include the discussion for completeness.

Using a Newton based constrained optimization method, the $O(rn^2d)$ constraints imply a per-iteration cost using standard matrix arithmetic of $(r^3n^6d^3)$ FLOPs. Using a Newton-like method, the interpolation and block convolution techniques will ultimately have a comparable per-iteration cost. Given this information, we use the block convolution formulation of the problem, although the interpolation approach is theoretically similar. In fact Henrion and Šebek [56] show that interpolation techniques and block convolution techniques are essentially the same, as interpolation techniques work on a permuted block convolution matrix for several operations.

3.3 Optimization Formulation Setup

Problem 3.3.1 (Lower Rank Approximation of Matrix Polynomials). *Given $\mathcal{A} \in \mathbb{R}[t]^{n\times n}$ non-singular of degree d and an integer $r \leq n-1$ determine $\Delta\mathcal{A} \in \mathbb{R}[t]^{n^2(d+1)\times 1}$, with $\boldsymbol{\Delta}(\cdot) = \boldsymbol{\Delta}_{entry}(\cdot)$, the structure of preserving the entry wise degree of $\mathcal{A}$, and r linearly independent vectors $\{b_k\}_{k=1}^r$ with $b_k \in \mathbb{R}[t]^{n\times 1}$, such that $\|\Delta\mathcal{A}\|$ is (locally) minimized, subject to the constraint that $(\mathcal{A} + \widetilde{\Delta}\mathcal{A})b_k = 0$ and $\|b_k\| = 1$.*

Note that this is minimizing a convex objective function subject to non-convex constraints. However, the equality constraints are linear in each argument. It is still not clear that Problem 3.3.1 is well-posed in the current form. We will prove that solutions exist, that is, there is an attainable global minimum value and not an infimum.

In this problem, $\widetilde{\Delta}\mathcal{A}(\cdot)$ converts $\Delta\mathcal{A} \in \mathbb{R}^{n^2(d+1)\times 1}$ into a (possibly padded with zero entries) matrix polynomial over $\mathbb{R}$ with dimension $n \times n$ and degree d. Recall that for unstructured perturbations of degree at most d, we can consider $\boldsymbol{\Delta}(\Delta\mathcal{A}) = \Delta\mathcal{A} \in \mathbb{R}[t]^{n\times n}$ where $\Delta\mathcal{A}$ has degree at most d, since $\boldsymbol{\Delta}$ behaves like an identity operator on the coefficients.

Writing $\mathcal{B} = (b_1, \ldots, b_r) \in \mathbb{R}[t]^{n\times r}$, Problem 3.3.1 becomes

$$\min \|\Delta\mathcal{A}\|_F^2 \text{ subject to } \begin{cases} (\mathcal{A} + \Delta\mathcal{A})\mathcal{B} = 0, \\ \operatorname{rank}(\mathcal{B}) = r, \\ \left\{\|b_k\|_2^2 = 1\right\}_{k=1}^r. \end{cases}$$

Lemma 3.3.2. *Given a non-singular $\mathscr{A} \in \mathbb{R}[t]^{n\times n}$, and $\widetilde{\Delta}\mathscr{A} \in \mathbb{R}[t]^{n\times n}$ such that $\mathscr{B} = \mathscr{A} + \widetilde{\Delta}\mathscr{A}$ is singular, it is the case that $\|\Phi_{nd}(\widetilde{\Delta}\mathscr{A})\|_2 \geq \sigma_{\min}(\Phi_{nd}(\mathscr{A}))$.*

Proof. This follows immediately by the SVD. □

Lemma 3.3.3. *The set of all matrices of rank at most $n-r$ over $\mathbb{R}[t]^{n\times n}$ of degree at most d is closed.*

Proof. This is a non-technical restatement of the previous lemma. □

Theorem 3.3.4 (Existence of Solutions)**.** *The minimization posed in Problem 3.3.1 has an attainable global minimum if $0 \in \operatorname{Range}(\boldsymbol{\Delta}(\cdot))$, i.e. $\boldsymbol{\Delta}(\cdot)$ is a linear structure.*

Proof. Let

$$\begin{aligned} S = &\left\{\mathscr{C} \in \mathbb{R}[t]^{n\times n} \mid \operatorname{rank}\mathscr{C} \leq n-r \wedge \deg\mathscr{C} \leq d\right\} \\ &\cap \left\{\mathscr{C} \in \mathbb{R}[t]^{n\times n} | \|\mathscr{C}\|_F^2 \leq \|\mathscr{A}\|_F^2\right\}. \end{aligned}$$

S is the intersection of a closed and bounded set and a closed set, hence S is closed and bounded. S is isomorphic to some closed and bounded subset of Euclidean space, hence by the Heine-Borel theorem, S is compact. To show the set is non-empty, we note that, by the degree assumption on $\widetilde{\Delta}\mathscr{A}$, $\widetilde{\Delta}\mathscr{A} = -\mathscr{A}$ is a feasible point independent of rank.

Let $\mathscr{C} \in S$ then $\|\mathscr{A} - \mathscr{C}\|_F^2 = \|\Delta\mathscr{A}\|_F^2$ is a continuous function over a compact set. By Weierstrass' theorem it has an attainable global minimum. □

It is important not to over-constrain the problem with a choice of $\widetilde{\Delta}\mathscr{A}$, since otherwise the feasible set might be empty. Another reasonable choice of $\boldsymbol{\Delta}(\cdot)$ which we can handle, is that the perturbation has the same coefficient structure/support as $\mathscr{A}$, that is, zero terms in polynomial entries are preserved. As long rank deficient matrices with the prescribed structure exist, a solution to the optimization problem exists.

We note that this result says nothing about uniqueness or separation of solutions or any local properties. All that has been shown is that if the perturbations are in the same space as the input, and one seeks a rank at most approximation, then there is an attainable global minimum value, i.e. not an infimum. If one wants a minimal solution with the rank being exactly $n-r$, then there is no guarantee that there is an attainable global minimum to Problem 3.3.1.

3.4 Rank Factorizations

A natural formulation of the problem that encompasses the rank implicitly is to perform a rank factorization and write $\mathcal{A} + \widetilde{\Delta}\mathcal{A} = \mathcal{U}\mathcal{V} \in \mathbb{R}[t]^{n\times n}$ for $\mathcal{U} \in \mathbb{R}[t]^{n\times(n-r)}$ and $\mathcal{V} \in \mathbb{R}(t)^{(n-r)\times n}$. Here $\mathcal{U}\mathcal{V}$ is subject to some constraints that preserve the structure of $\widetilde{\Delta}\mathcal{A}$ (i.e., that we do not perturb any coefficients we are not allowed to, that is $\mathcal{U}\mathcal{V} \in \text{Range}(\mathbf{\Delta}(\cdot))$. This is a non-linear least squares problem. However, solutions are not unique. Indeed, if $\mathcal{Z} \in \mathbb{R}[t]^{(n-r)\times(n-r)}$ is unimodular (i.e., $\det(\mathcal{Z}) \in \mathbb{R}\backslash\{0\}$), then $\mathcal{U}\mathcal{Z}$, $\mathcal{Z}^{-1}\mathcal{V}$ is another rank $n-r$ factorization, and we obtain an infinite family.

In this section we investigate some theoretical properties of rank factorizations and use them to derive separation bounds for the objective function. The rank factorization is a technique suitable for obtaining an initial guess for a Newton-like method.

While normalizing over matrix polynomial rank-factorizations is difficult, it is much easier to exploit the quasi-distance preserving property of $\|\cdot\|_F$ and look at rank-factorizations of $\Phi_{nd}(\mathcal{A})$, that do not necessarily correspond to $\mathcal{U}$ and $\mathcal{V}$.

Definition 3.4.1 (R-Embedding). *We say that $\widehat{A} \in \mathsf{R}^{M\times N}$ is an* R*-embedding of $\mathcal{A} \in \mathsf{R}[t]^{m\times n}$ if for $\mathcal{B} = \begin{pmatrix} b_1 & \cdots & b_r \end{pmatrix} \in \mathsf{R}[t]^{n\times r}$ and $\mathcal{X} \in \mathsf{R}[t]^{n\times r}$ we have*

$$\mathcal{A}\mathcal{B} = \mathcal{X} \iff \widehat{A}\begin{pmatrix} \text{vec}(b_1) & \cdots & \text{vec}(b_r) \end{pmatrix} = \begin{pmatrix} \text{vec}(\mathcal{X}_1) & \cdots & \text{vec}(\mathcal{X}_r) \end{pmatrix}.$$

Obviously $\Phi_{nd}(\mathcal{A})$ is an embedding of $\mathcal{A}$ with respect to kernel vectors of $\mathcal{A}$.

3.4.1 Embedded Rank Factorization

Definition 3.4.2. *Let $\mu = nd$, $M = (\mu + d)n$, $N = n\mu$ and $R > 0$. A rank factorization of $\Phi_\mu(\mathcal{A} + \widetilde{\Delta}\mathcal{A})$ is given by writing $\Phi_\mu(\mathcal{A} + \widetilde{\Delta}\mathcal{A}) = UV$ where $U \in \mathbb{R}^{M\times R}$ and $V \in \mathbb{R}^{R\times N}$ are arbitrary (unstructured) matrices over $\mathbb{R}$.*

If $\mathcal{A}$ is rank deficient, then $\Phi_\mu(\mathcal{A})$ will be rank deficient, and all rank deficient matrices admit a rank factorization. The matrix V implicitly parametrizes the kernel. Our goal is to find U, V with appropriate dimensions which minimize

$$\|\Phi_\mu(\Delta\mathcal{A})\|_F = \|\Phi_\mu(\mathcal{A}) - UV\|_F$$

and such that UV has the correct Toeplitz-block and coefficient structure (i.e., it is an $\mathbb{R}$-embedding of a matrix polynomial).

This is a problem with a non-convex objective function (that is convex in each argument) and non-convex constraints. We note that U and V have no direct connection with $\mathcal{U}$ and $\mathcal{V}$ discussed earlier.

One may always write $\Phi_\mu(\mathcal{A} + \widetilde{\Delta}\mathcal{A})$ this way via the SVD for fixed $\mathcal{A}$ and $\Delta\mathcal{A}$, so in particular the optimal solution can be written as a rank factorization. The problem $\min \|\Phi_\mu(\mathcal{A}) - UV\|_F^2$ such that UV has the same structure as $\Phi_\mu(\widetilde{\Delta}\mathcal{A})$ for some $\Delta\mathcal{A}$ is generally ill-posed and needs to be constrained to do any meaningful analysis, as there are numerous degrees of freedom. At first glance, optimizing over rank factorizations appears to be a harder problem than the original. However, it is helpful to perform analysis on this formulation. In particular, we are able to prove that optimal values of $\Delta\mathcal{A}$ are separated by a constant amount, and that equivalence classes of solutions are isolated. Additionally, this formulation of the problem is convex in each argument (but not jointly convex) and is amenable to block coordinate descent methods.

We next need to demonstrate that the condition that the matrix $\Delta\widehat{A} = \Phi_\mu(\mathcal{A}) - UV$ is the $\mathbb{R}$-embedding of some matrix polynomial $\widetilde{\Delta}\mathcal{A} \in \mathbb{R}[t]^{n\times n}$ can be phrased as a single polynomial being zero. $\Phi_\mu(\mathcal{A})$ is generated by a linear structure $\sum_{i=1}^{L} c_i \widehat{A}^{(i)}$ where $c_i \in \mathbb{R}$ and $\{\widehat{A}^{(1)}, \dots, \widehat{A}^{(L)}\} \subseteq \mathbb{R}^{M\times N}$. Define the structural enforcement function

$$\Gamma : \mathbb{R}^{M\times R} \times \mathbb{R}^{R\times N} \to \mathbb{R} \text{ as } \Gamma(U,V) = \left\| \sum_{i=1}^{L} c_i \widehat{A}^{(i)} - \Delta\widehat{A} \right\|_F^2 .$$

We note that there exist c_i such that $\Gamma(\Delta\widehat{A}) = 0$ if and only if $\Delta\widehat{A}$ is an $\mathbb{R}$-embedding of a matrix polynomial. The idea can be generalized in a straightforward way for arbitrary $\boldsymbol{\Delta}(\cdot)$.

Problem 3.4.3. *With $\mathcal{A}, U, V$ as above, the constrained $\mathbb{R}$-embedded rank factorization problem consists of computing $\min \|\widehat{A} - UV\|_F^2$ subject to the constraints that $U^T U - I = 0$ and $\Gamma(U,V) = 0$ where $\widehat{A} = \Phi_\mu(\mathcal{A})$. If $R = M - 1$, then this encodes all rank deficient matrix polynomials.*

It is still not clear that Problem 3.4.3 is well-posed, as there are many degrees of freedom in V, and this matrix can have arbitrary rank. The enforcement of U as an orthogonal matrix ($U^T U - I = 0$) is allowed for without loss of generality. Informally, we are looking at all rank factorizations where U is orthogonal and $\Gamma(U,V) = 0$, that is, the product satisfies the block-Toeplitz structure on $\Delta\widehat{A}$.

We employ the machinery of non-linear optimization to describe the geometry of the minimal solutions, and hence the nearest appropriately structured matrices.

Lemma 3.4.4. *Suppose that $\Delta\widehat{A}$ and $\Delta\widehat{A}^\star$ are distinct local solutions of minimal rank to Problem 3.3.1, then*

$$\|\Delta\widehat{A} - \Delta\widehat{A}^\star\|_2 > 0.$$

Proof. First we note that

$$\|\Delta\widehat{A} - \Delta\widehat{A}^\star\|_2 = \|\widehat{A} + \Delta\widehat{A} - \widehat{A} - \Delta\widehat{A}^\star\|_2.$$

For any $x \in \ker(\widehat{A} + \Delta\widehat{A})$ that satisfies $\|x\|_2 = 1$ and $x \notin \ker(\widehat{A} + \Delta\widehat{A}^\star)$ then

$$\begin{aligned}\|\widehat{A} + \Delta\widehat{A} - \widehat{A} - \Delta\widehat{A}^\star\|_2 &\geq \|(\widehat{A} + \Delta\widehat{A})x - (\widehat{A} + \Delta\widehat{A}^\star)x\|_2\\ &= \|(\widehat{A} + \Delta\widehat{A}^\star)x\|_2\\ &> 0,\end{aligned}$$

which follows from the fact that given $A \in \mathbb{C}^{M\times N}$ and $x \in \mathbb{C}^{N\times 1}$ we recall that for $x \neq 0$,

$$\|A\|_2 = \max_{x^Tx=1} \|Ax\|_2 = \max_{x\neq 0} \frac{\|Ax\|_2}{\|x\|_2} \geq \frac{\|Ax\|_2}{\|x\|_2}. \qquad \square$$

The best x would be one that is orthogonal to $\ker(\widehat{A} + \Delta\widehat{A}^\star)$, although this may not always be possible to find. Instead, the best x to generate a lower bound can be derived by formulating a Procrustes problem where one seeks the x that is "closest" to being orthogonal to the kernel. This is only useful in theory though, as the bound depends on the solution, which we do not know in advance.

We note that $\ker(\widehat{A} + \Delta\widehat{A}) \neq \ker(\widehat{A} + \Delta\widehat{A}^\star)$, since if their kernels were the same, then this implies that the solutions are not distinct. In such a scenario we could write $K = \ker(\widehat{A} + \Delta\widehat{A}) = \ker(\widehat{A} + \Delta\widehat{A}^\star)$. The total least squares solution to $(\widehat{A} + \Delta\widehat{A})K = 0$ for a fixed K and variable $\Delta\widehat{A}$ is unique, since it is a linear system of equations in the entries of $\Delta\widehat{A}$. For the instance of arbitrary affine structures (assuming a solution exists) then the solution can be normalized by other means, since it is a convex optimization problem when K is fixed. Of course the bound from Lemma 3.4.4 is *unstructured*, but is adequate to show that solutions are isolated. be used in instances where the separation theorem is not particularly insightful. In general rank reducing perturbations of minimal norm do not have rank one. We characterize when this occurs by analyzing the KKT conditions of a different formulation of the problem in Section 3.6.2.

Corollary 3.4.5. *All locally optimal solutions of minimal rank are isolated modulo equivalence classes.*

Proof. Suppose the contrary, that is that $(U, V) \in \mathbb{R}^{M\times R}\times\mathbb{R}^{R\times N}$ is a solution corresponding to $\Delta\widehat{A}$ and $(U^\star, V^\star) \in \mathbb{R}^{M\times R} \times \mathbb{R}^{R\times N}$ is a solution corresponding to $\Delta\widehat{A}^\star$. Let c_{sep} be the best lower bound from Lemma 3.4.4. The objective function and constraints are locally Lipschitz continuous, so let $s > 0$ be a Lipschitz constant with respect to $\|\cdot\|_F$ in some open neighborhood.

If we take $0 < \varepsilon < \frac{c_{sep}}{s}$ such that $\left\| \begin{pmatrix} U \\ V \end{pmatrix} - \begin{pmatrix} U^\star \\ V^\star \end{pmatrix} \right\|_F < \varepsilon$ then we have that

$$\begin{aligned} c_{sep} &\leq \|\Delta\widehat{A} - \Delta\widehat{A}^\star\|_2 \\ &\leq s\left\| \begin{pmatrix} U \\ V \end{pmatrix} - \begin{pmatrix} U^\star \\ V^\star \end{pmatrix} \right\|_F \\ &\leq s\varepsilon \\ &< c_{sep}, \end{aligned}$$

which is a contradiction to Lemma 3.4.4. □

Implicitly the matrix V parametrizes the kernel of $\widehat{A}$. If we normalize the kernel of $\widehat{A}$ to contain $\mathbb{R}$-embeddings of primitive kernel vectors then the matrix V can be made locally unique, although we do not employ this in the rank-factorization formulation directly. What this result says is that solutions in the (U, V) coordinates are only arbitrarily close to each other if they correspond to the same minimal perturbation (because V is not normalized, solutions in both coordinates are not necessarily unique).

While there are too many degrees of freedom to easily obtain a (locally) quadratically convergent minimization over the rank factorization, the rank factorization does yield non-trivial insights into the geometry of the solution space. In particular, the isolation of solutions indicates first-order (gradient) methods will perform well on the problem. In the next section we will introduce a locally quadratically convergent algorithm for an equivalent form of Problem 3.3.1 that reduces each equivalence class of solutions to a single solution.

3.4.2 Lagrange Multipliers and Optimality Conditions

While Section 3.4.1 discusses solutions to the problem of computing the nearest singular matrix polynomial, it is not particularly insightful into the instance of a lower-rank approximation. The issue is that a lower rank approximation of $\widehat{A} + \Delta\widehat{A}$ may correspond to a rank deficient $\mathscr{A} + \Delta\mathscr{A}$ that is not adequately rank deficient. We recall that this occurs because there exists a basis of kernel vectors where each kernel vector is a cyclic shift of some of the other kernel vectors. To overcome this problem, we can iteratively build lower rank approximations for different values of R by computing a local solution to the optimization problem for each value of R. This avoids the tricky issue of normalizing V and choosing an embedding that is "minimal" with respect to the input.

We can proceed to solve the optimization problem

$$\min \|\Delta\mathscr{A}\|_F^2 \text{ subject to } \widehat{A} + \Delta\widehat{A} - UV = 0, \tag{3.1}$$

directly by using Newton's method, alternating directions or alternating optimizations, since the optimization problem is convex in U and V (but not jointly convex). Alternating least squares (or a similar block Gauss-Seidel type routine) is particularly straightforward to implement as an initial guess routine, although in our implementation we will use the SVD and post-refine with Newton's method[2] to compute a feasible point for an initial guess.

The bi-linearity of the problem means that the curvature of the constraints is constant, that is, the second-order derivative of the constraints is a constant tensor. Accordingly, if we normalize U and V (and eliminate some redundant constraint equations), we can expect the second-order sufficient conditions to hold at a solution, although we will not do this, since we generally satisfy a constant rank constraint qualification (which implies that Newton-like methods will have rapid local convergence with a suitable initial guess). In general, the problem can be normalized to ensure a constraint qualification is satisfied. Later in this chapter we will show that another formulation of the problem satisfies the second-order sufficient conditions. Accordingly due to the regularity conditions holding, we will generally have *local quadratic convergence* with several Newton-like methods.

We note that there are two important parameters that the user can prescribe;

1. The first parameter is μ, which is the degree bound for some of the kernel vectors of a solution. We can take $\mu = (n-r)d$ where r is the desired rank reduction of $\mathscr{A}$. This is because if $\operatorname{rank}(\mathscr{A}) = n - r$, then kernel vectors of minimum degree satisfy $\deg(\ker(\mathscr{A})) \leq \mu$.

2. The other parameter is R, which is an upper bound on the rank of the sought embedding. If the embedding is minimal, then it is easy to see that taking $R = n - r$

[2]We use a regularized Gauss-Newton method to solve $F(x) = 0$, we do not use "Newton's method for optimization".

will produce the desired result. Unfortunately, computing a minimal embedding may not be practical without a very good initial guess. To rectify this, independent of the choice of μ, we can compute a local solution $\widehat{A}+\Delta\widehat{A}^\star$ that corresponds to $\mathcal{A}+\Delta\mathcal{A}^\star$. If $\operatorname{rank}(\mathcal{A}+\Delta\mathcal{A}^\star) \leq n-r$, then we are done. Otherwise, $\operatorname{rank}(\mathcal{A}+\Delta\mathcal{A}^\star) > n-r$ and we can compute a new embedding of rank at most $R-1$, using $\mathcal{A}+\Delta\mathcal{A}^\star$ as an initial guess. We can proceed iteratively by reducing R on each computed solution if it is not adequate. Note that we can choose $R^{next} = \operatorname{rank}(\widehat{A}+\Delta\widehat{A}^\star) - 1$. The quantity $R - R^{next}$ depends implicitly on μ.

For our implementation we choose to implement a Newton-like method to solve $\nabla L = 0$, where

$$L = \|\Delta\mathcal{A}\|_F^2 + \lambda^T \operatorname{vec}(\widehat{A}+\Delta\widehat{A} - UV).$$

Recall that L is the Lagrangian and λ is a vector of Lagrange multipliers. Note that there are $O(NM) = O(n^2\mu^2) = O(n^4d^2)$ Lagrange multipliers in the worst case. The size of this problem is large because U and V require several orders of magnitude more variables than the size of $\mathcal{A}$.

Remark 3.4.6. *When U and V have full rank at a solution to (3.1) then the problem can be normalized so that Lagrange multipliers exist. The Jacobian of the constraints is (up to permutation)*

$$J = \nabla \operatorname{vec}(\widehat{A}+\Delta\widehat{A} - UV) = \begin{pmatrix} \nabla_{\Delta\mathcal{A}} \operatorname{vec}(\widehat{A}+\Delta\widehat{A}) & -I\otimes U & -V^T \otimes I \end{pmatrix}. \tag{3.2}$$

If U and V have locally constant rank in some open neighborhood around a solution, then J will have (or can be normalized) to have locally constant rank. The expression $\widehat{A}+\Delta\widehat{A} - UV = 0$ has prescribed zero entries, which means that $\nabla_{\Delta\mathcal{A}} \operatorname{vec}(\widehat{A}+\Delta\widehat{A}) = \dfrac{\partial \operatorname{vec}(\widehat{A}+\Delta\widehat{A})}{\partial \operatorname{vec}(\Delta\mathcal{A})}$ is a constant matrix. If R and μ are chosen to be "minimal" then, regularity conditions will hold (either the constant rank or linearly independent constraint qualification will hold).

Furthermore, if R and μ are not minimal, Lagrange multipliers will exist because one can always set some multipliers to zero to obtain the minimal solution.

3.4.3 The Hessian

We recall that

$$\nabla^2 L = \begin{pmatrix} \nabla^2_{xx} L & J^T \\ J & 0 \end{pmatrix},$$

where J is defined as in (3.2). So, it remains to compute $\nabla^2_{xx}L$.

Recall that $U \in \mathbb{R}^{M\times R}$, $V \in \mathbb{R}^{R\times N}$ and $\lambda \in \mathbb{R}^{NM}$ (with some entries possibly fixed to zero). We note that

$$\nabla^2_{xx}L = \begin{pmatrix} 2I & 0 & 0 \\ 0 & 0 & \dfrac{\partial}{\partial \operatorname{vec}(V)}(V \otimes I_M)\lambda \\ 0 & \dfrac{\partial}{\partial \operatorname{vec}(V)}\lambda^T(V^T \otimes I_M) & 0 \end{pmatrix}.$$

Lemma 3.4.7. *The matrix $\nabla^2_{xx}L$ has closed-form expression*

$$\begin{pmatrix} 2I & 0 & 0 \\ 0 & 0 & E \\ 0 & E^T & 0 \end{pmatrix} \quad \textit{where} \quad E = (\lambda^T \otimes I_{MR})(I_M \otimes \boldsymbol{K}_{M,R} \otimes I_N)(I_{NR} \otimes \operatorname{vec}(I_M)).$$

Proof. By symmetry, it is adequate to compute the expression

$$\begin{aligned}\frac{\partial}{\partial \operatorname{vec}(V)} \operatorname{vec}(\lambda^T(V^T \otimes I_N)) &= \frac{\partial}{\partial \operatorname{vec}(V)}(\lambda^T \otimes I_{MR}) \operatorname{vec}(V \otimes I_M) \\ &= (\lambda^T \otimes I_{MR})(I_N \otimes \boldsymbol{K}_{M,R} \otimes I_M)\frac{\partial}{\partial \operatorname{vec}(V)}(I_{NR} \otimes \operatorname{vec}(I_M)) \operatorname{vec}(V) \\ &= (\lambda^T \otimes I_{MR})(I_M \otimes \boldsymbol{K}_{M,R} \otimes I_N)(I_{NR} \otimes \operatorname{vec}(I_M)).\end{aligned}$$

□

Thus, we now have a closed-form expression for the Hessian matrix that reveals the underlying sparsity and kernel structure. The mixed partials are not generally square, so $\nabla^2_{xx}L$ will not have full rank (but the rank will generally be locally constant). In general the second-order sufficient condition will not hold for this formulation of the problem (however the problem can be modified so that the second-order sufficient condition will hold).

Together Remark 3.4.6 and Lemma 3.4.7 have the following corollary.

Corollary 3.4.8. *Suitable Newton-like methods to solve $\nabla L = 0$ will have local quadratic convergence with a suitable initial guess when R is minimal with respect to the parameter μ.*

Note that one such Newton-like method is discussed in detail in Chapter 5.4.3. Alternatively one may employ a Newton-like method that resembles the method of Wright [109] in this instance, as we do later in this chapter. To obtain quadratic convergence we do not need to ensure that μ is minimal, but we do need to ensure that R is minimal, otherwise U and V can be rank deficient. Reducing R implies that μ *can be reduced*, but this is not necessary, however this improvement reduces the size of the problem.

To compute the Hessian efficiently, we note that $\nabla_{xx} L$ is a linear function in λ, and J is a linear function in U and V. J is straightforward to compute since

$$\frac{\partial}{\partial \operatorname{vec}(\Delta \mathcal{A})} \operatorname{vec}(\widehat{A} + \Delta \widehat{A})$$

is a constant and only needs to be computed once. $I \otimes U$ and $V^T \otimes I$ are straightforward to compute. The block $\nabla^2_{xx} L$ is a linear function in λ. Thus, computing

$$\frac{\partial}{\partial \operatorname{vec}(V)} \operatorname{vec}(V \otimes I) \quad \text{or the analogous expression} \quad \frac{\partial}{\partial \operatorname{vec}(U)} \operatorname{vec}(I \otimes U^T)$$

needs to be done once, and then it may be scaled by $(\lambda^T \otimes I)$ to obtain the appropriate block in $\nabla^2_{xx} L$.

Note that computing $(I_M \otimes \boldsymbol{K}_{M,R} \otimes I_N)$ naively[3] is generally unwise, as this is a $M^2RN \times M^2RN$ matrix, where as the Hessian has dimension $O(MNR) \times O(MNR)$. We carefully note that $(I_M \otimes \boldsymbol{K}_{M,R} \otimes I_N)$ is a permutation matrix (hence has M^2NR non-zero entries), thus a careful application of the requisite permutation can be done quickly.

Given that L is a tri-linear scalar function, automatic differentiation is a reasonable choice if the procedure is modified to exploit the Kronecker product block structure to reduce the number of Lagrangian computations required.

3.4.4 Implementation Notes

Recall that $\widehat{A} \in \mathbb{R}^{M \times N}$ where $N, M \in O(n^2 d)$, so when $R \in \Theta(N)$ then U and V will have $O(n^4 d^2)$ variables. This means that a Newton method that relies of matrix inversion (or non-fast linear system solving) would require $\Omega(n^{12} d^6)$ FLOPs to compute a rank at most $n-1$ approximation of $\mathcal{A}$.

[3] Assuming one uses $(V \otimes I)$ in lieu of $(I \otimes U^T)$ to compute the second-order mixed partials. The quantities are continuously differentiable so the mixed partials must agree.

However, if R is large, then the rank of $\mathcal{A}$ is small. In the instance when $n-r \in o(n)$, i.e. we want a *very low rank approximation* of $\mathcal{A}$ then U and V require $O(n\mu \times R)$ variables. As noted earlier, if μ is minimal, then $R \approx n-r$, so the number of variables needed for U and V is $O(n^2 d(n-r))$. When $r \approx n$ then U and V require $O(n^2 d)$ variables, which is asymptotically the same as $\Delta\mathcal{A}$. Thus, for very low rank approximations, the the runtime is more accurately reflected as $\Omega(n^6 d^3)$ FLOPS if the technique relies on naive matrix inversion. Now, to account for the Lagrange multipliers, if $n \approx r$ then $N \approx n$ and $M \approx n$ so the number of Lagrange multipliers will be proportional to $n^2 d$ instead of $n^4 d^2$, thus a suitable Newton-like method will require $O(n^6 d^3)$ FLOPS, and a variant of Newton's method relying on standard linear system solving or matrix inverse would require $\Theta(n^6 d^3)$ FLOPs. In other words, the rank factorization will be quasi-optimal (with respect to naive matrix inversion) for very low rank factorizations, and perform relatively poorly for approximations of relatively high rank (but not full).

In general, we will use a Newton-like method to solve

$$\begin{pmatrix} x^{k+1} \\ \lambda^{k+1} \end{pmatrix} = \begin{pmatrix} x^k + \Delta x^k \\ \lambda^k + \Delta\lambda^k \end{pmatrix} \text{ such that } \nabla^2 L \begin{pmatrix} \Delta x \\ \Delta\lambda \end{pmatrix} = -\nabla L. \tag{3.3}$$

The quantities in question will generally have constant rank, so a Newton iteration is typically well-defined in a local sense with a feasible initial guess.

3.5 Evaluated Rank Factorization

A technique analogous to the previous section is to perform several rank factorizations on $nd+1$ images simultaneously. If $\operatorname{rank}((\mathcal{A}+\Delta\mathcal{A})(\omega_j)) \leq n-r$ for $nd+1$ distinct ω_j then $\mathcal{A}+\Delta\mathcal{A}$ must have rank $n-r$.

The following optimization problem yields a rank $n-r$ approximation to $\mathcal{A} \in \mathsf{R}[t]^{n\times n}$ of degree at most d is

$$\min \|\Delta\mathcal{A}\|_F^2 \text{ subject to } \begin{cases} \{(\mathcal{A}+\widetilde{\Delta\mathcal{A}})(\omega_j) = U_j V_j\}_{j=0}^{nd}, \\ U_j \in \mathsf{R}^{n\times(n-r)}, \\ V_j \in \mathsf{R}^{(n-r)\times n}. \end{cases} \tag{3.4}$$

For the analysis in this section we will take $\mathsf{R} = \mathbb{R}$. The evaluation points ω_j can be chosen as real numbers or complex numbers, irrespective of R. Complex roots of unity are a suitable choice because of the Fast Fourier Transform (FFT) and the useful property that the largest terms appearing will be on the order of $O((d+1)\|\operatorname{vec}(\mathscr{A}+\Delta\mathscr{A})\|_\infty)$ instead of $O(d|\omega|^d\|\operatorname{vec}(\mathscr{A}+\Delta\mathscr{A})\|_\infty)$. An alternative strategy is to employ a linearization, such as the companion linearization. If only some parts of the arising pencil are perturbed (corresponding to the input), then the exponential scaling is removed in exchange for a larger problem.

3.5.1 Lagrange Multipliers and Optimality Conditions

The theory of Lagrange multipliers and optimality conditions for this section is remarkably similar to that of Section 3.4.2. Since all of the quantities are constant. The $\operatorname{vec}(\cdot)$ operator will not pad zero entries for this analysis, again, since the images are all matrix polynomials of degree zero. For ease of analysis we will assume that real evaluation points are used. The ideas transfer over in a straightforward manner for complex evaluation points.

For our implementation, we choose to implement a Newton-like method to solve $\nabla L = 0$, where

$$L = \|\Delta\mathscr{A}\|_F^2 + \lambda^T \begin{pmatrix} \operatorname{vec}((\mathscr{A}+\widetilde{\Delta}\mathscr{A})(\omega_0) - U_0V_0) \\ \operatorname{vec}((\mathscr{A}+\widetilde{\Delta}\mathscr{A})(\omega_1) - U_1V_1) \\ \vdots \\ \operatorname{vec}((\mathscr{A}+\widetilde{\Delta}\mathscr{A})(\omega_{nd}) - U_{nd}V_{nd}) \end{pmatrix}.$$

Remark 3.5.1. *There exist $\{U_j\}_{j=0}^{nd}$ and $\{V_j\}_{j=0}^{nd}$ of full rank at a solution to (3.4) where Lagrange multipliers exist.*

Let $Z_j = \begin{pmatrix} -I\otimes U_j & -V_j^T\otimes I \end{pmatrix}$ be a block matrix. The Jacobian matrix of the constraints may be written (up to permutation) as

$$J = \begin{pmatrix} \nabla(\operatorname{vec}(\mathscr{A}+\widetilde{\Delta}\mathscr{A})(\omega_0)) & Z_0 & & & \\ \nabla(\operatorname{vec}(\mathscr{A}+\widetilde{\Delta}\mathscr{A})(\omega_1)) & & Z_1 & & \\ \vdots & & & \ddots & \\ \nabla(\operatorname{vec}(\mathscr{A}+\widetilde{\Delta}\mathscr{A})(\omega_{nd})) & & & & Z_{nd} \end{pmatrix}. \qquad (3.5)$$

The matrices $\nabla(\operatorname{vec}(\mathscr{A}+\widetilde{\Delta}\mathscr{A})(\omega_j))$ are constant, and the blocks consisting of the Z_j are clearly linearly independent. If each block $\begin{pmatrix}\nabla(\operatorname{vec}(\mathscr{A}+\widetilde{\Delta}\mathscr{A})(\omega_j)) & 0 & Z_j\end{pmatrix}$ has locally constant rank, then J will have locally constant rank, thus Lagrange multipliers will exist via the constant rank constraint qualification.

The matrices U_j and V_j^T both have rank $n - r$ (they have n rows and $n - r$ linearly independent columns), and can be normalized so that the rank of each row block is constant. If the U_j and V_j did not have rank $n - r$, then the rank could not be locally constant.

In fact, if some U_j or V_j did not have full rank, then we could perform a compression that reduces the number of columns of U_j and rows of V_j, to obtain matrices that have full rank. This statement is essentially a corollary of applying mathematical induction to Lemma 3.4.6.

If complex evaluation points are used then Lagrange multipliers also exist.

Remark 3.5.2. *An analogous expression to (3.5) exists after one separates the complex and real parts from the equation*

$$\Re((\mathcal{A} + \widetilde{\Delta}\mathcal{A})(\omega_j)) = \Re(U_j V_j) \quad \textit{and} \quad \Im((\mathcal{A} + \widetilde{\Delta}\mathcal{A})(\omega_j)) = \Im(U_j V_j).$$

Accordingly, Lagrange multipliers will generally exist when complex evaluation points are used.

3.5.2 The Hessian

The Hessian matrix of the evaluated rank factorization is a generalization of the Hessian matrix appearing in Section 3.4.3. For ease of analysis we will suppose that real evaluation points are used.

The Hessian matrix is essentially the same as the one appearing in Lemma 3.4.7, except that there are now several more blocks that are decoupled.

Lemma 3.5.3. *Let $\boldsymbol{\lambda}_j$ be a vector of the Lagrange multipliers corresponding to the constraint*

$$\operatorname{vec}((\mathcal{A} + \widetilde{\Delta}\mathcal{A})(\omega_j) - U_j V_j) = 0.$$

The matrix $\nabla_{xx}^2 L$ may be written (up to permutation) as

$$\begin{pmatrix} 2I & & & & \\ & F_0 & & & \\ & & F_1 & & \\ & & & \ddots & \\ & & & & F_{nd} \end{pmatrix} \quad \textit{where} \quad F_j = \begin{pmatrix} 0 & E_j \\ E_j^T & 0 \end{pmatrix} \quad \textit{and} \quad E_j = \frac{\partial}{\partial \operatorname{vec}(V_j)} (V_j \otimes I)\boldsymbol{\lambda}_j.$$

Proof. This follows by applying mathematical induction on Lemma 3.4.7. □

It should be noted that a closed-form expression for the Hessian can be derived by applying Lemma 3.4.7 on each block F_j.

The Hessian matrix for the evaluated rank projection is sparse under the assumption that $\nabla_{\Delta\mathscr{A}} \operatorname{vec}(\mathscr{A} + \widetilde{\Delta}\mathscr{A}(\omega_j))$ is sparse. This condition holds for several perturbation structures we care about, such as when perturbations are unstructured. This sparsity pattern may be exploited for fast linear system solving. If complex evaluation points are used, then the Hessian will consist of diagonal blocks that correspond to the real part and the imaginary part and the analysis is similar, and thus omitted. Complex evaluation points result in a system that is roughly four times as large.

3.5.3 Implementation Notes

There are $O(nd)$ matrix factorization constraints, each of which has $O((n-r)^2)$ variables and $O(n^2)$ constraints assuming the kernel has rank r. The total number of variables is $O(n^2d + nd(n-r)^2) = O(n^2d + n^3d)$. There are $O(n^3d)$ Lagrange multipliers, arising from the n^2 constraints on $nd+1$ images. A reasonable Newton method would require $O(n^6d^3 + n^3d^3(n-r)^6 + n^9d^3) = O(n^9d^3)$ FLOPs per iteration, which is a substantial improvement over the embedded rank factorization. This improvement comes almost exclusively because instead of having a rank factorization of one large matrix, we have rank factorizations of $O(nd)$ small matrices, which reduces the number of variables needed from $O(n^4d^2)$ to $O(n^3d)$ in the worst case. Rapid local convergence can be expected, since the Jacobian will have (or can be normalized) to have (locally) constant rank under these assumptions. An initial guess for U_j and V_j is easily obtained from applying a truncated SVD on $(\mathscr{A} + \widetilde{\Delta}\mathscr{A})(\omega_j)$ and projecting to a feasible point, and performing a row/column compression if necessary. The matrices J and $\nabla^2_{xx}L$ are sparse, in that $\nabla^2 L$ has $O(n^4d)$ non-zero entries but has dimension $O(n^3d) \times O(n^3d)$. A fast iterative solver can reduce the per-iteration cost to $O(n^7d^2)$ FLOPs (almost quadratic in the output size), since computing the Hessian can be done in $O(n^5d^2 + n^6d)$ FLOPs.

In general, we will use a Newton-like method to solve

$$\begin{pmatrix} x^{k+1} \\ \lambda^{k+1} \end{pmatrix} = \begin{pmatrix} x^k + \Delta x^k \\ \lambda^k + \Delta\lambda^k \end{pmatrix} \quad \text{such that} \quad \nabla^2 L \begin{pmatrix} \Delta x \\ \Delta\lambda \end{pmatrix} = -\nabla L. \tag{3.6}$$

The quantities in question will generally have constant rank, so a Newton iteration is typically well-defined in a local sense with a feasible initial guess.

A lurking detail in a software implementation that is irrelevant in theory is the precision used to compute ω_j and $\mathcal{A}(\omega_j)$. The highest possible precision of a solution depends on how accurately the quantities ω_j and $\mathcal{A}(\omega_j)$ are computed. While the system is sparse and in theory may be solved quickly using an iterative solver, this technique may fail to compute answers to extremely high accuracy because of errors arising from computing powers of ω_j. To overcome the precision issue, instead of using roots of unity, one could post-refine the roots of unity to be complex numbers *close* to roots of unity. The danger is that evaluation points may become excessively large in magnitude or be perturbed to the point that they are indistinguishable from each other.

3.6 Explicit Kernel Iterative Algorithm for Lower Rank Approximation

In this section we propose an iterative algorithm to solve Problem 3.3.1 based on Newton's method for constrained optimization. Sufficient conditions for quadratic convergence are that the second-order sufficiency condition holds [109] and local Lipschitz continuity of the objective and constraints. We ensure these conditions hold for non-degenerate problems by working on a restricted space of minimal $\mathbb{R}$-embeddings that remove degrees of freedom.

3.6.1 Minimal System of Equations

In order to compute a nearby rank $n-r$ approximation we want to solve the non-convex optimization problem

$$\min \|\Delta\mathcal{A}\|_F^2 \text{ subject to } \begin{cases} (\mathcal{A}+\widetilde{\Delta}\mathcal{A})\mathcal{B}=0, \\ \operatorname{rank}(\mathcal{B})=r. \end{cases} \tag{3.7}$$

In the instance of (structured) scalar matrices the rank constraint can be enforced by ensuring that $\mathcal{B}$ has orthogonal columns[4] or is in a CREF. In the instance of matrix polynomials this is not sufficient, since polynomial multiples of the same vector will have linearly independent combined coefficient vectors. In order to apply these normalizations on the coefficient vectors of $\mathcal{B}$ we require that the columns be represented with a minimal number of equations with respect to $\mathcal{B}$.

[4]This normalization alone is not sufficient for rapid convergence.

Definition 3.6.1 (Minimal $\mathbb{R}$-Embedding). *Suppose $\mathcal{A} \in \mathbb{R}[t]^{n\times n}$ with $\mathbb{R}$-embedding $\widehat{A}$. The vector $\mathscr{b} \in \mathbb{R}[t]^{n\times 1}$, with $\mathbb{R}$-embedding $\widehat{b} = \operatorname{vec}(\mathscr{b})$, is said to be* minimally $\mathbb{R}$-embedded *in $\widehat{A}$ if* $\ker(\widehat{A}) = \langle \widehat{b} \rangle$ *(i.e., a dimension 1 subspace). We say that $\widehat{b}$ is* minimally degree $\mathbb{R}$-embedded *in $\widehat{A}$ if*

1. *$\widehat{b}$ is minimally $\mathbb{R}$-embedded in $\widehat{A}$, and*
2. *$\widehat{b}$ corresponds to a primitive kernel vector $\mathscr{b}$, that is* $\gcd(\mathscr{b}_1, \ldots, \mathscr{b}_n) = 1$.

We note that this definition ensures minimally $\mathbb{R}$-embedded vectors are unique (up to a scaling factor), or that $(\widehat{A}^{(j)} + \Delta\widehat{A}^{(j)})\widehat{B}_{*,j} = 0$ has a (locally) unique solution for fixed $\Delta\mathcal{A}$ and $\widehat{B} = \begin{pmatrix} \widehat{b}^{(1)} & \widehat{b}^{(2)} & \cdots & \widehat{b}^{(r)} \end{pmatrix}$. In the minimal embedding, we will assume, without loss of generality, that redundant or equations known in advance, such as $0 = 0, \Delta\widehat{A}_{ij} = 0$ or $\widehat{B}_{ijk} = 0$ corresponding to known entries are removed for some indices of i, j and k. For example, if we assume that $\mathscr{B}$ is primitive and $\widehat{B}$ is in CREF, then this will satisfy the minimal embedding requirements. The degree of the pivot reveals the degrees of the other entries, thus by knowing the pivot, one implicitly knows the degree of each entry. Some of these trivial equations occur because of the CREF assumption, while others occur from over-estimating degrees of entries.

This allows us to reformulate $(\mathcal{A} + \widetilde{\Delta}\mathcal{A})\mathscr{B} = 0$ as a (bi-linear) system of equations

$$\{(\widehat{A}^{(j)} + \Delta\widehat{A}^{(j)})\widehat{B}_{*j} = 0\}_{j=1}^{r}, \tag{3.8}$$

where the j^{th} column of $\mathscr{B}$ is minimally degree embedded in the system $(\widehat{A}^{(j)} + \Delta\widehat{A}^{(j)})$. We also note that assuming $\mathscr{B}$ is in a column-reduced echelon form essentially requires us to guess the pivots in advance of the optimal solution, which is possible with a good initial guess. The benefit of this approach is that if the pivots are not guessed correctly, we are still able to compute a $n - r$ approximation of $\mathcal{A}$ (that will not be globally optimal).

In order to exclude trivial solutions, we can assume that the pivot elements of $\mathscr{B}$ have a norm bounded away from zero. Let $\mathcal{N}(\widehat{b}^{(j)})$ be a normalization vector such that $\mathcal{N}(\widehat{b}^{(j)})^T\widehat{b}^{(j)} = 1$ which implies that the CREF pivots are bounded away from zero. For example, take the pivot to have unit norm. Note that other normalization vectors are possible, such as $\mathcal{N}(\widehat{b}^{(j)}) = \widehat{b}^{(j)}$ (which corresponds to each column having a unit norm) if the initial guess is adequately close, or we could take the pivot element to be a monic polynomial. Of course there are several other permissible normalizations.

Define the matrix $\widehat{A}^{(j)}$ to have the column $\widehat{b}^{(j)} = \widehat{B}_{*,j}$ minimally degree embedded. We can express (3.8) in a vector-matrix form as follows

$$\left(\begin{array}{cccc} \widehat{A}^{(1)} + \Delta\widehat{A}^{(1)} & & & \\ & \widehat{A}^{(2)} + \Delta\widehat{A}^{(2)} & & \\ & & \ddots & \\ & & & \widehat{A}^{(r)} + \Delta\widehat{A}^{(r)} \\ \hline \mathcal{n}(\widehat{b}^{(1)})^T & & & \\ & \mathcal{n}(\widehat{b}^{(2)})^T & & \\ & & \ddots & \\ & & & \mathcal{n}(\widehat{b}^{(r)})^T \end{array}\right) \begin{pmatrix} \widehat{b}^{(1)} \\ \widehat{b}^{(2)} \\ \vdots \\ \widehat{b}^{(r)} \end{pmatrix} = \left(\begin{array}{c} 0 \\ 0 \\ \vdots \\ 0 \\ \hline 1 \\ 1 \\ \vdots \\ 1 \end{array}\right). \tag{3.9}$$

The minimal system has a (locally) unique solution for fixed $\Delta\mathcal{A}$. Note that the upper block in this system is remarkably similar to $(\mathcal{A} + \widetilde{\Delta}\mathcal{A})\mathcal{B} = 0$, which is equivalent to

$$[I \otimes (\mathcal{A} + \widetilde{\Delta}\mathcal{A})]\,\mathrm{pvec}(\mathcal{B}) = 0 \implies \Phi_{nd}([I \otimes (\mathcal{A} + \widetilde{\Delta}\mathcal{A})])\,\mathrm{vec}(\mathcal{B}) = 0,$$

except that several rows and columns of each diagonal block of $\Phi_{nd}([I \otimes (\mathcal{A} + \widetilde{\Delta}\mathcal{A})])$ are deleted. These rows and columns are deleted to account for known zero entries in the kernel or over-padding of zero entries in the block convolution matrix.

3.6.2 Lagrange Multipliers and Optimality Conditions

Let $M(\Delta\mathcal{A}, \mathcal{B})$ be the vector of residuals corresponding to (3.9), then the Lagrangian is defined as

$$L = \|\Delta\mathcal{A}\|_F^2 + \lambda^T M(\Delta\mathcal{A}, \mathcal{B}), \tag{3.10}$$

where λ is a vector of Lagrange multipliers.

We will find it convenient to define $x = \begin{pmatrix} \mathrm{vec}(\Delta\mathcal{A}) \\ \mathrm{vec}(\mathcal{B}) \end{pmatrix}$ to be the combined vector of unknowns corresponding to $\mathcal{A}$ and $\mathcal{B}$. Let $\nabla^2_{xx}L$ denote the Hessian matrix of L with respect to x and J be the Jacobian of the residuals of the constraints, i.e. $J = \nabla_x M(\Delta\mathcal{A}, \mathcal{B})$. Recall the necessary optimality conditions at a point (x^*, λ^*) are that

$$\nabla L = 0 \text{ and } \ker(J)^T\,\nabla^2_{xx}L\,\ker(J) \succeq 0. \tag{3.11}$$

Sufficient conditions for optimality at the same point are that

$$\nabla L = 0 \text{ and } \ker(J)^T\,\nabla^2_{xx}L\,\ker(J) \succ 0. \tag{3.12}$$

We note that (3.12) implies that minimal solutions are locally unique, and will fail to hold if minimal solutions are not locally unique. The idea is to show that (3.12) holds in the minimal embedding, which allows us to construct an algorithm with rapid local convergence.

Does There Always Exist an Optimal Rank One Perturbation?

An interesting observation here is that minimal perturbations to the nearest (structured) singular matrix polynomial problem will generally have rank exceeding one (which is different from the unstructured scalar instance). In particular, the strategy of looking for an optimal rank one perturbation to a matrix polynomial will almost never be a good idea. We will also see that the same lesson will hold for structured scalar matrices.

Theorem 3.6.2. *Suppose that $\mathcal{A} \in \mathbb{R}[t]^{n\times n}$ has degree d and that $\Delta\mathcal{A} \in \mathbb{R}^{n^2(d+1)}$ is a minimal unstructured perturbation (i.e. $\boldsymbol{\Delta}(\cdot) = \boldsymbol{\Delta}_{degree}(\cdot)$) to $\mathcal{A}$ that does not increase the degree of $\mathcal{A}$ with $\mathscr{b}$ and $\lambda = \begin{pmatrix}\lambda_1\\ \lambda_2\end{pmatrix}$ corresponding to a regular solution (i.e. the Jacobian of the constraints has full rank) to a rank at most $n-1$ approximation to (3.7), then*

$$2\operatorname{vec}(\Delta\mathcal{A}) = -[\Phi_d(\mathscr{b}^T \otimes I)]^T\lambda_1.$$

Proof. From the KKT conditions we have that if we write $\lambda = \begin{pmatrix}\lambda_1\\ \lambda_2\end{pmatrix}$ then

$$\begin{aligned}\nabla_{\Delta\mathcal{A}}L = 0 \implies 2\operatorname{vec}(\Delta\mathcal{A})^T &= -\nabla_{\Delta\mathcal{A}}\lambda_1^T\operatorname{vec}((\mathcal{A}+\Delta\mathcal{A})\mathscr{b})\\ &= -\nabla_{\Delta\mathcal{A}}\lambda_1^T\operatorname{vec}([\mathscr{b}^T\otimes I]\operatorname{pvec}(\mathcal{A}+\Delta\mathcal{A}))\\ &= -\lambda_1^T\frac{\partial\operatorname{vec}([\mathscr{b}^T\otimes I]\operatorname{pvec}(\mathcal{A}+\Delta\mathcal{A}))}{\partial\operatorname{vec}(\Delta\mathcal{A})}\\ &= -\lambda_1^T\Phi_d(\mathscr{b}^T\otimes I)\frac{\partial\operatorname{vec}(\mathcal{A}+\Delta\mathcal{A})}{\partial\operatorname{vec}(\Delta\mathcal{A})}\\ &= -\lambda_1^T\Phi_d(\mathscr{b}^T\otimes I).\end{aligned}$$

□

Now, when $d=0$ we have that $2\operatorname{vec}(\Delta\mathcal{A}) = -(\mathscr{b}\otimes I)\operatorname{vec}(\lambda_1)$, which is equivalent[5] to $2\Delta\mathcal{A} = \lambda_1\mathscr{b}^T$, thus the optimal (unstructured) $\Delta\mathcal{A}$ is a rank one perturbation. This is

[5]This equivalence occurs because it is irrelevant in the unstructured scalar instance if $\Delta\mathcal{A}$ is a vector or a matrix, as the two are isomorphic. This distinction and abuse of notation is not permissible for structured matrices or structured matrix polynomials in this context, because the isomorphism may not exist.

not surprising, as the SVD implies the same result. However, if $\|\mathscr{b}\| = 1$, then there exist Lagrange multipliers λ where the block λ_1 satisfies $\|\lambda_1/2\|_2 = \|\Delta\mathscr{A}\|_F$. In other words, we can bound the size of the Lagrange multipliers (corresponding to $(\mathscr{A} + \Delta\mathscr{A})\mathscr{b} = 0$) in terms of the size of a minimal perturbation.

Note that $\lambda^T \Phi_d(\mathscr{b}^T \otimes I)$ does not in general admit a "nice" factorization interms of the Kronecker product, other than the instances of $d = 0$ or $\deg(\mathscr{b}) = 0$. When $\deg(\mathscr{b}) = 0$ we have that $\phi_d(\mathscr{b}) = \mathscr{b} \otimes I_{d+1}$. Accordingly, if a kernel vector of a local minimizer has degree zero, then it is generated by a rank one perturbation. If we suppose that $\lambda_1 = \operatorname{vec}(\mathscr{z})$ for some $\mathscr{z} \in \mathbb{R}[t]^{n\times 1}$ of degree at most d then we have

$$\begin{aligned} 2\operatorname{vec}(\Delta\mathscr{A}) &= -\Phi_d(\mathscr{b}^T \otimes I_n)^T \operatorname{vec}(\mathscr{z}) \\ &= -(\mathscr{b} \otimes I_{n(d+1)}) \operatorname{vec}(\mathscr{z}) \\ &= -\Phi_d(\mathscr{b} \otimes I_n) \operatorname{vec}(\mathscr{z}), \end{aligned}$$

so we have that

$$2\operatorname{pvec}(\Delta\mathscr{A}) = -(\mathscr{b} \otimes I_n)\mathscr{z} \iff 2\Delta\mathscr{A} = -\mathscr{z}\mathscr{b}^T,$$

which implies that $\Delta\mathscr{A}$ is a rank one perturbation. It is important to note that in general $\Phi_d(\mathscr{b}^T \otimes I)^T \neq \Phi_d(\mathscr{b} \otimes I)$. This relationship does hold when $\deg(\mathscr{b}) = 0$ though.

If the coefficient matrices of $\mathscr{A} + \Delta\mathscr{A}$ have a common kernel vector, then there exists a kernel vector of degree zero, and so it must be generated by a rank one perturbation. If $\deg(\mathscr{A} + \Delta\mathscr{A}) = 1$ and there is a degree zero kernel vector, then ΔA_0 and $\Delta\mathscr{A}_1$ have rank (at most) one, which can be easily seen from evaluating $\operatorname{rank}((\mathscr{A} + \Delta\mathscr{A})(0))$ and $\operatorname{rank}((t(\mathscr{A}_0 + \Delta\mathscr{A}_0) + \mathscr{A}_1 + \Delta\mathscr{A}_1)(0))$. These statements are a more general and direct proof to some of the claims made in [50] about low rank perturbations, which only considered matrix pencils (degree one matrix polynomials).

Next we observe that if perturbations are structured, i.e. we are solving $(\mathscr{A} + \widetilde{\Delta}\mathscr{A})\mathscr{b} = 0$ then

$$\nabla_{\Delta\mathscr{A}} \operatorname{vec}((\mathscr{A} + \widetilde{\Delta}\mathscr{A})\mathscr{b}) = \frac{\partial \operatorname{vec}((\mathscr{A} + \widetilde{\Delta}\mathscr{A})\mathscr{b})}{\partial \operatorname{vec}(\Delta\mathscr{A})} \neq \Phi_d(\mathscr{b}^T \otimes I),$$

thus we cannot naively think of $\Delta\mathscr{A} \in \mathbb{R}[t]^{n\times n}$. From the perspective of matrix calculus, $\Delta\mathscr{A}$ is just a collection of at most $n^2(d+1)$ variables that is sometimes isomorphic to a matrix polynomial.

In the instances of support or entry degree preserving (but otherwise unstructured) perturbations, the requisite derivative will be a sub-matrix of $\Phi_d(\mathscr{b}^T \otimes I)$ (several columns would be deleted, corresponding to the fixed zero entries). An analogous theorem would hold, and the following corollary can be modified in a straightforward manner. If some entries were weighted, then again similar results hold (columns of the derivative are now scaled).

Theorem 3.6.2 leads to the following bounds on the Lagrange multipliers.

Corollary 3.6.3. *Suppose that $\Delta\mathcal{A}$ is a minimal unstructured perturbation to $\mathcal{A}$ that does not increase the degree of $\mathcal{A}$ with $\mathcal{b}$ and $\lambda = \begin{pmatrix}\lambda_1\\ \lambda_2\end{pmatrix}$ corresponding to a regular solution to a rank at most $n-1$ approximation to (3.7), then*

$$\frac{2\|\Delta\mathcal{A}\|_F}{\|(\Phi_d(\mathcal{b}^T\otimes I))\|_2} \leq \|\lambda_1\|_2 \leq \frac{2\|\Delta\mathcal{A}\|_F}{\sigma_{\min}(\Phi_d(\mathcal{b}^T\otimes I))}.$$

Furthermore, we have that when $\|\mathcal{b}\|=1$ that

$$\frac{2\|\Delta\mathcal{A}\|_F}{d+1} \leq \|\lambda_1\|_2.$$

Proof. First, $\mathcal{b}$ is a non-zero vector, so $\mathcal{b}$ has full rank. Thus $\mathcal{b}^T\otimes I$ has full rank, and $\Phi_d(\mathcal{b}^T\otimes I)$ will have full rank as well. Thus, the smallest non-trivial singular value of $\Phi_d(\mathcal{b}^T\otimes I)$ is non-zero.

The instance of $d=0$ follows by the preceding discussion.

In the instance when $d>0$ we have that $-\left[\Phi_d(\mathcal{b}^T\otimes I)\right]^T\lambda = 2\operatorname{vec}(\Delta\mathcal{A})$, so

$$\frac{2\|\Delta\mathcal{A}\|_F}{\|(\Phi_d(\mathcal{b}^T\otimes I))\|_2} \leq \|\lambda_1\|_2 \leq \frac{2\|\Delta\mathcal{A}\|_F}{\sigma_{\min}(\Phi_d(\mathcal{b}^T\otimes I))}.$$

The second-statement follows by Lemma 2.5.11. □

So, if the linearly independent constraint qualification does not hold, we will later show that the problem can be modified so that some other regularity condition holds. Thus there will exist a Lagrange multiplier λ that satisfies Theorem 3.6.2 and the associated corollary.

In general for arbitrary structured perturbations, we can write

$$2\operatorname{vec}(\Delta\mathcal{A})^T = -\lambda^T\Phi_d(\mathcal{b}^T\otimes I)\frac{\partial\operatorname{vec}(\widetilde{\Delta\mathcal{A}})}{\partial\operatorname{vec}(\Delta\mathcal{A})}.$$

The preceding results generalize if $\Phi_d(\mathcal{b}^T\otimes I)$ is replaced with $\Phi_d(\mathcal{b}^T\otimes I)\frac{\partial\operatorname{vec}(\widetilde{\Delta\mathcal{A}})}{\partial\operatorname{vec}(\Delta\mathcal{A})}$, which is scaling the block-convolution matrix of $\mathcal{b}^T\otimes I$ by the derivative of $\mathbf{\Delta}(\cdot)$ with respect to $\Delta\mathcal{A}$.

3.6.3 The Jacobian

Definition 3.6.4. *The matrix $\psi(\widehat{b})$ is an alternative form of $(\widehat{A}+\Delta\widehat{A})\widehat{b}=0$ that satisfies $\psi(\widehat{b})\,\mathrm{vec}(\mathcal{A}+\widetilde{\Delta}\mathcal{A})=0$. That is, $\psi(\widehat{b})$ satisfies*

$$\psi(\widehat{b})\cdot\mathrm{vec}(\mathcal{A}+\widetilde{\Delta}\mathcal{A})=0 \iff (\widehat{A}+\Delta\widehat{A})\widehat{b}=0.$$

We will adopt that notation that $\psi(\widehat{b}^{(j)})$ corresponds to $\psi(\widehat{b}^{(j)})\,\mathrm{vec}(\widehat{A}_i+\Delta\widehat{A}_i)=0$. Here we use the bi-linearity of (3.9) to write the same system using a matrix with entries from $\widehat{B}$ instead of $\mathrm{vec}(\mathcal{A}+\widetilde{\Delta}\mathcal{A})$.

Note that $\psi(\widehat{b})$ can be computed explicitly. We can write

$$(\mathcal{A}+\widetilde{\Delta}\mathcal{A})\mathscr{b}=0 \iff [\mathscr{b}^T\otimes I]\,\mathrm{pvec}((\mathcal{A}+\widetilde{\Delta}\mathcal{A}))=0,$$

which reduces to

$$\Phi_d([\mathscr{b}^T\otimes I])\,\mathrm{vec}(\mathcal{A}+\widetilde{\Delta}\mathcal{A})=0.$$

Now from the minimal embedding some columns or rows will be removed, and $\psi(\cdot)$ is a sub matrix obtained from $\Phi_d([\mathscr{b}^T\otimes I])$. So, as long as we know the degrees of the kernel vector of minimal degree, $\psi(\widehat{b})$ will have full rank if this kernel vector is used and we do not over-pad with zero entries for $\mathbf{\Delta}(\cdot)$ that preserve zero coefficients, the degree of each entry or the degree of the matrix polynomial. For other structures (such as entries being coupled or weighted), this rank condition may not hold.

The closed-form expression for the Jacobian of the residuals (up to permutation) in (3.9) is (assuming $\mathcal{N}(\widehat{b}_j)^T\widehat{b}_j=\|\widehat{b}_j\|_2^2$) given by

$$J=\left(\begin{array}{c|cccc}
\psi(\widehat{b}_1) & \widehat{A}^{(1)}+\Delta\widehat{A}^{(1)} & & & \\
\psi(\widehat{b}_2) & & \widehat{A}^{(2)}+\Delta\widehat{A}^{(2)} & & \\
\vdots & & & \ddots & \\
\psi(\widehat{b}_r) & & & & \widehat{A}^{(r)}+\Delta\widehat{A}^{(r)} \\
\hline
0 & 2\mathcal{N}(\widehat{b}^{(1)})^T & & & \\
0 & & 2\mathcal{N}(\widehat{b}^{(2)})^T & & \\
\vdots & & & \ddots & \\
0 & & & & 2\mathcal{N}(\widehat{b}^{(r)})^T
\end{array}\right). \tag{3.13}$$

If we look at the problem when $r=1$, then J will have full row rank, as the bi-linear form is represented with the minimal number of equations.

Unlike the case of a single kernel vector, J may be rank deficient since some equations corresponding to low (high) index entries may be redundant at the solution. The Lagrange multipliers will not be unique in this particular scenario and the rate of convergence may degrade if Newton's method is used.

Theorem 3.6.5. *Suppose that $r = 1$ and $\widehat{b}^{(1)}$ is minimally degree $\mathbb{R}$-embedded in $\widehat{A}_1$, then J has full rank when (3.11) holds.*

Proof. We show that J has full row rank by contradiction. If this matrix was rank deficient, then one row is a linear combination of the others. This means that one of the equations in the constraints is redundant, trivial or the solution is not regular. As we are only concerned about regular solutions, this contradicts the minimal $\mathbb{R}$-embedding. $\square$

From the discussion earlier, all we need to know in a generic sense is the degrees of the entries of a kernel vector that is primitive. A kernel vector of minimum degree will of course satisfy this requirement. The minimal embedding assumption ensures linear independence if there are multiple possible vectors of the same degree, thus no constraints may be redundant. As mentioned earlier, it is straight forward to compute such a vector via orthogonal (unitary) transformations. We note that for arbitrary linear and affine structures that this result may not hold, as the structure may imply some additional redundancy in the constraints.

The corollary to this is that in the minimal embedding regularity conditions hold and it is straight forward to obtain rapid local convergence. If the kernel vector is not minimally embedded, then there will exist a linearly independent subset of the rows which determine the Lagrange multipliers. I.e. $\nabla_x \|\Delta\mathscr{A}\|_F^2 = 2(\text{vec}(\Delta\mathscr{A})^T, 0) = -\lambda^T J$ will be surjective at a solution, as the rows that are redundant have $\lambda_j = 0$ as a Lagrange multiplier. In general, it is not always possible to compute a minimal embedding for the solution to the optimization problem from the initial guess. In practice, one makes a reasonable (possibly over-restricted guess), then relaxes the degree and or dimension requirements until an improvement cannot be made.

When $r \geq 1$ we have that at the solution some constraints are redundant. If these constraints were removed, then the bi-linear form would also be minimal. Thus, $\lambda^T J$ in general will be surjective, i.e. if the solution is finite then Lagrange multipliers exist. J can be rank deficient because some kernel vectors have non-pivot entries that are the same (some constraints are redundant). Note that ∇J is a constant tensor, i.e. the Hessian of the constraints is a constant, so the constraints have constant curvature.

3.6.4 The Hessian

The Hessian matrix, $\nabla^2 L$ is straight forward to compute as

$$\nabla^2 L = \begin{pmatrix} \nabla^2_{xx} L & J^T \\ J & 0 \end{pmatrix}.$$

The following theorem shows that second-order sufficiency holds for the instance of $r = 1$. The case of $r > 1$ follows immediately by induction. This is in contrast to Theorem 3.6.5, which does not always hold for $r > 1$.

Note that $\nabla^2_{xx} L$ is a linear function in terms of the entries of λ, thus computing

$$\frac{\partial}{\partial \operatorname{vec}(\Delta \mathcal{A})} \operatorname{vec}(\Delta \widehat{A}^{(j)}) \text{ and } \frac{\partial}{\partial \operatorname{vec}(\widehat{b}^{(j)})} \operatorname{vec}(\psi(\widehat{b}^{(j)}))$$

can be done once, then the quantities can be scaled by $(\lambda^T \otimes I)$ each time $\nabla \operatorname{vec}(J^T \lambda)$ is required to be updated. Of course given the linear structure, other reasonable methods are applicable as well.

Theorem 3.6.6 (Second-Order Sufficiency Holds)**.** *Suppose that $\widehat{A} + \Delta \widehat{A}$ has a minimally degree $\mathbb{R}$-embedded kernel vector $\widehat{b}$, i.e. $r = 1$ in (3.10), then at a minimal solution, the second-order sufficient condition (3.12) holds in the minimal embedding of $\widehat{b}$.*

Proof. If $\|\Delta A\| = 0$ at the local minimizer (x^*, λ^*) then

$$\nabla^2_{xx} L(x^\star, \lambda^\star) = \begin{pmatrix} 2I & \\ & 0 \end{pmatrix} \text{ and } K = \ker(\nabla^2_{xx} L(x^\star, \lambda^\star)) = \operatorname{span} \begin{pmatrix} 0 & \\ & I \end{pmatrix}.$$

We have that for $y \in \operatorname{span}(K)$ such that $Jy = 0$ implies that $\widehat{A}y = 0$ and $\mathcal{N}(\widehat{b})^T y = 0$. It follows that $\ker(\widehat{A}) = \operatorname{span}(\widehat{b})$, thus we have $y = \widehat{b}$ or $y = 0$ via the minimal degree $\mathbb{R}$-embedding, thus $y = 0$ as $\widehat{b} \notin \operatorname{span}(K)$. Hence, second-order sufficiency holds, as $\ker(J) \cap K = 0$.

If $\|\Delta \mathcal{A}\| \neq 0$ then we have that

$$\nabla^2_{xx} L(x^\star, \lambda^\star) = \underbrace{\begin{pmatrix} 2I & 0 \\ 0 & 0 \end{pmatrix}}_{\mathcal{H}} + \underbrace{\begin{pmatrix} 0 & E^T \\ E & 0 \end{pmatrix}}_{\mathcal{E}}.$$

The matrix $\mathscr{E}$ is linear in λ, however the precise tensor decomposition is irrelevant to the proof. If E has full rank, then $\nabla^2_{xx}L$ has full rank and we are done, so suppose that E is rank deficient. If E is rank deficient, then one can eliminate a row of E and column of E^T without affecting $\mathscr{H}$ via symmetric row and column updates. We observe that $\ker(\mathscr{H} + \mathscr{E}) \subseteq \ker(\mathscr{H})$ and the result follows. □

Corollary 3.6.7. *Suppose that $r > 1$ in (3.10) and $\mathscr{B}$ is minimally degree embedded, then second-order sufficiency (3.12) holds.*

Proof. The proof is almost the same as Theorem 3.6.6 and follows by induction on r since each block is decoupled. □

We now have all of the ingredients for an iterative method with rapid local convergence. Note that Theorem 3.6.6 is generally satisfied for arbitrary linear and affine structures assuming the kernel vector is minimally embedded some how. All that was needed was minimality of the kernel vector. There is no dependency on the magnitude of $\Delta\mathscr{A}$, which differs from similar results that will be presented later in this thesis, that is they assume $\|\Delta\mathscr{A}\|$ is sufficiently small to show that the second-order sufficient condition holds.

3.6.5 Implementation Notes

Newton's method for equality constrained minimization problems can be interpreted as solving the non-linear system of equations $\nabla L = 0$. Newton's method is based on the iterative update scheme:

$$\begin{pmatrix} x^{k+1} \\ \lambda^{k+1} \end{pmatrix} = \begin{pmatrix} x^k + \Delta x^k \\ \lambda^k + \Delta\lambda^k \end{pmatrix} \quad \text{such that} \quad \nabla^2 L \begin{pmatrix} \Delta x \\ \Delta\lambda \end{pmatrix} = -\nabla L. \tag{3.14}$$

If $r = 1$ then $\nabla^2 L$ has full rank and the iteration is well defined by matrix inversion. If $r > 1$ then we consider the Newton method defined as

$$\begin{pmatrix} x^{k+1} \\ \lambda^{k+1} \end{pmatrix} = \begin{pmatrix} x^k + \Delta x^k \\ \lambda^k + \Delta\lambda^k \end{pmatrix} \quad \text{such that} \quad \begin{pmatrix} \nabla^2_{xx}L & J^T \\ J & -\nu_k I \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta\lambda \end{pmatrix} = -\nabla L, \tag{3.15}$$

for a suitably chosen parameter ν_k. Taking $\nu_k = \|\nabla L(x^k, \lambda^k)\|_1$ one has provably quadratic convergence [109, Theorem 4.2] with x^k and λ^k chosen sufficiently close to the optimal solution.

Theorem 3.6.8. *The iteration* (3.15) *converges quadratically to* $(x^\star, \lambda^\star)$ *if* (x^0, λ^0) *are chosen sufficiently close to* $(x^\star, \lambda^\star)$.

The size of $\nabla^2 L$ is $O(n^2d + rn^2d) \times O(n^2d + rn^2d) = O(r^2n^4d^2)$ and accordingly each iteration has a cost of $O(r^3n^6d^3)$ FLOPs using standard matrix multiplication, where r is the dimension of the kernel. In general the number of variables for r kernel vectors with $n \times (n-r)d$ is $O(rn(n-r)d) = O(rn^2d)$. The worst-case is $r = O(n)$, so the number of FLOPs per iteration would be $O(n^9d^3)$, which is cubic in the output.

We now have a method to compute a nearby rank deficient matrix polynomial with a rate of convergence that is quadratic, provided that the initial values of x are chosen to be sufficiently close to the optimal solution.

3.7 Implementation and Examples

In this subsection we discuss implementation details and demonstrate our implementation for computing the nearest rank deficient matrix polynomial. All algorithms are implemented in Maple.

3.7.1 Description of Algorithms

We now describe several algorithms for computing the nearest matrix polynomial of a prescribed rank with a suitable initial guess. The variables x, λ and z are defined in the respective section that each algorithm references.

Algorithm 1 attempts to compute a nearby embedding of a matrix polynomial of reduced rank. Usually reducing the rank of the embedding will lead to a reduced rank matrix polynomial, but this is not necessarily the case. To ensure that $\mathscr{A} + \Delta\mathscr{A}$ has the desired rank one may need to iteratively build several lower rank approximations of $\widehat{A}$.

If C has full rank or has the wrong structure (for example, after projecting to a reduced rank matrix via a truncated SVD), we use a variant of Newton's method[6] to compute a feasible point of reduced rank to begin the Newton-like iteration (for optimization). After computing a feasible point, we re-initialize U and V by performing a thin SVD so that both U and V have full rank. If U or V become rank deficient then one simply re-initializes after performing a thin SVD, thus by Corollary 3.4.8, Algorithm 1 will typically have local quadratic convergence.

[6]We use a regularized Gauss-Newton method to project C to a feasible point.

```
Algorithm 1 : Iterative Embedded Rank Factorization Post-Refinement
Input:
  • Matrix polynomial embedding $\widehat{A} \in \mathbb{R}^{M \times N}$.
  • (Approximately) Rank deficient $C \in \mathbb{R}^{M \times N}$ of the desired degree/displacement structure and rank deficiency.
  • Displacement structure matrix $\widetilde{\Delta}\mathcal{A}$ to optimize over.
Output:
  • $\widehat{A} + \Delta\widehat{A}$ of prescribed rank deficiency or an indication of failure.
1: Compute $U$ and $V$ from a rank factorization of $C$.
2: Compute Lagrangian $L$ from Section 3.4.2.
3: Initialize $\lambda$ via linear least squares from $\nabla L|_{x^{init}} = 0$.
4: Compute $\begin{pmatrix} x + \Delta x \\ \lambda + \Delta\lambda \end{pmatrix}$ by solving (3.3) until $\left\| \begin{pmatrix} \Delta x \\ \Delta\lambda \end{pmatrix} \right\|_2$ or $\|\nabla L(x, \lambda)\|_2$ is sufficiently small or divergence is detected.
5: Return the locally optimal $\Delta\mathcal{A}$ or an indication of failure.
```

```
Algorithm 2 : Iterative Evaluated Rank Factorization Post-Refinement
Input:
  • Matrix polynomial $\mathcal{A} \in \mathsf{R}^{n \times n}$.
  • Upper bound $r$ on the dimension of the kernel of a solution.
  • (Approximately) Rank deficient $\mathcal{C} \in \mathbb{R}^{n \times n}$ of the desired degree/displacement structure and rank deficiency.
  • Displacement structure matrix $\widetilde{\Delta}\mathcal{A}$ to optimize over.
Output:
  • $\mathcal{A} + \widetilde{\Delta}\mathcal{A}$ of prescribed rank deficiency or an indication of failure.
1: Choose $nd + 1$ suitable evaluation points $\{\omega_j\}_{j=0}^{nd}$.
2: Compute $U_j$ and $V_j$ from an (approximate) rank factorization of $\mathcal{C}(\omega_j)$.
3: Compute Lagrangian $L$ from Section 3.5.1.
4: Initialize $\lambda$ via linear least squares from $\nabla L|_{x^{init}} = 0$.
5: Compute $\begin{pmatrix} x + \Delta x \\ \lambda + \Delta\lambda \end{pmatrix}$ by solving (3.6) until $\left\| \begin{pmatrix} \Delta x \\ \Delta\lambda \end{pmatrix} \right\|_2$ or $\|\nabla L(x, \lambda)\|_2$ is sufficiently small or divergence is detected.
6: Return the locally optimal $\Delta\mathcal{A}$ or an indication of failure.
```

Algorithm 2 is basically Algorithm 1 applied to $nd + 1$ small instances of the problem. To compute an initial guess we perform a truncated SVD and post-refine the computed

U_j and V_j with a Newton-like method, analogous to earlier. It is important to recall that the precision used to compute the ω_j determines how accurate the solution will be. Maple uses ten digits of accuracy, so we can expect convergence up to approximately ten digits with default precision[7]. Our implementation uses complex roots of unity. This procedure can be iterated several times to obtain a lower-rank approximation of desired rank.

```
Algorithm 3 : Iterative Kernel Post-Refinement
Input:
  • Full rank matrix polynomial $\mathscr{A} \in \mathbb{R}[t]^{n\times n}$.
  • (Approximately) Rank deficient matrix polynomial $\mathscr{C} \in \mathbb{R}[t]^{n]\times n}$.
  • Approximate kernel vectors $c_1, \dots, c_r \in \mathbb{R}[t]^{n\times 1}$ of the desired degree/displacement structure.
  • Displacement structure matrix $\widetilde{\Delta\mathscr{A}}$ to optimize over.
Output:
  • Singular matrix $\mathscr{A} + \widetilde{\Delta\mathscr{A}}$ with $\mathscr{B} \subseteq \ker(\mathscr{A} + \Delta\mathscr{A})$ or an indication of failure.
1: $\mathbb{R}$-Embed $\mathscr{A}, \mathscr{C}, c_1, \dots, c_r$ and $\Delta\mathscr{A}$.
2: Compute Lagrangian $L$ from Section 3.6.2.
3: Initialize $\lambda$ via linear least squares from $\nabla L|_{x^{init}} = 0$.
4: Compute $\begin{pmatrix} x+\Delta x \\ \lambda+\Delta\lambda \end{pmatrix}$ by solving (3.15) until $\left\| \begin{pmatrix} \Delta x \\ \Delta\lambda \end{pmatrix} \right\|_2$ or $\|\nabla L(x,\lambda)\|_2$ is sufficiently small or divergence is detected.
5: Return the locally optimal $\Delta\mathscr{A}$ and $\mathscr{B}$ or an indication of failure.
```

All experiments using Algorithm 3 are done using quad precision floating point arithmetic, with about 35 decimal digits of accuracy or hardware precision. We compare our techniques to some degree one examples to the recent results of [50].

To compute an approximate kernel vector, first we use the SVD to compute an approximate kernel of an $\mathbb{R}$-embedded (nearly) rank deficient matrix polynomial. Next we use structured orthogonal elimination RQ (LQ) decomposition to produce a minimally (degree) $\mathbb{R}$-embedded vector from the kernel. In the case of several kernel vectors we use a modified Gaussian elimination on an embedding of an approximate kernel obtained by the SVD and approximate GCD to find nearby approximate kernel vectors that are primitive.

Algorithm 3 has no global convergence guarantees, however a globally convergent (although not necessarily optimal) algorithm can be developed in a straight forward manner

[7]This arises to truncations due to the use of `evalf()` in the implementation, and other implementations may not exhibit this behavior.

via augmenting our second-order algorithm with a first-order one, and removing content from kernel vectors if necessary.

3.7.2 Linear and Affinely Structured Matrix Examples

In this section we consider Examples 2.10, 2.11 and 2.12 from [50], where we compare our results to real perturbations. Note that complex perturbations are a straight-forward generalization of the theory presented here, and can be re-formulated as a problem over $\mathbb{R}$.

The technique of [50] poses computing a nearby rank deficient linear matrix pencil by verifying that sufficiently many images of the matrix polynomial are singular, so that $\det(\mathscr{A} + \widetilde{\Delta\mathscr{A}}) \equiv 0$. The problem is then posed as a solution to a system of Ordinary Differential Equations (ODE), assuming that certain genericity conditions on the eigenvalues of the solution hold[8]. They consider the instances of computing A_0 and A_1 with a common kernel vector, and the instance where A_0 and A_1 do not have a common kernel. Additionally, perturbations affecting only one of A_0 and A_1 are considered. We note that the solutions to the ODEs do not necessarily satisfy necessary optimality conditions (3.11), and accordingly will generally not be local minimizers. Another distinction between both methods is that we view the problem in terms of the coefficients, whereas [50] views the problem through the context of interpolation.

Affine Structured Examples I

Consider first the matrix polynomial

$$\mathscr{A} = \underbrace{\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}}_{A_1} t + \underbrace{\begin{pmatrix} 0 & 0.0400 & 0.8900 \\ 0.1500 & -0.0200 & 0 \\ 0.9200 & 0.1100 & 0.06600 \end{pmatrix}}_{A_0}$$

coming from Examples 2.10 and 2.12 of [50]

Example 3.7.1. *If we assume that A_1 is constant, then this is finding the (locally) nearest matrix polynomial with an affine structure since A_1 has non-zero fixed constants. First let's assume that zero entries are preserved, this is a linear structure on A_0.*

[8]Our algorithm and convergence theory does not explicitly rely on genericity assumptions or other properties of eigenvalues, however we do exploit generic properties in formulating initial guesses.

To compute an initial guess for $\mathscr{b}$ we use the SVD on $\widehat{A}$ and extract a guess from the smallest singular vector. This gives us

$$\mathscr{b}_{init} = \begin{pmatrix} -0.41067t^3 + 0.50576t^2 - 0.26916t - 0.035720 \\ 0.38025t^2 - 0.51139t + 0.30674 \\ 0.027012t^2 - 0.028083t + 0.010715 \end{pmatrix}.$$

For an initial guess on $\mathscr{A}$ we take $\mathscr{A}_{init} = \mathscr{A}$. Note that we do not need an initial guess that is singular, it just needs to be "sufficiently close" to a singular matrix polynomial.

If we do not allow perturbations to zero-coefficients, that is, $A_0[1,1]$ and $A_0[2,3]$ may not be perturbed, then after five iterations of plain Newton's method (see [38]) we compute

$$\Delta A_0 \approx \begin{pmatrix} 0.0 & -0.094149 & -0.0057655 \\ -0.093311 & 0.026883 & 0.0 \\ 0.0057142 & -0.0016462 & -0.00010081 \end{pmatrix}$$

with perturbation $\|\Delta\mathscr{A}\|_F \approx 0.135507$.

A corresponding (approximate) kernel vector is

$$\mathscr{b} \approx \begin{pmatrix} 0.73073t + 0.082126 \\ -0.67644 \\ -0.041424 \end{pmatrix}.$$

Example 3.7.2. *If we allow perturbations to zero-coefficients in A_0 then after five rounds of plain Newton's method we compute*

$$\Delta A_0 \approx \begin{pmatrix} 0.0 & -0.094179 & -0.0057705 \\ -0.093280 & 0.026786 & 0.0016412 \\ 0.0057154 & -0.0016412 & -0.00010056 \end{pmatrix}$$

with perturbation $\|\Delta\mathscr{A}\|_F \approx 0.135497$, which is a marginal improvement over the previous example. A corresponding approximate kernel vector is

$$\mathscr{b} \approx \begin{pmatrix} 0.73073t + 0.082131 \\ -0.67644 \\ -0.041447 \end{pmatrix}.$$

[50] report an upper-bound on the distance to singularity allowing *complex perturbations*, that is $\Delta\mathscr{A} \in \mathbb{C}[t]^{n\times n}$ of $\|\Delta^{\mathbb{C}}\mathscr{A}\|_F \approx 0.1357$ in Example 2.10. In Example

2.12, [50] report an upper-bound on the distance to singularity allowing *real perturbations*, $\|\Delta^{\mathbb{R}}\mathcal{A}\|_F \approx 0.1366$. Although we only consider real perturbations, both bounds are improved. We conjecture that the complex bound can be improved further.

If we allow perturbations to A_0 and A_1, then this is some form of finding the nearest rank deficient matrix polynomial. The question is whether to allow degree or support preserving perturbations. Again, we will use the same initial guesses as the previous example.

Matrix degree preserving perturbations are of the form

$$\Delta^{deg}\mathcal{A} = \begin{pmatrix} tA_{1,1,1} + A_{1,1,0} & tA_{1,2,1} + A_{1,2,0} & tA_{1,3,1} + A_{1,3,0} \\ tA_{2,1,1} + A_{2,1,0} & tA_{2,2,1} + A_{2,2,0} & tA_{2,3,1} + A_{2,3,0} \\ tA_{3,1,1} + A_{3,1,0} & tA_{3,2,1} + A_{3,2,0} & tA_{3,3,1} + A_{3,3,0} \end{pmatrix},$$

where as support preserving perturbations are of the form

$$\Delta^{sup}\mathcal{A} = \begin{pmatrix} 0 & A_{1,2,0} & A_{1,3,0} \\ A_{2,1,0} & A_{2,2,0} & A_{2,3,1}t \\ A_{3,1,0} & tA_{3,2,1} + A_{3,2,0} & A_{3,3,0} \end{pmatrix}.$$

Example 3.7.3. *In the instance of degree preserving perturbations we compute after five iterations of Newton's method*

$$\Delta^{deg}\mathcal{A} \approx \begin{pmatrix} 0.0036502 & 0.0039174t - 0.066405 & 0.00011839t - 0.0020069 \\ -0.066897 & 0.058993t + 0.029807 & 0.0017829t + 0.00090082 \\ 0.0059893 & -0.0053098t - 0.0024133 & -0.00016047t - 0.000072934 \end{pmatrix}$$

with $\|\Delta^{deg}\mathcal{A}\| \approx 0.115585$.

A corresponding approximate kernel vector is

$$\mathcal{b} \approx \begin{pmatrix} -0.72941t - 0.080355 \\ 0.67903 \\ 0.020522 \end{pmatrix}.$$

Example 3.7.4. *In the instance of support preserving we compute after five iterations of Newton's method,*

$$\Delta^{sup}\mathcal{A} \approx \begin{pmatrix} 0.0 & -0.094311 & -0.0057928 \\ -0.092552 & 0.026973 & 0.0051028t \\ 0.0057434 & -0.0051554t - 0.0016739 & -0.00010281 \end{pmatrix}$$

with $\|\Delta^{sup}\mathcal{A}\| \approx 0.135313$. *A corresponding approximate kernel vector is*

$$\mathcal{b} \approx \begin{pmatrix} -0.72895t - 0.082339 \\ 0.67832 \\ 0.041664 \end{pmatrix}.$$

[50] report an upper-bound on the distance to singularity of $\|\Delta^{deg}\mathcal{A}\|_F \approx 0.1193$ in Example 2.12. This bound is larger than the one computed in Example 3.7.3.

3.7.3 Affine Structured Examples II

Example 3.7.5. *Next we consider the the matrix polynomial $\mathcal{A}$ in Example 2.11 of [50] defined as*

$$\mathcal{A} = \underbrace{\begin{pmatrix} -1.79 & 0.10 & -0.6 \\ 0.84 & -0.54 & 0.49 \\ -0.89 & 0.3 & 0.74 \end{pmatrix}}_{A_0} + \underbrace{\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}}_{A_1} t.$$

To compute an initial guess for we take $\mathcal{A}_{init} = A$ and take

$$\mathscr{b}^{init} = \begin{pmatrix} -0.16001t^3 - 0.10520t^2 + 0.15811t + 0.11409 \\ 0.14980t^3 - 0.51289t^2 - 0.18616t + 0.54098 \\ 0.20801t^3 + 0.26337t^2 - 0.44619t - 0.027979 \end{pmatrix}.$$

$\mathscr{b}^{init}$ is computed from the smallest singular vector of $\widehat{A}$.

We note that this initial guess does not attempt to find a nearby singular matrix polynomial for the initial guess, all that is needed is $\nabla L(x^{init}, \lambda^{init})$ is reasonably small to obtain convergence.

Using a globalized variant of Newton's method we compute a stationary point

$$\Delta\mathcal{A} \approx \begin{pmatrix} 0.047498t + 0.17772 & 0.44989t + 0.12420 & -0.091945t - 0.068210 \\ 0.20979t + 0.078872 & -0.094205t + 0.41583 & -0.037916t - 0.094081 \\ 0.082862t - 0.15413 & -0.58334t + 0.12940 & 0.081637t + 0.017208 \end{pmatrix},$$

with $\|\Delta\mathcal{A}\|_F \approx 0.949578$. The corresponding approximate kernel vector is

$$\mathscr{b} = \begin{pmatrix} -0.29258t - 0.21491 \\ 0.044825t - 0.90281 \\ 0.068189t + 0.21562 \end{pmatrix}.$$

If we use the result of [50] as the initial guess, then we compute

$$\mathscr{b}^{init} = \begin{pmatrix} 0.16409t^2 + 0.25146t + 0.12362 \\ -4.5353 \times 10^{-14}t^2 + 0.23740t + 0.55516 \\ 1.2457 \times 10^{-13}t^2 - 0.48688t - 0.0060443 \end{pmatrix}.$$

We will assume the entries of $\mathscr{b}$ are of degree at most two.

After five iterations of Newton's method we obtain

$$\Delta\mathscr{A} \approx \begin{pmatrix} 0.17257 & 0.12237t + 0.25225 & -0.46902t + 0.087147 \\ 0.21449 & 0.15210t + 0.31353 & -0.58296t + 0.10832 \\ -0.055963 & -0.039685t - 0.081803 & 0.15210t - 0.028261 \end{pmatrix},$$

with $\|\Delta\mathscr{A}\|_F \approx 0.94356416$.

The corresponding approximate kernel vector is

$$\mathscr{b} = \begin{pmatrix} 0.18971t^2 + 0.29750t + 0.14667 \\ 0.27896t + 0.66186 \\ -0.58143t - 0.0079694 \end{pmatrix}.$$

The previously noted small quadratic terms were at roughly machine precision (the computation is done with 35 digits of precision) and truncated.

It is noted that in [50] a result on this past example that produces an upper bound on the distance to singularity of 0.9438619. The computation [50] is accurate to seven decimal points, and accordingly our post-refinement has an improvement of about 0.000297. This is not surprising, since we solve the necessary conditions (3.11) directly with a reasonable initial guess.

3.7.4 Lower Rank Approximation of a 4×4 Matrix Polynomial

In this following example we consider computing a lower-rank approximation to a given matrix polynomial. Consider the 4×4 matrix polynomial $\mathscr{A}$, defined as

$$\mathscr{A} = A_0 + A_1 t + A_2 t^2 + A_3 t^3, \text{ where}$$

$$A_0 = \begin{pmatrix} 0.09108776 & -0.05442464 & 0.3645006 & 0.01821543 \\ -0.1456436 & 0.03647524 & -0.07277662 & 0.07305016 \\ 0.05478714 & -0.05444916 & 0.4373220 & 0.05478385 \\ -0.1274211 & 0.09124859 & -0.6556615 & -0.05446850 \end{pmatrix},$$

$$A_1 = \begin{pmatrix} 0.09116729 & 0.00001797690 & 0.2550857 & 0.05475106 \\ 0.0001156514 & 0.00001659159 & 0.09108906 & -0.05447104 \\ 0.05470823 & 0.03662426 & 0.1276959 & 0.03650378 \\ 0.05472202 & -0.1091389 & 0.1458359 & -0.09090507 \end{pmatrix},$$

$$A_2 = \begin{pmatrix} 0.01833149 & 0.03661770 & 0.01824331 & 0.03660918 \\ 0.01837542 & -0.05442525 & 0.0 & 0.01832234 \\ 0.01841784 & 0.00003900436 & 0.0 & 0.01836515 \\ 0.01840752 & 0.00001508311 & 0.01839699 & 0.03659170 \end{pmatrix},$$

$$A_3 = \begin{pmatrix} 0.0 & 0.01837967 & 0.0 & 0.0 \\ 0.0 & 0.01843603 & 0.0 & 0.0 \\ 0.0 & 0.01829203 & 0.0 & 0.0 \\ 0.0 & 0.01842778 & 0.0 & 0.0 \end{pmatrix}.$$

Example 3.7.6. *We will consider a displacement structure on the kernel as well in this example, where higher-order zero terms are not perturbed from the initial guess. For the entries of $\Delta\mathcal{A}$ we preserve higher-order zero terms, and allow low-order terms to be perturbed. This is a linearly structured problem, on both the main variable $\Delta\mathcal{A}$ and the auxiliary kernel variable $\mathcal{B}$.*

To ensure the rank constraint holds, we will additionally assume that the kernel, $\widehat{B}$ is in a CREF (while $\mathcal{B}$ is obviously not) and the columns have unit norm. This normalization is (locally) equivalent to the ones discussed in Section 3.6.2. Having $\widehat{B}$ in a CREF ensures that the two kernel vectors are locally linearly independent during the iteration. Of course perturbing both pivots to zero is possible (although this is sub-optimal). In such a scenario linear independence can no longer be guaranteed, and the iteration would need to be reininitialized.

For the initial guess we use $\mathcal{A}^{init} = \mathcal{A}$ and take $\mathcal{B}^{init}$ as

$$\begin{pmatrix} 0.1954059t^2 & 0.0 \\ -0.2526800t - 0.7681472 & -0.06131396t^2 - 0.1839419t + 0.7357675 \\ -0.05727413t^2 - 0.01010720t - 0.1280246 & -0.06131396t^3 - 0.06131396t + 0.1226279 \\ 0.05727413t^2 + 0.4683004t + 0.2560491 & 0.06131396t^3 + 0.4905117t^2 - 0.3065698t - 0.2452558 \end{pmatrix}.$$

Using Algorithm 3 we compute after nine iterations

$$\Delta A_0 \approx \begin{pmatrix} 0.00003841866 & -0.0001970606 & -0.00002444167 & -0.000003273264 \\ 0.00001831140 & -0.00009026377 & 0.00002067189 & -0.0001255102 \\ -0.0001265513 & -0.0001595407 & 0.00003425737 & -0.00007523197 \\ -0.00007666528 & -0.0002773970 & 0.00004057408 & -0.0001720881 \end{pmatrix},$$

$$\Delta A_1 \approx \begin{pmatrix} 0.00001508776 & 0.00003166597 & 0.00004647888 & -0.0001142308 \\ -0.00005872595 & -0.00004487730 & 0.00004547421 & -0.0001483973 \\ 0.00002056901 & -0.0001596527 & -0.000006413632 & -0.00006541721 \\ -0.00003695701 & -0.0001773889 & 0.00004119722 & -0.0002159825 \end{pmatrix},$$

$$\Delta A_2 \approx \begin{pmatrix} -0.00003352295 & -0.0001190577 & 0.00005687700 & -0.0001783770 \\ 0.00001768442 & -0.0001467423 & 0.0 & -0.00008587235 \\ -0.00006506345 & 0.00005243135 & 0.0 & -0.0001686619 \\ -0.0001471227 & -0.0001295490 & -0.00001105246 & -0.0001124559 \end{pmatrix},$$

$$\Delta A_3 \approx \begin{pmatrix} 0.0 & -0.0001025690 & 0.0 & 0.0 \\ 0.0 & -0.0001315095 & 0.0 & 0.0 \\ 0.0 & -0.00002763942 & 0.0 & 0.0 \\ 0.0 & -0.0001877673 & 0.0 & 0.0 \end{pmatrix},$$

with $\|\Delta\mathcal{A}\|_F \approx 0.0007844$.

An approximate kernel, $\mathcal{B}$ *is given by*

$$\begin{pmatrix} 0.1955493t^2 + 0.0006874986t - 0.001013023 & 0.0 \\ -0.2542383t - 0.7686061 & -0.06128819t^2 - 0.1818298t + 0.7368313 \\ -0.05698735t^2 - 0.01004111t - 0.1276311 & -0.06125293t^3 - 0.0002486115t^2 - 0.06112324t + 0.1226783 \\ 0.05795811t^2 + 0.4677475t + 0.2541290 & 0.06151690t^3 + 0.4894569t^2 - 0.3069667t - 0.2452396 \end{pmatrix}.$$

A natural question is what happens if we change the displacement structure on the kernel? To investigate this behavior, we consider an equivalent representation of the previously used kernel, except that $\mathcal{B}$ is in a CREF directly.

Example 3.7.7. *If we change the kernel* $\mathcal{B}^{init}$ *to be*

$$\begin{pmatrix} 0.1581139t^3 + 0.1581139t - 0.3162278 & 0.03965258t^3 + 0.3172206t^2 - 0.1982629t - 0.1586103 \\ -0.1581139t^2 - 0.4743417t - 0.6324556 & -0.03965258t^2 - 0.4361784t - 0.7930516 \\ 0.0 & 0.07930516t - 0.07930516 \\ 0.3162278t - 0.3162278 & 0.0 \end{pmatrix},$$

used in the initialization of the previous example, then we compute a perturbation with $\|\Delta\mathcal{A}\|_F \approx 0.0008408$.

In either case, we obtain comparable answers that are a reasonable lower-rank approximation, and can likely be improved by relaxing restrictions on the displacement structure on $\mathscr{B}$ or $\widehat{B}$. It is important to note that relaxing the degree bounds to be $(n-r)d$ in general on all non-zero entries (where entries are zero if they are in the same row as a CREF pivot) will likely lead to a better approximation, however one may lose quadratic convergence if doing so, since iterates may no longer have primitive kernel vectors, and (3.12) will no longer hold. As discussed in Section 3.6, it is generally difficult to determine the CREF pivots of the kernel unless the initial guess is very accurate.

The structure of the kernel is an important consideration when deciding upon an initial guess. It is preferable to restrict fewer coefficients, however the iteration requires a better initialization due to the increased number of possible descent directions. In such scenarios for maximum flexibility, a globalized variant of Newton's method is required. Like-wise, the structure for $\Delta\mathscr{A}$ is also an important choice. Restricting which terms can be changed has a large influence on the (approximate) distance to singularity (of prescribed kernel dimension).

Embedded and Evaluated Rank Factorization

Another way to approach the lower-rank approximation problem is to use the factorization in Section 3.4 using Algorithm 1. In these examples hardware precision is used. The advantage of the rank factorization is that it is kernel-free, in a sense that one does not need to have any information about the kernel to compute a lower-rank approximation. In these examples we use default precision (unless noted) in Maple, which is about ten digits.

Example 3.7.8. *Consider the same problem as Example 3.7.7 and Example 3.7.6 without imposing any structure on the kernel and taking $\Delta\mathscr{A}^{init}$ to be the computed perturbation from Example 3.7.7 as an initial guess.*

We compute the same solution of Example 3.7.6. In fact, initializing with the solution to Example 3.7.6 yields a stationary point immediately.

The solution is computed using an implementation of Algorithm 1 based on a regularized Newton method, that requires 12 iterations to reach approximately 13 digits of accuracy.

Example 3.7.9. *Consider the same problem as Example 3.7.7and Example 3.7.6 without imposing any structure on the kernel and taking $\Delta\mathscr{A}^{init}$ to be the computed perturbation from Example 3.7.7 as an initial guess.*

We compute the same solution of Example 3.7.6. In fact, initializing with the solution to Example 3.7.6 yields a stationary point immediately.

*The solution is computed using an implementation of Algorithm 2 using Newton's method, that requires one iteration to reach approximately 12 digits of accuracy with 15 digits of precision. We note that if the default precision is used, then two iterations of Newton's method yields an answer that is accurate to roughly 11 digits. This discrepancy arises from the use of **`evalf`** during intermediate computations in Maple.*

Example 3.7.10. *Using the rank factorization and the previously computed rank 2 approximation, we can compute a rank 1 approximation via the rank factorization, without normalizing the kernel.*

We compute the following rank 1 approximation by iteratively constructing four lower-rank approximations via Algorithm 3.

A rank 1 approximation of $\mathcal{A}$ is $\mathcal{A} + \Delta\mathcal{A}$ where

$$\Delta A_0 \approx \begin{pmatrix} -0.009816368 & 0.002504998 & 0.001000535 & 0.004258998 \\ 0.1228952 & -0.02194259 & -0.02952957 & -0.07934090 \\ 0.04005767 & -0.006141763 & -0.01077717 & -0.02855589 \\ -0.01808597 & 0.001707572 & 0.001273578 & 0.01423062 \end{pmatrix},$$

$$\Delta A_1 \approx \begin{pmatrix} -0.02321372 & 0.004133564 & 0.006380596 & 0.01062149 \\ 0.01411873 & -0.02242337 & -0.01471757 & 0.04536903 \\ -0.01495545 & -0.006513157 & -0.0004289664 & 0.02884990 \\ -0.03764955 & 0.01307587 & 0.009973064 & 0.01222832 \end{pmatrix},$$

$$\Delta A_2 \approx \begin{pmatrix} -0.01131022 & -0.007078220 & -0.01824331 & -0.001345216 \\ -0.01632458 & 0.06305340 & 0.0 & -0.008022115 \\ -0.01500028 & 0.01433914 & 0.0 & -0.001200648 \\ -0.01422351 & 0.01758763 & -0.01839699 & -0.01557774 \end{pmatrix},$$

$$\Delta A_3 \approx \begin{pmatrix} 0.0 & -0.01837967 & 0.0 & 0.0 \\ 0.0 & -0.01843603 & 0.0 & 0.0 \\ 0.0 & -0.01829203 & 0.0 & 0.0 \\ 0.0 & -0.01842778 & 0.0 & 0.0 \end{pmatrix},$$

with $\|\Delta\mathcal{A}\|_F \approx 0.2007488$.

The final iterative approximation converges to hardware precision in three steps once $\|\nabla L(x^{(k)})\|_2 \approx 5.9 \times 10^{-5}$. Prior to this, the rate of convergence is approximately linear. The Newton-like method used required roughly 86 iterations to obtain convergence (although this can be improved using various heuristics).

3.8 Conclusion

We have shown that finding lower-rank approximations of matrix polynomials can be established as a numerically well-posed problem and is amenable to first and second-order optimization methods. The existence and isolation of solutions is established along with algorithms exploiting the affine structures to obtain locally quadratic convergence under mild normalization assumptions.

Along with considering the lower-rank approximation of matrix polynomials, we present a generalization of the theory to matrix polynomials with an arbitrary affine structure. We provide examples of how the structure of permissible perturbations and prescribed kernel structure impacts the distance to solutions. The rank factorization does not depend on the structure of the kernel, however the trade off is a cost that can be prohibitively large. This can be remedied in practice by using evaluation and computing rank factorizations of several images simultaneously that are several orders of magnitude smaller. Of course the issue here is with the precision of the computed solution, which means the method may be fast but not accurate. There is no free lunch: of all of the methods studied rely on some form of normalization or are prohibitively expensive.

We also regard this chapter as a first step towards a formally robust approach to nonlinear matrix polynomials, in the spirit of recent work with symbolic-numeric algorithms for polynomials. Problems such as approximate matrix polynomial division, GCRD and factorization all have applications which can benefit from these modern tools.

Chapter 4

Matrix Polynomial Determinants, Adjoints, and their Derivatives

This chapter discusses computing the determinant and adjoint of a matrix polynomial in a numerically robust and efficient manner. Computing the adjoint matrix, sometimes known as the classical adjoint or adjugate, is a classical linear algebra problem. The results presented in this chapter are based on a manuscript of [52] and the preliminary section of [39]. The determinant and adjoint matrix appear naturally in optimization problems related to the spectral structure of scalar and polynomial matrices, and are a necessary component of Chapter 5.

4.1 Introduction

The problem of computing the adjoint matrix over an arbitrary field is well understood as a symbolic computation problem [101] and exact quasi-optimal $\widetilde{O}(n^3d)$ field operation algorithms exist, even when $\mathcal{A}$ is rank deficient. Likewise for the determinant, fast $\widetilde{O}(n^3d)$ field operation algorithms exist [67] as well. The floating point matrix polynomial adjoint has not been extensively studied previously in the literature to the best of the knowledge of the author. Suppose that we are given $\mathcal{A} \in \mathsf{R}[t]^{n\times n}$ of degree at most d. When $\mathcal{A}$ has full rank, a popular algorithm in control theory involves computing the adjoint matrix to compute $\mathcal{A}^{-1} \in \mathsf{R}(t)^{n\times n}$. In this instance the inverse is completely defined by the relationship $\det(\mathcal{A})^{-1}\operatorname{Adj}(\mathcal{A}) = \mathcal{A}^{-1}$. While the adjoint matrix can provide insight into the behavior of the inverse of a matrix polynomial, our interest is in using the entries of the adjoint matrix as a constraint in optimization problems. This naturally necessitates computing the adjoint matrix quickly in a robust manner using floating point arithmetic, and computing the first two derivatives. The entries of the adjoint matrix reveal partial information about the spectral structure of a matrix polynomial (see Chapter 5 for a detailed discussion). These applications arise since the adjoint matrix consists of the $(n-1) \times (n-1)$ minors of $\mathscr{A}$.

In the setting of floating point arithmetic, most of the previous work for computing adjoint matrices was done for the scalar instance of $d = 0$ by Stewart [100]. The work of Stewart [100] consists of a forward error analysis and some techniques to approximate the adjoint matrix of $A \in \mathsf{R}^{m \times n}$. The general idea to compute the adjoint relies on the SVD, QR decomposition, or some other combination of triangular or diagonal rank revealing factorization. These numerical routines typically require $O(mn^2)$ FLOPs.

Other techniques for the scalar instance involve computing the determinant numerically and using automatic differentiation. The determinant when computed using floating arithmetic is typically accomplished via a modified QR decomposition using Householder reflections that compute $\det(Q)$ iteratively via the scaling of the Householder reflection used in the orthogonal (unitary) elimination. Using the identity [84, Chapter 8] that $\nabla \det(A) = \operatorname{vec}(\operatorname{Adj}(A)^T)^T$, computing the determinant is effectively the same as computing the adjoint matrix when backwards mode automatic differentiation is used [89]. Computing the adjoint in this manner requires $O(n^3)$ FLOPs, assuming $m = n$.

The matrix polynomial determinant was historically computed with a modified QR factorization on a linearized matrix pencil. Several of the algorithms found in the literature typically require computing n^2 determinants of $(n-1)\times(n-1)$ matrix polynomials of degree at most d. If one assumes a determinant cost of $\widetilde{O}(n^4 d)$ FLOPs or $O(n^3 d^3)$ FLOPs, then these routines to compute the adjoint typically cost $\widetilde{O}(n^2 \times n^4 d)$ FLOPs or $O(n^2 \times n^3 d^3)$ FLOPs. Note that the determinant can be computed using the stated number of FLOPs via an interpolation based approach or an algorithm analogous to the QZ decomposition. The cost of these naive techniques are far from the information theoretical lower bound of $\Omega(n^3 d)$, since $\operatorname{Adj}(\mathscr{A})$ generally has $\Theta(n^3 d)$ coefficients.

At the moment there are no known (to the author) *robust* algorithms for matrix polynomial adjoint that use $\widetilde{O}(n^3 d)$ FLOPs. We review some existing algorithms that require $\widetilde{O}(n^4 d)$ FLOPs and discuss algorithms that can be faster in some special cases. We propose some algorithms based on two sided unitary decompositions that require $\widetilde{O}(n^3 d^3)$ FLOPs. In general, we can compute the adjoint matrix in $\widetilde{O}(n^4 d)$ or $\widetilde{O}(n^3 d^3)$ FLOPs using "off the shelf" numerical linear algebra routines in most instances.

4.1.1 Outline

This chapter has the following objectives:

1. Derive closed-form expressions for the first-order derivatives of the determinant and adjoint matrices.
2. Use the closed-form derivative information to obtain condition number estimates for $\det(\cdot)$ and $\mathrm{Adj}(\cdot)$.
3. Discuss several algorithms for computing the adjoint of a matrix polynomial that are numerically stable and reasonably fast.
4. Discuss how to compute the first two derivatives of adjoint matrix using floating point arithmetic.
5. Use fast determinant and adjoint computation results to derive a formulation of the nearest singular square matrix polynomial without auxiliary variables as an application of these techniques.

4.2 Overview of Existing Results and Techniques

This section reviews some basic results about the adjoint matrix and determinant computation.

Lemma 4.2.1 ([100])**.** *Suppose that $A \in \mathbb{C}^{n\times n}$ and $A = U\Sigma V^*$ is a singular value decomposition of A. Then*

$$\mathrm{Adj}(A) = \det(U)\det(V)V\Gamma U^*,$$

where $\Gamma = \mathrm{diag}(\gamma_1, \ldots, \gamma_n)$ and $\gamma_j = \prod_{\substack{i=1\\ i\neq j}}^{n} \sigma_i$. If A has full rank then this is equivalent to writing

$$\mathrm{Adj}(A) = \det(U)\det(V)\det(\Sigma)V\Sigma^{-1}U^*.$$

Note that this result holds independent of the rank of A by continuity. Since $\det(U)$ and $\det(V)$ are just complex numbers on the unit disk, this is an SVD of $\mathrm{Adj}(A)$. If U and V are real, i.e. orthogonal instead of unitary, then $\det(U), \det(V) \in \{\pm 1\}$. Stewart [100] discusses several algorithms for computing (approximating) the adjoint matrix and provides a first-order perturbation theory. We do not explicitly make use of these algorithms, but it is important to mention that they exist and can compute (approximate) the adjoint matrix even if A is (nearly) singular.

While [100] essentially performs a forward error analysis on the problem, a backwards error analysis is missing from the literature. In several meaningful instances, computing the adjoint is not even backwards stable. For a problem to be backwards stable, it must compute the exact solution to a "nearby" problem. The issue with this is that the adjoint operator is not *surjective* [11], even when A has full rank. What this means is that there may not exist a $C \in \mathbb{R}^{n\times n}$ such that $C = \mathrm{Adj}(B)$ for some $B \in \mathbb{R}^{n\times n}$. Fortunately, this only occurs when n is odd, i.e. $n = 2k+1 \in \mathbb{Z}_{\geq 0}$.

Beslin [11] shows that for every $A \in \mathsf{R}^{n\times n}$ ($\mathsf{R} = \mathbb{R}$ or $\mathbb{C}$) of full rank we have that $A = \mathrm{Adj}(\omega A^{-1})$ where $\omega \in \mathbb{C}$ satisfies $\omega^{n-1} = \det(A)$. Furthermore:

1. For every $A \in \mathbb{C}^{n\times n}$ of full rank, there exists $C \in \mathbb{C}^{n\times n}$ such that $\mathrm{Adj}(C) = A$.
2. For every $A \in \mathbb{R}^{2n\times 2n}$ of full rank, there exists $C \in \mathbb{R}^{2n\times 2n}$ such that $\mathrm{Adj}(C) = A$.
3. For every $A \in \mathbb{R}^{n\times n}$ of full rank, there exists $C \in \mathbb{C}^{n\times n}$ such that $\mathrm{Adj}(C) = A$.

For the case of full rank scalar matrices, the problem will generally be backwards stable if it is well-conditioned and we permit complex perturbations to the coefficients. In the special case of matrices of even dimension, the problem is backwards stable over $\mathbb{R}$.

To illustrate why the problem is not backwards stable when n is odd, we will provide an example.

Example 4.2.2. *Let $A \in \mathbb{R}^{3\times 3}$ with $\det(A) = -1$. Suppose that there exists $C \in \mathbb{R}^{3\times 3}$ such that $\mathrm{Adj}(C) = A$. Then $\det(C)^2 = \det(A) = -1$, and so one would conclude that $(-1)^{1/2} \in \mathbb{R}$, which is clearly false.*

While we can simply tweak the domain of computation with scalar matrices to obtain satisfactory results, this is not necessarily the case with matrix polynomials. Again, we will illustrate this with an example.

Example 4.2.3. *Let $\mathcal{A} \in \mathbb{R}[t]^{3\times 3}$ with $\det(A) = -t$. Suppose that there exists $\mathcal{C} \in \mathbb{R}[t]^{3\times 3}$ such that $\mathrm{Adj}(\mathcal{C}) = \mathcal{A}$. By assumption we have that $\det(\mathcal{A}) = -t$. However, $\det(\mathcal{C})^2 = \det(\mathcal{A}) = -t$, so one concludes that $(-t)^{1/2} \in \mathbb{R}[t]$, which is false.*

In the case of matrix polynomials, it is not even adequate to consider perturbations over $\mathbb{C}[t]^{n\times n}$. Rather, we would need to work over a far more complicated algebraic domain including radicals. In this scenario, it makes more sense to study a model of mixed stability. Formally, if $\widetilde{\mathrm{Adj}}(\cdot)$ is an approximation to $\mathrm{Adj}(\cdot)$, i.e. $\widetilde{\mathrm{Adj}}(\mathcal{A}) = \mathrm{Adj}(\mathcal{A}) + \mathcal{B}$ for some $\mathcal{B} \in \mathsf{R}[t]^{n\times n}$ is a numerical approximation to $\mathrm{Adj}(\cdot)$. Then there exists $\mathcal{C} \in \mathsf{R}[t]^{n\times n}$ such that $\mathrm{Adj}(\mathcal{A}+\mathcal{C}) = \mathrm{Adj}(\mathcal{A}) + \mathcal{B}$ and $\|\mathcal{B}\|$ and $\|\mathcal{C}\|$ are sufficiently small.

In order to study the notion of backwards and mixed stability in detail, we will need to analyze the behavior of $\nabla \mathrm{Adj}(\cdot)$, that is how the derivative of $\mathrm{Adj}(\cdot)$ behaves as a vector-valued function from $\mathsf{R}^{n^2(d+1)} \to \mathsf{R}^{n^2((n-1)d+1)}$ (note that $d = 0$ includes the scalar case without loss of generality). We will also study computing the second-order derivatives of $\mathrm{Adj}(\cdot)$ later, since it is of interest for optimization problems involving the adjoint.

4.3 The First Derivative of the Determinant

The first task at hand is to understand the Jacobian of the determinant. If $\mathcal{A} \in \mathsf{R}[t]^{n\times n}$ has degree at most d then $\det(\cdot) : \mathsf{R}^{n^2(d+1)} \to \mathsf{R}^{n^2(nd+1)}$, since $\deg(\det(\mathcal{A})) \le nd$, with equality holding generically. In our applications we generally take $\mathsf{R} = \mathbb{R}$ since we are primarily concerned with problems over the real numbers. The theory developed will hold [59] if one considers $\mathsf{R} = \mathbb{C}$ as well, and so we present the results in the most general form. If $A \in \mathsf{R}^{n\times n}$ has full rank, then it is well known that $\nabla(\det(A)) = \mathrm{vec}(\mathrm{Adj}(A)^T)^T$, with the result holding for $\mathsf{R} = \mathbb{R}$ or $\mathsf{R} = \mathbb{C}$.

Intuitively one would suspect that it is straight-forward to extract the first-order derivative from the adjoint operator by the multi-linearity of the determinant operator. The issue is each entry of $\mathcal{A}$ is essentially a vector of $d+1$ entries, which complicates the matter somewhat. Informally, one can make a linear substitution and apply the chain rule to obtain the derivative of the $d+1$ entries. Using this idea, one notes that

$$\det(\mathcal{A} + \Delta\mathcal{A}) = \det(\mathcal{A}) + \mathrm{pvec}(\mathrm{Adj}(\mathcal{A})^T)^T\,\mathrm{pvec}(\Delta\mathcal{A}) + O(\|\Delta\mathcal{A}\|_F^2),$$

or the scalar expression (ignoring higher-order terms)

$$\mathrm{vec}(\det(\mathcal{A} + \Delta\mathcal{A})) \approx \mathrm{vec}(\det(\mathcal{A})) + \mathrm{vec}(\mathrm{pvec}(\mathrm{Adj}(\mathcal{A})^T)^T\,\mathrm{pvec}(\Delta\mathcal{A})).$$

Recall from Definition 3.1.1 that $\Delta\mathcal{A}$ is an unstructured perturbation.

The Jacobian can be extracted by (padding with zero coefficient entries as necessary) writing $\mathrm{vec}(\mathrm{pvec}(\mathrm{Adj}(\mathcal{A})^T)^T\,\mathrm{pvec}(\Delta\mathcal{A})) = J_{\det}\,\mathrm{vec}(\Delta\mathcal{A})$ as a matrix-vector product. Thus, using block-convolution matrices we have

$$\frac{\partial\,\mathrm{vec}(\det(\mathcal{A}))}{\partial\,\mathrm{vec}(\mathcal{A})} = \nabla(\det(\mathcal{A})) = \Phi_d(\mathrm{pvec}(\mathrm{Adj}(\mathcal{A})^T)^T).$$

Theorem 4.3.1. *Let $\mathcal{A} \in \mathsf{R}[t]^{n\times n}$ have degree at most d, then*

$$J_{\det} = \frac{\partial \operatorname{vec}(\det(\mathcal{A}))}{\partial \operatorname{vec}(\mathcal{A})} = \Phi_d(\operatorname{pvec}(\operatorname{Adj}(\mathcal{A})^T)^T) \in \mathsf{R}^{(nd+1)\times n^2(d+1)}.$$

Proof. The proof follows immediately from the discussion above, as the Jacobian provides the best first-order approximation. □

It is not surprising that the matrix polynomial determinant admits a derivative similar to the scalar case given the similar structure. In fact, we can derive the same expression via interpolation of the first-order approximation and using the result of the degree zero instance. Later we will need to compute the derivative of the adjoint operator, and this is essentially computing the second derivative of the determinant. In several instances $J_{\det}$ can be padded with rows of zeros, such as when the determinant has non-zero degree.

4.3.1 First-Order Perturbation Bounds for the Matrix Polynomial Determinant

Corollary 4.3.2. *Theorem 4.3.1 implies the first-order local perturbation bound (ignoring higher-order terms)*

$$\frac{\|\det(\mathcal{A}) - \det(\mathcal{A}+\Delta\mathcal{A})\|_2}{\|\det(\mathcal{A})\|_2} \leq \frac{\|J_{\det}\operatorname{vec}(\Delta\mathcal{A})\|_2}{\|\det(\mathcal{A})\|_2} \leq \frac{(d+1)\|\operatorname{Adj}(\mathcal{A})\|_F\|\Delta\mathcal{A}\|_F}{\|\det(\mathcal{A})\|_2}.$$

Now, we can re-write this bound by noting that

$$\operatorname{Adj}(\mathcal{A})\mathcal{A} = \det(\mathcal{A})I \implies \sqrt{n}\|\det(\mathcal{A})\|_2 = \|\operatorname{Adj}(\mathcal{A})\mathcal{A}\|_F,$$

thus

$$\frac{\|\det(\mathcal{A}) - \det(\mathcal{A}+\Delta\mathcal{A})\|_2}{\|\det(\mathcal{A})\|_2} \leq (d+1)\sqrt{n}\frac{\|\operatorname{Adj}(\mathcal{A})\|_F\|\Delta\mathcal{A}\|_F}{\|\mathcal{A}\operatorname{Adj}(\mathcal{A})\|_F}.$$

If $A \in \mathsf{R}^{n\times n}$, then $\dfrac{\|\operatorname{Adj}(A)\|_2}{\|A\operatorname{Adj}(A)\|_2} = \|A^{-1}\|_2$. Due to our choice of coefficient norm we are not able to handle matrices of rational functions, however the ratio

$$\frac{\|\mathcal{A}\|_F\|\operatorname{Adj}(\mathcal{A})\|_F}{\|\mathcal{A}\operatorname{Adj}(\mathcal{A})\|_F} = \frac{\|\mathcal{A}\|_F\|\det(\mathcal{A})\mathcal{A}^{-1}\|_F}{\sqrt{n}\|\det(\mathcal{A})\|_F}$$

can be thought of as an analog to the classical scalar (degree zero matrix polynomial instance) condition number. It should be noted that one can use Hadamard's inequality to obtain a (poor) bound [1] on $\| \operatorname{Adj}(\mathscr{A})\|$ and the other related quantities interms of $\|\mathscr{A}\|$ and n. In the scalar instance using $\| \cdot \|_2$, our bounds are the same as the ones obtained in [64] (accounting for the choice of $\| \cdot \|$).

4.4 The First Derivative of the Adjoint

While we have a perturbation theory for the scalar adjoint, we are missing one for the matrix polynomial adjoint. Naturally, a first-order series expansion would provide insights into the problem.

Recall that $\det(\mathscr{A})I = \mathscr{A} \operatorname{Adj}(\mathscr{A})$, although this relationship does not define the adjoint when $\mathscr{A}$ is rank deficient. If $\mathscr{A}$ is rank deficient, then either $\operatorname{Adj}(\mathscr{A})$ has rank one or is the zero matrix. For the purposes of deriving an error analysis, we are generally not concerned with these cases.

We can write

$$\Phi_{(n-1)d}(I \otimes \mathscr{A}) \operatorname{vec}(\operatorname{Adj}(\mathscr{A})) = \operatorname{vec}(\det(A)I),$$

which is a scalar matrix equation, and from which estimates from $\dfrac{\partial \operatorname{vec}(\operatorname{Adj}(\mathscr{A}))}{\partial \operatorname{vec}(\mathscr{A})}$ can be extracted. We note that $\Phi_{(n-1)d}(I \otimes \mathscr{A})$ is pseudo-invertible if $\mathscr{A}$ has full rank. The Jacobian of $\operatorname{Adj}(\cdot)$ can be extracted from this form by performing matrix calculus (see [84, Chapter 9]).

4.4.1 Computing the First Derivative

Now, we formally state the result and include a technical proof.

Theorem 4.4.1. *Let $\mathscr{A} \in \mathsf{R}[t]^{n\times n}$ have degree at most d and rank n. The Jacobian of $\operatorname{Adj}(\mathscr{A})$ is $J_{\operatorname{Adj}} \in \mathsf{R}^{n^2((n-1)d+1)\times n^2(d+1)}$ with*

$$J_{\operatorname{Adj}} = \left[\Phi_{(n-1)d}(I \otimes \mathscr{A})\right]^+ \left[\Phi_d(\operatorname{pvec}(I) \operatorname{pvec}(\operatorname{Adj}(\mathscr{A})^T)^T) - \Phi_d(\operatorname{Adj}(\mathscr{A})^T \otimes I)\right],$$

where I is understood to be the $n \times n$ identity matrix and for a scalar matrix A of full rank, A^+ is the Moore-Penrose pseudo-inverse arising from the SVD.

[1]The bound is exponential and too large to be of practical use.

Proof. First recall that if $\mathscr{A}$ has full rank, then $\mathscr{A}\operatorname{Adj}(\mathscr{A}) = \operatorname{Adj}(\mathscr{A})\mathscr{A} = \det(\mathscr{A})I$. This expression defines the adjoint matrix when $\mathscr{A}$ has full rank. We can write

$$\operatorname{pvec}(\mathscr{A}\operatorname{Adj}(\mathscr{A})) = (\operatorname{Adj}(\mathscr{A})^T \otimes I)\operatorname{pvec}(\mathscr{A}) = (I \otimes \mathscr{A})\operatorname{pvec}(\operatorname{Adj}(\mathscr{A})),$$

thus converting to a linear system over R produces

$$\operatorname{vec}(\mathscr{A}\operatorname{Adj}(\mathscr{A})) = \Phi_{(n-1)d}(I \otimes \mathscr{A})\operatorname{vec}(\operatorname{Adj}(\mathscr{A})) = \Phi_d(\operatorname{Adj}(\mathscr{A})^T \otimes I)\operatorname{vec}(\mathscr{A}).$$

Applying the product rule yields

$$\partial\operatorname{vec}(\mathscr{A}\operatorname{Adj}(\mathscr{A})) = (\partial\Phi_{(n-1)d}(I\otimes\mathscr{A}))\operatorname{vec}(\operatorname{Adj}(\mathscr{A})) + \Phi_{(n-1)d}(I\otimes\mathscr{A})\partial\operatorname{vec}(\operatorname{Adj}(\mathscr{A})). \quad (4.1)$$

Next we observe that (4.1) has the same coefficients as the expression

$$\operatorname{vec}((\partial\mathscr{A})\operatorname{Adj}(\mathscr{A}) + \mathscr{A}(\partial\operatorname{Adj}(\mathscr{A})))$$

which is equivalent to

$$\operatorname{vec}((\operatorname{Adj}(\mathscr{A})^T \otimes I)\operatorname{pvec}(\partial\mathscr{A}) + (I \otimes \mathscr{A})\operatorname{pvec}(\partial\operatorname{Adj}(\mathscr{A}))),$$

which reduces to

$$\Phi_d((\operatorname{Adj}(\mathscr{A})^T \otimes I))\operatorname{vec}(\partial\mathscr{A}) + \Phi_{(n-1)d}(I \otimes A)\operatorname{vec}(\partial\operatorname{Adj}(\mathscr{A})). \quad (4.2)$$

We now have the derivative of the left hand side of the expression $\mathscr{A}\operatorname{Adj}(\mathscr{A}) = \det(\mathscr{A})I$. Differentiation of the right hand side yields

$$\partial\operatorname{vec}(\det(\mathscr{A})I) = \operatorname{vec}(\partial\operatorname{pvec}(\det(\mathscr{A})I)),$$

which is equivalent to the expression

$$\operatorname{vec}(\partial\operatorname{pvec}(\det(\mathscr{A})I)) = \operatorname{vec}(\operatorname{pvec}(I)\operatorname{pvec}(\operatorname{Adj}(\mathscr{A})^T)^T\operatorname{pvec}(\partial\mathscr{A})). \quad (4.3)$$

Converting (4.3) into a linear system over R leads to

$$\operatorname{vec}(\operatorname{pvec}(I)\operatorname{pvec}(\operatorname{Adj}(\mathscr{A})^T)^T)\operatorname{pvec}(\partial\mathscr{A}) = \Phi_d(\operatorname{pvec}(I)\operatorname{pvec}(\operatorname{Adj}(\mathscr{A})^T)^T)\operatorname{vec}(\partial\mathscr{A}), \quad (4.4)$$

which is the derivative of the right-hand side.

Combining (4.2) and (4.4) we have

$$\Phi_{(n-1)d}(I \otimes \mathcal{A}) \frac{\partial \operatorname{vec}(\operatorname{Adj}(\mathcal{A}))}{\partial \operatorname{vec}(\mathcal{A})} = \Phi_d(\operatorname{pvec}(I) \operatorname{pvec}(\operatorname{Adj}(\mathcal{A})^T)^T) - \Phi_d(\operatorname{Adj}(\mathcal{A})^T \otimes I).$$

Assuming that $\mathcal{A}$ has full rank so $\Phi_{(n-1)d}(\operatorname{pvec}(I \otimes \mathcal{A}))$ is pseudo-invertible, we can write

$$J_{\mathrm{Adj}} = \left[\Phi_{(n-1)d}(I \otimes \mathcal{A})\right]^+ \left[\Phi_d(\operatorname{pvec}(I) \operatorname{pvec}(\operatorname{Adj}(\mathcal{A})^T)^T) - \Phi_d(\operatorname{Adj}(\mathcal{A})^T \otimes I)\right],$$

which completes the proof. □

An observation that is important later is that the derivative of the adjoint has a Toeplitz-block structure. More importantly, the bandwidth is $O(d)$, and we only need to compute $O(n^2)$ columns instead of $O(n^2 d)$. We also note that J_{Adj} may be padded with zeros, since $\mathcal{A}$ may not have generic degrees.

Corollary 4.4.2. *If $\mathcal{A}$ has full rank then J_{Adj} has full rank.*

Proof. The matrix $\Phi_{(n-1)d}(I \otimes \mathcal{A})$ has full rank since $I \otimes \mathcal{A}$ has full rank. The matrix

$$\operatorname{pvec}(I) \operatorname{pvec}(\operatorname{Adj}(\mathcal{A})^T)^T - \operatorname{Adj}(\mathcal{A})^T \otimes I = -\left(-\operatorname{pvec}(I) \operatorname{pvec}(\operatorname{Adj}(\mathcal{A})^T)^T + \operatorname{Adj}(\mathcal{A})^T \otimes I\right) \tag{4.5}$$

is a rank one update to a matrix polynomial. By evaluating (4.5) at a complex number ω that is not an eigenvalue of $\mathcal{A}$ we can show that (4.5) has full rank. Let $A = \mathcal{A}(\omega)$, so $A \in \mathbb{C}^{n \times n}$ has full rank.

Using the Sherman-Morrison formula [57, pg. 487] for rank 1 updates to a matrix, we need to verify that

$$1 - \operatorname{vec}(\operatorname{Adj}(A)^T)^T \left[\left(\operatorname{Adj}(A)^T\right)^{-1} \otimes I\right] \operatorname{vec}(I) \neq 0,$$

in order to ensure that (4.5) has full rank. We have that

$$\begin{aligned} \operatorname{vec}(\operatorname{Adj}(A)^T)^T \left[\left(\operatorname{Adj}(A)^T\right)^{-1} \otimes I\right] \operatorname{vec}(I) &= \operatorname{vec}(\operatorname{Adj}(A)^T)^T \operatorname{vec}\left(\operatorname{Adj}(A)^T)^{-1}\right) \\ &= \operatorname{Tr}\left(\operatorname{Adj}(A)^T \left(\operatorname{Adj}(A)^T\right)^{-1}\right) \\ &= n, \end{aligned}$$

thus (4.5) has full rank. Note we have used the identities for matrices X, Y and Z of appropriate dimension, that $\operatorname{vec}(XYZ) = (Z^T \otimes X)\operatorname{vec}(Y)$ and $\operatorname{vec}(X^T)^T \operatorname{vec}(Y) = \operatorname{Tr}(XY)$. Again, we have that

$$\Phi_d(\operatorname{pvec}(I)\operatorname{pvec}(\operatorname{Adj}(\mathcal{A})^T)^T) - \Phi_d(\operatorname{Adj}(\mathcal{A})^T \otimes I)$$

has full rank, thus J_{Adj} is a product of two matrices of full rank, and so J_{Adj} must also have full rank. □

Corollary 4.4.2 implies that Lagrange multipliers will exist to several optimization problems involving the adjoint matrix as a constraint, since the Jacobian matrix of the adjoint has full rank. The linear independent constraint qualification or the constant rank constraint qualification will hold for several optimization problems of the form

$$\min \|\Delta\mathcal{A}\| \ \text{ subject to } \ \operatorname{Adj}(\mathcal{A} + \Delta\mathcal{A}) = \mathcal{F},$$

for some reasonably prescribed $\mathcal{F} \in \mathbb{R}[t]^{n\times n}$.

Remark 4.4.3. *If $\mathcal{A}$ is rank deficient, then the derivative is still defined, but not necessarily by Theorem 4.4.1. If $\operatorname{rank}(\mathcal{A}) \le n-3$ then $J_{\mathrm{Adj}} = 0$, since all $(n-3)\times(n-3)$ minors vanish (J_{Adj} consists of the coefficients of these minors). If $\operatorname{rank}(\mathcal{A}) = n-1$ or $\operatorname{rank}(\mathcal{A}) = n-2$ then J_{Adj} is still defined and in both cases $J_{\mathrm{Adj}} \neq 0$. However J_{Adj} is not necessarily described by Theorem 4.4.1.*

For several affine or linear perturbation structures (such as ones that preserve the degree of entries or the support of entries), Theorem 4.4.1 and the associated Corollary 4.4.2 will hold (after deleting some extraneous rows or columns).

For the purpose of differentiation techniques, we need at most $O(n^2)$ adjoint calls to compute the first derivative and at most $O(n^4)$ adjoint calls to compute the second derivative. Since $\operatorname{Adj}(\cdot) : \mathsf{R}^{n^2(d+1)} \to \mathsf{R}^{n^2((n-1)d+1)}$, forward mode automatic differentiation is ideal. If the cost to compute the adjoint is $\widetilde{O}(n^4d^2)$ FLOPs, then with respect to using a Newton method on a problem with constraints involving the adjoint matrix, this is quasi-optimal with respect to naive (cubic) matrix inversion. Note that the adjoint has a size of $O(n^3d)$ (assuming floating point representation), so the Hessian would have dimension $\Omega(n^3d) \times \Omega(n^3d)$. Using standard (non asymptotically fast) linear algebra techniques, this would require roughly $\Omega(n^9d^3)$ FLOPs to invert the Hessian matrix. For the purpose of a naive second-order optimization technique that works on the Hessian, we do not need quasi-optimal adjoint computation.

Again, a similar observation holds for the determinant as well and we exploit this in our implemented algorithms.

4.4.2 First-Order Perturbation Bounds for the Matrix Polynomial Adjoint

Theorem 4.4.1 has the following corollary.

Corollary 4.4.4. *If we let* $c = \left\| \left[\Phi_{(n-1)d}(I \otimes \mathcal{A}) \right]^{+} \right\|_2$*, then we have the relative first-order perturbation bound (ignoring higher-order terms)*

$$\begin{aligned}\frac{\| \operatorname{Adj}(\mathcal{A}) - \operatorname{Adj}(\mathcal{A} + \Delta\mathcal{A}) \|_F}{\| \operatorname{Adj}(\mathcal{A}) \|_F} &\leq \frac{c(n+\sqrt{n})(d+1) \| \operatorname{Adj}(\mathcal{A}) \|_F}{\| \operatorname{Adj}(\mathcal{A}) \|_F} \| \Delta\mathcal{A} \|_F \\ &= c(n+\sqrt{n})(d+1) \| \Delta\mathcal{A} \|_F.\end{aligned}$$

This quantity also implicitly bounds the second-order condition number[2] of $\det(\cdot)$ when $\operatorname{rank}(\mathcal{A}) = n$.

Corollary 4.4.4 is analogous to the degree zero case [84], where

$$\frac{\partial \operatorname{vec}(\operatorname{Adj}(A))}{\partial \operatorname{vec}(A)} = \det(A) \left[\operatorname{vec}(A^{-1}) \operatorname{vec}((A^{-1})^T)^T - ((A^{-1})^T \otimes A^{-1}) \right].$$

In fact, if we take $d = 0$, then both results agree with each other (up to the choice of norm).

This produces the perturbation bound (ignoring higher-order terms)

$$\begin{aligned}&\| \operatorname{Adj}(A) - \operatorname{Adj}(A + \Delta A) \|_2 \\ \approx &\left\| \left(\frac{1}{\det(A)} \operatorname{vec}(\operatorname{Adj}(A)) \operatorname{vec}(\operatorname{Adj}(A)^T)^T - \operatorname{Adj}(A)^T \otimes \operatorname{Adj}(A) \right) \operatorname{vec}(\Delta A) \right\|_2 \\ \leq &\left\| \left(\frac{1}{\det(A)} \operatorname{vec}(\operatorname{Adj}(A)) \operatorname{vec}(\operatorname{Adj}(A)^T)^T - \operatorname{Adj}(A)^T \otimes \operatorname{Adj}(A) \right) \right\|_2 \| \operatorname{vec}(\Delta A) \|_F \\ \leq &\frac{1}{|\det(A)|} (n+\sqrt{n}) \| \operatorname{Adj}(A) \|_2^2 \| \Delta A \|_2 \\ \leq &(n+\sqrt{n}) \| \operatorname{Adj}(A) \|_2 \| A^{-1} \|_2 \| \Delta A \|_2.\end{aligned}$$

If we analyze the first-order perturbation bound directly arising from the instance of $d = 0$, we have

$$\frac{\| \operatorname{Adj}(A) - \operatorname{Adj}(A + \Delta A) \|_2}{\| \operatorname{Adj}(A) \|_2} \leq (n+\sqrt{n}) \| A^{-1} \|_2 \| \Delta A \|_2.$$

[2]The second-order condition number is the "condition number of the condition number". See [58] for a detailed discussion of second-order (and higher) condition numbers.

Note that this is different from [100], which has the first-order bound (ignoring higher-order terms)

$$\frac{\| \operatorname{Adj}(A) - \operatorname{Adj}(A + \Delta A)\|_2}{\|A\|_2} \leq (n - 1 + \sqrt{n-1}) \frac{\|A\|_2}{\sigma_{n-1}(A)} \frac{\|\Delta A\|_2}{\|A\|_2}.$$

Thus, there is possibly room for improvement for $d \geq 1$. However the tools used by [100] (mainly the singular value decomposition) are not available for matrix polynomials. One could proceed by interpolation for example as, the result would hold for the images.

4.5 Floating Point Algorithms for Matrix Polynomial Adjoint

In this section we discuss families of floating point techniques to compute the adjoint of a matrix polynomial, that does not necessarily have full rank.

Naive methods involve computing $O(n^2)$ determinants with a cost of $\Omega(n^3 d)$ FLOPs for each determinant, for a total cost of $\Omega(n^5 d)$ FLOPs. Other naive methods might use floating point polynomial linear algebra directly and use a variant of Gaussian elimination over $\mathsf{R}[t]$ to compute the adjoint. This is generally unstable since Gaussian elimination (or a fraction-free variant) amounts to performing the Euclidean algorithm using floating arithmetic.

Instead, it is often preferred to either solve the underlying polynomial system by expanding the coefficients into a larger linear system and seeking a least squares solution by performing linear algebra over R. An alternative method to this is to solve the problem by evaluation and interpolation. Using complex roots of unity as evaluation points leads to stable interpolations, so the stability depends entirely on the conditioning of the problem at the evaluation points.

Another family of algorithms that have not received much attention is performing a computation using exact arithmetic with a symbolic method, then rounding to the required precision. Such a technique is only viable if the intermediate bit complexity is proportional to the working floating point precision. The advantage is that we have absolute control over how errors are propagated and the problem is straight forward to analyze. The trade off here is that the computation will likely take longer than an algorithm that does not use symbolic computation techniques.

For the purpose of this thesis and arising optimization problems involving the adjoint matrix, we need to be able to compute $\operatorname{Adj}(\mathscr{A})$ in $O(n^4 d^2)$ FLOPs. Recall that $\operatorname{Adj}(\mathscr{A})$

has $O(n^3d)$ entries (as a finite dimensional vector space), and so inverting the blocks of the Hessian with derivatives of the adjoint terms appearing would require $O(n^9d^3)$ FLOPs using standard arithmetic. Computing these blocks in the Hessian would require $O(n^2 \times n^2 \times n^4d^2) = O(n^8d^2)$ FLOPs under this assumption. Most of the algorithms discussed will be able to accomplish this, or they can accomplish this in some special cases (such as d is a constant or significantly smaller than n as a parameter).

4.5.1 Exact Symbolic-Numeric Method

The symbolic-numeric method for computing the adjoint is to treat the input as exact, execute a symbolic-algorithm then round to the required precision. The algorithm presented by Storjohann [101] can be used, which will succeed with high probability. Alternatively, any fast deterministic determinant and inverse algorithm can be used. If the algorithm of Storjohann is used naively, then the number of FLOPs required is proportional [80] to $\widetilde{O}(n^3d(\log(n^n(d+1)^n\|\mathcal{A}\|_\infty^n))) = \widetilde{O}(n^4d^2)$ FLOPs. The issue is that the bit complexity of intermediate terms may be larger than a constant number of words. Hadamard's bound (Goldstein-Graham variant) provides a very rough upper bound of $\|\operatorname{Adj}(\mathcal{A})\|_\infty \in O(\text{Poly}(n,d)\|\mathcal{A}\|_\infty^{n-1}(d+1)^{n-1}n^{(n-1)/2})$. Given that $\|\mathcal{A}\|_\infty$ is constant for numerical purposes, the issue is that after rounding, the size in terms of the bit complexity of the problem is $O(n^3d)$, but the size of the exact solution is $O(n^4d^2)$.

The technique of [101] is unstable if one attempts to translate it into a purely numerical algorithm since it relies on several GCD and remainder operations. If these operations are stabilized using approximate GCD techniques, then the cost is no longer $\widetilde{O}(n^3d)$ FLOPs.

4.5.2 Floating Point Interpolation Method

The problem can also be approached from numeric interpolation (see [62, 63]). The algorithm is simply to evaluate $\mathcal{A}$ at $(n-1)d+1$ (or another suitable upper bound on the degree of the determinant) equidistant roots of unity and use a floating point scalar matrix adjoint method. The algorithm may be briefly summarized as:

1. Compute $O(nd)$ images of $\mathcal{A}$ for a cost of $\widetilde{O}(n^2d)$ FLOPs via the Fast Fourier Transform (FFT).

2. Compute $O(nd)$ scalar matrix adjoints for a cost of $O(n^3)$ FLOPs, for a total cost of $O(n^4d)$ FLOPs.

3. Interpolate the $O(nd)$ adjoint images for a cost of $\widetilde{O}(n^2d)$ FLOPs via the FFT.

The cost of such a procedure is $\widetilde{O}(n^4d)$ FLOPs, and is suboptimal by a factor of $O(n)$, since the adjoint has $O(n^3d)$ entries.

Let ε_{ijk} be the error from computing $\mathscr{B}_{ij}(\omega_k)$ where $\mathscr{B} = \mathrm{Adj}(\mathscr{A})$ and ω_k is an $nd-1$ root of unity. The quantity ε_{ijk} is the error of the underlying scalar adjoint problem. This quantity is also backwards stable if $\mathscr{A}$ is not poorly conditioned. The error propagated from interpolating $\mathscr{B}_{ij}(\omega)$ is the error of solving the linear system

$$V_{ij}(\omega_0, \ldots, \omega_{(n-1)d}) \begin{pmatrix} \mathscr{B}_{ijk_0} \\ \vdots \\ \mathscr{B}_{ijk_{(n-1)d}} \end{pmatrix} = \left(\begin{pmatrix} \mathscr{B}_{ij}(\omega_0) \\ \vdots \\ \mathscr{B}_{ij}(\omega_{(n-1)d}) \end{pmatrix} + \begin{pmatrix} \varepsilon_{ij0} \\ \vdots \\ \varepsilon_{ij((n-1)d)} \end{pmatrix} \right),$$

The Vandermonde matrices $V_{ij}(\omega_0, \ldots, \omega_{(n-1)d})$ are unitary, so the error propagated (using $\|\cdot\|_2$) in each instance will be proportional to $O(\sqrt{nd})\varepsilon_k$, where $\varepsilon_k = \max_k\{\varepsilon_{ijk}\}$. The total error propagation is $O(n^{3/2}d^{1/2}\varepsilon_{\max})$, where $\varepsilon_{\max} = \max_k\{\varepsilon_k\}$. This method is not always backwards stable (but each sub-problem can typically be solved in a backwards stable manner), because $\mathrm{Adj}(\cdot)$ is not surjective over $\mathsf{R}[t]^{n\times n}$. Thus, the error propagation will be proportional to the condition number of $\mathrm{Adj}(\cdot)$ and the interpolation point with the largest error.

4.5.3 Linear System Solving over R

If we assume that $\mathscr{A}$ has full rank, then we can pose computing the adjoint matrix as a solution to

$$\mathscr{A}\mathscr{X} = \det(\mathscr{A})I.$$

The problem can be reduced to solving

$$\Phi_{(n-1)d}(I \otimes \mathscr{A})\,\mathrm{vec}(\mathscr{X}) = \mathrm{vec}(\det(\mathscr{A})I).$$

Given the block structure present, it is sufficient to compute $\det(\mathscr{A})$ once (which we assume can be done fast enough) and QR factor $\Phi_{(n-1)d}(\mathscr{A})$ for $\widetilde{O}(n^6d^3)$ FLOPs. After $\Phi_{(n-1)d}(\mathscr{A})$ has been QR factored, n linear system solves need to be performed (at a cost of $O(n^4d^2)$ FLOPs each), so the total cost of $\widetilde{O}(n^6d^3)$ FLOPs are required. This method is too slow to use in practice. However it is easy to analyze as it is a linear algebra problem over R (and the QR decomposition is backwards stable).

4.5.4 Automatic Differentiation

If we assume that we can compute $\det(\mathscr{A})$ in $\widetilde{O}(n^4 d)$ FLOPs, then we can extract $\mathrm{Adj}(\mathscr{A})$ from $\nabla \det(\mathscr{A})$. This requires $O(nd)$ determinant calls using backwards mode automatic differentiation, for a total cost of $\widetilde{O}(n^5 d^2)$ FLOPs. The stability of such an algorithm is proportional to the dimension, machine round off and the condition number of $\mathrm{Adj}(\cdot)$.

In general, any method to compute the determinant in $O(f(n,d))$ FLOPs leads to a trivial $O(nd\, f(n,d))$ FLOP algorithm to compute the adjoint via reverse mode automatic differentiation.

4.5.5 QZ Decomposition

It is possible to break the $\widetilde{O}(n^4)$ barrier in exchange for increasing the cost with respect to d in some special instances. If we linearize $\mathscr{A} \in \mathsf{R}[t]^{n\times n}$ of degree at most d into the pencil $\mathscr{P} \in \mathsf{R}[t]^{nd\times nd}$ then we can QZ factor $\mathscr{P}$ in $\widetilde{O}(n^3 d^3)$ FLOPs. This means we can compute the adjoint of $\mathscr{P}$ in $O(n^3 d^3)$ FLOPs, which is often sufficiently good in many problems. In some special instances, we can also compute the adjoint matrix of $\mathscr{A}$ from this linearization.

Suppose that $\mathscr{P} = Q\mathscr{R}Z$ for $Q, Z \in \mathbb{C}^{nd\times nd}$, $Q^*Q = I$, $Z^*Z = I$ and $\mathscr{R} \in \mathbb{C}[t]^{nd\times nd}$ with $\mathscr{R}$ being a triangular matrix. Then we can write

$$\det(\mathscr{A}) = \det(Q)\det(\mathscr{R})\det(Z).$$

Analogous to the QR decomposition, we can compute $\det(Q)$, $\det(Z)$ and if needed, their derivatives once while performing the QZ decomposition. Note that Q and Z are scalar matrices, whose determinant is always a degree zero polynomial, so their derivative with respect to the entries of $\mathscr{A}$ can be computed in $O(n^3 d^3)$ FLOPs. Now we can compute $\det(\mathscr{R})$ in $\widetilde{O}(nd)$ FLOPs.

For a suitable $\omega \in \mathbb{C}$, we may invert $\mathscr{R}(\omega)$ in $O(n^2 d^2)$ FLOPs (as $\mathscr{R}$ is triangular) and compute $\det(\mathscr{R}(\omega))$ in $\widetilde{O}(nd)$ FLOPs. We can compute $\mathrm{Adj}(\mathscr{R})$ in $\widetilde{O}(n^3 d^3)$ FLOPs via interpolation on $O(nd)$ images. We can also compute $\mathrm{Adj}(Q)$ and $\mathrm{Adj}(Z)$ in $O(n^3 d^3)$ FLOPs, thus $\mathrm{Adj}(\mathscr{P})$ may be computed in $\widetilde{O}(n^3 d^3)$ FLOPs (if $\mathscr{A}$ is a degree one matrix polynomial, this is $\widetilde{O}(n^3)$ FLOPs).

Alternatively, one may use automatic differentiation (reverse mode) on the $O(nd)$ images of $\det(\mathscr{R}(\omega))$ in conjunction with the derivatives of $\det(Q)$ and $\det(Z)$ to obtain

$O(nd)$ images of $\dfrac{\partial \det(\mathscr{A}(\omega))}{\partial \operatorname{vec}(\mathscr{A}(\omega))} = \operatorname{vec}(\operatorname{Adj}(\mathscr{A}(\omega))^T)^T$ in a comparable amount of time if $\mathscr{A} \in \mathsf{R}[t]^{n\times n}$ is not a degree one matrix polynomial.

In terms of numerical stability, the QZ decomposition uses unitary operations so the error from the QZ decomposition is proportional to the dimension, unit round-off error and the condition number of $\operatorname{Adj}(\cdot)$. Note that the QZ decomposition is a modified QR decomposition and behaves in almost the same way with respect to numerical stability. If some eigenvalues are repeated or the spectral structure of $\mathscr{A}$ is non-trivial then the QZ algorithm may not be stable in some implementations. The technique of [7] can be used in lieu of a pure QZ decomposition, of which the analysis holds since it is another two-sided unitary factorization.

4.6 Error Analysis of Matrix Polynomial Adjoint Computations

In this section we discuss the problem of estimating the backwards error of an algorithm that computes the adjoint of a matrix polynomial. The main application is a symbolic algorithm that computes the adjoint exactly, then rounds to working precision. We provide a detailed analysis of the instance of scalar matrices ($d = 0$), then generalize the idea using block-convolution matrices to the instance of matrix polynomials.

4.6.1 The Scalar Instance

Given $A \in \mathsf{R}^{n\times n}$ of full rank, there exists a $B \in \mathsf{R}^{n\times n}$ such that $AB = \det(A)I$. One such method to approximate the adjoint matrix, B, is to compute the exact adjoint of A then round to the required working precision. Alternatively, one can use a technique mentioned in [100] with sufficiently high precision.

We want to find a bound on the minimal (unstructured) perturbation ΔA such that $(A+\Delta A)(B+\Delta B) = \det(A+\Delta A)I$ where ΔB is the rounding error arising from computation (floating point error). One notes that $B = \operatorname{Adj}(A)$ in this formulation of the problem for ease of reading and notation. In general if $\mathsf{R} = \mathbb{R}$ or $\mathsf{R} = \mathbb{C}$, we can write $\Delta A = \omega(B+\Delta B)^{-1} - A$, where $\omega^{n-1} = \det(B+\Delta B)$. Note that ω^{n-1} need not be real, even if $\mathsf{R} = \mathbb{R}$.

Estimating the Backwards Error Directly

The goal is to estimate the backwards error $\|\Delta A\|_2 = \|\omega(B + \Delta B)^{-1} - A\|_2$ in terms of $\|\Delta B\|_2$.

We can write a first-order series expansion [61]

$$(B - (-\Delta B))^{-1} \approx B^{-1}(I - \Delta B\, B^{-1}).$$

Using a first-order Taylor expansion we can also write

$$\omega^{n-1} \approx \det(A)^{n-1} + \operatorname{Tr}(\operatorname{Adj}(B)\Delta B) = \det(A)^{n-1}\left(1 + \frac{\operatorname{Tr}(A\Delta B)}{\det(A)}\right).$$

Thus, $\omega \approx \det(A)\left(1 + \operatorname{Tr}(\operatorname{Adj}(A)^{-1}\Delta B)\right)^{1/(n-1)}$, since $\operatorname{Adj}(\operatorname{Adj}(A)) = \det(A)^{n-2}A$ and $A/\det(A) = \operatorname{Adj}(A)^{-1}$.

This leads to the approximation

$$\begin{aligned}\Delta A &= \omega(B + \Delta B)^{-1} - A \\ &\approx \det(A)\left(1 + \operatorname{Tr}(\operatorname{Adj}(A)^{-1}\Delta B)\right)^{1/(n-1)} \frac{A}{\det(A)}\left(I - \Delta B\, B^{-1}\right) - A\big) \\ &= \left(1 + \operatorname{Tr}(\operatorname{Adj}(A)^{-1}\Delta B)\right)^{1/(n-1)} A\left(I - \Delta B \frac{A}{\det(A)}\right) - A.\end{aligned}$$

If we consider $c = \left(1 + \operatorname{Tr}(\operatorname{Adj}(A)^{-1}\Delta B)\right)^{1/(n-1)}$, then $c \approx 1 + \dfrac{\operatorname{Tr}(\operatorname{Adj}(A)^{-1}\Delta B)}{n-1}$, again using a first-order expansion.

Taking norms we obtain

$$\begin{aligned}\|\Delta A\|_2 &\approx \left\| cA - A - cA\Delta B \frac{A}{\det(A)} \right\|_2 \\ &\leq \underbrace{\|A\|_2 |c-1|}_{O(\|\Delta B\|_2)} + \underbrace{\frac{|c|}{\det(\Sigma)}\|A\Delta B A\|_2}_{O(\|\Delta B\|_2^2)}.\end{aligned}$$

Theorem 4.6.1. *Ignoring higher-order terms, we have*

$$\|\Delta A\|_2 \leq \frac{n}{n-1}\frac{\|A\|_2^2}{\det(\Sigma)}\|\Delta B\|_2 \approx \kappa_2(A)\frac{1}{\sigma_2 \cdots \sigma_{n-1}}\|\Delta B\|_2.$$

Estimating the Backwards Error via Optimization

Computing the backwards error can also be expressed as the optimization problem

$$\min \|\Delta A\|_F \ \text{ subject to } \ (A+\Delta A)(B+\Delta B) - \det(A+\Delta A)I = 0,$$

which is a non-linear least squares problem in ΔA. By considering $\mathsf{R} = \mathbb{C}$ we can always compute some notion of a backwards error, as the constraint can be satisfied.

Recall that $\nabla \det(\cdot)$ is essentially the adjoint operator as

$$\left(\frac{\partial}{\partial \operatorname{vec}(A)} \det\right)(A) = \operatorname{vec}(\operatorname{Adj}(A)^T)^T.$$

The non-linear part of this problem is $\det(A) - \det(A+\Delta A)$ which can be approximated by a first-order approximation $\det(A+\Delta A) - \det(A) \approx \operatorname{vec}(\operatorname{Adj}(A)^T)^T \operatorname{vec}(\Delta A)$. Using the identity that $\operatorname{vec}(X^T)^T \operatorname{vec}(Y) = \operatorname{Tr}(XY)$, we can write $\det(A+\Delta A) - \det(A) \approx \operatorname{Tr}(\operatorname{Adj}(A)\Delta A)$.

If we suppose that $\|\Delta A\|_F$ is sufficiently small, then the constraint is approximately

$$-\operatorname{vec}(\operatorname{Adj}(A)^T)^T \operatorname{vec}(\Delta A) I + \Delta A(B+\Delta B) + A\Delta B + O(\|\Delta A\|_F^2) = 0.$$

If we ignore higher-order terms, we have a linear system of the form

$$\begin{aligned} &\Delta A(\operatorname{Adj}(A) + \Delta B) + A\Delta B - \operatorname{Tr}(\operatorname{Adj}(A)\Delta A)I = 0 \\ \iff &\left(\left[\operatorname{Adj}(A)^T + \Delta B^T\right] \otimes I - \operatorname{vec}(I)\operatorname{vec}(\operatorname{Adj}(A)^T)^T\right)\operatorname{vec}(\Delta A) + \operatorname{vec}(A\Delta B) = 0. \end{aligned}$$

The coefficient matrix $\left(\left[\operatorname{Adj}(A)^T + \Delta B^T\right] \otimes I - \operatorname{vec}(I)\operatorname{vec}(\operatorname{Adj}(A)^T)^T\right)$ is the sum of a full rank matrix and a rank one matrix. If we assume that $\|\Delta B\|_2$ is sufficiently small, then we can largely ignore it, since ΔB is essentially a random perturbation of a sufficiently small magnitude and will generally not perturb the spectral structure into singularity. Using the Sherman-Morrison formula for rank 1 updates to a matrix, we need to verify that

$$1 - \operatorname{vec}(\operatorname{Adj}(A)^T)^T \left[\left(\operatorname{Adj}(A)^T\right)^{-1} \otimes I\right] \operatorname{vec}(I) \neq 0,$$

in order to ensure that the coefficient matrix has full rank. We have that

$$\begin{aligned} \operatorname{vec}(\operatorname{Adj}(A)^T)^T \left[\left(\operatorname{Adj}(A)^T\right)^{-1} \otimes I\right] \operatorname{vec}(I) &= \operatorname{vec}(\operatorname{Adj}(A)^T)^T \operatorname{vec}\left(\operatorname{Adj}(A)^T)^{-1}\right) \\ &= \operatorname{Tr}\left(\operatorname{Adj}(A)^T \left(\operatorname{Adj}(A)^T\right)^{-1}\right) \\ &= n, \end{aligned}$$

so for $n \geq 2$ the coefficient matrix will have full rank. Note we have used the identities $\operatorname{vec}(XYZ) = (Z^T \otimes X)\operatorname{vec}(Y)$ and $\operatorname{vec}(X^T)^T \operatorname{vec}(Y) = \operatorname{Tr}(XY)$.

Now, the matrix $C = \operatorname{Adj}(A)^T \otimes I$ is invertible and the rank one update $C - \operatorname{vec}(I)\operatorname{vec}(\operatorname{Adj}(A)^T)^T$ was also shown earlier to be invertible. Given this, we can write

$$\begin{aligned}(C - \operatorname{vec}(I)\operatorname{vec}(\operatorname{Adj}(A)^T)^T)^{-1} &= C^{-1} + \frac{C^{-1}\operatorname{vec}(I)\operatorname{vec}(\operatorname{Adj}(A)^T)^T C^{-1}}{1 - \operatorname{vec}(\operatorname{Adj}(A)^T)^T C^{-1}\operatorname{vec}(I)} \\ &= C^{-1} + \frac{\operatorname{vec}((\operatorname{Adj}(A)^T)^{-1})\operatorname{vec}(I)^T}{1-n}.\end{aligned}$$

We can now derive some bounds on the conditioning of the coefficient matrix in terms of A.

$$\begin{aligned}\|C - \operatorname{vec}(I)\operatorname{vec}(\operatorname{Adj}(A)^T)^T)^{-1}\|_2 &\leq \sigma_{\min}(Z)^{-1} + \frac{\|I\|_F\|\operatorname{Adj}(A)^{-1}\|_F}{n-1} \\ &\leq \sigma_{\min}(Z)^{-1} + \frac{(\sqrt{n})\,(\sqrt{n}\|\operatorname{Adj}(A)^{-1}\|_2)}{n-1} \\ &\leq \sigma_{\min}(C)^{-1}\left(1 + \frac{n}{n-1}\right) \\ &\approx 2\sigma_{\min}(C)^{-1}.\end{aligned}$$

We note that

$$\left[\operatorname{Adj}(A)^T \otimes I\right]^{-1} = \left[(\operatorname{Adj}(A)^T)^{-1} \otimes I\right] \quad \text{and} \quad \left\|\left[(\operatorname{Adj}(A)^T)^{-1} \otimes I\right]\right\|_2 = \|(\operatorname{Adj}(A)^T)^{-1}\|_2.$$

Now, by a similar argument we have that

$$\|C - \operatorname{vec}(I)\operatorname{vec}(\operatorname{Adj}(A)^T)^T)\|_2 \leq \|C\|_2(n+1).$$

Thus, the 2 norm condition number is bounded above by

$$\kappa_2(C - \operatorname{vec}(I)\operatorname{vec}(\operatorname{Adj}(A)^T)^T)) \leq 2(n+1)\kappa_2(\operatorname{Adj}(A)).$$

An immediate application of Lemma 4.2.1 yields that $\kappa_2(\operatorname{Adj}(A)) = \kappa_2(A)$.

Now we ignored the ΔB term earlier, which we need to consider. If we have that $\|\Delta B\|$ is sufficiently small, then it seems obvious that the impact on the conditioning will be minimal. Since the entries of ΔB are proportional to the machine round-off error, we can in theory increase the precision to be sufficiently high so that errors do not affect the computation. The question is how much precision is needed? We recall that if $\|(\mathrm{Adj}(A)^T)^{-1}\Delta B\|_2 < 1$ then

$$\|(\mathrm{Adj}(A)^T + \Delta B)^{-1}\|_2 \leq \frac{\|\mathrm{Adj}(A)^{-1}\|_2}{1 - \|(\mathrm{Adj}(A)^T)^{-1}\Delta B\|_2},$$

so if we force $\|(\mathrm{Adj}(A)^T)^{-1}\Delta B\|_2 < 1/2$ we have $\|((\mathrm{Adj}(A)^T) + \Delta B)^{-1}\|_2 \leq 2\|\mathrm{Adj}(A)^{-1}\|_2$. Note that this follows from a first-order series expansion of $(B + \Delta B)^{-1}$.

To ensure that $\|\Delta B\|$ is sufficiently small, we note that

$$\|(\mathrm{Adj}(A)^T)^{-1}\Delta B\|_2 \leq \|\mathrm{Adj}(A)^{-1}\|_2\|\Delta B\|_2 = \frac{\|A\|_2}{\det(\Sigma)}\|\Delta B\|_2.$$

So, if we choose $\|\Delta B\|_2 \leq \frac{1}{2}\frac{\det(\Sigma)}{\|A\|_2}$, we can estimate the backwards error using this approximation. Note that computing the SVD of A is adequate to determine if computing the adjoint will have a modest backwards error.

Theorem 4.6.2. *If $\|\Delta A\|_F$ is sufficiently small, then*

$$\|\mathrm{vec}(\Delta A)\|_2 = \|\Delta A\|_F \approx \left\|\left(\left[\mathrm{Adj}(A)^T + \Delta B^T\right] \otimes I - \mathrm{vec}(I)\,\mathrm{vec}(\mathrm{Adj}(A)^T)^T\right)^{-1}\mathrm{vec}(A\Delta B)\right\|_2.$$

Furthermore, if $\|\Delta B\|_2$ is sufficiently small, then ignoring $O(\|\Delta B\|_2^2)$ terms,

$$\|\Delta A\|_2 \leq \|\Delta A\|_F \leq (\|(B + \Delta B)^{-1}\|_2 + \|B^{-1}\|_2)\|\mathrm{vec}(A\Delta B)\|_2 \leq \frac{2\sqrt{n}\kappa_2(A)}{\sigma_2 \cdots \sigma_{n-1}}\|\Delta B\|_2.$$

Proof. The term

$$\left\|\left(\left[\mathrm{Adj}(A)^T + \Delta B^T\right] \otimes I - \mathrm{vec}(I)\,\mathrm{vec}(\mathrm{Adj}(A)^T)^T\right)^{-1}\right\|_2 \leq \|(B + \Delta B)^{-1}\|_2 + \|B^{-1}\|_2$$

follows from the earlier analysis in this section, as this is a rank one update to a matrix. We note that

$$\|\mathrm{vec}(A\Delta B)\|_2 = \|A\Delta B\|_F \leq \|A\|_F\|B\|_2 \leq \sqrt{n}\|A\|_2\|\Delta B\|_2.$$

□

Corollary 4.6.3. *If $\|\Delta B\|_2$ is sufficiently small, then*

$$\kappa_2\left(\left[\operatorname{Adj}(A)^T + \Delta B^T\right] \otimes I - \operatorname{vec}(I)\operatorname{vec}(\operatorname{Adj}(A)^T)^T\right) \in O(n \times \kappa_2(A)).$$

The proof follows immediately from the preceding discussion.

Note that the two analysis of the backwards error agree up to a constant factor of $2\sqrt{n}$, which arises due to the vectorization and resulting linear system of equations. If we used $\|\cdot\|_F$ for our first analysis, then the constant factor becomes 2, which arose from the triangle inequality. We observe from analysis in both instances that it is sufficient to have $\|\Delta B\|_2 \leq \frac{1}{2}\det(\Sigma)/\|A\|_2$ to ensure that the series approximation $(B+\Delta B)^{-1}$ is meaningful in both instances.

It is important to note that the analysis in this section generalizes very easily to matrix polynomials since it is "inverse-free" until the last step.

4.6.2 The Matrix Polynomial Instance

Suppose that we are given $\mathcal{A} \in \mathsf{R}[t]^{n\times n}$ of full rank, so there exists $\mathcal{B} \in \mathsf{R}[t]^{n\times n}$ such that $\mathcal{A}\mathcal{B} = \det(\mathcal{A})I$. We would like a bound on the minimal (unstructured) perturbation $\Delta\mathcal{A}$ such that $(\mathcal{A}+\Delta\mathcal{A})(\mathcal{B}+\Delta\mathcal{B}) = \det(\mathcal{A}+\Delta\mathcal{A})I$ where $\Delta\mathcal{B}$ is the rounding error arising from the computation (floating point error). We will assume with some loss of generality that there exists $\mathcal{C} \in \mathsf{R}[t]^{n\times n}$ such that $\operatorname{Adj}(\mathcal{C}) = \mathcal{B}+\Delta\mathcal{B}$. In theory, the rounded answer could be perturbed to a nearby matrix polynomial with a pre-image under the adjoint operator. However, we assume that this does not need to be done.

The analysis of the previous section can be applied directly here by an interpolation argument. Alternatively, we can generate a linear system of equations over R and solve this problem instead.

Like earlier, we are now looking to compute a solution to

$$\left(\left[\operatorname{Adj}(\mathcal{A})^T + \Delta\mathcal{B}^T\right] \otimes I - \operatorname{pvec}(I)\operatorname{pvec}(\operatorname{Adj}(\mathcal{A})^T)^T\right)\operatorname{pvec}(\Delta\mathcal{A}) + \operatorname{pvec}(\mathcal{A}\Delta\mathcal{B}) = 0, \quad (4.6)$$

which is equivalent to solving the linear system

$$\Phi_d\left(\left(\left[\operatorname{Adj}(\mathcal{A})^T + \Delta\mathcal{B}^T\right] \otimes I - \operatorname{pvec}(I)\operatorname{pvec}(\operatorname{Adj}(\mathcal{A})^T)^T\right)\right)\operatorname{vec}(\Delta\mathcal{A}) + \Phi_d\left(\Delta\mathcal{B}^T \otimes I\right)\operatorname{vec}(A) = 0.$$

Note that the coefficient matrix in (4.6) has full rank by essentially the same argument as before if $\mathcal{A}$ has full rank. By assumption a solution exists (otherwise in theory one could perturb $\Delta\mathcal{B}$ to the nearest point in which a solution does exist).

Theorem 4.6.4. *If $\|\Delta\mathcal{A}\|_F$ is sufficiently small and there exists $\mathcal{C} \in \mathsf{R}[t]^{n\times n}$ such that* $\operatorname{Adj}(\mathcal{C}) = \mathcal{B} + \Delta\mathcal{B}$*, then* $\operatorname{vec}(\Delta\mathcal{A}) \approx$

$$-\left[\Phi_d\left(\left(\left[\operatorname{Adj}(\mathcal{A})^T + \Delta\mathcal{B}^T\right] \otimes I - \operatorname{pvec}(I)\operatorname{pvec}(\operatorname{Adj}(\mathcal{A})^T)^T\right)\right)\right]^+ \Phi_d\left(\Delta\mathcal{B}^T \otimes I\right)\operatorname{vec}(A).$$

To answer the precision problem we draw our attention to the instance of $d = 0$. We can determine how much precision is needed by evaluating the problem at $O(nd)$ complex roots of unity that are uniformly distributed along the unit circle. An upper bound on the precision needed is easily derived from the image that requires the most precision.

4.7 Computing the First and Second Derivatives of the Determinant and Adjoint

In this section we discuss some methods for computing first and second-order derivatives of the adjoint matrix when working over $\mathbb{R}[t]^{n\times n}$. The degree zero case is well studied (see [12, 64] for example).

As we saw earlier, the adjoint matrix is basically the first-order derivative of the determinant. Accordingly, it should not be a surprise that we can think of the first two derivatives of the adjoint as the second and third order derivatives of the determinant. In the scalar case, this is quite straight forward to compute, however in the matrix polynomial case the generalization is not quite immediate. We will assume that computing the adjoint matrix numerically costs $\widetilde{O}(n^4 d)$ FLOPs, as this is easily accomplished with existing numerical linear algebra routines currently available.

4.7.1 Polynomial-Time Symbolic Differentiation of the Matrix Polynomial Determinant

If we consider same degree unstructured perturbations to $\mathcal{A} \in \mathsf{R}[t]^{n\times n}$ of the form $\Delta\mathcal{A} \in \mathbb{R}[t]^{n\times n}$, then computing derivatives is straight forward. A naive method would be to compute $\det(\mathcal{A} + \Delta\mathcal{A})$ as a vector of $nd + 1$ multivariate polynomials of total degree nd. Obviously there are at least exponentially many coefficients in the entries of $\det(\mathcal{A}+\Delta\mathcal{A})$, so this will not yield a fast algorithm for differentiation.

Alternatively, one could use Theorem 4.3.1 to compute the derivatives. The cost of computing the first derivative with such a technique is $\widetilde{O}(n^4 d + n^3 d^2)$ FLOPs. To obtain the second derivative a naive computation using Theorem 4.4.1 would cost roughly

$O(n^6d^3 + n^5d^3)$ FLOPs after expanding the Toeplitz-block structure. A faster algorithm would require $\widetilde{O}(n^5d + n^5d^3)$ FLOPs by computing the first derivative of the adjoint directly then expanding the Toeplitz-block structure. In practice we would not compute the tensor $\nabla^2 \det(\mathscr{A})$ directly, since we generally need expressions of the form $\nabla^2\lambda^T \det(\mathscr{A})$ where $\lambda \in \mathbb{R}^{(nd+1)\times 1}$. It is important to note that automatic differentiation is very efficient for evaluating expressions of the form $\nabla^2\lambda^T f(x)$. However, in several situations the tools of automatic differentiation may not be available.

We can use a hybrid-symbolic algorithm to compute $\det(\mathscr{A})$ in the case where $\mathscr{A}$ has an affine structure $\mathbf{\Delta}(\cdot)$ of some kind. Since $\det(\cdot)$ is a multi linear function in the entries of $\Delta\mathscr{A}$, i.e. $\det(\cdot)$ is multi-linear in $\mathsf{R}[\Delta\mathscr{A}_{110}, \dots \Delta\mathscr{A}_{nnd}]$, we can store it as a sequence of multi-linear polynomials. Again, this would require an exponential amount of storage in n and d. To compute $\dfrac{\partial \det(\mathscr{A} + \Delta\mathscr{A})}{\partial \Delta\mathscr{A}_{ijk}}$ it is adequate to compute $\dfrac{\partial \det(\mathscr{A} + \Delta\mathscr{A}_{ijk} \cdot E_{ijk})}{\partial \Delta\mathscr{A}_{ijk}}$, where $E_{ijk} \in \mathsf{R}[t]^{n\times n}$ is a matrix from an appropriate basis for the perturbation structure, assuming we are evaluating the derivative at $\Delta\mathscr{A} = 0$ (by performing a shift we can generalize to other evaluation points, as we will demonstrate for the adjoint). Likewise, evaluating the derivative at a particular point follows a similar idea. The cost to compute $\det(\mathscr{A} + \Delta\mathscr{A}_{ij} \cdot E_{ijk})$ is $\widetilde{O}(n^4d^2)$ FLOPs using naive symbolic methods, so computing all n^2d derivatives would have a worst case cost of $\widetilde{O}(n^6d^3)$ FLOPs. We can improve this slightly, by making n^2 determinant calls, by computing the derivatives of $\Delta\mathscr{A}_{ij} \cdot E_{ij}$ in the same pass (one simply makes a linear substitution, computes the derivative, then invokes the chain rule on the computed result and performs a scalar polynomial multiplication). A reasonable implementation avoids the multivariate polynomial swell, so the resulting cost remains polynomial. The derivative cost is reduced to $\widetilde{O}(n^6d^2)$ FLOPs. If one were to assume that $\Delta\mathscr{A}$ has no structure, i.e. $\mathscr{A}$ has degree at most d and any entry may be arbitrarily perturbed, then computing $\operatorname{Adj}(\mathscr{A})$ (symbolically) requires $\widetilde{O}(n^4d)$ FLOPs, which would be faster in the *unstructured case* or the case of *some special affine structure*. If we had a quasi-optimal algorithm to compute the adjoint matrix that required $\widetilde{O}(n^3d)$ FLOPs then we could compute $\nabla \det(\mathscr{A})$ in $\widetilde{O}(n^3d^2)$ FLOPs (which is quasi-optimal).

Computing the two remaining higher-order derivatives remains to be done, and this technique generalizes in the obvious way. The run time is still polynomial, except more determinant calls are required. We explain in detail how to do this for the adjoint matrix, of which the theory is nearly identical.

4.7.2 Polynomial Time Symbolic Differentiation of the Matrix Polynomial Adjoint

Computing the derivatives of the matrix polynomial adjoint is similar to the determinant, however we will include a proper discussion for completeness. While $\det(\cdot) : \mathsf{R}^{n^2(d+1)} \to \mathsf{R}^{nd+1}$, we have that $\operatorname{Adj}(\cdot) : \mathsf{R}^{n^2(d+1)} \to \mathsf{R}^{n^2((n-1)d+1)}$. We will generally take $\mathsf{R} = \mathbb{R}$, because we will treat $\mathbb{C} \cong \mathbb{R}^2$ and the ideas will carry over with some care, since we are generally not performing complex differentiation (the end goal is to minimize $\| \cdot \|$ of some function subject to some constraints, of which real differentiation is exclusively used).

If we compute $\nabla \operatorname{Adj}(\cdot)$ using Theorem 4.3.1 directly, then using standard arithmetic this will cost $O(n^6 d^3)$ FLOPs which is too slow. Instead, we will discuss methods that behave like forward mode automatic differentiation, but are well suited towards matrix polynomials with an affine structure. If $\mathcal{A} \in \mathbb{R}[t]^{n\times n}$ has the structure of preserving the degree of each entry or the support of each entry then the following method has the same cost as the unstructured case (the unstructured Jacobian will be padded with several columns of zeros which can be deleted to obtain the structured Jacobian).

We can compute the first-order derivatives using the same idea as the determinant, that is

$$\left.\frac{\partial \operatorname{Adj}(\mathcal{A} + \Delta\mathcal{A})}{\partial \Delta\mathcal{A}_{ijk}}\right|_{\Delta\mathcal{A} = \Delta\mathcal{A}^{eval}} = \frac{\partial \operatorname{Adj}(\mathcal{A} + \Delta\mathcal{A}_{ijk} \cdot E_{ijk} + \Delta\mathcal{A}^{eval} - \Delta\mathcal{A}^{eval}_{ijk} \cdot E_{ijk})}{\partial \Delta\mathcal{A}_{ijk}},$$

from which vectorizing (and padding with zeros as needed) produces the desired vector of derivatives. Since $\operatorname{Adj}(\cdot)$ is a continuous function as a multivariate polynomial in $\Delta\mathcal{A}$, we can compute mixed partial derivatives in an analogous fashion. The "blocking" idea from earlier is equally applicable. We are essentially computing the adjoint symbolically of a matrix with entries over $\mathsf{R}[t][\Delta\mathcal{A}_{ijk}]$. The cost of doing this symbolically (i.e. performing the computation with symbols) is roughly $\widetilde{O}(n^4 d)$ FLOPs, since the polynomials in $\Delta\mathcal{A}_{ijk}$ are linear.

The expression

$$\operatorname{Adj}(\mathcal{A} + \Delta\mathcal{A}_{ijk} \cdot E_{ijk} + \Delta\mathcal{A}^{eval} - \Delta\mathcal{A}^{eval}_{ijk} \cdot E_{ijk})$$

may seem confusing at first, but all it does is evaluate $\Delta\mathcal{A}$ at the point $\Delta\mathcal{A}^{eval}$ except for the entries $\Delta\mathcal{A}_{ijk}$. This is the "shift" that would need to be applied to the determinant.

Computing the second-order derivatives is commonly found in optimization problems using sequential quadratic programming, or Newton-like methods. Generally, we will need to differentiate an expression of the form

$$\lambda^T \operatorname{vec}(\operatorname{Adj}(\mathcal{A} + \Delta\mathcal{A})) \tag{4.7}$$

where $\lambda \in \mathbb{R}^{n^2((n-1)d+1)\times 1}$ is a vector of *Lagrange multipliers*.

The second derivative of (4.7) is sometimes required, and we can write (differentiating with respect to $\Delta\mathcal{A}$)

$$\nabla^2 \lambda^T \operatorname{vec}(\operatorname{Adj}(\mathcal{A} + \Delta\mathcal{A})) = \nabla\lambda^T J, \quad \text{with} \quad J = \nabla \operatorname{vec}(\operatorname{Adj}(\mathcal{A} + \Delta\mathcal{A})).$$

If we define

$$A^{diff}_{ijk,uvw} = \operatorname{Adj}(\mathcal{A} + \Delta\mathcal{A}^{eval} + (\Delta\mathcal{A}_{ijk} - \Delta\mathcal{A}^{eval}_{ijk}) \cdot E_{ijk} + (\Delta\mathcal{A}_{uvw} - \Delta\mathcal{A}^{eval}_{uvw}) \cdot E_{uvw}),$$

then $A^{diff}_{ijk,uvw}$ represents a symbolic expression of $\operatorname{Adj}(\mathcal{A} + \Delta\mathcal{A})$ evaluated at $\Delta\mathcal{A}^{eval}$ except at the points corresponding to $\Delta\mathcal{A}_{uvw}$ and $\Delta\mathcal{A}_{ijk}$. Computing the quantity $\mathcal{A}^{diff}_{ijk,uvw}$ requires a symbolic matrix polynomial adjoint computation over $\mathsf{R}[t][\Delta\mathcal{A}_{ijk}, \Delta\mathcal{A}_{uvw}]$. The entries are bivariate multi-linear polynomials in $\mathsf{R}[\Delta\mathcal{A}_{ijk}, \Delta\mathcal{A}_{uvw}]$, so the corresponding computation will cost $\widetilde{O}(n^4 d)$ operations. Since there are two multivariate polynomials of total degree two, the overhead cost will generally be asymptotically constant for unstructured matrices or matrices with some special affine structures. This technique is feasible for computing a few low-order terms, but not all of the higher-order derivatives, as the cost will grow exponentially in the number order of the derivatives computed.

Now we note that

$$\left.\frac{\partial^2 \lambda^T \operatorname{vec}(\operatorname{Adj}(\mathcal{A} + \Delta\mathcal{A}))}{\partial\Delta\mathcal{A}_{ijk}\partial\Delta\mathcal{A}_{uvw}}\right|_{\Delta\mathcal{A}^{eval}} = \frac{\partial^2 \lambda^T \operatorname{vec}(\mathcal{A}^{diff}_{ijk,uvw})}{\partial\Delta\mathcal{A}_{ijk}\partial\Delta\mathcal{A}_{uvw}},$$

so computing the Hessian at a specific point can be done in a polynomial amount of time. Note that a reasonable implementation will compute $\nabla^2_{\Delta\mathcal{A}_{ijk},\Delta\mathcal{A}_{uvw}} \lambda^T \mathcal{A}^{diff}_{ijk,uvw}$ or something similar. Again we can use $O(n^2) \times O(n^2)$ adjoint calls if we work on blocks of each coefficient $\Delta\mathcal{A}_{ij}$. As mentioned earlier, these types of problems are well suited towards automatic differentiation, especially if a matrix-free method is used to approach the problem.

4.8 Optimization Problems Involving the Determinant

One particular application of Section 4.7.1 and the other ideas in this chapter is computing a nearby matrix polynomial with a prescribed eigenvalue, determinant or minors. The list of problems here are not exhaustive, but they fit within the theme of this thesis. The next chapter is dedicated to an optimization problem that depends on the derivatives of the adjoint matrix. If one had reason to compute additional minors, a very similar theory would hold.

4.8.1 Nearest Singular Matrix (Polynomial) Revisited

One such application in this family is finding the nearest singular matrix (polynomial) with an affine displacement structure, assuming it exists. Of course in the degree zero case with no particular structure this problem can be solved quickly with the SVD.

The optimization problem in Chapter 3 of computing the nearest rank deficient matrix polynomial can be formulated as

$$\min \|\Delta\mathscr{A}\|_F^2 \text{ subject to } \det(\mathscr{A}+\Delta\mathscr{A})=0, \tag{4.8}$$

without any particular affine structure. The quantity $\nabla\det(\mathscr{A}+\Delta\mathscr{A})$ will typically have (locally) constant rank. In general, Lagrange multipliers exist when $\operatorname{rank}(\mathscr{A}+\Delta\mathscr{A})\geq n-1$ since the adjoint, thus the derivative, does not vanish by Theorem 4.3.1 (recall that the adjoint has full rank or rank one in these scenarios). Note that the linear operator $\Phi_d(\cdot)$ does not deform the non-zero vector $\operatorname{pvec}(\operatorname{Adj}(\mathscr{A}+\Delta\mathscr{A})^T)^T$.

The constraint qualifications will fail to hold when $\operatorname{rank}(\mathscr{A}+\Delta\mathscr{A})\leq n-2$ since $\nabla\det(\mathscr{A}+\Delta\mathscr{A})\equiv 0$. One could prescribe that the $(n-r)\times(n-r)$ minors are zero to remedy this, although this is difficult to determine in advance. This observation can provide an alternative proof to the separation bounds and arguments to show that regularity conditions hold from Chapter 3. Note that no auxiliary variables are used in this formulation of the problem.

The idea of computing more minors is analogous to the ideas in Chapter 3 involving the minimal embedding. The minimal embedding shows that a constraint qualification can be satisfied if there are multiple kernel vectors. The analog in this instance is the number of leading zeros before or after the pivot element (depending on the permutation used) is the number of other kernel vectors.

We can formally summarize the procedure to compute the nearest singular matrix polynomial via the determinant in the following algorithm.

Using a standard variant of Newton's method, Algorithm 4 using non-fast linear system solving will require $\Omega(n^6d^3)$ FLOPs. In practice, computing the Hessian will require $\widetilde{O}(n^4d\times n^2d\times n^2d)=O(n^8d^3)$ FLOPs assuming the determinant is computed in $\widetilde{O}(n^4d)$

```
Algorithm 4 : Nearest Matrix Polynomial with Zero Determinant
Input:
  • Matrix polynomial $\mathcal{A} \in \mathbb{R}^{n\times n}$
  • (Approximately) Rank deficient $\mathcal{C} \in \mathbb{R}[t]^{n\times n}$ of the desired degree/displacement structure and rank deficiency.
  • Displacement structure matrix $\widetilde{\Delta}\mathcal{A}$ to optimize over.
Output:
  • $\mathcal{A} + \Delta\mathcal{A}$ that is singular or an indication of failure.
1: Compute $\Delta\mathcal{A}^{init}$ from $\mathcal{C}$
2: Define Lagrangian function $L$ as
       $L(x,\lambda) = \|\Delta\mathcal{A}\|_F^2 + \lambda^T \operatorname{vec}(\det(\mathcal{A}+\widetilde{\Delta}\mathcal{A}))$ where $x = \operatorname{vec}(\Delta\mathcal{A})$.
3: Initialize $\lambda \in \mathbb{R}^{(nd+1)\times 1}$ via linear least squares from $\nabla L|_{x^{init}} = 0$.
4: Compute
       $\begin{pmatrix} x+\Delta x \\ \lambda+\Delta\lambda \end{pmatrix}$ by solving $\nabla^2 L \begin{pmatrix} \Delta x \\ \Delta\lambda \end{pmatrix} = -\nabla L(x,\lambda)$ until
   $\left\| \begin{pmatrix} \Delta x \\ \Delta\lambda \end{pmatrix} \right\|_2$ or $\|\nabla L(x,\lambda)\|_2$ is sufficiently small, or divergence is detected.
5: Return the locally optimal $\Delta\mathcal{A}$ or an indication of failure.
```

FLOPs using standard techniques. Using reverse mode automatic differentiation on the Lagrangian should generally require $\widetilde{O}(n^4 d) \times O(n^2 d + nd) = \widetilde{O}(n^6 d^2)$ FLOPs, although our implementation does not do this. A reasonable implementation will require roughly $O(n^6 d^3)$ FLOPs, assuming "Newton's method" is used in some form. A first-order gradient algorithm would generally require $\widetilde{O}(n^4 d)$ FLOPs per iteration.

Theorem 4.8.1. *If $\|\Delta\mathcal{A}\|$ is sufficiently small and $\operatorname{Adj}(\mathcal{A}+\Delta\mathcal{A}) \neq 0$ then reasonable versions of Newton's method will converge quadratically in Algorithm 4 with a suitable initial guess .*

Proof. All the quantities are polynomials and thus locally Lipschitz. We have that $\nabla_{xx}^2 L = \big(2I + F\big)$, where F is a symmetric matrix with zero diagonal that is a linear function in λ and multi-linear in $\Delta\mathcal{A}$ that satisfies $F = 0$ if $\lambda = 0$. If $\|\lambda\|$ is sufficiently small, then $\sigma_{\min}(2I + F) > 0$ so $\nabla_{xx}^2 L$ has full rank and the second-order sufficient condition will hold at a minimizer, as the second-order necessary condition must hold. If $\operatorname{Adj}(\mathcal{A}+\Delta\mathcal{A}) \neq 0$ then Lagrange multipliers exist by an application of Theorem 4.3.1. □

If the solution has rank at most $n-2$ then regularity conditions are no longer satisfied. One would need to add the constraint that higher-order minors vanish in order to ensure that a constraint qualification held.

Example 4.8.2. *Let us consider $\mathcal{A}$ from Example 3.7.5, using the technique of 4.7.1 to solve the optimization problem (4.8) via Lagrange multipliers.*

We can compute a local minimizer[3] *by taking $\mathcal{A}^{init} = \mathcal{A}$ and performing 14 iterations of a Newton method to obtain an answer that is accurate to approximately 15 decimal points of accuracy.*

We compute $\|\Delta\mathcal{A}\| \approx 0.94356416746$ with

$$\Delta\mathcal{A}_0 \approx \begin{pmatrix} 0.1725690 & 0.2522491 & 0.08714713 \\ 0.2144912 & 0.3135280 & 0.1083178 \\ -0.05596349 & -0.08180346 & -0.02826149 \end{pmatrix},$$

$$\Delta\mathcal{A}_1 \approx \begin{pmatrix} 0.0 & 0.1223740 & -0.4690226 \\ 0.0 & 0.1521023 & -0.5829622 \\ 0.0 & -0.03968543 & 0.1521023 \end{pmatrix}.$$

This is the same solution computed earlier, up to rounding errors.

It should be noted that solving (4.8) with the same initial guesses in Section 3.7.2 and Section 3.7.3 computes the same solutions when $\operatorname{rank}(\mathcal{A} + \Delta A) = n-1$.

4.8.2 Nearest Matrix Polynomial with Prescribed Eigenvalue

An analogous problem is to find a nearby matrix polynomial with a prescribed *finite* eigenvalue by computing a (local) solution to

$$\min \|\Delta\mathcal{A}\|_F^2 \text{ subject to } \begin{cases} \det(\mathcal{A} + \Delta\mathcal{A}) = p(t)(t-\omega), \\ p(t) \in \mathsf{R}[t], \\ \omega \in \mathbb{C}. \end{cases}$$

In this problem $p(t)$ is an arbitrary polynomial and ω is prescribed by the user. If $\mathsf{R} = \mathbb{R}$ and $\omega \in \mathbb{C}\backslash\mathbb{R}$ then the right factor can be modified to be $(t-\omega)(t-\overline{\omega})$ so that a real irreducible quadratic factor is obtained. This problem generally has a solution since the set

[3]The second-order sufficient conditions are satisfied.

of all polynomials that do not have ω as a zero is open. I.e. if $p(\omega) \neq 0$ then $(p+\Delta p)(\omega) \neq 0$ by continuity for a Δp that is sufficiently small. Thus the set of all polynomials with ω as a zero is closed. $\Delta \mathscr{A} = -\mathscr{A}$ is such a solution, hence by Weierstrass' theorem a solution exists. As long as $\deg(\det(\mathscr{A} + \Delta \mathscr{A})) \neq -\infty$ (i.e. the solution has full rank) then a rank based constraint qualification will generally hold. Under these assumptions, we can expect Newton-like methods to have a local rate of convergence that is super linear and generally of order two.

A related problem is finding a nearby matrix polynomial with a prescribed *infinite* eigenvalue. It is well known that a matrix polynomial $\mathscr{A} \in \mathbb{R}[t]^{n\times n}$ of degree at most d has an infinite eigenvalue if and only if the leading coefficient matrix A_d is rank deficient. In other words, the matrix polynomial $t^d(\mathscr{A}(t^{-1}))$ has $t = 0$ as an eigenvalue. This problem is easily solved via the SVD if $\mathscr{A}$ is *unstructured*. If $\mathscr{A}$ is structured then this easily seen to be essentially the same problem as the finite case.

A natural question is how to compute a nearby matrix polynomial with a prescribed *spectral structure*. If possible, compute a nearby matrix polynomial that has a non-trivial Smith normal form or whose reversed polynomial has a prescribed spectral structure at the point $t = 0$. The next chapter is dedicated to this problem with an emphasis on the finite spectral structure.

4.9 Conclusion

We have shown that the matrix polynomial determinant and adjoint can be computed in a numerically robust manner using floating point operations that is reasonably fast. Furthermore theoretically, quasi-optimal symbolic numeric algorithms that compute the adjoint and determinant were discussed. Algorithms with a polynomial amount of FLOPs are discussed to compute the matrix polynomial adjoint and determinant in the instance of matrix polynomials with an affine structure. Also studied were the first two derivatives of the determinant, adjoint and some algorithms to compute them.

This chapter discussed the determinant and adjoint of matrix polynomials extensively, however the ideas here can be applied to performing differential calculus on a matrix consisting of the $(n-k) \times (n-k)$ minors. Importantly, the determinant is a multi-linear function, so computing higher-order minors can be accomplished via differentiation of the determinant. Most of the ideas presented in this chapter also generalize to matrix polynomials with an affine structure by considering directional derivatives. Alternatively, one may use Theorem 4.3.1 Theorem 4.4.1 or or Theorem 4.3.1 conjunction with the chain rule to compute the derivatives of affinely structured matrices. This amounts to a derivative computation and a matrix multiplication and can be more efficient in some instances.

We view this chapter as a first step towards a formally robust numerical algorithm to compute the adjoint of a matrix polynomial in a quasi-optimal number of FLOPs. The applications of this section have several applications in the study of the spectral properties of matrix polynomials and fast algorithms to find a nearby matrix polynomial with an interesting spectral structure. The next chapter will make use of these results extensively.

Chapter 5

The Approximate Smith Normal Form

This chapter discusses the problem of computing a nearby matrix polynomial with a prescribed spectral structure. The emphasis is on the finite spectral structure of a matrix polynomial. The techniques described for the finite spectral structure are easily generalized to handle the instance of the infinite spectral structure as a special case. Most of the results presented in this chapter are based on the conference paper [39] and a manuscript of the extended journal version [40]. This chapter is not about computing the Smith normal form of a matrix polynomial using floating point arithmetic; instead we discuss computing a nearby matrix polynomial with "an interesting" or non-generic Smith normal form. This chapter draws heavily on the ideas presented in Chapter 4, since several optimization problems involve the determinant and adjoint matrix.

5.1 Introduction

Matrix polynomials appear in many areas of computational algebra, control systems theory, differential equations and mechanics. The algebra of matrix polynomials is typically described assuming that the coefficients are from the field of real or complex numbers. However, in some applications, coefficients can come from measured data or contain some amount of uncertainty. As such, arithmetic may contain numerical errors and algorithms are prone to numerical instability.

One problem of computational importance is finding the Smith Normal Form (SNF, or simply Smith form) of a matrix polynomial. Given $\mathscr{A} \in \mathbb{R}[t]^{n\times n}$ of full rank, the Smith form $\mathcal{S}$ of $\mathcal{A}$ is a matrix polynomial

$$\mathcal{S} = \begin{pmatrix} s_1 & & & \\ & s_2 & & \\ & & \ddots & \\ & & & s_n \end{pmatrix} \in \mathbb{R}[t]^{n\times n},$$

where $s_1, \ldots, s_n$ are monic and $s_i \mid s_{i+1}$ for $1 \le i < n$, such that there exist unimodular $\mathcal{U}, \mathcal{V} \in \mathbb{R}[t]^{n\times n}$ (i.e., with determinants in $\mathbb{R}^*$) with $\mathcal{S} = \mathcal{U}\mathcal{A}\mathcal{V}$. The Smith form always exists and is unique though the matrices $\mathcal{U}, \mathcal{V}$ are not unique [44, 65]. The diagonal entries $s_1, \ldots, s_n$ are referred to as the *invariant factors* of $\mathcal{A}$.

The Smith form is important as it reveals the structure of the polynomial lattice of rows and columns, as well as the effects of localizing at individual eigenvalues. That is, it characterizes how the rank decreases as the variable t is set to various eigenvalues. The form is closely related to the more general *Smith-McMillan form* for matrices of rational functions, a form that reveals the structure of eigenvalues at infinity.

In an exact setting, computing the Smith form has been well studied and very efficient procedures are available (see [67] and the references therein). However, in the case that coefficients contain uncertainties, the problem is much less understood. Numerical methods to compute the Smith form of a matrix polynomial typically rely on linearization and orthogonal transformations [7, 26, 27, 28, 104] to infer the Smith form of a nearby matrix polynomial via the Jordan blocks in the Kronecker canonical form (see [65]). These linearization techniques are backwards stable, and for many problems this is sufficient to ensure that the computed solutions are computationally useful when a problem is continuous. However, the eigenvalues of a matrix polynomial are not necessarily continuous functions of the coefficients of the matrix polynomial, and backwards stability is not always sufficient to ensure computed solutions are useful in the presence of discontinuities. These methods are also unstructured in the sense that the computed non-trivial Smith form may not be the Smith form of a matrix polynomial with a prescribed coefficient structure. In extreme instances, the unstructured backwards error can be arbitrarily small, while the structured distance to an interesting Smith form is relatively large. This is often seen in problems with prescribed sparsity patterns or zero-coefficients. Numerical methods can also fail to compute meaningful results on some problems due to uncertainties. Examples of such problems include nearly rank deficient matrix polynomials, repeated eigenvalues or eigenvalues that are close together and other ill-posed instances. The above issues are largely resolved by our optimization-based approach, though at a somewhat higher computational cost.

The invariant factors $s_1, \ldots, s_n$ of a matrix $\mathcal{A} \in \mathbb{R}[t]^{n\times n}$ can also be defined via the *determinantal divisors* $\delta_1, \ldots, \delta_n \in \mathbb{R}[t]$, where

$$\delta_i = \mathrm{GCD}\left\{\text{all } i \times i \text{ minors of } \mathcal{A}\right\} \in \mathbb{R}[t].$$

Then $s_1 = \delta_1$ and $s_i = \delta_i/\delta_{i-1}$ for $2 \le i \le n$ (and $\delta_n = \det(\mathcal{A})$). In the case of 2×2 matrix polynomials, computing the nearest non-trivial Smith form is thus equivalent to finding the nearest matrix polynomial whose polynomial entries have a non-trivial GCD. This points to a significant difficulty: approximate GCD problems can have infima that are *unattainable*. That is, there are co-prime polynomials with nearby polynomials with a non-trivial GCD at distances arbitrarily approaching an infimum, while at the infimum itself the GCD is trivial (see, e.g., [36]). This issue extends to Smith forms as is seen in the following example.

Example 5.1.1. *Let $f = t^2 - 2t + 1$ and $g = t^2 + 2t + 2$. We first seek $\widetilde{f}, \widetilde{g} \in \mathbb{R}[t]$ of degree at most 2 such that $\gcd(\widetilde{f}, \widetilde{g}) = \gamma t + 1$ at a minimal distance $\|f - \widetilde{f}\|_2^2 + \|g - \widetilde{g}\|_2^2$ for some $\gamma \in \mathbb{R}$. Using the approach of Karmarkar & Lakshman [72] it is shown [51, Example 3.3.6] that this distance is $(5\gamma^4 - 4\gamma^3 + 14\gamma^2 + 2)/(\gamma^4 + \gamma^2 + 1)$. This distance has an infimum of 2 at $\gamma = 0$. However, at $\gamma = 0$ we have $\gcd(\widetilde{f}, \widetilde{g}) = 1$ even though $\deg \gcd(\widetilde{f}, \widetilde{g}) > 0$ for all $\gamma \neq 0$.*

Now consider the matrix $\mathcal{A} = \mathrm{diag}(f, g) \in \mathbb{R}[t]^{2\times 2}$. For $\mathcal{A}$ to have a non-trivial Smith form we must perturb f, g such that they have a non-trivial GCD, and thus any such perturbation must be at a distance of at least 2. However, the perturbation of distance precisely 2 has a trivial Smith form. There is clearly no merit to perturbing the off-diagonal entries of $\mathcal{A}$.

Our work indirectly involves measuring the sensitivity to the eigenvalues of $\mathcal{A}$ and the determinant of $\mathcal{A}$. Thus we differ from most sensitivity and perturbation analysis [4, 99] since we also study how perturbations affect the invariant factors, instead of the roots of the determinant. Additionally our theory is able to support the instance of $\mathcal{A}$ being rank deficient and having degree exceeding one. One may also approach the problem geometrically in the context of manifolds [29, 30]. We do not consider the manifold approach directly since it does not yield numerical algorithms.

5.1.1 Outline

We address two fundamental questions in this chapter:

1. What does it mean for a matrix polynomial $\mathscr{A}$ to have a non-trivial Smith form numerically?
2. How far is $\mathscr{A}$ from another matrix polynomial with an interesting or non-trivial Smith form?

We formulate the answers to these questions as solutions to continuous optimization problems. The main contributions of this chapter are deciding when $\mathscr{A}$ has an interesting Smith form, providing bounds on a "radius of triviality" around $\mathscr{A}$ and a structured stability analysis on iterative methods to compute a structured matrix polynomial with desired spectral properties.

The remainder of the chapter is organized as follows. In Section 5.2 we give the notation and terminology along with some needed background used in our work. Section 5.3 discusses the approximate Smith form computation as an optimization problem and provide some new bounds on the distance to non-triviality. We present an optimization algorithm in Section 5.4 with local stability properties and rapid local convergence to compute a nearby matrix polynomial with a non-trivial Smith form and discuss implementation details. A method to compute a matrix polynomial with a prescribed lower bound on the number of ones is discussed in Section 5.5. The instances of prescribing the whole spectral structure is dealt with in Section 5.6 and the prescribed Smith form and eigenvalues at infinity is dealt with in Section 5.7. The chapter ends with a discussion of our implementation and examples.

5.2 Preliminaries

In this section we explore the topology of the approximate Smith normal form and discuss basic results concerning the notion of both *attainable* and *unattainable* solutions. In the context of the KKT conditions, the unattainable solutions are irregular in the meaning that Lagrange multipliers generally do not exist. In the instance of irregular solutions, we describe how to *regularize* the optimization problem to obtain a problem where Lagrange multipliers exist.

In this chapter we are mainly concerned with preserving the zero structure of a matrix polynomial, that is we do not change zero-coefficients or increase the degrees of entries.

Recall that the *rank* of a matrix polynomial is the maximum number of linearly independent rows or columns as a vector space over $\mathbb{R}(t)$. This is the rank of the matrix $\mathscr{A}(\omega)$ for any $\omega \in \mathbb{C}$ except when ω is an eigenvalue of $\mathscr{A}(t)$.

Definition 5.2.1 (McCoy Rank and Non-Trivial SNF)**.** *The McCoy rank of $\mathcal{A}$ is* $\min_{\omega\in\mathbb{C}}\{\operatorname{rank}\mathcal{A}(\omega)\}$*, which is the lowest rank when $\mathcal{A}$ is evaluated at an eigenvalue (or any value in $\mathbb{C}$ for that matter). The McCoy rank is also the number of ones in the Smith form, or equivalently, if $\mathcal{A}$ has r non-trivial invariant factors, then the McCoy rank of $\mathcal{A}$ is $n-r$. The matrix polynomial $\mathcal{A}$ is said to have a* non-trivial Smith form *if the McCoy rank is at most $n-2$, or equivalently, has two or more invariant factors of non-zero degree.*

Problem 5.2.2 (Approximate SNF Problem)**.** *Given a matrix polynomial $\mathcal{A}\in\mathbb{R}[t]^{n\times n}$, find the distance to a non-trivial SNF and, if possible, a matrix polynomial $\widehat{\mathcal{A}}\in\mathbb{R}[t]^{n\times n}$ of prescribed coefficient structure that has a prescribed McCoy rank of $n-r$ for $r\geq 2$ such that $\|\mathcal{A}-\widehat{\mathcal{A}}\|$ is minimized under $\|\cdot\|$.*

We will consider $\|\cdot\| = \|\cdot\|_F$ to be the Frobenius norm. Computing the nearest (if it exists) McCoy rank $n-2$ matrix is the *approximate SNF*. Note that rank deficient matrix polynomials have a non-trivial SNF under this definition, although the SNF may be *unattainable* (the matrix polynomial is rank deficient and all of the non-zero invariant factors are one).

Problem 5.2.3 (Lower McCoy Rank Approximation Problem)**.** *Computing the nearest (if it exists) McCoy rank $n-r$ matrix for $r\geq 3$ is a lower McCoy rank approximation.*

In a generic sense, the nearest matrix polynomial with an interesting SNF will have McCoy rank $n-2$ with probability one, but many matrices arising from applications are expected to have more interesting (i.e. the invariant factors have a richer or non-generic multiplicity structure) Smith forms nearby.

As described in Section 5.1, it is possible that the distance to a non-trivial SNF is not attainable. That is, there is a solution that is approached asymptotically, but where the Smith form is trivial at the infimum. Fortunately, in most instances of interest, solutions will generally be attainable, and we will later discuss how to identify and compute unattainable solutions. Problem 5.2.2 and Problem 5.2.3 admit the nearest rank $n-1$ or rank $n-2$ matrix polynomial as a special case. However the computational challenges are fundamentally different for non-trivial instances.

5.2.1 Basic Results

In this subsection we review some basic results needed to analyze the topology of the approximate Smith form problem. We draw heavily on the notion of a generalized Sylvester matrix.

For a matrix $\mathcal{A} \in \mathbb{R}[x]^{n\times n}$, we know that $s_n = \delta_n/\delta_{n-1}$, the quotient of the determinant and the GCD of all $(n-1)\times(n-1)$ minors. Since these minors are precisely the entries of the adjoint matrix, it follows that $\mathcal{A}$ has a non-trivial Smith form if and only if the GCD of all entries of the adjoint is non-trivial, that is, $\deg(\gcd(\{\mathrm{Adj}(\mathcal{A})_{ij}\})) \geq 1$. In order to obtain bounds on the distance to a matrix having a non-trivial Smith form, we consider an approximate GCD problem of the form

$$\min \|\Delta\mathcal{A}\| \;\; \text{subject to} \;\; \deg\left(\gcd\left\{\mathrm{Adj}\left(\mathcal{A}+\Delta\mathcal{A}\right)_{ij}\right)\right\} \neq 1.$$

If this was a classical approximate GCD problem, then the use of Sylvester-like matrices would be sufficient. However, in our problem the degrees of the entries of the adjoint may change under perturbations. In order to perform an analysis, we need to study a family of generalized Sylvester matrices that allow higher-degree zero coefficients to be perturbed.

The computation of the GCD of many polynomials is typically embedded into a scalar matrix problem using the classical Sylvester matrix. However, in our case we want to look at GCDs of nearby polynomials but with the added wrinkle that the degrees of the entries of the individual polynomials may change under perturbations. In order to perform such an analysis, we need to study a family of generalized Sylvester matrices that allow higher-degree zero coefficients to be perturbed.

Let $\mathbf{f} = (f_1, \ldots, f_k) \in \mathbb{R}[t]^k$ be a vector of polynomials with degrees $\mathbf{d} = (d_1, \ldots, d_k)$ ordered as $d_j \geq d_{j+1}$ for $1 \leq j \leq k-1$. Set $d = d_1$ and $\ell = \max(d_2, \ldots, d_k)$ and suppose that for each $i \in \{2, \ldots, k\}$ we have $f_i = \sum_{1\leq j\leq \ell} f_{ij}t^j$. Recall that defined the *generalized Sylvester matrix* of $\mathbf{f}$ is defined as

$$\mathrm{Syl}(\mathbf{f}) = \mathrm{Syl}_{\mathbf{d}}(\mathbf{f}) = \begin{pmatrix} \phi_{\ell-1}(f_1)^T \\ \phi_{d-1}(f_2)^T \\ \vdots \\ \phi_{d-1}(f_k)^T \end{pmatrix} \in \mathbb{R}^{(\ell+(k-1)d)\times(\ell+d)}.$$

Some authors, e.g., [33, 106], refer to such a matrix as an expanded Sylvester matrix or generalized resultant matrix. The generalized Sylvester matrix has many useful properties pertaining to the Bézout coefficients. However, we are only concerned with the well known result that $\gcd(\mathbf{f}) = \gcd(f_1, \ldots, f_k) = 1$ if and only if $\mathrm{Syl}_{\mathbf{d}}(\mathbf{f})$ has full rank [106].

Sometimes treating a polynomial of degree d as one of larger degree is useful. This can be accomplished by constructing a similar matrix and padding rows and columns with zero entries. The generalized Sylvester matrix of degree at most $\mathbf{d}' \geq \mathbf{d}$ (component-wise) of $\mathbf{f}$

is defined analogously as $\operatorname{Syl}_{\mathbf{d}'}(\mathbf{f})$, taking d to be the largest degree entry and ℓ to be the largest degree of the remaining entries of $\mathbf{d}'$. Note that $\ell = d$ is possible and typical. If the entries of $\mathbf{f}$ have a non-trivial GCD (that is possibly unattainable) under a perturbation structure $\Delta\mathbf{f}$, then it is necessary that $\operatorname{Syl}_{\mathbf{d}'}(\mathbf{f})$ is rank deficient, and often this will be sufficient.

If we view the entries of $\mathbf{f}$ as polynomials of degree $\mathbf{d}'$ and $\mathbf{d}'_i > \mathbf{d}_i$ for all i, then the entries of $\mathbf{f}$ have an unattainable GCD of distance zero, typically of the form $1 + \varepsilon t$, which can be normalized to $t + \varepsilon^{-1}$. Accordingly, as $\varepsilon \to 0$ we have that $\|t + \varepsilon^{-1}\| \to \infty$. In other words, the underlying approximate GCD problem is ill-posed in a sense that the solution is unattainable, and the magnitude of the GCD approaches infinity. In order to study the theory of unattainable GCD's, sometimes referred to as GCD's at infinity, we need to study the notion of a degree reversed polynomial.

Lemma 5.2.4. *If* $\max(\mathbf{d}) = \max(\mathbf{d}')$ *then* $\operatorname{Syl}_{\mathbf{d}}(\mathbf{f})$ *has full rank if and only if and* $\operatorname{Syl}_{\mathbf{d}'}(\mathbf{f})$ *has full rank.*

Proof. Let d and ℓ be the largest and second largest entries of $\mathbf{d}$ and ℓ' be the second largest entry of $\mathbf{d}'$. The result follows from the main theorem of [106] by considering the case of $\ell' = d$. $\square$

This lemma characterizes the (generic) case when elements of maximal degree of $\mathbf{f}$ do not change under perturbations, in which case the generalized Sylvester matrix still meaningfully encodes GCD information. However, it is possible that $\operatorname{Syl}_{\mathbf{d}}(\mathbf{f})$ has full rank and $\operatorname{Syl}_{\mathbf{d}'}(\mathbf{f})$ is rank deficient but the distance to a non-trivial GCD is not zero. This can occur when $\mathbf{d}_j = \mathbf{d}'_j$ for some j and $\mathbf{d}' \geq \mathbf{d}$. To understand the most general case, we need to look at generalized Sylvester matrices of the reversal of several polynomials.

Definition 5.2.5. *The* degree d reversal *of* $f \in \mathbb{R}[t]$ *of degree at most* d *is defined as* $\operatorname{rev}_d(f) = t^d f(t^{-1})$. *For a vector of polynomials* $\mathbf{f} \in \mathbb{R}[t]^k$ *of degrees at most* $\mathbf{d} = (d_1, \dots, d_k)$ *the* degree $\mathbf{d}$ reversal *of* $\mathbf{f}$ *is the vector* $\operatorname{rev}_{\mathbf{d}}(\mathbf{f}) = (\operatorname{rev}_{d_1}(f_1), \dots, \operatorname{rev}_{d_k}(f_k))$.

The following theorem enables us to determine if unattainable solutions are occurring in an approximate GCD problem with an arbitrary (possibly non-linear) structure on the coefficients.

Theorem 5.2.6. *Let* $\mathbf{f}$ *be a vector of non-zero polynomials of degree at most* d. *Suppose that* $\operatorname{Syl}_{\mathbf{d}}(\mathbf{f})$ *has full rank and* $\operatorname{Syl}_{\mathbf{d}'}(\mathbf{f})$ *is rank deficient, where the perturbations* $\Delta\mathbf{f}$ *have degrees at most* $\mathbf{d}'$ *and the entries of* $\mathbf{f}$ *have degrees* $\mathbf{d}$. *Then* $\mathbf{f}$ *has an* unattainable *non-trivial GCD of distance zero under the perturbation structure* $\Delta\mathbf{f}$ *if and only if* $\operatorname{Syl}(\operatorname{rev}_{\mathbf{d}'}(\mathbf{f}))$ *is rank deficient.*

Proof. Suppose that $\mathrm{Syl}(\mathrm{rev}_{\mathbf{d}'}(\mathbf{f}))$ has full rank. Then $\gcd(\mathrm{rev}_{\mathbf{d}'}(\mathbf{f})) = 1$, hence $\mathbf{f}$ does not have an unattainable non-trivial GCD, since $\gcd(\mathbf{f}) = 1$. Conversely, suppose that $\mathrm{Syl}(\mathrm{rev}_{\mathbf{d}'}(\mathbf{f}))$ is rank deficient. Then, t is a factor of $\gcd(\mathrm{rev}_{\mathbf{d}'}(\mathbf{f}))$ but t is not a factor of $\gcd(\mathrm{rev}_{\mathbf{d}}(\mathbf{f}))$. Accordingly, all entries of $\mathbf{f}+\Delta\mathbf{f}$ may increase in degree and so the distance of $\mathbf{f}$ having a non-trivial GCD is zero, and so is unattainable. □

If the generalized Sylvester matrix of $\mathbf{f}$ has full rank, but the generalized Sylvester matrix that encodes the perturbations $\mathbf{f}+\Delta\mathbf{f}$ is rank deficient, then either there is an unattainable GCD, or the generalized Sylvester matrix is rank deficient due to over-padding with zeros. Theorem 5.2.6 provides a reliable way to detect this over-padding.

Definition 5.2.7. *We say that $\mathcal{A}$ has an* unattainable non-trivial Smith form *if* $\gcd(\mathrm{Adj}(\mathcal{A})) = 1$ *and* $\gcd(\mathrm{Adj}(\mathcal{A}+\widetilde{\Delta\mathcal{A}})) \neq 1$ *for an arbitrarily small perturbation $\Delta\mathcal{A}$ of prescribed affine structure $\boldsymbol{\Delta}(\cdot)$.*

We recall that $\boldsymbol{\Delta}(\cdot)$ and $\Delta\mathcal{A}$ are defined in the same was as in Definition 3.1.1. We have to carefully consider structured perturbations, because some matrix polynomials have an unattainable non-trivial SNF under unstructured perturbations, but have an attainable non-trivial SNF under structured perturbations (perturbations that preserve the degree of entries or support of entries are structured). Solutions that cannot be attained correspond to an eigenvalue at infinity of $\mathcal{A}$ with a non-trivial spectral structure. Such examples are easily constructed when $\det(\mathcal{A})$ or $\mathrm{Adj}(\mathcal{A})$ have non-generic degrees.

Example 5.2.8. *Let*

$$\mathcal{A} = \begin{pmatrix} t & t-1 \\ t+1 & t \end{pmatrix} \in \mathbb{R}[t]^{2\times 2} \quad \textit{and} \quad \mathcal{C} = \begin{pmatrix} \mathcal{A} & \\ & \mathcal{A} \end{pmatrix} \in \mathbb{R}[t]^{4\times 4}.$$

Then $\mathcal{C}$ has an unattainable non-trivial Smith form if all perturbations to $\mathcal{A}$ are support or degree preserving (i.e. they preserve zero entries or do not increase the degree of each entry), both linear structures. Note that $\mathcal{A}$ and $\mathcal{C}$ are both unimodular. However small perturbations to the non-zero coefficients of $\mathcal{A}$ make $\mathcal{A}+\Delta\mathcal{A}$ non-unimodular.

The Smith form of $\mathrm{rev}(\mathcal{C}) = t\mathcal{C}|_{t=t^{-1}}$ *is*

$$\mathrm{SNF}(\mathrm{rev}(\mathcal{C})) = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & t^2 & \\ & & & t^2 \end{pmatrix},$$

which implies that the eigenvalue at infinity of $\mathscr{A}$ has a non-trivial spectral structure. The eigenvalue at infinity having a non-trivial spectral structure implies that the SNF of $\mathcal{C}$ is unattainable. Note that this is equivalent to saying that $\mathcal{C}$ has a non-trivial Smith-McMillan form.

These examples are non-generic. Generically, the degree of all entries in the adjoint will be $(n-1)d$ and will remain unchanged locally under perturbations to the coefficients. Computing the distance to the nearest matrix polynomial with a non-trivial Smith form under a prescribed perturbation structure can be formulated as finding the nearest rank deficient (structured) generalized Sylvester matrix of the adjoint or the $\mathbf{d}'$ reversal of the adjoint.

5.3 Nearest Rank Deficient Structured Generalized Sylvester Matrix

Suppose that $\mathscr{A} \in \mathbb{R}[t]^{n\times n}$ of degree at most d has a trivial Smith form and does not have an unattainable non-trivial Smith form. Then one method to compute a lower bound on the distance the entries of $\mathscr{A}$ need to be perturbed to have an attainable or unattainable non-trivial Smith form is to solve

$$\inf \|\Delta\mathscr{A}\| \quad \text{subject to} \quad \begin{cases} \operatorname{rank}(\operatorname{Syl}_{\mathbf{d}'}(\operatorname{Adj}(\mathscr{A}+\widetilde{\Delta}\mathscr{A}))) < e, \\ e = \operatorname{rank}(\operatorname{Syl}_{\mathbf{d}'}(\operatorname{Adj}(\mathscr{A}))). \end{cases} \tag{5.1}$$

Here $\mathbf{d}'$ is the vector of the largest possible degrees of each entry of $\operatorname{Adj}(\mathscr{A}+\widetilde{\Delta}\Delta\mathscr{A})$, and $\widetilde{\Delta}\mathscr{A}$ is a prescribed linear or affine perturbation structure.

It is sufficient to compute $\max(\mathbf{d}')$, a quantity which will generically be $(n-1)d$. For non-generic instances we require the computation of $\mathbf{d}'$. This optimization problem is non-convex, but multi-linear in each coefficient of $\Delta\mathscr{A}$.

We do not attempt to solve this problem directly via numerical techniques, since it enforces a necessary condition that is often sufficient. Instead we use it to develop a theory of solutions which can be exploited by faster and more robust numerical methods.

Lemma 5.3.1. *Let* $\mathbf{f}$ *be a vector of polynomials with degrees* $\mathbf{d}$ *and admissible perturbations* $\Delta\mathbf{f}$ *of degrees* $\mathbf{d}'$ *where* $\max(\mathbf{d}) \le \max(\mathbf{d}')$*. Then the family of generalized Sylvester matrices* $\operatorname{Syl}_{\mathbf{d}'}(\mathbf{f})$ *of rank at least* e *form an open set under the perturbations* $\Delta\mathbf{f}$*.*

Proof. By the degree assumption on $\Delta\mathbf{f}$ we have that for an infinitesimal $\Delta\mathbf{f}$ that $\operatorname{Syl}_{\mathbf{d}'}(\mathbf{f})$ and $\operatorname{Syl}_{\mathbf{d}'}(\Delta\mathbf{f})$ have the same dimension. Accordingly, let us suppose that $\operatorname{Syl}_{\mathbf{d}'}(\mathbf{f})$ has rank at least e. Then the Sylvester matrix in question must have rank at least e in an open-neighborhood around it. In particular, when $\|\operatorname{Syl}_{\mathbf{d}'}(\Delta\mathbf{f})\|_2 < \sigma_e(\operatorname{Syl}_{\mathbf{d}'}(\mathbf{f}))$ then $\operatorname{rank}(\operatorname{Syl}_{\mathbf{d}'}(\mathbf{f}+\Delta\mathbf{f})) \geq \operatorname{rank}(\operatorname{Syl}_{\mathbf{d}'}(\mathbf{f}))$ and the result follows. $\square$

Theorem 5.3.2. *The optimization problem* (5.1) *has an attainable global minimum under linear perturbation structures.*

Proof. Let $\mathcal{S}$ be the set of all rank at most $e-1$ generalized Sylvester matrices of prescribed shape by $\mathbf{d}'$ and $\operatorname{Adj}(\mathcal{A})$. Lemma 5.3.1 implies that $\mathcal{S}$ is topologically closed.

Let $\mathcal{R} = \{\operatorname{Syl}_{\mathbf{d}'}(\operatorname{Adj}(\mathcal{C}))$ subject to $\|\mathcal{C}\| \leq \|\mathcal{A}\|\}$, where the generalized Sylvester matrices are padded with zeros to have the appropriate dimension if required. Since $\Delta\mathcal{A}$ has a linear perturbation structure, a feasible point is always $C = -\mathcal{A}$. By inspection $\mathcal{R}$ is seen to be a non-empty set that is bounded and closed.

The functional $\|\cdot\|$ is continuous over the non-empty closed and bounded set $\mathcal{S} \cap \mathcal{R}$. Let $\mathcal{B} \in \mathcal{S} \cap \mathcal{R}$. By Weierstrass's theorem $\|\mathcal{A} - \mathcal{B}\|$ has an attainable global minimum over $\mathcal{S} \cap \mathcal{R}$. $\square$

Note that if a feasible point exists under an affine perturbation structure, then a solution to the optimization problem exists as well. What this result says is that computing the distance to non-triviality is generally a well-posed problem, even though computing a matrix polynomial of minimum distance may be ill-posed (the solution is unattainable). The same results also hold when working over the $\mathbf{d}'$ reversed coefficients. A similar argument is employed by [71, Theorem 2.1].

5.3.1 Bounds on the Distance to non-triviality

Suppose that $\mathcal{A} \in \mathbb{R}[t]^{n\times n}$, of degree at most d, has a trivial Smith form and does not have an unattainable non-trivial Smith form. This section provides some basic bounds on the distance coefficients of $\mathcal{A}$ need to be perturbed to have a non-trivial Smith form. The bounds we derive are unstructured, although they can be generalized to several perturbation structures (such as ones that preserve the degree or support of entries) in a straight forward manner.

If we consider the mapping $\operatorname{Adj}(\cdot)$ as a vector-valued function from $\mathbb{R}^{n^2(d+1)} \to \mathbb{R}^{n^2((n-1)d+1)}$ (with some coordinates possibly fixed to zero), then we note that the mapping is locally

Lipschitz. More precisely, there exists $c > 0$ such that for a sufficiently small $\Delta\mathcal{A}$ that

$$\|\operatorname{Adj}(\mathcal{A}) - \operatorname{Adj}(\mathcal{A} + \Delta\mathcal{A})\| \leq c\|\Delta\mathcal{A}\|.$$

The quantity c can be bounded above by the (scalar) Jacobian matrix $\nabla \operatorname{Adj}(\cdot)$ evaluated at $\mathcal{A}$. A local upper bound for c is approximately $\|\nabla \operatorname{Adj}(\mathcal{A})\|_2$. We can invoke Theorem 4.4.1 if $\mathcal{A}$ has full rank. By considering $\hat{c} = \left\| \left[\Phi_{(n-1)d}(I \otimes \mathcal{A})\right]^{+} \right\|_2$, we obtain the (absolute) first-order *approximate* perturbation bound

$$\|\operatorname{Adj}(\mathcal{A}) - \operatorname{Adj}(\mathcal{A} + \Delta\mathcal{A})\|_F \leq \hat{c}(n + \sqrt{n})(d+1)\|\operatorname{Adj}(\mathcal{A})\|_F \|\Delta\mathcal{A}\|_F.$$

The entries of $\nabla \operatorname{Adj}(\mathcal{A})$ consist of the coefficients of the $(n-2) \times (n-2)$ minors of $\mathcal{A}$. This follows because $\operatorname{Adj}(\cdot)$ is a multi-linear vector mapping and the derivative of each entry is a coefficient of the leading coefficient with respect to the variable of differentiation. The size of each minor can be bounded above (albeit poorly) by Hadamard's inequality (Goldstein-Graham variant, see [80]). As such, we have the sequence of bounds

$$\|\nabla \operatorname{Adj}(\mathcal{A})\|_2 \;\leq\; n\sqrt{d+1}\|\nabla \operatorname{Adj}(\mathcal{A})\|_\infty \;\leq\; n^3(d+1)^{5/2}\|\mathcal{A}\|_\infty^{n-2}(d+1)^{n-2}n^{(n-2)/2},$$

where $\|\mathcal{A}\|_\infty$ is understood to be a vector norm and $\|\nabla \operatorname{Adj}(\mathcal{A})\|_\infty$ is understood to be a matrix norm. The bound in question can be used in conjunction with the SVD to obtain a lower bound on the distance to a matrix polynomial with a non-trivial Smith form.

Theorem 5.3.3. *Suppose that* $\mathbf{d}' = (\gamma, \gamma \ldots, \gamma)$ *and* $\operatorname{Syl}_{\mathbf{d}'}(\operatorname{Adj}(\mathcal{A}))$ *has rank* e. *Then a lower bound on the distance to non-triviality is*

$$\frac{1}{\gamma\|\nabla \operatorname{Adj}(\mathcal{A})\|_F}\sigma_e(\operatorname{Syl}_{\mathbf{d}'}(\operatorname{Adj}(\mathcal{A}))).$$

Proof. We note that for polynomials $\mathbf{f}$ with degrees $\mathbf{d}'$ that $\|\operatorname{Syl}_{\mathbf{d}'}(\mathbf{f})\|_F = \gamma\|\mathbf{f}\|_F$. Accordingly, if $\Delta\mathcal{A}$ is a minimal perturbation to non-triviality, then (ignoring higher-order terms)

$$\begin{aligned}\frac{1}{\gamma}\sigma_e(\operatorname{Syl}_{\mathbf{d}'}(\operatorname{Adj}(\mathcal{A}))) &\leq \|\operatorname{Adj}(\mathcal{A}) - \operatorname{Adj}(\mathcal{A} + \Delta\mathcal{A})\|_F \\ &\leq \|\nabla \operatorname{Adj}(\mathcal{A})\|_F \|\Delta\mathcal{A}\|_F,\end{aligned}$$

and the theorem follows by a simple rearrangement. Note that $\|\cdot\|_2 \leq \|\cdot\|_F$. $\square$

If $\mathbf{d}'$ has different entries, then $\ell\|\mathbf{f}\|_F \leq \|\operatorname{Syl}_{\mathbf{d}'}(\mathbf{f})\|_F \leq \gamma\|\mathbf{f}\|_F$, where γ and ℓ are the largest and second-largest entries of $\mathbf{d}'$. The lower bound provided can also be improved using the Karmarkar-Lakshman distance [72] in lieu of the smallest singular value of the generalized Sylvester matrix, the $\mathbf{d}'$ reversal of the adjoint or other approximate GCD lower bounds (e.g., [6]).

5.4 Approximate SNF via Optimization

In this section we formulate the approximate Smith form problem as the solution to a continuous constrained optimization problem. We assume that the solutions in question are attainable and develop a method with rapid local convergence. As the problem is non-convex, our convergence analysis will be local.

5.4.1 Constrained Optimization Formulation

An equivalent statement to $\mathscr{A}$ having a non-trivial attainable Smith form is that $\operatorname{Adj}(\mathscr{A}) = \mathscr{F}^* h$ where $\mathscr{F}^*$ is a vector (or matrix) of scalar polynomials and h is a divisor of $\gcd(\operatorname{Adj}(\mathscr{A}))$. This directly leads to the following optimization problem:

$$\min \|\Delta\mathscr{A}\|_F^2 \quad \text{subject to} \quad \begin{cases} \operatorname{Adj}(\mathscr{A}+\Delta\mathscr{A}) = \mathscr{F}^* h, & \mathscr{F}^* \in \mathbb{R}[t]^{n\times n}, h \in \mathbb{R}[t], \\ \mathcal{N}_h \operatorname{vec}(h) = 1, & \mathcal{N}_h \in \mathbb{R}^{1\times(\deg(h)+1)}. \end{cases} \tag{5.2}$$

This is a multi-linearly structured approximate GCD problem which is a non-convex optimization problem. Instead of finding a rank deficient Sylvester matrix, we directly enforce that the entries of $\operatorname{Adj}(\mathscr{A})$ have a non-trivial GCD. The normalization requirement that $\mathcal{N}_h \operatorname{vec}(h) = 1$ is chosen to force h to have a non-zero degree, so that h is not a scalar. One useful normalization is to define $\mathcal{N}_h$ such that $\operatorname{lcoeff}(h) = 1$ (that is $\operatorname{lcoeff}(\cdot)$ is the leading coefficient of a polynomial). Explicitly, we assume the degree of the approximate GCD is known and make it monic. Of course, other valid normalizations also exist.

Since we are working over $\mathbb{R}[t]$, there will always be a quadratic, linear or zero factor of attainable solutions. If $h = 0$ then the approximate SNF of $\mathscr{A}$ is rank deficient and computing approximate SNF reduces to the nearest rank at most $n-1$ or $n-2$ matrix polynomial problems, both of which are well-understood [37, 38]. Assuming that we are now working in the non-zero case, we can assume generically that $\deg(h) = 1$ or $\deg(h) = 2$.

5.4.2 Lagrange Multipliers and Optimality Conditions

In order to solve our problem we will employ the method of Lagrange multipliers. The Lagrangian is defined as

$$L = \|\Delta\mathscr{A}\|_F^2 + \lambda^T \begin{pmatrix} \operatorname{vec}(\operatorname{Adj}(\mathscr{A}+\Delta\mathscr{A}) - \mathscr{F}^* h) \\ \mathcal{N}_h \operatorname{vec}(h) - 1 \end{pmatrix},$$

where λ is a vector of Lagrange multipliers.

A necessary first-order condition (KKT condition, e.g. [10]) for a tuple $z^\star = z^\star(\Delta\mathscr{A}, \mathscr{F}^*, h, \lambda)$ to be a regular (attainable) minimizer is that the gradient of L vanishes, that is,

$$\nabla L(z^\star) = 0. \tag{5.3}$$

Let J be the Jacobian matrix of the constraints defined as

$$J = \nabla_{\Delta\mathscr{A}, \mathscr{F}^*, h} \left(\operatorname{vec}(\operatorname{Adj}(\mathscr{A} + \Delta\mathscr{A}) - \mathscr{F}^* h) \right).$$

The second-order sufficiency condition for optimality at a local minimizer $z^\star$ is that

$$\ker(J(z^\star))^T \, \nabla^2_{xx} L(z^\star) \, \ker(J(z^\star)) \succ 0, \tag{5.4}$$

that is, the Hessian with respect to $x = x(\Delta\mathscr{A}, \mathscr{F}^*, h)$ is positive definite over the kernel of the Jacobian of the constraints. The vector x corresponds to the variables in the affine structure of $\Delta\mathscr{A}$,$\mathscr{F}^*$, and h. If (5.3) and (5.4) both hold, then $z^\star$ is necessarily a local minimizer of (5.2). Of course, it is also necessary that $\ker(J(z^\star))^T \, \nabla^2_{xx} L(z^\star) \, \ker(J(z^\star)) \succeq 0$ at a minimizer, which is the second-order necessary condition. Our strategy for computing a local solution is to solve $\nabla L = 0$ using a Newton-like method.

5.4.3 An Implementation with Local Quadratic Convergence

A problem with Newton-like methods is that when the Hessian is rank deficient or ill-conditioned, then the Newton step becomes ill-defined or the rate of convergence degrades. The proposed formulation of our problem can encounter a rank deficient Hessian (this is due to over padding some vectors with zero entries or redundant constraints). Despite this we are still able to obtain a method with rapid local convergence under a very weak normalization assumption.

In order to obtain rapid convergence we make use of the Levenberg-Marquart (LM) algorithm. If $H = \nabla^2 L$, then the LM iteration is defined as repeatedly solving for $z^{(k+1)} = z^{(k)} + \Delta z^{(k)}$ by

$$(H^T H + \nu_k I)\Delta z^{(k)} = -H^T \nabla L(z^{(k)}) \text{ where } z = \begin{pmatrix} x \\ \lambda \end{pmatrix} \in \mathbb{R}^\ell,$$

for some $\ell > 0$ while using $\|\nabla L\|_2$ as a merit function. The speed of convergence depends on the choice of $\nu_k > 0$. Note that since LM is essentially a regularized Gauss-Newton method, when the Hessian is rank deficient then we may converge to a stationary point of the merit function. If convergence to a stationary point of the merit function is detected, then the method of [109] can be used to replace LM in several instances.

Yamashita and Fukushima [110] show that, under a local-error bound condition, a system of non-linear equations $g(z) = 0$ approximated by LM will converge quadratically to a solution with a suitable initial guess. Essentially, what this says is that to obtain rapid convergence it is sufficient for regularity (J having full rank) to hold or second-order sufficiency, but it is not necessary to satisfy both. Note that we assume Lagrange multipliers exist. However, unlike the case when J has full rank, the multipliers need not be unique. The advantage of LM over other Newton methods is that this method is globalized[1] in exchange for an extra matrix multiplication, as $H^T H + \nu_k I$ is always positive definite, and hence always a descent direction for the merit function. We make the choice of $\nu_k \approx \|g(z)\|_2$ based on the results of [32].

Definition 5.4.1 (Local Error Bound)**.** *Let $Z^\star$ be the set of all solutions to $g(z) = 0$ and X be a subset of $\mathbb{R}^\ell$ such that $X \cap Z^\star \neq \emptyset$. We say that $\|g(z)\|$ provides a local error bound on $g(z) = 0$ if there exists a positive constant c such that $c \cdot \operatorname{dist}(z, Z^\star) \leq \|g(z)\|$ for all $z \in X$, where $\operatorname{dist}(\cdot)$ is the distance between a point and a set.*

Theorem 5.4.2. *If the second-order sufficiency condition (5.4) holds at an attainable solution to (5.2), then the local error-bound property holds.*

Proof. This result follows immediately from Section 3 of [109] and the references therein. $\square$

The bounds of Wright can be used to infer when quadratic convergence occurs for Newton-like methods. In this problem, perturbations to x are important in understanding how the problem behaves locally.

Remark 5.4.3. *Let $z = z(x, \lambda)$ where x is a vector of variables and λ is a vector of Lagrange multipliers, and define $g(z) = \nabla L(z)$. First suppose that both the second-order sufficiency condition (5.4) and first-order necessary condition (5.3) hold at the point $z^\star$. We can write the first-order expansion*

$$g(z^\star + \Delta z) = H(z^\star)(\Delta z) + O(\|\Delta z\|_2^2) \approx H(z^\star)(\Delta z),$$

[1]Here "globalized" means that the method will converge to a stationary point of the merit function, not a local extremum of the problem.

noting that $g(z^\star) = 0$. It is useful to observe that

$$H(z^\star) = \begin{pmatrix} H_{xx}(z^\star) & J^T(z^\star) \\ J(z^\star) & 0 \end{pmatrix}.$$

If $\Delta x = 0$ then the error-bound from [60] (main theorem) applies and we have that there exists $c_{hof} > 0$ such that $c_{hof}\|\Delta\lambda\| \le \|g(x, \lambda + \Delta\lambda)\|$. If $\Delta x \neq 0$ then $\left\| \begin{pmatrix} H_{xx}(z^\star) \\ J(z^\star) \end{pmatrix} \Delta x \right\| \approx \|g(x + \Delta x, \lambda)\|$ and (5.4) implies that $H(z^\star)(\Delta z) = 0 \implies \Delta x = 0$, so

$$\sigma_{\min} \begin{pmatrix} H_{xx}(z^\star) \\ J(z^\star) \end{pmatrix} \|\Delta x\| \lesssim \|g(x + \Delta x, \lambda)\|,$$

so there exists $c_{\sigma_{\min}} > 0$ when $\|\Delta x\|$ is sufficiently small such that $c_{\sigma_{\min}}\|\Delta x\| \le \|g(x + \Delta x, \lambda)\|$. Note that $c_{\sigma_{\min}} \approx \sigma_{\min} \begin{pmatrix} H_{xx}(z^\star) \\ J(z^\star) \end{pmatrix}$.

The first-order approximation implies that when $\|\Delta z\|$ is sufficiently small that

$$g(z^\star + \Delta z) \approx H(z^\star)(\Delta z) = H(z^\star) \begin{pmatrix} \Delta x \\ 0 \end{pmatrix} + H(z^\star) \begin{pmatrix} 0 \\ \Delta\lambda \end{pmatrix} \approx g(x + \Delta x, \lambda) + g(x, \lambda + \Delta\lambda).$$

The key idea is to separate the problem into the cases of $\Delta x = 0$ and $\Delta x \neq 0$, and then derive error bounds for each case. The important part of the discussion is that if one can estimate $c_{\sigma_{\min}}$ then one can often infer when quadratic convergence occurs.

The second-order sufficiency assumption is not necessary to derive error bounds bounds. It is straightforward to show the local error bound property holds if $J(z^\star)$ has full rank, as the Lagrange multipliers will be (locally) unique, hence the solution is (locally) unique. Alternatively, if J had constant rank in a non-trivial open neighborhood around a solution, then a similar argument could be made about the local error-bound property.

Theorem 5.4.4. *The second-order sufficiency condition holds at minimal solutions with Lagrange multipliers of minimal norm if h is of maximal degree and monic and the minimal structured perturbation $\|\Delta\mathscr{A}^\star\|$ is sufficiently small.*

Proof. The Hessian of L with respect to $x = x(\Delta A, \mathscr{F}^*, h)$ is

$$\nabla^2_{xx} L = H_{xx} = \begin{pmatrix} F + 2I & & \\ & & E \\ & E^T & \end{pmatrix},$$

where F is a square matrix with zero diagonal whose entries are a multi-linear polynomial in λ and $\Delta\mathcal{A}$ and E^T is a matrix whose entries are homogeneous linear functions in λ.

If $\Delta\mathcal{A}^\star = 0$ then $\lambda^\star = 0$. Hence both $E = 0$ and $F = 0$ and so, if $y \in \ker(H_{xx}) \cap \ker(J)$ then $y = \begin{pmatrix} 0 & y_2 & y_3 \end{pmatrix}^T$. The Jacobian of the constraints may be written (up to permutation) as

$$J = \begin{pmatrix} * & \mathcal{C}_h & \mathcal{C}_{\mathcal{F}^*} \\ & & \mathcal{N}_h \end{pmatrix},$$

where $*$ are blocks corresponding to differentiation with respect to variables in $\Delta\mathcal{A}$ and the blocks $\mathcal{C}_{\mathcal{F}^*}$ and $\mathcal{C}_h$ consist of block convolution and convolution matrices that correspond to multiplication by $\mathcal{F}^*$ and h, respectively. The block $\mathcal{N}_h$ contains a normalization vector to ensure that h has the appropriate degree. $Jy = 0$ implies that there exists a vector of polynomials v and a polynomial u with the same degrees as $\mathcal{F}^*$ and h such that $\mathcal{F}^*u + vh = 0$ and $\mathcal{N}_h \operatorname{vec}(u) = 0$.

We have that h is a factor of both $\mathcal{F}^*u$ and vh. Since $\gcd(\mathcal{F}^*, h) = 1$ it must be that h is a factor of u. It follows that $\deg(u) = \deg(h)$, so there exists some $\alpha \neq 0$ such that $\alpha u = h$. Since h is monic, we have that $\mathcal{N}_h \operatorname{vec}(h) = 1$ but $\mathcal{N}_h \operatorname{vec}(u) = 0$, which implies that $\alpha = 0$, and so $u = 0$. We have that $vh = 0$ and this implies $v = 0$. Hence $\ker(J) \cap \ker(H_{xx}) = 0$ and second-order sufficiency holds when $\|\Delta\mathcal{A}^*\| = 0$.

If $\|\Delta\mathcal{A}^*\|$ is sufficiently small, then $\|F\|$ will be sufficiently small so that $F + 2I$ has full rank. Accordingly, we have that

$$\ker \begin{pmatrix} F + 2I & & \\ & 0 & E \\ & E^T & 0 \end{pmatrix} \subseteq \ker \begin{pmatrix} 2I & & \\ & 0 & \\ & & 0 \end{pmatrix}. \qquad \square$$

We remark that the techniques in the proof are very similar to those of [112] and [36] to show that a Jacobian matrix appearing in approximate GCD computations of two (or more) polynomials has full rank. If we over-estimated the degrees of $\mathcal{F}^*$ then H_{xx} would have some columns and rows consisting of zero (the block-convolution matrices would be padded with extra zero entries).

In the proof of Theorem 5.4.4 we note that

$$\nabla^2_{xx} L = \nabla^2_{xx} \|\Delta\mathcal{A}\|_F^2 + \nabla_x \lambda^T J.$$

The matrix $F = \nabla_{\Delta\mathcal{A}} \lambda^T J_{\text{Adj}}(\mathcal{A} + \Delta\mathcal{A})$ will consist of coefficients of the $(n-3) \times (n-3)$ minors of $\mathcal{A} + \Delta\mathcal{A}$ scaled by entries of λ. Accordingly, F will generally not have -2 as an eigenvalue.

Remark 5.4.5. *Thus far we have assumed that Lagrange multipliers exist at the current solutions of interest, which are attainable solutions that have full rank. Corollary 4.4.2 and the proof of Theorem 5.4.4 imply that Lagrange multipliers generally exist under these assumptions for several perturbation structures, since we need to solve*

$$\begin{pmatrix} 2\,\mathrm{vec}(\Delta\mathscr{A})^T & 0 \end{pmatrix} = -\lambda^T J,$$

of which J generally has (locally) constant or full rank. Of course if the solution was unattainable then the GCD constraints would break down as there is a "solution at infinity" in a sense that $\|h\| \to \infty$ as $\Delta\mathscr{A} \to \Delta\mathscr{A}^\star$.

The implication of the local error bound property holding is that one can reasonably approximate when quadratic convergence occurs by estimating $\sigma_{\min}\left(\left[H_{xx} \mid J^T\right]\right)$ and c_{hof}. In particular, these quantities act as a structured condition number on the system. A structured backwards-error analysis of existing techniques can be performed using these quantities. Additionally, it is somewhat generic that $F + 2I$ has full rank, hence the local error-bound will hold for most instances of the approximate SNF problem with an attainable solution. It is also important to note that we did not explicitly use the adjoint matrix. Indeed the result remains valid if we replace the adjoint with minors of prescribed dimension. Likewise, if $\mathscr{A}$ is an ill-posed instance of lower McCoy rank or approximate SNF without an attainable global minimum, then optimizing over a reversal of each entry of $\mathrm{Adj}(\mathscr{A} + \Delta\mathscr{A})$ would yield a non-trivial answer and the same stability properties would hold. Thus, poorly posed problems also remain poorly posed if slightly perturbed.

Corollary 5.4.6. *The LM algorithm for solving $\nabla L = 0$ has quadratic convergence under the assumptions of Theorem 5.4.2 and using $\nu_k = \|\nabla(L(z^k))\|_2$.*

Proof. The quantity ∇L is a multivariate polynomial, hence it is locally Lipschitz. Second-order sufficiency holds, thus we have the local error bound property is satisfied. The method converges rapidly with a suitable initial guess. $\square$

Note that for several perturbation structures if the adjoint has generic degrees, then the Jacobian of the constraints will have full rank, and a standard Newton iteration is also well-defined, and will converge quadratically as well.

In the next section we discuss a technique that possibly forgoes rapid local convergence, but has a polynomial per iteration cost to compute a low McCoy rank approximation.

5.4.4 Computational Challenges and Initial Guesses

The most glaring problem in deriving a fast iterative algorithm for the approximate Smith form problem is that the matrix $\operatorname{Adj}(\mathcal{A} + \Delta\mathcal{A})$ has exponentially many coefficients as a multivariate polynomial in $\Delta\mathcal{A}$. This means computing the adjoint matrix symbolically as an ansatz is not feasible. In order to solve (5.3) we instead approximate the derivatives of the coefficients of the adjoint numerically.

To compute an initial guess, we can use $\Delta\mathcal{A}_{init} = 0$ and take $\mathcal{F}^*$ and h to be a reasonable approximation to an approximate GCD of $\operatorname{Adj}(\mathcal{A})$, which will often be valid as per Theorem 5.4.2. To make sure the point is feasible, one can use a variant of Newton's method to project to a feasible point. Corollary 4.4.2 implies that with a suitable initial guess, reasonable variants of Newton's method (such as LM) will converge quadratically to a feasible point, assuming one exists.

Another technique is to take two rows or columns of $\mathcal{A}$ and perturb them so that the $2n$ entries have a non-trivial GCD. To find the best guess with this technique, $O(n^2)$ approximate GCD computations on $O(n)$ polynomials of degree d need to be performed. In the next section we will discuss more sophisticated techniques.

5.4.5 Attaining Unattainable Solutions

If a solution is unattainable then the degrees of all the entries of the adjoint matrix may change in an open neighborhood around a solution. If $\Delta\mathcal{A}^\star$ is an unattainable solution (of full rank) to (5.2) then $h(t) = t$ is clearly not a solution since $h(t) = t$ being a solution implies that such a solution would be attainable. Let d_{Adj} be the generic degree of $\operatorname{Adj}(\mathcal{A} + \Delta\mathcal{A})$. Then t is a factor of $\gcd(\operatorname{rev}_{d_{\mathrm{Adj}}}(\operatorname{Adj}(\mathcal{A} + \Delta\mathcal{A}^\star)))$. The reversed adjoint has no GCD at infinity by assumption, as such a GCD at infinity would be an attainable solution to the original problem. Accordingly, we note that Theorem 5.4.4 applies after some straightforward modifications, since

$$\nabla \operatorname{vec}(\operatorname{Adj}(\mathcal{A} + \Delta\mathcal{A})) \quad \text{and} \quad \nabla \operatorname{vec}(\operatorname{rev}_{d_{\mathrm{Adj}}}(\operatorname{Adj}(\mathcal{A} + \Delta\mathcal{A})))$$

are essentially (block) permutations of each other.

Since $\operatorname{rev}_{d_{\mathrm{Adj}}}(\operatorname{Adj}(\mathcal{A} + \Delta\mathcal{A})))$ achieves the generic degree, Lagrange multipliers should exist as we can apply Corollary 4.4.2 on $\nabla \operatorname{vec}(\operatorname{rev}_{d_{\mathrm{Adj}}}(\operatorname{Adj}(\mathcal{A}+\Delta\mathcal{A})))$ by permuting entries, and the underlying approximate GCD problem is well-posed. Thus the problem will also typically admit Lagrange multipliers.

The essential ingredient in Theorem 5.4.4 is the normalization of the underlying approximate GCD problem. This means that "backwards stable" algorithms will compute the exact SNF of a nearby matrix polynomial that has no meaning in the context of computation. This generally occurs because the radius of uncertainty, usually proportional to unit rounding errors, contains infinitely many matrix polynomials with a non-trivial SNF. The backwards stability is not meaningful in this context, because the instance of the problem is not continuous. In such instances, computing the SNF is most likely the wrong problem to be considering. Instead, computing the spectral structure of eigenvalues at infinity is most likely the appropriate problem. However there exist instances where both problems could be simultaneously poorly conditioned.

If the reversed problem has a radius of stability with respect to Theorem 5.4.4, then the original problem has a radius of instability, meaning that the iterates will converge to a point where $\|h\|$ is excessively large. In other words, if an instance of a problem is ill-posed, then it cannot be regularized — the finite and infinite eigenvalues and their spectral structure is indistinguishable in floating point arithmetic — in the context of the QZ decomposition, GUPTRI [26, 27] or similar algorithms. There are some instances where attempting to compute the SNF numerically is not possible and should not be attempted. In the context of an optimization problem, we can of course regularize the problem as we have just described. In fact, Van Dooren [103] suggests that ill-posed problems should be formulated as an optimization problem as a means of regularization to overcome some of the numerical difficulties.

5.5 Lower McCoy Rank Approximation

In this section we describe how to find a nearby matrix polynomial of lower McCoy. Another way to formulate $\mathcal{A}$ having a non-trivial SNF is to solve the minimization problem

$$\min \|\Delta\mathcal{A}\|_F^2 \;\text{ subject to }\; \begin{cases} (\mathcal{A}(\omega) + \Delta\mathcal{A}(\omega))\, B = 0, \\ B^*B = I_2, \\ \text{for some } \omega \in \mathbb{C} \text{ and } B \in \mathbb{C}^{n\times 2}, \end{cases} \tag{5.5}$$

where $\Delta\mathcal{A}$ must have the appropriate structure. Essentially this finds the smallest perturbation of $\mathcal{A}$ with an eigenvalue that lowers the rank by at least 2. The auxiliary variables ω and B are used to enforce this constraint. Here B^* is the conjugate transpose of B, and $B^*B = I_2$ ensures that the kernel vectors are linearly independent and do not tend towards zero.

The optimization is unstable if ω is reasonably large, since the largest terms appearing are proportional to $O((d+1)\|\mathscr{A}\|_\infty |\omega|^d)$. To remedy this, if we assume that a solution to the optimization problem (5.5) exists and has full rank, then we may transform $\mathscr{A}+\Delta\mathscr{A}$ into a degree-one matrix polynomial (also known as a matrix pencil) with the same spectral properties, known as a *linearization*. If there is no full-rank solution one can simply take a lower-rank approximation [37] and extract a square matrix polynomial of full rank that may be linearized. Alternatively, one may forgo the linearization and work directly with a problem that is more poorly conditioned. For the rest of this section we will assume, without loss of generality, that $\mathscr{A}$ and the solutions to the low McCoy rank problem have full rank.

We can encode the spectral structure and SNF of $\mathscr{A}$ as the following degree-one matrix polynomial (sometimes referred to as the *companion linearization* [44]) of the form $\mathscr{P} \in \mathbb{R}[t]^{nd\times nd}$, defined as

$$\mathscr{P} = \begin{pmatrix} I & & \\ & \ddots & \\ & & A_d \end{pmatrix} t - \begin{pmatrix} & I & & \\ & & \ddots & \\ -A_0 & -A_1 & \cdots & -A_{d-1} \end{pmatrix}.$$

This particular linearization encodes the SNF of $\mathscr{A}$, as $\mathrm{SNF}(\mathscr{P}) = \mathrm{diag}(I, I, \ldots, I, \mathrm{SNF}(\mathscr{A}))$. It follows that $\mathscr{A}$ has a non-trivial SNF if and only if $\mathscr{P}$ has a non-trivial SNF. If we preserve the affine structure of $\mathscr{P}$ and only perturb blocks corresponding to $\mathscr{A}$, then the reduction to a pencil will be sufficient. Other linearizations are possible as well. The pencil is generally better behaved numerically since the largest entry upon evaluation at a $\omega \in \mathbb{C}$ is proportional to $O(d\|\mathscr{A}\|_\infty |\omega|)$ rather than $O(\|A\|_\infty |\omega|^d)$, albeit with matrices that are d times larger.

5.5.1 Fast Low McCoy Rank via Optimization

One way to approach the lower McCoy rank approximation problem is to study all the minors (or sufficiently many) of a matrix polynomial. This method immediately generalizes from the previous section, however is not practical for computational purposes since the number of minors grows exponentially in the dimension. Instead, we can approach the problem by formulating it as an optimization problem, one that is remarkably similar to structured lower rank approximation of scalar matrices. This similarity facilitates computing an initial guess for the following optimization problem using the SVD.

The lower McCoy rank approximation problem may be formulated as the following *real optimization problem*: to find the nearest matrix polynomial to $\mathscr{A} \in \mathbb{R}[t]^{n\times n}$ with McCoy rank $n-r$, find the perturbation $\Delta\mathscr{A} \in \mathbb{R}[t]^{n\times n}$ which minimizes

$$\min \|\Delta\mathscr{A}\|_F^2 \text{ subject to } \begin{cases} \Re((\mathscr{P}+\widetilde{\Delta}\mathscr{P})(\omega)B) = 0, \\ \Im((\mathscr{P}+\widetilde{\Delta}\mathscr{P})(\omega)B) = 0, \\ \Re(B^*B) = I_r, \\ \Im(B^*B) = 0, \end{cases} \qquad \text{for some } \omega \in \mathbb{C} \text{ and } B \in \mathbb{C}^{nd\times r}. \tag{5.6}$$

Note that the perturbation $\Delta\mathscr{A}$ is *real valued* in this problem and $\widetilde{\Delta}\mathscr{P}$ are structured (affine) perturbations that preserve the linearization structure. The unitary constraint on B ensures that $\operatorname{rank}(B) = r$ and each column of B remains away from zero. Accordingly, $\omega \in \mathbb{C}$ will be an eigenvalue of $(\mathscr{P}+\widetilde{\Delta}\mathscr{P})(\omega)$ since $\operatorname{rank}((\mathscr{P}+\widetilde{\Delta}\mathscr{P})(\omega)) \leq nd - r$, and thus the McCoy rank of $\mathscr{A}+\Delta\mathscr{A}$ is at most $n-r$.

Real matrix polynomials can have complex eigenvalues and so complex numbers must necessarily appear in the constraints. The constraints arising from the complex numbers may be divided into real parts and imaginary parts, denoted as $\Re(\cdot)$ and $\Im(\cdot)$, respectively. By dividing the constraint into real and imaginary parts, we are able to solve an equivalent optimization problem completely with real variables. This ensures that $\Im(\Delta\mathscr{A}) = 0$, that is, the perturbations are real. Since $\mathscr{A}+\Delta\mathscr{A}$ may have complex eigenvalues (but entries with real coefficients), we require that $\operatorname{SNF}(\mathscr{A}+\Delta\mathscr{A})$ has entries from $\mathbb{R}[t]$. Accordingly, we need to interpret the auxiliary variable ω. The instance of $\Im(\omega) = 0$ corresponds to $t-\omega$ as an invariant factor, while $\Im(\omega) \neq 0$ corresponds to the real irreducible quadratic $(t-\omega)(t-\overline{\omega})$. Thus at a solution, we are able to recover a real invariant factor regardless if ω has a non-zero imaginary part.

In order to approach the problem using the method of Lagrange multipliers we define the Lagrangian as

$$L = \|\Delta\mathscr{A}\|_F^2 + \lambda^T \operatorname{vec}\begin{pmatrix} \Re(\mathscr{P}+\widetilde{\Delta}\mathscr{P})(\omega)B) \\ \Im(\mathscr{P}+\widetilde{\Delta}\mathscr{P})(\omega)B) \\ \Re(B^*B) - I_r \\ \Im(B^*B) \end{pmatrix},$$

and proceed to solve $\nabla L = 0$. In our implementation we again make use of the LM method, although given the relatively cheap gradient cost, a first-order method will often be sufficient and faster. The problem is essentially tri-linear, and structurally similar to affinely structured low rank approximation, of which Lagrange multipliers will exist for most instances.

It is important to note that an attainable solution to this problem is not guaranteed, as it is possible for $\|\omega\| \to \infty$ as $\Delta\mathscr{A} \to \Delta\mathscr{A}^\star$. Such an instance is an unattainable solution in the context of Section 5.4.5. These solutions behave like an infinite eigenvalue and can be handed by specifically considering the eigenvalue $t = 0$ of the reversed matrix polynomial.

5.5.2 Computing an Initial Guess

In order to compute an initial guess to (5.6) we exploit the pseudo tri-linearity of the problem. If two of $\Delta\mathscr{A}$, ω and B are fixed then the problem is linear (or a linear surrogate can be solved) in the other variable. Despite the unitary constraint on B being non-linear, it is not challenging to handle. Any full rank B is suitable for an initial guess, since we may orthonormalize B to satisfy the constraint that $B^*B = I_r$.

First we approximate the determinant of $\mathscr{A}$ and consider initial guesses where $\sigma_{n-r}(\mathscr{A}(\omega^{init}))$ is reasonably small. If $\sigma_{n-r}(\mathscr{A}(\omega^{init}))$ is reasonably small, then ω^{init} is (approximately) an eigenvalue of a nearby matrix polynomial of reduced McCoy rank. The zeros and local extrema of $\det(\mathscr{A})$ are suitable candidates for computing an initial guess for ω. The kernel B^{init} can be approximated from the smallest r singular vectors of $\mathscr{A}(\omega^{init})$. This ensures that B^{init} is unitary and spans the kernel of a nearby rank deficient (scalar) matrix.

To compute an initial guess for $\Delta\mathscr{A}$ we can take $\Delta\mathscr{A}^{init} = 0$, or solve a linear least squares problem where B and ω are fixed. Alternatively, one may project to a feasible point by using a variant of Newton's method, using $\Delta\mathscr{A}^{init} = 0$, ω^{init} and B^{init} as an initial guess for the Newton iteration to solve $(\mathscr{A} + \Delta\mathscr{A})(\omega)B = 0$ and $B^*B = I_r$. A feasible point computed by Newton's method tends not to perturb $\Delta\mathscr{A}$ very much, whereas the least squares approximation may perturb $\mathscr{A}$ by an unnecessarily large amount.

5.5.3 Convergence and Prescribed Spectral Structure

The linearization may converge to a solution where the invariant factors are reducible quadratics or degree larger than two. Accordingly, the rate of convergence will generally be linear with a first-order method and super-linear (but not always quadratic) with Newton-like methods. To obtain a prescribed spectral structure one simply adds constraints of the form (5.6) in conjunction with a "staircase form" constraint [104] to force invariant factors to be repeated or have higher degree. We defer discussion to Section 5.6.

5.5.4 About Global Optimization Methods

The problems previously discussed are NP hard to solve exactly and to approximate with coefficients from $\mathbb{Q}$. This follows since affinely structured low rank approximation [14, 91] is a special case. If we consider a *structured* matrix polynomial of degree zero, then this is a scalar matrix with an affine structure. The approximate SNF will be a matrix of rank at most $n-2$, and finding the nearest affinely structured singular matrix is NP hard.

Despite the problem being intractable in the worst case, not all instances are necessarily hard. The formulation (5.6) is multi-linear and polynomial, hence amenable to the sum of squares hierarchy. Lasserre's sum of squares hierarchy [76] is a global framework for polynomial optimization that asymptotically approximates a lower bound. Accordingly, if $\|\omega^{opt}\|$ is bounded, then sum of squares techniques should yield insight into the problem.

5.6 The Theory of Prescribed Spectral Structure

In several instances it may be practical to prescribe the degree structure, also called the *structural supports*, of the eigenvalues or the invariant factors of a nearby matrix polynomial. A lower McCoy rank approximation just ensures that $(t-\omega)$ is an eigenvalue and a divisor of an invariant factor. However in some instances we may want $(t-\omega)^2$ or $(t-\omega)(t-\overline{\omega})$ to divide an invariant factor. In general, it may be desired that $(t-\omega)^{\alpha_j}$ is a divisor of the invariant factor s_j for $\alpha_j \geq 0$ and $\alpha_j \leq \alpha_{j+1}$.

Example 5.6.1. *If $\mathcal{A} \in \mathbb{R}[t]^{2\times 2}$ with $\det(\mathcal{A}) = t^4$, then there are three possible SNF's for $\mathcal{A}$:*

$$\mathcal{S}_1 = \begin{pmatrix} 1 & \\ & t^4 \end{pmatrix}, \quad \mathcal{S}_2 = \begin{pmatrix} t^2 & \\ & t^2 \end{pmatrix} \quad \text{or} \quad \mathcal{S}_3 = \begin{pmatrix} t & \\ & t^3 \end{pmatrix}.$$

Both $\mathcal{S}_2$ and $\mathcal{S}_3$ imply that $\mathcal{A}$ has McCoy rank zero. However their degree structures, sometimes referred to as the structural supports, are different.

The tools of Section 5.4 can help answer this problem, as well as encode the structural support information as a constraint in the GCD of the entries. Of course computing exponentially many minors is not practical, but the theory generalizes immediately from Section 5.4 in a very straightforward manner.

The most general form of the problem is described as follows.

Problem 5.6.2 (Nearest Matrix Polynomial with Prescribed Structural Supports)**.** *Given $\mathcal{A} \in \mathbb{R}[t]^{n\times n}$ of degree at most d with a trivial SNF and $(\alpha_1, \ldots, \alpha_n) \in \mathbb{Z}^n_{\geq 0}$, compute a (local) solution (in terms of $\Delta\mathcal{A}$, h and $\{s_j\}_{j=1}^n$) to the optimization problem*

$$\inf \|\Delta\mathcal{A}\|_F^2 \quad \textit{subject to} \quad \begin{cases} \mathrm{SNF}(\mathcal{A}+\Delta\mathcal{A}) = \mathcal{S} = \mathrm{diag}(s_1, \ldots, s_n), \\ h^{\alpha_j} \textit{ is a factor of } s_j, \\ s_j \in \mathbb{R}[t] \textit{ are the invariant factors of } \mathcal{A}+\Delta\mathcal{A}, \\ h \in \mathbb{R}[t] \textit{ is monic.} \end{cases}$$

One needs to find the h and s_j that satisfy this problem, as they are not generally prescribed. In Section 5.7 we discuss a variation of the problem where h is prescribed.

In this problem formulation we implicitly require that $\det(\mathcal{A}+\Delta\mathcal{A}) \leq nd$. Thus if the structural supports or McCoy rank are over prescribed, then the zero matrix is the only feasible point. It is also necessary that $\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_n$ for $j = 1, \ldots, n$ to ensure that the divisibility property of the SNF is maintained. Accordingly, formulating an approximate GCD problem in the minors may be feasible for several instances and is generally more robust than the alternatives described later. Like all approximate GCD-type problems, solutions may be unattainable (irregular). However in this work we will only consider the special case where the solutions are regular (so the inf is assumed to be a min for the instances concerned here).

We can approach prescribing the entire spectral structure as an approximate GCD problem in terms of the minors of $\mathcal{A}+\Delta\mathcal{A}$ in a manner analogous to Section 5.4, with most of the results holding. Unfortunately, using exponentially many minors is only useful for a theoretical understanding and does not lead to an efficient implementation. The ideas in Section 5.5 in the presently described form cannot handle prescribing the spectral structure, despite being relatively faster for lower McCoy rank approximations. However, we are able to augment the technique in theory by specifying a "staircase" constraint on an appropriate matrix pencil.

Recall that Kronecker's Canonical Form [65, § 6.3] reveals the spectral structure, hence the Smith normal form. Instead of adding the constraint that the KCF is non-trivial (which would require one to "know" the entire KCF in advance), we can instead encode a constraint that implies the KCF is non-trivial (which is computationally more feasible). A staircase constraint is a sufficient condition to ensure this occurs.

To approach this problem numerically, we can also formulate an approximate GCD type optimization problem in terms of the unimodular multipliers and the SNF. Such a problem formulation is analogous to polynomial approximate GCD with some determinant constraints.

5.6.1 Stair Case Constraints

We now discuss how to incorporate staircase constraints into the optimization problems discussed in Section 5.5. The notions in this section are presented mainly for theoretical completeness, as they are generally not suitable for a robust implementation.

Lemma 5.6.3 ([103])**.** *Let $\mathscr{P} = Dt - C \in \mathbb{R}^{nd\times nd}$ be a matrix pencil with an eigenvalue $t = 0$. Then there exists unitary matrices $Q, Z \in \mathbb{C}^{nd\times nd}$ such that*

$$\begin{aligned} Q\mathscr{P}Z &= \begin{pmatrix} tD_{\ell+1,\ell+1} - C_{\ell+1,\ell+1} & 0 \\ * & tD_\alpha - C_\alpha \end{pmatrix} \qquad (5.7) \\ &= \begin{pmatrix} tD_{\ell+1,\ell+1} - C_{\ell+1,\ell+1} & & & & \\ tD_{\ell+1,\ell} - C_{\ell+1,\ell} & tD_{\ell,\ell} & & & \\ \vdots & \vdots & \ddots & & \\ tD_{\ell+1,2} - C_{\ell+1,2} & tD_{\ell,2} - C_{\ell,2} & \cdots & tD_{2,2} & \\ tD_{\ell+1,1} - C_{\ell+1,1} & tD_{\ell,1} - C_{\ell,1} & \cdots & tD_{2,1} - C_{2,1} & tD_{1,1} \end{pmatrix}, \end{aligned}$$

where

1. *$C_{\ell+1,\ell+1}$ has full rank,*
2. *$D_{j,j} \in \mathbb{C}^{r_j\times r_j}$ have full rank for $j = 1, \ldots, \ell$ and*
3. *$C_{j,j-1} \in \mathbb{C}^{r_j\times r_{j-1}}$ have full column rank for $j = 2, \ldots, \ell$.*

Furthermore, $r_j - r_{j+1} = \alpha_j \geq 0$ for $j = \ell, \ldots, 1$.

The assumption that $t = 0$ is an eigenvalue is without loss of generality, as one can perform the "deflation" $(t - \omega)D - (C - \omega D)$ which shifts the problem for an arbitrary eigenvalue $\omega \in \mathbb{C}$.

From this unitary decomposition, we have enough information to determine the blocks in the KCF that reveal the spectral structure of the eigenvalue $t = \omega$. In particular, this tells us the multiplicity of $t = \omega$ in each of the invariant factors of the SNF. The assumption that the $C_{j,j-1}$ and $D_{j,j}$ have full column rank is important, as this ensures that Jordan blocks in the KCF are non-trivially block-diagonal. Note that these blocks will locally have full rank under small perturbations, so the staircase constraint is only valid in a local context.

Note that $\mathscr{P}$ is equivalent [103] to the matrix pencil (in triangular block form)

$$T\mathscr{P}S = \left(\begin{array}{c|cccc} tD_{\ell+1,\ell+1} - C_{\ell+1,\ell+1} & 0 & 0 & \cdots & 0 \\ \hline * & tJ_\ell & & & \\ * & -K_{\ell-1} & \ddots & & \\ * & & \ddots & \ddots & \\ * & & & -K_1 & tJ_1 \end{array}\right), \tag{5.8}$$

where $T, S \in \mathbb{C}^{nd\times nd}$, $J_j = I_{r_j}$ and $K_j = \begin{pmatrix} I_{r_\ell+1} \\ 0 \end{pmatrix} \in \mathbb{R}^{(r_\ell+1)\times r_\ell}$. The transformation matrices T and S are not unitary, and this reduction can be poorly conditioned as it amounts to a Gaussian-like elimination *without* pivoting. We could add (5.8) as a constraint to the Lower McCoy rank approximation problem, but computing T and S for an initial guess could be ill-posed.

Two similar techniques in the literature that both require $O(n^4d^4)$ FLOPs to compute the form of (5.7) are those of Van Dooren [103] and Demmel and Kågström [26, 27]. There is a faster variation due to Beelen and Van Dooren [7] that requires $O(n^3d^3)$ FLOPs using a rank-revealing QR decomposition in lieu of the SVD for intermediate rank computations. All of these decompositions compute a two-sided unitary transformation.

We can also formulate an optimization problem to find a nearby matrix polynomial with McCoy rank at most $n - r$. Given $\mathscr{A} \in \mathbb{R}[t]^{n\times n}$ of degree at most d and a desired staircase of the form (5.7), compute a (local) solution to

$$\min \|\Delta\mathscr{A}\|_F^2 \text{ subject to } \begin{cases} (\mathscr{P} + \widetilde{\Delta}\mathscr{P})(\omega)B = 0, \\ B^*B = I_r, \qquad B \in \mathbb{C}^{nd\times r}, \qquad \omega \in \mathbb{C}, \\ Q(\mathscr{P} + \widetilde{\Delta}\mathscr{P})Z \text{ is in a staircase form (5.7) around } t = \omega, \\ Q^*Q = I, \qquad Z^*Z = I, \\ Q \in \mathbb{C}^{nd\times nd}, \qquad Z \in \mathbb{C}^{nd\times nd}. \end{cases} \tag{5.9}$$

Analogous to Section 5.5, we need to find B, ω, Q, Z such that the constraints are satisfied. As in Section 5.5, we are looking for real perturbations, that is $\Im(\Delta\mathscr{A}) = 0$.

We note that $\widetilde{\Delta}\mathscr{P} = \mathbf{\Delta}(\Delta\mathscr{A})$ is a structure-preserving perturbation that maintains the companion linearization analogous to Section 5.5. This problem is also similar to an affinely structured SLRA, and so we can expect Lagrange multipliers to exist for attainable solutions.

The optimization problem (5.9) leads to an algorithm to compute a nearby matrix pencil with $(t-\omega)_j^{\alpha}$ as a factor of the invariant factor s_j if the blocks in (5.7) are chosen according to Lemma 5.6.3. The first constraint, that $(\mathcal{P}+\widetilde{\Delta\mathcal{P}})(\omega)B = 0$ (which is separated into real and imaginary parts as in Section 5.5) implies that $t=\omega$ is an eigenvalue, so the staircase form is valid. Likewise, the constraints that Q^*Q and Z^*Z may be separated into real and imaginary parts, so that the entire problem is formulated as a real optimization problem. The constraint that $Q(\mathcal{P}+\widetilde{\Delta\mathcal{P}})Z$ is in a staircase form prescribes the structural supports of the eigenvalue $t=\omega$.

Alternatively, we could specify that the staircase constraint is of the form (5.8) and replace Q and Z with S and T. In this case the staircase form is guaranteed, but S and T could be poorly conditioned. In both formulations, it is possible that at the solution the blocks in the staircase are not of maximal size for the eigenvalue ω. It is also possible that other blocks of the same or larger size could exist for a different eigenvalue $\hat{\omega}$. For example, an instance where this occurs is when $\hat{\omega}=\bar{\omega}\neq\omega$, that is, ω is a solution with a non-trivial complex conjugate.

5.6.2 A Direct Approach with Unimodular Multipliers

A direct method to approach Problem 5.6.2 is to study another approximate GCD type problem that also optimizes over the unimodular multipliers. Given $\mathcal{A}\in\mathbb{R}[t]^{n\times n}$ of degree at most d and $(\alpha_1,\ldots,\alpha_n)$, we can formulate the following approximate GCD-like problem as

$$\min \|\Delta\mathcal{A}\|_F^2 \ \text{ subject to } \begin{cases} \mathcal{U}(\mathcal{A}+\Delta\mathcal{A})\mathcal{V}=\mathcal{S}, \\ \mathcal{S}=\operatorname{diag}(s_1', s_1's_2',\ldots,s_1's_2'\cdots s_n')\operatorname{diag}(h^{\alpha_1},\ldots,h^{\alpha_n}), \\ h\in\mathbb{R}[t] \text{ is monic}, \\ \det(\mathcal{U})=c_0\in\mathbb{R}\backslash\{0\}, \\ \det(\mathcal{V})=c_1\in\mathbb{R}\backslash\{0\} \end{cases} \tag{5.10}$$

where $\mathcal{U},\mathcal{V}\in\mathbb{R}[t]^{n\times n}$ have degrees at most nd and $s_j'\in\mathbb{R}[t]$.

The objective of this problem is to find $\{s_j'\}_{j=1}^n$, h, $\mathcal{U}$, $\mathcal{V}$. The constraint that

$$\mathcal{S}=\operatorname{diag}(s_1', s_1's_2',\ldots,s_1's_2'\cdots s_n')\operatorname{diag}(h^{\alpha_1},\ldots,h^{\alpha_n})$$

ensures that $\mathcal{A}+\Delta\mathcal{A}$ has a non-trivial SNF with a divisor of the invariant factors h with the prescribed degree structure. Note that h is not prescribed, but the degree structure of h is prescribed. The requirements that $\det(\mathcal{U})$ and $\det(\mathcal{V})$ have a non-zero determinant ensures that $\mathcal{U}$ and $\mathcal{V}$ are unimodular, so $\mathcal{S}$ is indeed the SNF of $\mathcal{A} + \Delta\mathcal{A}$. The quantities c_0 and c_1 can be taken to be any reasonable non-zero constant in practice, as the s'_j will absorb the re-scaling.

Generically, if we are working over $\mathbb{R}[t]$ then h will either be an irreducible degree one or degree two polynomial. The structural supports α_j are prescribed and constant with respect to the problem. There will likely be some redundancy, as generically, for $j = 1 \ldots, n-1$, we have $\deg(s'_j) = 0$. In the most general form, $\deg(s'_j) \leq nd$. There are no normalization assumptions on $\mathcal{S}$ other than that a divisibility property holds and that there is an upper bound on the degrees of the entries. To extract the SNF we can make the entries of $\mathcal{S}$ monic after the computation. We could also instead prescribe that $\Delta\mathcal{A}$ has some affine coefficient structure $\widetilde{\Delta\mathcal{A}}$ with the analysis being similar for various instances we care about (such as perturbations that preserve the support of each entry or do not increase the degrees of each entry).

Regularity and Existence of Lagrange Multipliers

The Jacobian of the constraints of (5.10) will generally have constant rank (after accounting for the redundancy of constraints and over-padding with zero coefficients of some polynomials) around a solution, as it behaves both like a determinant constrained problem and a multi-linear constrained approximate GCD problem. We do not investigate a second-order stability theory explicitly, because Theorem 5.4.4 can be generalized using more minors to reveal the same type of information. Accordingly, solutions in $\Delta\mathcal{A}$ are isolated when the residual of the problem is sufficiently small and quadratic convergence for Newton-like methods is expected when the problem is normalized in a reasonable manner.

Lemma 5.6.4. *Lagrange multipliers exist for the constraints* $\det(\mathcal{U}) = c_0$ *and* $\det(\mathcal{V}) = c_1$ *in* (5.10).

Proof. Suppose without loss of generality that that $\deg(\mathcal{U}) \leq nd$ and $\deg(\mathcal{V}) \leq nd$. By Theorem 4.3.1 we can write $\nabla \operatorname{vec}(\det(\mathcal{U})) = \Phi_{nd}(\operatorname{pvec}(\operatorname{Adj}(\mathcal{U})^T)^T)$.

The matrix $\Phi_{nd}(\operatorname{pvec}(\operatorname{Adj}(\mathcal{U})^T)^T)$ will generally have linearly independent non-zero rows, so long as one of the $(n-1) \times (n-1)$ minors of $\mathcal{U}$ achieves the generic degree. Unimodular matrices always have a full rank trailing coefficient matrix, since

$$\det(\mathcal{U}) = \det(\mathcal{U}|_{t=0}) = \det(U_0) \in \mathbb{R}\backslash\{0\}.$$

Accordingly, the block could be rank deficient by over-estimating the degree of $\mathcal{U}$. If we knew the minimal degree of $\mathcal{U}$, then Lagrange multipliers exist if the degrees are minimal, since $\nabla \operatorname{vec}(\det(\mathcal{U}_{\min}))$ would have full rank. By setting some Lagrange multipliers to zero, the Lagrange multipliers to the reduced problem can be obtained as a subset of the Lagrange multipliers of the unreduced problem.

Lagrange multipliers existing for the other derivative corresponding to $\mathcal{V}$ has the same reasoning by symmetry. Accordingly, the problem can be normalized so that the linear independent or constant rank constraint qualifications holds. Lagrange multipliers must exist for attainable solutions. $\square$

For reasonable affine perturbation structures, we can generally expect Lagrange multipliers to exist. Likewise, if we restrict $\mathcal{U}$ or $\mathcal{V}$ in some way (such as forcing one of them to be triangular), then Lagrange multipliers will also generally exist so long as the problem is not over constrained.

Remark 5.6.5. *We can write the Jacobian of the constraints (up to permutation) in a block form (using Theorem 4.3.1 to compute the derivatives of the determinant constraints) as*

$$J = \left(\begin{array}{ccccc} \partial \Delta\mathcal{A} & \partial \mathcal{U} & \partial \mathcal{V} & \partial \mathcal{S} & \partial h \\ \hline *_{\Delta\mathcal{A}} & *_{\mathcal{U}} & *_{\mathcal{V}} & \mathcal{C}_{\mathcal{S}} & \mathcal{C}_h \\ 0 & \Phi_{nd}(\operatorname{pvec}(\operatorname{Adj}(\mathcal{U})^T)^T) & 0 & 0 & 0 \\ 0 & 0 & \Phi_{nd}(\operatorname{pvec}(\operatorname{Adj}(\mathcal{V})^T)^T) & 0 & 0 \end{array}\right).$$

Here the precise expressions for $*_{\Delta\mathcal{A}}, *_{\mathcal{U}}$, and $*_{\mathcal{V}}$ are mostly irrelevant (they are permuted block-convolution matrices or permuted block convolution matrices with several zero columns removed). For suitable values of $\mu_{\mathcal{U}}, \mu_{\mathcal{A}}$ and $\mu_{\mathcal{V}}$ we can write,

$$\frac{\partial \operatorname{vec}(\mathcal{U}(\mathcal{A}+\Delta\mathcal{A})\mathcal{V})}{\partial \operatorname{vec}(\mathcal{U})} = \Phi_{\mu_{\mathcal{U}}}(((\mathcal{A}+\Delta\mathcal{A})\mathcal{V})^T \otimes I)$$

and

$$\frac{\partial \operatorname{vec}(\mathcal{U}(\mathcal{A}+\Delta\mathcal{A})\mathcal{V})}{\partial \operatorname{vec}(\mathcal{V})} = \Phi_{\mu_{\mathcal{V}}}(I \otimes \mathcal{U}(\mathcal{A}+\Delta\mathcal{A})).$$

Both of these matrices have full column rank if $\mathcal{A}+\Delta\mathcal{A}$ has full rank, which will hold locally around a solution. Next, we can write

$$\frac{\partial \operatorname{vec}(\mathcal{U}(\mathcal{A}+\Delta\mathcal{A})\mathcal{V})}{\partial \operatorname{vec}(\Delta\mathcal{A})} = \Phi_{\mu_{\mathcal{A}}}(\mathcal{V}^T \otimes \mathcal{U}),$$

which has full rank since $\mathcal{U}$ and $\mathcal{V}$ are unimodular. If $\Delta\mathcal{A}$ had some prescribed affine perturbation structure $\mathbf{\Delta}(\cdot)$, then the derivative would be $\Phi_{\mu_{\mathcal{A}}}(\mathcal{V}^T \otimes \mathcal{U})\dfrac{\partial \operatorname{vec}(\widetilde{\Delta\mathcal{A}})}{\partial \operatorname{vec}(\Delta\mathcal{A})}$. If the perturbations preserve the zero structure, then the derivative has full rank (the derivative is a sub-matrix of $\Phi_{\mu_{\mathcal{A}}}(\mathcal{V}^T \otimes \mathcal{U})$ with some columns deleted).

The blocks $\mathcal{C}_{\mathcal{S}}$ and $\mathcal{C}_h$ are closely related to the GCD constraint from Section 5.4. Accordingly, if h is normalized properly (analogous to Theorem 5.4.4, i.e. degree of h is known, h is monic, h^{α_j} is of maximal degree, the degrees of the entries of $\mathcal{S}$ are known and symmetry is removed), then this block will have locally constant rank. Since the approximate GCD constraint is feasible (the solution is attainable), the matrix $\nabla \operatorname{vec}(\mathcal{U}(\mathcal{A}+\Delta\mathcal{A})\mathcal{V}-\mathcal{S})$ can typically be normalized to have constant rank, thus Lagrange multipliers generally exist.

Of course unattainable (irregular) solutions are possible when some of the minors of $\mathcal{A}+\Delta\mathcal{A}$ have an unattainable non-trivial SNF. In such a scenario, $\|s_j'\| \to \infty$ or $\|h\| \to \infty$ as $\Delta\mathcal{A} \to \Delta\mathcal{A}^\star$, where $\Delta\mathcal{A}^\star$ is a minimal perturbation. The optimization problem can be regularized by dividing the constraint further into a regular part and an irregular part. The irregular part corresponds to an eigenvalue at infinity, a case dealt with in the next section.

Computing an Initial Guess

Unless the minimal perturbation $\|\Delta\mathcal{A}^\star\|$ is sufficiently small, computing an initial guess is a difficult problem. In some special instances, one can use an exact algorithm to compute the SNF of $\mathcal{A}$ with the multipliers and use this as an initial guess. Generally in a floating point environment, computing unimodular multipliers is a difficult problem since the exact algorithms relying on Gaussian elimination or related variants are not numerically stable. If the residual is small, then we can infer the SNF of a nearby matrix polynomial and use linear-least squares to approximate the unimodular multipliers.

Lemma 5.6.6. *Suppose* $\operatorname{SNF}(\mathcal{A}) = \mathcal{S}$ *and* $\mathcal{U}\mathcal{A}\mathcal{V} = \mathcal{S}$. *Then we can write* $\mathcal{U}\mathcal{A} = \mathcal{S}\widetilde{\mathcal{V}}$ *with* $\widetilde{\mathcal{V}}$ *unimodular. One may then compute a feasible* $\mathcal{U}$ *and* $\mathcal{V}$ *by solving the linear least squares problem*

$$\min_{\mathcal{U},\widetilde{\mathcal{V}}} \|\mathcal{U}\mathcal{A} - \mathcal{S}\widetilde{\mathcal{V}}\|_F \quad \textit{subject to} \quad \widetilde{\mathcal{V}} = \begin{pmatrix} 1 & * & \cdots & * \\ & 1 & \ddots & \vdots \\ & & \ddots & * \\ & & & 1 \end{pmatrix}. \tag{5.11}$$

Proof. We note that $\mathrm{Adj}(\widetilde{\mathcal{V}}) = \mathcal{V}$, so the other multiplier may be recovered in a straightforward manner. A solution must always exist because there always exists a unimodular $\mathcal{U}$ such that $\mathcal{U}\mathcal{A} = \mathcal{H}$ where $\mathcal{H}$ is in a triangular form that can be reduced into the SNF via column operations. We can perform column operations on a triangular matrix with the same SNF as $\mathcal{A}$, resulting in a triangular post-multiplier matrix that reduces $\mathcal{H}$ into $\mathcal{S}$, and the existence follows. □

The normalization that $\widetilde{\mathcal{V}}$ is upper-triangular with constant diagonal enforces that $\mathcal{U} \neq 0$, $\mathcal{V} \neq 0$ and that $\mathcal{U}$ is unimodular. Of course other normalizations are possible, as the unimodular multipliers are not unique. Setting the diagonal entries to be 1 is theoretically valid (as any re-scaling is absorbed by $\mathcal{U}$), but may not be wise from a numerical perspective.

Remark 5.6.7. *The residual to* (5.11) *is necessarily (approximately)* 0 *if* $\mathrm{SNF}(\mathcal{A}) = \mathcal{S}$. *To observe this, we can write*

$$\mathcal{U}\mathcal{A}\mathcal{V} = \mathcal{S} \iff \mathcal{U}\mathcal{A} = \mathcal{S}\mathcal{V}^{-1}.$$

Since $\mathcal{V}$ *is assumed to be unimodular, the inverse is also a matrix polynomial. If the residual is non-zero but sufficiently small, then* $\mathcal{U}$ *will be sufficiently close to a unimodular matrix polynomial by continuity.*

Lemma 5.6.6 provides a suitable means for computing an initial guess (and computing a pair of unimodular multipliers, that are not necessarily of minimal degree). If the residual is relatively large, then the guess can be projected to a feasible point using a variant of Newton's method. We recall that $\mathcal{S}$ can be approximated to high precision using GUPTRI [26, 27] or two sided unitary transformations [7, 104].

5.7 Prescribed Smith Normal Form

Our focus is now brought to the instance when divisors of the SNF are prescribed (that is, divisors of the invariant factors are polynomials with prescribed coefficients). Fixing every invariant factor of an entire SNF is typically ill-posed because the invariant factor s_n will generally be of the form $s_n = p(t)s_\varepsilon$ where $p(t) \in \mathbb{R}[t]$ is divided by some of the other invariant factors and $s_\varepsilon \in \mathbb{R}[t]$ is relatively prime with the other invariant factors. Specifying that $s_\varepsilon = 1$ (either explicitly or implicitly) can lead to problems where the only feasible points are the zero matrix or some other rank deficient matrix polynomial.

Problem 5.7.1. *Given $\mathcal{A} \in \mathbb{R}[t]^{n\times n}$ of degree at most d with a trivial SNF, compute a (local) solution to the optimization problem*

$$\min \|\Delta\mathcal{A}\|_F^2 \quad \textit{subject to} \quad \begin{cases} \mathrm{SNF}(\mathcal{A}+\Delta\mathcal{A}) = \mathcal{S}'\mathcal{H}, \\ \mathcal{H} = \mathrm{diag}(h_1, h_1h_2, \ldots, h_1h_2\cdots h_n), \quad h_j \in \mathbb{R}[t] \quad \textit{are prescribed,} \\ \mathcal{S}' = \mathrm{diag}(s_1', s_1's_2', \ldots, s_1's_2'\cdots s_n'), \quad s_j' \in \mathbb{R}[t]. \end{cases}$$

The use of "min" here is intentional, as solutions to this problem always exist. Generically, at a solution to this problem $\Delta\mathcal{A}^\star$, we will have that $\mathrm{SNF}(\mathcal{A}+\Delta\mathcal{A}^\star) = \mathcal{H}\,\mathrm{diag}(1,\ldots,1,s_\varepsilon)$. This problem can be handled as a special case of the previous section. However it is different enough to merit a section on its own.

Theorem 5.7.2. *A solution to Problem 5.7.1 exists under linear perturbation structures.*

The result also holds under affine perturbation structures where feasible points also exist. Note that Problem 5.7.1 assumes that $\mathrm{SNF}(\mathcal{A})$ is trivial, but this is not strictly required.

Proof. Note first that the feasible set is non-empty and bounded since $\Delta\mathcal{A} = -\mathcal{A}$ is a feasible point. It is also implicit that we can take $\|\Delta\mathcal{A}\| \leq \|\mathcal{A}\|$.

Note also that the set of all polynomials that do not have h_j as a factor is topologically open. A generalized Sylvester matrix formed from a non-zero set of polynomials that do not have h_j as a factor and h_j does not decrease in rank when perturbed by a small non-zero amount, analogous to Theorem 5.3.2.

This implies that the set of matrix polynomials of degree at most d that do not have a prescribed divisor h_j of the entries in the SNF is also topologically open, since the invariant factors are polynomials and can be defined as a GCD problem with respect to sufficiently many $j\times j$ minors of $\mathcal{A}$ for $j = 1\ldots,n$. Therefore, the set of all matrix polynomials of degree at most d that have prescribed divisors of invariant factors is topologically closed. We note that the minors are continuous functions in the coefficients of the entries of $\mathcal{A}$, and are thus bounded in size if perturbations to $\mathcal{A}$ are bounded (one can apply a variant of Hadamard's inequality to show this for example, the actual bound is irrelevant to the proof).

By Weierstrass' theorem, we can optimize a continuous function over the intersection of a closed, non-empty and bounded set and a non-empty closed set, hence an attainable global minimum exists. □

This does not contradict the earlier problems, because in the underlying approximate GCD problem, the GCD is a priori that is prescribed and the co-factors and invariant factors are variable. The problem is structurally similar to the approximate GCD type problem over $\mathbb{R}$ or $\mathbb{C}$ where h is prescribed along with f and g. The minimization problem for $f, g, h \in \mathbb{R}[t]$ (or $\mathbb{C}[t]$) is

$$\min_{f^*, g^*} \|f - f^* h\|_2^2 + \|g - g^* h\|_2^2,$$

which has an attainable global minimum that is easy to compute by applying linear least squares.

The computational techniques of the previous section can be modified in a straightforward manner to handle this case, thus we omit a technical discussion.

5.7.1 Prescribed Infinite Spectral Structure

Recall that the infinite eigenvalues of $\mathcal{A} \in \mathbb{R}[t]^{n \times n}$ of degree at most d are characterized by the spectral structure of $t = 0$ of $\mathrm{rev}_d(\mathcal{A})$. Accordingly, to prescribe the infinite spectral structure of $\mathcal{A} + \Delta\mathcal{A}$ it is sufficient to prescribe the spectral structure of the eigenvalue $t = 0$ of $\mathrm{rev}_d(\mathcal{A} + \Delta\mathcal{A})$. The previously discussed techniques can be applied nearly verbatim to accomplish this. Since the eigenvalue at infinity is always "known", the matrix polynomial with the nearest non-trivial infinite spectral structure always exists. Accordingly, a nearest matrix polynomial with an interesting SNF always exists if eigenvalues at infinity are permitted, which is consistent with Theorem 5.3.2.

5.8 Implementation and Examples

We have implemented our algorithms and techniques in the Maple computer algebra system. All computations are done using hardware precision and measured in floating point operations, or FLOPs. The input size of our problem is measured in the dimension and degree of $\mathcal{A}$, which are n and d respectively. The cost of most Newton methods is roughly proportional to inverting the Hessian matrix, which is $O(\ell^3)$, where ℓ is the number of variables in the problem.

5.8.1 Nearest Interesting SNF and Lower McCoy Rank Approximation

We use the variant of Levenberg-Marquardt discussed in Section 5.4 in several instances to solve the first-order necessary condition. The method of Section 5.4 requires approximately $O((n^3d)^3) = O(n^9d^3)$ FLOPs per iteration in an asymptotically optimal implementation with cubic matrix inversion, which is the cost of inverting the Hessian. Computing the Hessian costs roughly $\widetilde{O}(n^4d^2 \times (n^2)^2) = \widetilde{O}(n^8d^2)$ FLOPs using a blocking procedure, assuming the adjoint computation runs in $\widetilde{O}(n^4d)$ FLOPs (which can be done via interpolation in a straightforward manner). There are $O(n^3d)$ Lagrange multipliers since the adjoint has degree at most $(n-1)d$. Using reverse-mode automatic differentiation to compute $\nabla^2 L$, this can be accomplished in $\widetilde{O}(n^4d \times n^3d) = \widetilde{O}(n^7d^2)$ FLOPs.

The method of Section 5.5 has a Hessian matrix of size $O(n^2d^2) \times O(n^2d^2)$ in the case of a rank zero McCoy rank approximation. Accordingly, the per iteration cost is roughly $O(n^6d^6)$ FLOPs. If the linearization is not performed, then the per-iteration cost is $O(n^6d^3)$ FLOPs. Given the lack of expensive adjoint computation, a first-order method will typically require several orders of magnitude fewer FLOPs per iteration (ignoring the initial setup cost), with local linear convergence.

Example 5.8.1 (Nearest Interesting SNF)**.** *Consider the matrix polynomial $\mathcal{A}$ with a trivial SNF*

$$\begin{pmatrix} t^2+.1t+1 & 0 & .3t-.1 & 0 \\ 0 & .9t^2+.2t+1.3 & 0 & .1 \\ .2t & 0 & t^2+1.32+.03t^3 & 0 \\ 0 & .1t^2+1.2 & 0 & .89t^2+.89 \end{pmatrix}$$

of the form $\operatorname{diag}(1,\ldots,1,\det(\mathcal{A}))$*.*

If we prescribe the perturbations to leave zero coefficients unchanged, then using the methods of Section 5.4 and Section 5.5 results in a local minimizer $\mathcal{A}+\Delta\mathcal{A}_{opt}$ given by

$$\begin{pmatrix} 1.0619t^2+.018349t+.94098 & 0 & .27477t-.077901 & 0 \\ 0 & .90268t^2+.22581t+1.2955 & 0 & .058333 \\ .13670t & 0 & .027758t^3+.97840t^2+1.3422 & 0 \\ 0 & .10285t^2+1.1977 & 0 & .84057t^2+.93694 \end{pmatrix},$$

with $\|\Delta\mathcal{A}_{opt}\|_F \approx .164813183138322$. The SNF of $\mathcal{A}+\Delta\mathcal{A}_{opt}$ is approximately

$$\operatorname{diag}(1,1,s_1,s_1(t^5+35.388t^4+6.4540t^3+99.542t^2+5.6777t+70.015)),$$

where $s_1 \approx t^2 + 0.0632934647739423t + 0.960572576466186$*. The factor* s_1 *corresponds to* $\omega_{opt} \approx -0.0316467323869714 - 0.979576980535687i$*.*

The method discussed in Section 5.4 converges to approximately 14 *decimal points of accuracy*[2] *after* 69 *iterations and the method of Section 5.5 converges to the same precision after approximately* 34 *iterations. The initial guess used in both instances was* $\Delta\mathcal{A}_{init} = 0$*. The initial guesses of* $\mathcal{F}^*$ *and* h *were computed by an approximate GCD routine. For the initial guess of* ω *we chose a root or local extrema of* $\det(\mathcal{A})$ *that minimized the second-smallest singular value of* $\mathcal{A}(\omega)$*, one of which is* $\omega_{init} \approx -.12793 - 1.0223i$*.*

Example 5.8.2 (Lowest McCoy Rank Approximation)**.** *Let* $\mathcal{A}$ *be as in the previous example and consider the* 0*-McCoy rank approximation problem with the same prescribed perturbation structure.*

In this case we compute a local minimizer $\mathcal{A} + \Delta\mathcal{A}_{opt}$ *given by*

$$\begin{pmatrix} .80863t^2 + 1.1362 & 0 & 0 & 0 \\ 0 & .91673t^2 + 1.2881 & 0 & 0 \\ 0 & 0 & .95980t^2 + 1.3486 & 0 \\ 0 & .60052t^2 + .84378 & 0 & .71968t^2 + 1.0112 \end{pmatrix},$$

with $\|\Delta\mathcal{A}_{opt}\|_F \approx .824645447014665$ *after* 34 *iterations to* 14 *decimal points of accuracy. We compute* $\omega_{opt} \approx -1.18536618732372i$ *which corresponds to the single invariant factor* $s_1 \approx t^2 + 1.4051$*. The SNF of* $\mathcal{A} + \Delta\mathcal{A}_{opt}$ *is of the form* (s_1, s_1, s_1, s_1)*.*

5.8.2 Prescribed Structural Supports

Now we consider some examples whereby we compute a nearby matrix polynomial with a prescribed spectral structure, including the structural supports. The algorithm implemented assumes that the structural supports are the generic instance where the SNF is of the form

$$\mathcal{S} = \begin{pmatrix} s_1 h^{\alpha_1} & & \\ & \ddots & \\ & & s_n h^{\alpha_n} \end{pmatrix}$$

$$\text{where } \{\deg(s_j)\}_{j=0}^{n-1} \subseteq \{0, -\infty\} \text{ and } \deg(s_n) = nd - \deg(h)\sum_{j=1}^{n} \alpha_j.$$

[2] $\nabla L = 0$ is solved to 14 digits of accuracy; the extracted quantities are accurate to approximately the same amount.

As this assumption is with loss of generality, we can make the appropriate modifications that are particular to the instance at hand. Recall that the issue with the non-generic instances is that $\{s_j\}_{j=0}^{n-1}$ may have some symmetry or other redundancy if not handled properly, possibly preventing rapid local convergence. If these variables are handled properly, then rapid local convergence will occur. In our experiments, we use hardware precision and attempt to solve $\nabla L = 0$ to at-least 10 digits of precision[3]. Local super linear convergence to precision occurs in all examples.

To demonstrate the convergence properties we take an exact instance with an exact non-trivial SNF and then perturb the coefficients by a prescribed amount of noise. We then attempt to compute a nearby matrix polynomial whose SNF has prescribed structural supports. In the examples we make the assumption that the degree of each entry of $\mathcal{U}$ and $\mathcal{V}$ does not increase from the initial guess[4], and is not necessarily the generic bounds of nd. If a reasonable technique (such as automatic differentiation) is used to compute the derivatives of the determinant of $\mathcal{U}$ and $\mathcal{V}$ then Newton methods will have a polynomial per-iteration cost.

[3]Ten digits of precision is the standard Digits of accuracy in Maple for software floats. In practice, the computed solutions are accurate between 12 to 14 decimal points.

[4]This reduces the dimension of the problem considerably. The auxiliary variables to enforce the spectral structure constraints can typically be several orders of magnitude larger than the input size of the problem.

```
Algorithm 5 : Nearest Matrix Polynomial with Prescribed Spectral Structure
Input:
  • Matrix polynomial $\mathcal{A} \in \mathbb{R}^{n\times n}$.
  • Initial guesses $\Delta\mathcal{A}^{init}$, $\mathcal{U}^{init}$, $\mathcal{V}^{init}$, $h^{init}$ and $\mathcal{S}^{init}$.
  • (Optional) Displacement structure matrix $\widetilde{\Delta}\mathcal{A}$ to optimize over.
  • Vector of structural supports $\alpha$ and degree bound for $h$.
Output:
  • $\mathcal{A} + \widetilde{\Delta}\mathcal{A}$ with a non-trivial SNF prescribed by $h$ and $\alpha$ or an indication of failure.
  • Unimodular multipliers $\mathcal{U}, \mathcal{V}$ and a diagonal matrix $\mathcal{S}$ such that
      $$\mathcal{U}(\mathcal{A} + \widetilde{\Delta}\mathcal{A})\mathcal{V} \approx \mathcal{S},$$
    where $\mathcal{S}$ is a constant re-scaling of SNF($\mathcal{A} + \widetilde{\Delta}\mathcal{A}$).
1: Define $x = x(\Delta\mathcal{A}, \mathcal{U}, \mathcal{V}, h, \mathcal{S})$ to be a combined vector of variables.
2: Define Lagrangian function $L$ as
      $$L(x,\lambda) = \|\Delta\mathcal{A}\|_F^2 + \lambda^T \operatorname{vec}(c(x)) \quad \text{where} \quad c(x) \quad \text{is defined from} \quad (5.10).$$
3: Initialize $\lambda$ via linear least squares from $\nabla L(\Delta x^{init}) = 0$.
4: Compute
      $$\begin{pmatrix} x+\Delta x \\ \lambda + \Delta\lambda \end{pmatrix} \quad \text{by solving} \quad \nabla^2 L \begin{pmatrix} \Delta x \\ \Delta\lambda \end{pmatrix} = -\nabla L(x,\lambda) \quad \text{until}$$
   $\left\| \begin{pmatrix} \Delta x \\ \Delta\lambda \end{pmatrix} \right\|_2$ or $\|\nabla L(x,\lambda)\|_2$ is sufficiently small or divergence is detected.
5: Return the locally optimal $\Delta\mathcal{A}$ or an indication of failure.
```

The method to solve the KKT equations we use is a combination of plain Newton's method via truncated SVD or a variant of Levenberg-Marquardt as described in Section 5.4.3. Solving the KKT equations is sufficient in practice by Lemma 5.6.4. The per-iteration cost mostly depends on the degrees of the unimodular multipliers. If we assume the most generic instance where $\deg(\mathcal{U}) = \deg(\mathcal{V}) = nd$ then $\deg(\det(\mathcal{U})) = \deg(\det(\mathcal{V})) = O(n^2 d)$. Accordingly, there will be $O(n^2 d)$ constraints and $O(n^2 d) + O(nd) + O(n^3 d) = O(n^3 d)$ variables. The resulting Hessian matrix will have dimension $O(n^3 d) \times O(n^3 d)$ and the per-iteration cost will be roughly $O(n^9 d^3)$ FLOPs assuming that derivatives of $\det(\mathcal{U})$ and $\det(\mathcal{V})$ are computed sufficiently fast. If we assume that $\det(\cdot)$ requires $\widetilde{O}(n^4 d)$ FLOPs to compute the determinant of a $n \times n$ matrix polynomial of degree d, then computing the derivatives of $\det(\mathcal{U})$ and $\det(\mathcal{V})$ should require $\widetilde{O}(n^5 d \times n^3 d \times n^3 d) = \widetilde{O}(n^{11} d^3)$ FLOPs. Using reverse-mode automatic differentiation on the Lagrangian[5], this cost can be reduced to $\widetilde{O}(n^5 d \times n^3 d) = \widetilde{O}(n^8 d^2)$ FLOPs to compute $\nabla^2 L$, which is sufficiently fast. In some special instances, Theorem 4.3.1 and Theorem 4.4.1 can be used as well.

Example 5.8.3. *Consider the matrix polynomial* $\mathcal{A} = \sum_{j=1}^{4} t^j A_j$ *where*

$$A_0 = \begin{pmatrix} 11.00005819 & 12.00084512 & -1.399997946 \\ -1.499479482 & -7.999005049 & -7.699516738 \\ -2.999119009 & 4.000860204 & 7.000347109 \end{pmatrix},$$

$$A_1 = \begin{pmatrix} -22.99958387 & -19.99925876 & 8.400477800 \\ -10.99927786 & 3.500374705 & 9.100640203 \\ 12.00087464 & -0.9993746387 & -6.999133373 \end{pmatrix},$$

$$A_2 = \begin{pmatrix} 17.00052443 & 8.000906180 & -12.59919322 \\ 21.00020391 & 15.50006404 & 3.500285016 \\ -8.999824450 & -7.999667605 & -5.599217955 \end{pmatrix},$$

$$A_3 = \begin{pmatrix} -8.999008650 & -3.999657217 & 5.600342707 \\ -4.999925327 & -7.499143553 & -4.899086933 \\ -3.999653167 & 1.000675922 & 5.600679017 \end{pmatrix},$$

$$A_4 = \begin{pmatrix} 4.000080910 & 4.000127572 & 0.0 \\ -3.499011335 & -3.499217500 & 0.0 \\ 4.000732625 & 4.000376565 & 0.0 \end{pmatrix}.$$

If we consider perturbations to $\mathcal{A}$ *that do not perturb zero coefficients then for our initial guess we use a feasible* $\Delta\mathcal{A}^{init}$ *with an absolute distance of* $\|\Delta\mathcal{A}^{init}\|_F \approx 0.004065107497$. *This is a relative error of approximately* 0.00006748779330. *We extract an initial guess where* $\|L(x^{init}, \lambda^{init})\|_2 \approx 8.130214 \times 10^{-3}$. *The prescribed structural supports are* $\alpha = (1, 2, 2)$ *for* $\deg(h) = 1$.

Five iterations of Newton's method[6] *yields a stationary point satisfying second-order*

[5]Recall that $L : \mathbb{R}^{O(n^3 d)} \to \mathbb{R}$ is a scalar function, where a single function call costs $\widetilde{O}(n^5 d)$ FLOPs, since it is dominated by determinant operations.

[6]The implementation uses a truncated SVD to solve the linear system. The computed solution is verified to be a solution to the KKT equations by checking a regularized linear system as well.

necessary conditions. The computed $\Delta\mathcal{A}$ *satisfies* $\|\Delta\mathcal{A}\|_F \approx 0.00374266893916940$ *with*

$$\Delta A_0 \approx \begin{pmatrix} -0.0003844000 & -0.0007351014 & -0.0003249840 \\ -0.0004893421 & -0.0007075497 & -0.0005221761 \\ -0.0005834209 & -0.0006799116 & -0.0006610957 \end{pmatrix},$$

$$\Delta A_1 \approx \begin{pmatrix} -0.0003844000 & -0.0007351014 & -0.0003249840 \\ -0.0004893421 & -0.0007075497 & -0.0005221761 \\ -0.0005834209 & -0.0006799116 & -0.0006610957 \end{pmatrix},$$

$$\Delta A_2 \approx \begin{pmatrix} -0.0004029894 & -0.0006319195 & -0.0003966100 \\ -0.0004392784 & -0.0005271485 & -0.0008297230 \\ -0.0005550320 & -0.0004622514 & -0.0008856620 \end{pmatrix},$$

$$\Delta A_3 \approx \begin{pmatrix} -0.0005315500 & -0.0003651453 & -0.0005821823 \\ -0.0002555722 & -0.0005416553 & -0.0006660183 \\ -0.0002162420 & -0.0007004980 & -0.0006262513 \end{pmatrix},$$

$$\Delta A_4 \approx \begin{pmatrix} -0.0004079370 & -0.0003551865 & 0.0 \\ -0.0006757246 & -0.0005565483 & 0.0 \\ -0.0008509183 & -0.0003954382 & 0.0 \end{pmatrix}.$$

The Smith form is a re-scaling of

$\mathrm{diag}(h, h^2, h^2(0.998439140885453t + 0.998660090266538))$ *where* $h \approx t - 1.00000113481449$.

Note that the infinite eigenvalues were not meaningfully perturbed.

Example 5.8.4. *Consider the matrix polynomial* $\mathcal{A} = \sum_{j=1}^{4} t^j A_j$ *where*

$$A_0 = \begin{pmatrix} 13.00581869 & 6.008090944 & 1.434278386 \\ -1.452220013 & -7.979609270 & -7.693595603 \\ -2.935979688 & 4.087464486 & 7.086020397 \end{pmatrix},$$

$$A_1 = \begin{pmatrix} 23.04161255 & 15.08451185 & 2.812757318 \\ -10.41932202 & -25.49253255 & -12.51435541 \\ -4.971498360 & 15.01755487 & 14.06253613 \end{pmatrix},$$

$$A_2 = \begin{pmatrix} 7.052443015 & 12.07412422 & 1.400205449 \\ -14.46572923 & -24.90113358 & -2.021749851 \\ -0.9086934350 & 18.03468319 & 7.033239490 \end{pmatrix},$$

$$A_3 = \begin{pmatrix} -2.900864980 & 3.090617872 & 0.0 \\ -3.447948247 & -5.400504917 & 2.848326232 \\ 1.088099075 & 7.073262447 & 0.0 \end{pmatrix},$$

$$A_4 = \begin{pmatrix} 0.0 & 0.0 & 0.0 \\ 2.072213842 & 2.037470532 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{pmatrix}.$$

If we consider perturbations to $\mathcal{A}$ *that do not perturb zero coefficients then for our initial guess we use a feasible* $\Delta\mathcal{A}^{init}$ *with an absolute distance of* $\|\Delta\mathcal{A}^{init}\|_F \approx 0.3739284133$*. This is a relative error of approximately* 0.005862300872*. We extract an initial guess where* $\|L(x^{init}, \lambda^{init})\|_2 \approx 7.478568 \times 10^{-1}$*. The prescribed structural supports are* $\alpha = (2, 2, 2)$ *for* $\deg(h) = 1$*.*

Six iterations of Newton's method yields a stationary point satisfying second-order nec-

essary conditions. The computed $\Delta\mathcal{A}$ *satisfies* $\|\Delta\mathcal{A}\|_F \approx 0.195106246768620$ *with*

$$\Delta A_0 \approx \begin{pmatrix} 0.003672710 & 0.004785086 & -0.02245739 \\ -0.005187912 & -0.03621277 & 0.007193448 \\ -0.06403311 & -0.08485719 & -0.04527277 \end{pmatrix},$$

$$\Delta A_1 \approx \begin{pmatrix} -0.009980157 & -0.01509224 & 0.01007845 \\ -0.002422992 & 0.008174672 & -0.01519851 \\ 0.02881208 & 0.01774652 & 0.01493851 \end{pmatrix},$$

$$\Delta A_2 \approx \begin{pmatrix} -0.008520615 & 0.02778100 & 0.01080944 \\ 0.002975467 & 0.006109075 & 0.01583740 \\ 0.02104305 & 0.02636789 & 0.003488096 \end{pmatrix},$$

$$\Delta A_3 \approx \begin{pmatrix} -0.07735493 & -0.04525647 & 0.0 \\ -0.04655637 & -0.001737837 & -0.01108606 \\ -0.03307563 & -0.04490523 & 0.0 \end{pmatrix},$$

$$\Delta A_4 \approx \begin{pmatrix} 0.0 & 0.0 & 0.0 \\ -0.06830108 & -0.01322433 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{pmatrix}.$$

The Smith form is

$$\mathrm{diag}(h^2, h^2, h^2) \quad \textit{where} \quad h \approx t + 1.00028560646861.$$

Note that the infinite eigenvalues were not meaningfully perturbed.

Example 5.8.5. *Consider the matrix polynomial* $\mathcal{A} = \sum_{j=1}^{4} t^j A_j$ *where*

$$A_0 = \begin{pmatrix} 4.005818693 & 6.084511854 & 0.3335387824 \\ 3.052051754 & -0.9005049160 & 2.381659565 \\ 4.088099075 & 4.073262446 & 4.067592248 \end{pmatrix}$$

$$A_1 = \begin{pmatrix} 2.124945887 & 2.157457552 & 0.4644466538 \\ 7.155547175 & 1.120803866 & 3.480686979 \\ 8.087464486 & 8.086020397 & 4.037656444 \end{pmatrix},$$

$$A_2 = \begin{pmatrix} 3.219109680 & 3.257284539 & 0.4973446486 \\ 9.187057395 & 3.173071064 & 3.445168307 \\ 8.017554870 & 8.062536135 & 4.034710873 \end{pmatrix},$$

$$A_3 = \begin{pmatrix} -0.7341983133 & -2.799054947 & 0.1176041064 \\ 6.174134120 & 4.252311257 & 1.174639898 \\ 4.034683193 & 4.033239487 & 0.0 \end{pmatrix},$$

$$A_4 = \begin{pmatrix} 1.091424277 & 1.096090651 & 0.0 \\ 2.182199749 & 2.161583482 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{pmatrix}.$$

If we consider perturbations to $\mathcal{A}$ *that do not perturb zero coefficients then for our initial guess we use a feasible* $\Delta\mathcal{A}^{init}$ *with an absolute distance of* $\|\Delta\mathcal{A}^{init}\|_F \approx 0.3834240788$*. This is a relative error of approximately* 0.01439260580*. We extract an initial guess where* $\|L(x^{init}, \lambda^{init})\|_2 \approx 7.668482 \times 10^{-1}$*. The prescribed structural supports are* $\alpha = (1, 1, 1)$ *for* $\deg(h) = 2$*.*

To compute a solution we use a hybrid Newton method where Newton's method is used for the first two iterations and a regularized Newton method based on Levenberg-Marquardt is used for eight iterations to compute a local minimizer[7]*. The local minimizer* $\Delta\mathcal{A}$ *satisfies*

[7] The computed point satisfies second-order sufficient conditions, however the Jacobian of the constraints and projected Hessian are both rank deficient.

$\|\Delta\mathscr{A}\|_F \approx 0.133615685603406$ *and is given by*

$$\Delta A_0 \approx \begin{pmatrix} -0.006991058 & 0.003723321 & 0.02165736 \\ 0.01237565 & -0.03888976 & -0.04065646 \\ -0.01008063 & 0.005678707 & -0.02034014 \end{pmatrix},$$

$$\Delta A_1 \approx \begin{pmatrix} 0.005138626 & -0.004803751 & 0.006353215 \\ -0.01570427 & -0.003329195 & -0.01394313 \\ 0.003091133 & 0.01993365 & -0.006293378 \end{pmatrix},$$

$$\Delta A_2 \approx \begin{pmatrix} 0.001515364 & 0.001382172 & -0.02789106 \\ 0.004169284 & 0.04187736 & 0.05550295 \\ 0.003999821 & -0.02276316 & 0.02018658 \end{pmatrix},$$

$$\Delta A_3 \approx \begin{pmatrix} -0.01775122 & 0.01464300 & -0.001669402 \\ 0.02062815 & -0.04831373 & -0.01978488 \\ -0.006211543 & 0.008028807 & 0.0 \end{pmatrix},$$

$$\Delta A_4 \approx \begin{pmatrix} 0.004236129 & 0.01890377 & 0.0 \\ -0.04154365 & 0.001907364 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{pmatrix}.$$

The Smith form is a re-scaling of

$$\operatorname{diag}(h, h, hs_n) \quad \textit{where} \quad h \approx t^2 + 1.00362940720781t + 1.00553122631092 \quad \textit{and}$$

$$s_n \approx 0.000058187t^9 - 0.000070334t^8 - 0.000033959t^7 + 0.00016269t^6 \\ - 0.00014110t^5 - 0.000081232t^4 - 0.0010299t^3 - 0.0054643t^2 + 1.0420t - 7.8367.$$

We note that unlike the previous two examples, the Smith form of the computed solution perturbed eigenvalues at infinity to be non-zero. Making s_n monic is not necessarily a wise normalization, as the trailing coefficient of s_n will become quite large. In particular, computing the SNF with the associated multipliers may be an ill-conditioned problem numerically, but inferring the SNF via unitary transformations can still be reasonably behaved.

5.9 Conclusion

The optimization problem of computing a nearby matrix polynomial with a non-trivial spectral structure was shown to be amenable to local optimization techniques. Three broad families of techniques were proposed based on;

1. Computing the minors of a matrix polynomial that while not practical, yields theoretical insights into the problem in a straightforward manner.

2. A technique that resembles a weighted instance of affine or linear structured lower rank approximations of matrix polynomials that is efficient in computing a lower McCoy rank approximation with respect to the per-iteration cost. A modification with staircase constraints is proposed to handle the most general form of the problem.

3. A direct approach yielding the unimodular multipliers and Smith form at the same time. The most generic version is discussed, although the results are applicable to other instances with some minor modifications.

While all of the methods solve the same problem, their challenges and theoretical considerations regarding the KKT conditions are different.

In general, the more structure that is demanded, the harder the problem becomes to solve. Finding a nearby matrix polynomial with an interesting Smith form has a larger region of convergence than finding a nearby matrix polynomial of reduced McCoy rank, which in turn has a larger region of convergence than finding a nearby matrix polynomial where the entire spectral structure is prescribed. In particular, our view of the problems are almost entirely local. While a global view of the problem may be desirable, this is perhaps too much to request. We recall that the eigenvalues are not always continuous, and any continuity tends to be local. Irregular solutions exist and solving the KKT conditions may not make sense without any information about the solution. To ensure existence of Lagrange multipliers, one must know in advance how eigenvalues at infinity at the solution behave, which is a local property and not a global one.

Regularity conditions were shown to hold for most instances of the problems in question, ensuring that Lagrange multipliers exist. When Lagrange multipliers do not exist, alternative formulations that satisfy Lagrange multipliers are proposed and shown to be theoretically robust with a suitable initial guess. In general, Newton methods will have rapid local convergence under normalization assumptions for all the problems considered.

Chapter 6

Approximate Greatest Common (Right) Divisors of Differential Operators

In this chapter we revisit some classic problems in symbolic-numeric computation with the goal of demonstrating that the techniques proposed earlier are applicable to other problems. We will study the approximate Greatest Common Right Divisor (GCRD) of differential polynomials over $\mathbb{R}[t][\partial;']$ from the perspective of computing a nearby matrix polynomial with a linear structure.

The approximate GCRD problems is a non-commutative generalization of approximate GCD. Approximate GCRD is structurally similar to the approximate GCD of two or more polynomials, but the ground ring of computation is $\mathbb{R}[t]$ instead of $\mathbb{R}$ or $\mathbb{C}$. This means that instead of computing the nearest rank deficient Sylvester matrix, we need to compute the nearest rank deficient *differential Sylvester matrix*, which is a Sylvester-like matrix with coefficients over $\mathbb{R}[t]$. In other words, this is an application of computing the nearest singular matrix polynomial with a linear structure [38, 51]. This chapter documents some improvements to the theory of the approximate GCRD of differential operators and some analogs to approximate GCD type problems over similar domains of computation.

6.1 Introduction

The problem of computing the Greatest Common Right Divisor (GCRD) in a symbolic and exact setting dates back to [90], who presents a Euclidean-like algorithm. See [15] for an elaboration of this approach. [79] introduces a differential-resultant-based algorithm which makes computation of the GCRD very efficient using modular arithmetic. The technique of [79] is an extension of ideas presented by [48] for computing GCRDs of differential operators.

The analogous approximate GCD problem for usual (commutative) polynomials has been a key topic of research in symbolic-numeric computing since its inception. A full survey is not possible here, but we note the deep connection between our current work and that of [21]; see also [72], [95], and [112]. Also important to this current work is the use of so-called structured (numerical) matrix methods for approximate GCD, such as structured total least squares (STLS) and structured total least norm (STLN); see [13] and [68]. More directly employed later in this chapter is the multiple polynomial approximate GCD method of [69]. This latter paper also provides a nice survey of the current state of the art in approximate GCDs. Finally, we modify the proof of [70], an optimization approach to computing the GCD of multiple, multivariate *commutative* polynomials, to prove the existence of a globally nearest GCRD.

The goal of this chapter is to devise an efficient, numerically robust algorithm to compute the GCRD when the coefficients in $\mathbb{R}$ are given approximately. Given $f, g \in \mathbb{R}(t)[\partial;']$, we wish to find $\widetilde{f}, \widetilde{g} \in \mathbb{R}(t)[\partial;']$, where $\widetilde{f}$ is *near* f and $\widetilde{g}$ is *near* g, such that $\deg_\partial(\operatorname{gcrd}(\widetilde{f}, \widetilde{g})) \geq 1$, where *near* is taken with respect to a distributed Euclidean norm. That is, $\widetilde{f}$ and $\widetilde{g}$ have an exact, non-trivial GCRD.

Linear differential polynomials and GCRD's are key tools in finding closed form symbolic solution of systems of linear differential equations in modern computer algebra systems like Maple and Mathematica (see, e.g., [94] and [2]). Equations with real (floating point) coefficients or parameters are regularly encountered and it is important to understand the stability of this fundamental tool in this case. Moreover, floating arithmetic is potentially much faster than managing large rational coefficients. We regard this work [36] as a positive and important initial exploration of this topic.

6.1.1 Outline

The main theoretical improvements to the theory involve studying the approximate GCRD problem as a constrained optimization problem, opposed to an unconstrained optimization problem as was done in the past [38, 51]. This chapter establishes:

1. A rigorous application of the theory of Chapter 3 to the problem to improve several degree bounds.
2. Characterization of the existence and local uniqueness of regular and irregular solutions.
3. Regularization for special cases with irregular solutions via nearest singular matrix polynomial.
4. A brief summary of how the constrained optimization approach improves upon unconstrained optimization techniques with an example, whereby a hybrid technique is used.

We commence with necessary preliminaries and well-known results that we expand upon in the remainder of this introductory section. In Section 6.3 we describe a linear algebra formulation of the approximate GCRD problem and that can be used in conjunction with truncated SVD [21, 35, 51] to compute nearby polynomials with an exact GCRD. Section 6.4 reformulates the approximate GCRD problem as a continuous unconstrained optimization problem. Sufficient conditions for existence of a solution are provided with an example showing that when this sufficient condition is not satisfied there is no solution. These results are complemented by showing that the Jacobian of the residuals has full rank and under ideal circumstances Newton iteration will converge quadratically. We generalize some results of [111] and [112] to a non-commutative Euclidean domain showing that the problem is locally well-posed. In Section 5.8 we present our algorithms explicitly, discuss their complexity and evaluate the numerical robustness of our implementation on examples of interest.

A part of this work, presenting the SVD-based approach to approximate GCRD, but without the proof of existence of a nearest solution or analysis of the corresponding optimization, is presented in the workshop paper [35]. Section 6.5 studies the optimization problem in the context of structured matrices further drawing upon the ideas of Chapter 3.

6.2 Preliminaries

We review some well known results [90] and [16] on differential polynomials.

The ring of differential (Ore) polynomials $\mathbb{R}(t)[\partial;']$ over the real numbers $\mathbb{R}$ provides a (non-commutative) polynomial ring structure to the linear ordinary differential operators. Differential polynomials have found great utility in symbolic computation, as they allow us to apply algebraic tools to the simplification and solution of linear differential equations; see [15] for a nice introduction to the mathematical and computational aspects.

Let $\mathbb{R}(t)[\partial;']$ be the ring of differential polynomials over the function field $\mathbb{R}(t)$. $\mathbb{R}(t)[\partial;']$ is the ring of polynomials in ∂ with coefficients from the commutative field of rational functions, under the usual polynomial addition along with the non-commutative multiplication defined by

$$\partial y(t) = y(t)\partial + y'(t) \text{ for } y(t) \in \mathbb{R}(t).$$

Here $y'(t)$ is the usual derivative of $y(t)$ with respect to t.

There is a natural action of $\mathbb{R}(t)[\partial;']$ on the space $C^\infty[\mathbb{R}]$ of infinitely differentiable functions $y(t) : \mathbb{R} \to \mathbb{R}$. In particular, for any $y(t) \in C^\infty[\mathbb{R}]$,

$$f(\partial) = \sum_{0\le i\le M} f_i(t)\partial^i \text{ acts on } y(t) \text{ as } \sum_{0\le i\le M} f_i(t)\frac{d^i}{dt^i}y(t).$$

We maintain a right canonical form for all $f \in \mathbb{R}(t)[\partial;']$ by writing

$$f = \frac{1}{f_{-1}} \sum_{0\le i\le M} f_i\partial^i, \tag{6.1}$$

for polynomials $f_{-1}, f_0, \ldots, f_M \in \mathbb{R}[t]$. That is, with coefficients in $\mathbb{R}(t)$ always written to the left of powers of ∂. An analogous left canonical form exists as well.

A primary benefit of viewing differential operators in this way is that they have the structure of a left (and right) Euclidean domain. In particular, for any two polynomials $f, g \in \mathbb{R}(t)[\partial;']$, there is a unique polynomial $h \in \mathbb{R}(t)[\partial;']$ of maximal degree in ∂ such that $f = f^*h$ and $g = g^*h$ for $f^*, g^* \in \mathbb{R}(t)[\partial;']$ (i.e., h divides f and g exactly on the right). This polynomial h is called the Greatest Common Right Divisor (GCRD) of f and g and it is unique up to multiplication by a unit (non-zero element) of $\mathbb{R}(t)$ (we could make this GCRD have leading coefficient 1, but this would introduce denominators from $\mathbb{R}[t]$, as well as potential numerical instability, as we shall see).

An important geometric interpretation of GCRDs is that the GCRD h of differential polynomials f and g is a differential polynomial whose solution space is the intersection of the solution spaces of f and g.

Approximations require a norm, so we need a proper definition of the *norm* of a differential polynomial.

Definition 6.2.1. *We define a* distributed coefficient norm *for differential polynomials as follows: for $f = \sum_{0 \le i \le M} f_i \partial^i \in \mathbb{R}[t][\partial;']$, define*

$$\|f\| = \|f\|_2 = \left(\sum_{0 \le i \le M} \|f_i\|_2^2 \right)^{1/2}.$$

We could extend the above definition of norm over $\mathbb{R}(t)$ and $\mathbb{R}(t)[\partial;']$. However it turns out that this is unnecessary and somewhat complicating. In practice, we perform most of our computations over $\mathbb{R}[t]$. In the cases where we are unable to avoid working over $\mathbb{R}(t)$, we simply solve an associate problem. This is done by clearing denominators and performing intermediate computations over $\mathbb{R}[t]$, then converting back to the representation over $\mathbb{R}(t)$. Note that the algebraic problem is always computing GCRDs and co-factors in $\mathbb{R}(t)[\partial;']$, and not the more intricate algebraic domain $\mathbb{R}[t][\partial;']$; see the discussion below.

Problem 6.2.2 (Approximate GCRD)**.** *Given $f, g \in \mathbb{R}[t][\partial;']$ such that* $\mathrm{gcrd}(f, g) = 1$ *we wish to compute $\widetilde{f}, \widetilde{g} \in \mathbb{R}[t][\partial;']$ with the same coefficient degree structure*[1] *as f and g such that $h = \mathrm{gcrd}(\widetilde{f}, \widetilde{g})$ with $D = \deg_\partial(h) \ge 1$ and*

1. *$\|f - \widetilde{f}\|_2^2 + \|g - \widetilde{g}\|_2^2 = \varepsilon$ is minimized, and*
2. *D is the largest possible for the computed distance ε.*

The differential polynomial h is said to be an approximate GCRD *of f and g if these conditions are satisfied. In general it is not easy to minimize ε, so instead we take a local optimization approach and compute an upper bound on this quantity. These upper-bounds will agree with the global minimum if ε is sufficiently small. The algorithmic considerations will generally assume D is fixed without loss of generality, since we can vary D from 1 to* $\min\{M, N\}$ *to determine the (local) optimal value.*

The approximate GCRD problem is a generalization of computing an ε-*GCD* [21, 31, 72, 96] in the commutative case. The requirement that the GCRD has maximal degree is difficult to certify outside the exact setting, however this usually is not a problem in our experiments. We prove that our formulation of the approximate GCRD problem has a solution with a minimal ε (opposed to an infimum). Furthermore, if D is fixed, then for a computed pair of nearby differential polynomials, we are able to certify that ε is reasonably close to the optimal value through a condition number.

[1]The polynomial coefficients of ∂^i have the same degree, i.e. $\deg(\widetilde{f_i}) \le \deg(f_i)$ and $\deg(\widetilde{g_i}) \le \deg(g_i)$.

In our approach to the approximate GCRD problem we devise methods of performing division and computing an exact GCRD numerically. These tools are used in conjunction with our algorithm for computing a nearby pair of differential polynomials with an exact GCRD via the SVD. The nearby differential polynomials with an exact GCRD are used as an initial guess in a post-refinement Newton iteration.

It will also be necessary to define a partial ordering on differential polynomials. In later sections we will need to make use of this partial ordering to preserve structure.

Definition 6.2.3. *Let* $\operatorname{dvec} : \mathbb{R}[t][\partial;'] \to \mathbb{Z}^{M+1}$ *be the degree vector function defined as*

$$\operatorname{dvec}(f) = (\deg_t(f_0), \deg_t(f_1), \ldots, \deg_t(f_M)), \text{ for } f_0, \ldots, f_M \in \mathbb{R}[t].$$

For $f, g \in \mathbb{R}[t][\partial;']$ *with* $\deg_\partial(f) = \deg_\partial(g) = M$ *we write*

$$\operatorname{dvec}(f) < \operatorname{dvec}(g) \text{ if } \deg_t f_i \leq \deg_t g_i \text{ for } 0 \leq i \leq M.$$

We define $\operatorname{dvec}(f) = \operatorname{dvec}(g), \operatorname{dvec}(f) < \operatorname{dvec}(g), \operatorname{dvec}(f) \geq \operatorname{dvec}(g)$ *and* $\operatorname{dvec}(f) > \operatorname{dvec}(g)$ *analogously.*

We note that differential polynomials are written in a canonical ordering with highest degree coefficients appearing to the left in our examples. The degree vector function and most linearizations will appear in reverse order as a result. For convenience, we will assume that $\deg(0) = -\infty$. Note that in this instance our vectorizations of differential operators are defined as the transpose of what one would expect from earlier chapters. This is because we are performing linear algebra on the left, opposed to the right, that is, we are solving equations that resemble $\boldsymbol{x}^T\mathcal{A} = 0$.

Definition 6.2.4. *Let* $f \in \mathbb{R}[t][\partial;']$ *where* $\deg_\partial(f) = M$ *is in standard form. The* content *of* f *is given by* $\operatorname{cont}(f) = \gcd(f_0, f_1, \ldots, f_M)$. *If* $\operatorname{cont}(f) = 1$, *we say that the differential polynomial is* primitive.

Proposition 6.2.5. *The ring* $\mathbb{R}(t)[\partial;']$ *is a non-commutative principal left (and right) ideal domain. For* $f, g \in \mathbb{R}(t)[\partial;']$, *with* $\deg_\partial(f) = M$ *and* $\deg_\partial(g) = N$, *we have the following properties* [90].

(i) $\deg_\partial(fg) = \deg_\partial(f) + \deg_\partial(g)$, $\deg_\partial(f+g) \leq \max\{\deg_\partial(f), \deg_\partial(g)\}$.

(ii) There exist unique $q, r \in \mathbb{R}(t)[\partial;']$ *with* $\deg_\partial(r) < \deg_\partial(g)$ *such that* $f = qg + r$ *(right division with remainder).*

(iii) *There exists $h \in \mathbb{R}(t)[\partial,']$ of maximal degree in ∂ with $f = f^*h$ and $g = g^*h$. h is called the GCRD (Greatest Common Right Divisor) of f and g, written* $\mathrm{gcrd}(f, g) = h$. *f^* and g^* are called the left co-factors of f and g. The GCRD is unique up to multiplication from a unit belonging to $\mathbb{R}(t)$.*

(iv) *There exist $\sigma, \tau \in \mathbb{R}(t)[\partial;']$ such that $\sigma f = \tau g = \ell$ for ℓ of minimal degree. ℓ is called the LCLM (Least Common Left Multiple) of f and g, written* $\mathrm{lclm}(f, g) = \ell$. *The LCLM is unique up to multiplication from a unit belonging to $\mathbb{R}(t)$.*

(v) $\deg_\partial(\mathrm{lclm}(f, g)) = \deg_\partial(f) + \deg_\partial(g) - \deg_\partial(\mathrm{gcrd}(f, g))$.

In an algebraic context we can clear denominators of our inputs and assume without loss of generality that our GCRD belongs to $\mathbb{R}[t][\partial;']$. We will also assume our inputs and output are primitive. Again, this is not algebraically necessary but will be important for the convergence of our subsequent optimization formulation (see Section 6.4.2). It is important to note that the co-factors of the GCRD need not belong to $\mathbb{R}[t][\partial;']$ even if we have $f, g, h \in \mathbb{R}[t][\partial;']$ such that $\mathrm{gcrd}(f, g) = h$. This is not unexpected, as a similar situation occurs when computing GCD's over $\mathbb{Z}[x]$, where co-factors in the GCD of primitive polynomials may well lie in $\mathbb{Q}[x] \setminus \mathbb{Z}[x]$. In essence, this is a computational technique to narrow the input domain, not a change to the problem being considered.

A related but considerably more difficult problem is computing ideal bases and factorizations completely within $\mathbb{R}[t][\partial;']$. This has been dealt with algebraically and in terms of exact computation by a number of authors, though not with respect to approximate coefficients; see for example [9, 41, 53].

Most of our results involve transforming a representation of $f \in \mathbb{R}(t)[\partial;']$ into a representation over $\mathbb{R}(t)^{1\times K}$ for $K \geq \deg_\partial(f)$. We make extensive use of the following map.

Definition 6.2.6. *For $f \in \mathbb{R}(t)[\partial;']$ of degree M in ∂ as in (6.1), and $K > M$, we define*

$$\Psi_K(f) = \frac{1}{f_{-1}}(f_0, f_1, \ldots, f_M, 0, \ldots, 0) \in \mathbb{R}(t)^{1\times K}.$$

That is, Ψ_K maps polynomials in $\mathbb{R}(t)[\partial;']$ of degree (in ∂) less than K into $\mathbb{R}(t)^{1\times K}$.

It will be useful to linearize (differential) polynomials, that is, express them as an element of Euclidean space. For $p \in \mathbb{R}[t]$ with $\deg_t(p) = d$ we write

$$\mathrm{vec}(p) = (p_0, p_1, \ldots, p_d) \in \mathbb{R}^{1\times(d+1)}$$

For $f = f_0 + f_1\partial + \cdots f_M\partial^M \in \mathbb{R}[t][\partial;']$ with $\deg_\partial(f) = M$ and $\deg_t(f_i) = d_i$ we write

$$\mathrm{vec}(f) = (\mathrm{vec}(f_0), \ldots, \mathrm{vec}(f_M)) \in \mathbb{R}^{1\times L},$$

where $L = (d_0 + 1) + \cdots + (d_M + 1)$. If $d \geq \max\{d_i\}$ we will sometimes pad each $\mathrm{vec}(f_i)$ with zeros to have precisely $d + 1$ coefficients, and by a slight abuse of notation regard

$$\mathrm{vec}(f) \in \mathbb{R}^{1\times(M+1)(d+1)}.$$

We will not do this unless specifically stated.

6.3 Computing the GCRD via Linear Algebra

In this section we demonstrate how to reduce the computation of the GCRD to that of linear algebra over $\mathbb{R}(t)$, and then over $\mathbb{R}$ itself. This approach has been used in the exact computation of GCRDs [79] and differential Hermite forms [42], and has the benefit of reducing differential, and more general Ore problems, to a system of equations over a commutative field. Here we will show that it makes our approximate version of the GCRD problem amenable to numerical techniques. We note that for computing approximate GCRDs of differential polynomials, much as for computing approximate GCDs of standard commutative polynomials, the Euclidean algorithm is numerically unstable, and thus we employ resultant-based techniques, as described below.

Since $\mathbb{R}(t)[\partial;']$ is a right (and left) Euclidean domain [90], a GCRD may be computed by solving a Diophantine equation corresponding to the Bézout coefficients. Using the sub resultant techniques of [78], we are able to transform the non-commutative problem over $\mathbb{R}(t)[\partial;']$ into a commutative linear algebra problem over $\mathbb{R}(t)$. This is done through a Sylvester-like resultant matrix. By using resultant-like matrices we are able to express the Bézout coefficients as a linear system over $\mathbb{R}(t)$ and compute a GCRD via nullspace basis computation.

Lemma 6.3.1. *Suppose $f, g \in \mathbb{R}(t)[\partial;']$ with $\deg_\partial(f) = M$ and $\deg_\partial(g) = N$. Then $\deg_\partial(\mathrm{gcrd}(f, g)) \geq 1$ if and only if there exist $u, v \in \mathbb{R}(t)[\partial;']$ such that $\deg_\partial(u) < N$, $\deg_\partial(v) < M$, and $uf + vg = 0$.*

Proof. This follows immediately from Proposition 6.2.5. □

Using Lemma 6.3.1 we can solve a Bézout-like system to compute a GCRD of two differential polynomials. This is characterized by the differential Sylvester matrix, based on the sub resultant method of [79].

Definition 6.3.2. *Suppose* $h \in \mathbb{R}[t][\partial;']$ *has* $\deg_\partial(h) = D$. *For any* $K \in \mathbb{N}$, *the matrix*

$$\mathcal{C}_K^{\mathbf{R}}(h) = \begin{pmatrix} \Psi_{K+D+1}(h) \\ \Psi_{K+D+1}(\partial h) \\ \vdots \\ \Psi_{K+D+1}(\partial^K h) \end{pmatrix} \in \mathbb{R}[t]^{(K+1)\times(K+D+1)}$$

is the K^{th} right differential convolution matrix *of* h. *We note that the entries of* $\mathcal{C}_K^{\mathbf{R}}(h)$ *are written in their* right canonical form, *where the* ∂*'s appear to the right (polynomials in* $\mathbb{R}[t]$ *appear to the left). We note that* $\deg_t(\partial^i h) = \deg_t(h)$, *so the degree in* t *of all entries of* $\mathcal{C}_K^{\mathbf{R}}(h)$ *is at most* $\deg_t(h)$.

We analogously define the K^{th} left differential convolution matrix *of* h *as* $\mathcal{C}_K^{\mathbf{L}}(h)$ *as*

$$\mathcal{C}_K^{\mathbf{L}}(h) = \begin{pmatrix} \Psi_{K+D+1}(h) \\ \Psi_{K+D+1}(h\partial) \\ \vdots \\ \Psi_{K+D+1}(h\partial^K) \end{pmatrix} \in \mathbb{R}[t]^{(K+1)\times(K+D+1)},$$

where elements are written in their left canonical form, *where the* ∂*'s appear to the left (polynomials in* $\mathbb{R}[t]$ *always appear to the right).*

Both right and left differential convolution matrices can be used to perform multiplication. Suppose $f^* \in \mathbb{R}(t)[\partial;']$, $h \in \mathbb{R}(t)[\partial;']$ and $f = f^*h \in \mathbb{R}[t][\partial;']$, with

$$f = \sum_{0\le i\le M} f_i\partial^i,\ f^* = \sum_{0\le i\le M-D} f_i^*\partial^i \text{ and } h = \sum_{0\le i\le D} h_i\partial^i, \tag{6.2}$$

with $f_i, h_i \in \mathbb{R}[t]$ and $f_i^* \in \mathbb{R}(t)$. We can express the product of f^* and h as

$$(f_0, f_1, \ldots, f_M) = (f_0^*, \ldots, f_{M-D}^*)\mathcal{C}_{M-D}^{\mathbf{R}}(h).$$

Similarly, we may write

$$(f_0, f_1, \ldots, f_M)^T = \mathcal{C}_D^{\mathbf{L}}(f^*)(h_0, h_1, \ldots, h_D)^T.$$

In keeping with our canonical ordering, we express our results in terms of right differential convolution matrices. We carefully observe that both the right and left differential convolution matrices described correspond to *right multiplication*. Left multiplication can be formulated in a similar manner.

Let $f, g \in \mathbb{R}(t)[\partial;']$ with $\deg_\partial(f) = M$ and $\deg_\partial(g) = N$. Then by Lemma 6.3.1 we have that $\deg_\partial(\operatorname{gcrd}(f, g)) \geq 1$ if and only if there exist $u, v \in \mathbb{R}(t)[\partial;']$ such that $\deg_\partial(u) < N, \deg_\partial(v) < M$ and $uf + vg = 0$. We can encode the existence of u, v as an $(M+N) \times (M+N)$ matrix over $\mathbb{R}(t)$ in what we will call the differential Sylvester matrix.

Definition 6.3.3. *The matrix*

$$\mathcal{S} = \operatorname{Syl}_\partial(f, g) = \begin{pmatrix} \mathcal{C}^{\mathbf{R}}_{N-1}(f) \\ \mathcal{C}^{\mathbf{R}}_{M-1}(g) \end{pmatrix} \in \mathbb{R}(t)^{(M+N)\times(M+N)}$$

is the differential Sylvester matrix of f and g.

This matrix [79] is analogous to the Sylvester matrix of real polynomials; see [34, Chapter 6]. As expected, many useful properties of the Sylvester matrix over real polynomials still hold with the differential Sylvester matrix. These similarities become evident when we consider

$$w = (u_0, u_1, \ldots, u_{N-1}, v_0, v_1, \ldots, v_{M-1}) \in \mathbb{R}(t)^{1\times(M+N)}.$$

Then $uf + vg = 0$ implies that $w\mathcal{S} = 0$, hence w is a non-trivial vector in the (left) nullspace of $\mathcal{S}$. In particular, this solution is equivalent to saying that $\mathcal{S}$ is singular. Clearing denominators of f and g, we may assume that $u, v \in \mathbb{R}[t][\partial;']$, i.e., they have polynomial coefficients, which implies that $\mathcal{S} \in \mathbb{R}[t]^{(M+N)\times(M+N)}$. Moreover, for $f, g \in \mathbb{R}[t][\partial;']$ with $\deg_t(f) \leq d$ and $\deg_t(g) \leq d$ then $\deg_t(\mathcal{S}_{ij}) \leq d$.

We summarize these results in the following lemma.

Lemma 6.3.4. *Suppose $f, g \in \mathbb{R}[t][\partial;']$, where $\deg_\partial(f) = M$, $\deg_\partial(g) = N$, $\deg_t(f) \leq d$ and $\deg_t(g) \leq d$.*

(i) *$\mathcal{S} = \operatorname{Syl}_\partial(f, g)$ is singular if and only $\deg_\partial(\operatorname{gcrd}(f, g)) \geq 1$.*

(ii) *$\deg_\partial(\operatorname{gcrd}(f, g)) = \dim \ker_\ell(\mathcal{S})$, where $\ker_\ell(\mathcal{S})$ is the left nullspace of $\mathcal{S}$.*

(iii) *For any $w = (u_0, \ldots, u_{N-1}, v_0, \ldots, v_{M-1}) \in \mathbb{R}(t)^{1\times(M+N)}$ such that $w\mathcal{S} = 0$, we have $uf + vg = 0$, where $u = \sum_{0\leq i<N} u_i\partial^i$ and $v = \sum_{0\leq i<M} v_i\partial^i$.*

(iv) *Suppose that $\deg_\partial(\operatorname{gcrd}(f, g)) \geq 1$. Then there exists $\mathfrak{w} \in \mathbb{R}[t]^{1\times(M+N)}$ such that $\mathfrak{w}\mathcal{S} = 0$ and $\deg_t(\mathfrak{w}) \leq \mu = 2(M+N)d$.*

Proof. Part (i) – (iii) follow from Lemma 6.3.1 and the discussion above. Part (iv) follows from Lemma 2.5.9. □

In practice, the bound in part (iv) can be improved in several instances by applying Lemma 3.2.1.

6.3.1 Linear Algebra over $\mathbb{R}$

Let $\mathcal{S} \in \mathbb{R}[t]^{(M+N)\times(M+N)}$ be the differential Sylvester matrix of $f, g \in \mathbb{R}[t][\partial;']$ of degrees M and N respectively in ∂, and degrees at most d in t. From Lemma 6.3.4 we know that if a GCRD of f and g exists, then there is a $w \in \mathbb{R}[t]^{1\times(M+N)}$ such that $w\mathcal{S} = 0$, with $\deg_t(w) \leq \mu = (M+N)d$.

Definition 6.3.5. *Given the $(M+N)\times(M+N)$ differential Sylvester matrix $\mathcal{S}$, we apply $\Phi_\mu(\cdot)$ to $\mathcal{S}$ to with $\mu \leq nd$ to obtain $\Phi_\mu(\mathcal{S})^T = \widehat{S} \in \mathbb{R}^{(\mu+1)(M+N)\times(M+N)(\mu+d+1)}$. We refer to $\widehat{S}$ as the* inflated differential Sylvester matrix *of f and g.*

Lemma 6.3.6. *Let $f, g \in \mathbb{R}[t][\partial;']$ have differential Sylvester matrix $\mathcal{S} \in \mathbb{R}[t]^{(M+N)\times(M+N)}$ and inflated differential Sylvester matrix*

$$\widehat{S} \in \mathbb{R}^{(M+N)(\mu+1)\times(M+N)(\mu+d+1)}.$$

There exists a $w \in \mathbb{R}[t]^{1\times(M+N)}$ such that $w\mathcal{S} = 0$, if and only if there exists a $\widehat{w} \in \mathbb{R}^{(\mu+d+1)\times(M+N)(\mu+1)}$ such that $\widehat{w}\widehat{S} = 0$.

Proof. This follows by Lemma 2.5.9 and Lemma 3.2.1. □

The degree of the GCRD can be computed using any matrix polynomial kernel basis technique from Chapter 3.

6.3.2 Division Without Remainder

While multiplication of differential polynomials with approximate numerical coefficients is straightforward, division is somewhat more difficult. We will generally require a division *without remainder*, for the computation of which we use a least squares approach. Given $f, h \in \mathbb{R}[t][\partial;']$ as in (6.2), we wish to find an $f^* \in \mathbb{R}(t)[\partial;']$ such that $\|f - f^*h\|$ is minimized. We will assume as usual that $\deg_\partial(f) = M$, $\deg_\partial(h) = D$ and $\deg_t(f), \deg_t(h) \leq d$.

Much as in the (approximate polynomial) commutative case, we do this by setting the problem up as a linear system and then finding a least squares solution. Let us assume for now that $f = f^*h$ is exact, so this can be expressed as a linear system over $\mathbb{R}(t)$ by writing

$$(f_0, f_1, \ldots, f_M) = (f_0^*, \ldots, f_{M-D}^*)\mathcal{C}_{M-D}^{\mathbf{R}}(h). \tag{6.3}$$

This system of equations is over-constrained (over $\mathbb{R}(t)$), but we note that the submatrix formed from the last $M-D+1$ columns of $\mathcal{C}_{M-D}^{\mathbf{R}}(h)$ is lower triangular, with diagonal entry $h_D \in \mathbb{R}[t]$. Thus, any exact quotient $h \in \mathbb{R}[t][\partial;']$ such that $f = f^*h$, in lowest terms, must have denominators dividing h_D^{M-D+1}, and in particular have denominators of degree at most $(M-D+1)\deg_t(h_D) \leq (M-D+1)d$. Equivalently, $h_D^{M-D+1}f^* \in \mathbb{R}[t]^{M-D+1}$. By applying Cramer's rule on the last $M-D+1$ columns of $\mathcal{C}^{\mathbf{R}}_{M-D}(h)$, the degrees of the numerators in f^* must be at most $(M-D+1)d$. Using this information we can formulate an associated problem with coefficients from $\mathbb{R}[t]$ and avoid performing linear algebra over $\mathbb{R}(t)$.

Now let $v_{-1}, v_0, \ldots, v_{M-D}$ be generic polynomials in t, with indeterminate coefficients of degree at most $(M-D+1)d$. I.e.,

$$v_i = \sum_{j=0}^{d(M-D+1)} v_{ij}t^j, \quad i = -1 \ldots M-D,$$

for indeterminate v_{ij} with $v_{-1} \neq 0$. Then we are seeking to solve the linear system of equations

$$v_{-1} \cdot (f_0, \ldots, f_M) = (v_0, \ldots, v_{M-D})\ \mathcal{C}^{\mathbf{R}}_{M-D}(h)$$

for the v_{ij}. For each entry f_i we have $(M-D+1)d+d+1$ equations; this is the degree $(v_0, \ldots, v_{M-D})\mathcal{C}^{\mathbf{R}}_{M-D}(h)$ plus one, and we get one equation per coefficient. Hence there are $(M+1)((M-D+1)d+d+1)$ equations in $(M-D+2)(M-D+1)d$ unknowns. We then use a standard linear least squares solution to find the v_i which minimizes the residual, and thus minimizes $\|f - f^*h\|$.

It may be desirable to find the lowest degree v_{-1} which meets this criteria, for which we can use a simple binary search for a lower degree with reasonable residual (or alternatively use an SVD-based identification procedure).

Finally, a more straightforward approach to solving (6.3) is to simply use the solution from the last $M-D+1$ columns of $\mathcal{C}^{\mathbf{R}}_{M-D}(h)$. The last $M-D+1$ columns of $\mathcal{C}^{\mathbf{R}}_{M-D}(h)$ are lower triangular, with diagonal entries consisting of $h_D \in \mathbb{R}[t]$. While this does not yield a solution to the least squares normal equations, it is usually sufficiently good in practice, and considerably easier to formulate.

6.4 Unconstrained Optimization Formulation of Approximate GCRD

First we standardize some notation and assumptions. We assume that $f, g, \widetilde{f}, \widetilde{g}, h \in \mathbb{R}[t][\partial;']$ and $f^*, g^* \in \mathbb{R}(t)[\partial;']$. Moreover, we assume that $\widetilde{f} = f^*h$ and $\widetilde{g} = g^*h$ and

$h = \operatorname{gcrd}(\widetilde{f}, \widetilde{g})$. Intuitively, f, g are our "input polynomials" and we will be identifying "nearby" $\widetilde{f}, \widetilde{g}$ with a non-trivial GCRD h. Note that f^*, g^* have rational function coefficients. Later we will find it useful to clear fractions and work with a primitive associate.

We also assume degree bounds as follows: $\deg_\partial(f) = \deg_\partial(\widetilde{f}) = M$, $\deg_t(f), \deg_t(\widetilde{f}) \leq d$, $\deg_\partial(g), \deg_\partial(\widetilde{g}) = N$, $\deg_t(g), \deg_t(\widetilde{g}) \leq d$, $\deg_\partial(h) = D$, $\deg_\partial(f^*) = M - D$ and $\deg_\partial(g^*) = N - D$.

Using the method of [35], essentially the generalization of the SVD-based method of [21] to differential polynomials, we will make an initial guess for $\widetilde{f}, \widetilde{g}$; details are described in Section 6.6 of this chapter. We then use optimization techniques to hone in on polynomials with minimal distance. While the techniques in that paper are not particularly effective at providing a nearest solution, they do provide a suitable initial guess, which we employ here.

We next describe how to formulate an objective function $\mathbf{\Phi}$ that, when minimized, corresponds to a solution to the approximate GCRD problem.

Problem 6.4.1 (Unconstrained Approximate GCRD)**.** *Define the objective function* $\mathbf{\Phi} : \mathbb{R}[t][\partial;'] \times \mathbb{R}(t)[\partial;']^2 \to \mathbb{R}$ *as*

$$\mathbf{\Phi}(h, f^*, g^*) = \|f - f^*h\|_2^2 + \|g - g^*h\|_2^2.$$

In keeping up with our notation from earlier, we observe that $\widetilde{f} = f^*h$ *and* $\widetilde{g} = g^*h$ *in the context of the objective function* $\mathbf{\Phi}$*, as* f *and* g *will typically be relatively prime.*

To compute guesses for the co-factors given h, we will perform an approximate division without remainder using the method of Section 6.3.2. We only require an initial guess for f^* and g^* to minimize $\mathbf{\Phi}$, so this factorization doesn't need to be exact, in the event that $\operatorname{gcrd}(f, g) = h$.

We show that $\mathbf{\Phi}$ has an attainable global minimum under appropriate assumptions. More precisely, there exist non trivial $\widetilde{f}$ and $\widetilde{g}$ such that

$$\|f - \widetilde{f}\|_2^2 + \|g - \widetilde{g}\|_2^2$$

is minimized. Furthermore, we will show that the approximate GCRD problem is locally well-posed.

6.4.1 Existence of Solutions

Lemma 6.4.2. *Let $f, h \in \mathbb{R}[t][\partial;']$, with monic leading coefficients, be not necessarily primitive, such that $f = f^*h$ for $f^* \in \mathbb{R}[t][\partial;']$ with $\deg_\partial(f) = M$ and $\deg_\partial(h) = D$. Then $\|f^*\|$ is bounded above.*

Proof. It follows that f^* is bounded by the computing the solution to (6.3) using the last $M - D + 1$ columns of $\mathcal{C}^{\mathbf{R}}_{M-D}(h)$ and Cramer's rule. □

As an observation, we relax the assumption that f is primitive (we work with an associate instead) in order to guarantee that $f^* \in \mathbb{R}[t][\partial;']$. This can be taken without loss of generality as the quantity $\|\operatorname{cont}(f)\|_2^2$ is bounded above and away from zero (as its leading coefficient is monic). Thus we may divide by it without affecting the quality of the results, as $\|f - f^*h\|$ is still well defined.

We first state a general version of the theorem where a logical predicate $\Xi : \mathbb{R}^k \to \{\text{true}, \text{false}\}$ (for some k) can be chosen to impose additional constraints on the problem. For the rest of this section let

$$\phi : \mathbb{R}[t][\partial;']^2 \to \mathbb{R}^{(M+N+2)(d+1)}$$

be the combined coefficient vector function, i.e. for arbitrary $f, g \in \mathbb{R}[t][\partial;']$ we write $\phi(f, g) = (\operatorname{vec}(f), \operatorname{vec}(g))$, where $\operatorname{vec}(f)$ and $\operatorname{vec}(g)$ are padded with zeros to have the desired dimensions.

The following lemma and its proof are analogous to [70, Theorem 2], which in turn generalizes the univariate argument of [71, Theorem 1].

Theorem 6.4.3 (Existence of Global Minima)**.** *Let $f, g \in \mathbb{R}[t][\partial;']\backslash\{0\}$, let $d = \max\{\deg_t(f), \deg_t(g)\}$, $\deg_\partial(f) = M$, $\deg_\partial(g) = N$ and $D \leq \min\{M, N\}$. Furthermore, let $\Xi : \mathbb{R}^{(M+N+2)(d+1)} \to \{\text{true}, \text{false}\}$ be a predicate on $\phi(f, g)$. We assume that the preimage $\Xi^{-1}(\text{true})$ is a topologically closed set in $\mathbb{R}^{(M+N+2)(d+1)}$ with respect to the Euclidean norm. For a given $\Omega \in \mathbb{R}_{>0}$ we define the set of possible solutions by*

$$\mathscr{F}_\Omega = \left\{ \begin{array}{lc} (\widetilde{f}, \widetilde{g}) \in \mathbb{R}[t][\partial;']^2 \text{ such that} & \deg_\partial(\widetilde{f}) = M, \\ & \deg_\partial(\widetilde{g}) = N, \\ & \deg_\partial(\widetilde{h}) \geq D, \\ & \widetilde{h} = \operatorname{gcrd}(\widetilde{f}, \widetilde{g}), \\ & \|\widetilde{h}\| \leq \Omega, \\ & \operatorname{lcoeff}_t(\operatorname{lcoeff}_\partial(\widetilde{h})) = 1, \\ & \text{and } \Xi(\phi(\widetilde{f}, \widetilde{g})) = \text{true} \end{array} \right\}.$$

Suppose that $\mathcal{F}_\Omega \neq \emptyset$. Then the minimization problem

$$\min_{(\widetilde{f},\widetilde{g})\in\mathcal{F}_\Omega} \|f-\widetilde{f}\|_2^2 + \|g-\widetilde{g}\|_2^2 \tag{6.4}$$

has an attainable global minimum.

Proof. Without loss of generality, we assume that $M \leq N$. Then we iterate the minimization over all $\ell \in \mathbb{Z}_{\geq 0}$ such that $D \leq \ell \leq M$ and coefficients $\rho \subset \mathbb{R}[t]^\ell$. Let $\mathcal{H}_{\ell,\rho}$ denote the set of all differential polynomials over $\mathbb{R}[t][\partial;']$ of degree ℓ with coefficients from ρ. We optimize over the continuous real objective function

$$\mathbf{\Phi}(h, f^*, g^*) = \|f - f^*h\|_2^2 + \|g - g^*h\|_2^2,$$

for $h \in \mathcal{H}_{\ell,\rho}$, $\deg_\partial(f^*) \leq M - D$ and $\deg_\partial(g^*) \leq N - D$. We fix the leading coefficient of h with respect to ∂ to be monic, that is $\operatorname{lcoeff}_t(\operatorname{lcoeff}_\partial(h)) = 1$.

Since the leading coefficient of h is monic, we can write $G = \operatorname{gcrd}(f^*h, g^*h)$ with $\deg_\partial(G) \geq D$. Since G is a multiple of h, we normalize G so that $\operatorname{lcoeff}_t(\operatorname{lcoeff}_\partial(G)) = 1$, i.e. the leading coefficient of G is also monic. The restriction on h that the leading coefficient of h is monic enforces that $\deg_\partial(G) \geq D$. Furthermore, we restrict the domain of our function $\mathbf{\Phi}$ to those h, f^* and g^* for which $(f^*h, g^*h) \in \mathcal{F}_\Omega$. If there is no such common factor h and co-factors f^* and g^*, then this pair of ℓ and ρ does not occur in the minimization (6.4). By assumption we have that $\mathcal{F}_\Omega \neq \emptyset$, so there must be at least one possible case. We note that if $(0,0) \in \mathcal{F}_\Omega$, then $f^* = g^* = 0$.

Now suppose that for the given ℓ and ρ, there are $\widetilde{h} \in \mathcal{H}_{\ell,\rho}$ and $\widetilde{f}^*, \widetilde{g}^*$ satisfying $\deg_\partial(\widetilde{f}^*) \leq M - \ell$ and $\deg_\partial(\widetilde{g}^*) \leq N - \ell$ such that $(\widetilde{f}^*\widetilde{h}, \widetilde{g}^*\widetilde{h}) \in \mathcal{F}_\Omega$. We shall prove that the function $\mathbf{\Phi}$ has a value on a closed and bounded set (i.e., compact with respect to the Euclidean metric) that is smaller than elsewhere. Hence $\mathbf{\Phi}$ attains a global minimum by Weierstrass" theorem.

Clearly any solution $\widetilde{h} \in \mathcal{H}_{\ell,\rho}$ and $\widetilde{f}^*, \widetilde{g}^*$ with $(\widetilde{f}^*h, \widetilde{g}^*h) \in \mathcal{F}_\Omega$ but with $\mathbf{\Phi}(\widetilde{h}, \widetilde{f}^*, \widetilde{g}^*) > \mathbf{\Phi}(h, f^*, g^*)$ can be discarded. So the norm of the products $\|\widetilde{f}^*\widetilde{h}\|_2$ and $\|\widetilde{g}^*\widetilde{h}\|_2$ can be bounded from above. We have that $\|\widetilde{h}\|$ is bounded above by Lemma 6.4.2 because it is a right factor of $\widetilde{G} = \operatorname{gcrd}(\widetilde{f}^*\widetilde{h}, \widetilde{g}^*\widetilde{h})$ with $\|\widetilde{G}\| \leq \Omega$. We note that $\widetilde{h}$ has a monic leading coefficient, so $\|\widetilde{h}\| \geq 1$. We have that $\|\widetilde{f}^*\|$ and $\|\widetilde{g}^*\|$ (or the appropriate associate) are both bounded above by Lemma 6.4.2.

Thus we can restrict the domain of $\mathbf{\Phi}$ to values that lie within a sufficiently large closed ball B. The function ζ that maps (h, f^*, g^*) to the combined coefficient vector $\phi(f^*h, g^*h)$

of f^*h and g^*h is continuous. We minimize over $\zeta^{-1}(\Xi^{-1}(\text{true}) \cap \zeta(B))$, which is a compact set. $\square$

For the less general version of the theorem, given arbitrary $f, g \in \mathbb{R}[t][\partial;']$, we define

$$\mathcal{S} = \mathcal{S}(f,g) = \left\{ \phi(\widetilde{f}, \widetilde{g}) \mid \widetilde{f}, \widetilde{g} \in \mathbb{R}[t][\partial;'] \text{ such that } \operatorname{dvec}(\widetilde{f}) \leq \operatorname{dvec}(f) \operatorname{dvec}(\widetilde{g}) \leq \operatorname{dvec}(g) \right\}.$$

We observe that $\mathcal{S}$ is a closed subset of $\mathbb{R}^{(M+N+2)(d+1)}$, where $\deg_\partial(f) = M$, $\deg_\partial(g) = N$ and $d = \max\{\deg_t(f), \deg_t(g)\}$. The set $\mathcal{S}$ corresponds to the combined coefficient vectors of $\widetilde{f}$ and $\widetilde{g}$ that have the same degree structure as f and g.

Corollary 6.4.4. *Let $f, g \in \mathbb{R}[t][\partial;']\backslash\{0\}$, let $d = \max\{\deg_t(f), \deg_t(g)\}$, $\deg_\partial(f) = M$, $\deg_\partial(g) = N$ and $D \leq \min\{M, N\}$. For a given $\Omega \in \mathbb{R}_{>0}$ we define the set of possible solutions by*

$$\mathcal{F}_\Omega = \left\{ \begin{array}{ll} (\widetilde{f}, \widetilde{g}) \in \mathbb{R}[t][\partial;'] \times \mathbb{R}[t][\partial;'] \textit{ such that} & \deg_\partial \widetilde{f} = M, \\ & \deg_\partial \widetilde{g} = N, \\ & \phi(\widetilde{f}, \widetilde{g}) \in \mathcal{S}, \\ & \deg_\partial \widetilde{h} \geq D, \\ & \widetilde{h} = \operatorname{gcrd}(\widetilde{f}, \widetilde{g}), \\ & \|\widetilde{h}\| \leq \Omega, \\ & \textit{and } \operatorname{lcoeff}_t(\operatorname{lcoeff}_\partial(\widetilde{h})) = 1 \end{array} \right\}.$$

Suppose that $\mathcal{F}_\Omega \neq \emptyset$. Then the minimization problem

$$\min_{(\widetilde{f}, \widetilde{g}) \in \mathcal{F}_\Omega} \|f - \widetilde{f}\|_2^2 + \|g - \widetilde{g}\|_2^2$$

has an attainable global minimum.

We note that Theorem 6.4.3 does not guarantee a unique minimum of $\mathbf{\Phi}$, merely that $\mathbf{\Phi}$ has an attainable minimum (as opposed to an infimum). The choice of h, f^* and g^* that we optimize over is important. If $\operatorname{lcoeff}_t(\operatorname{lcoeff}_\partial(h))$ vanishes or $\|h_0\|, \ldots, \|h_{D-1}\|$ are quite large, then f^* and g^* can be ill-conditioned in the approximate GCRD problem. Furthermore, choosing overly large, small or poor degree structure in t for h can result in a $\mathbf{\Phi}$ that cannot be minimized for the specified structure, but would otherwise have a minimum for a different choice of h.

Example 6.4.5. *Consider $f = \partial^2 - 2\partial + 1$ and $g = \partial^2 + 2\partial + 2$ (see [70] for an example with complex perturbations). Then f and g do not have a degree 1 approximate GCRD. That is, we show that there does not exist $\widetilde{f}, \widetilde{g} \in \mathbb{R}[t][\partial;']$ where $\deg_\partial(\mathrm{gcrd}(\widetilde{f}, \widetilde{g})) = 1$ and $\|f - \widetilde{f}\|_2^2 + \|g - \widetilde{g}\|_2^2$ is minimized.*

The real monic Karmarkar-Lakshman distance [72, 73] of

$$\|f - \widetilde{f}\|_2^2 + \|g - \widetilde{g}\|_2^2$$

occurs when the rational function

$$\frac{2h_0^4 + 14h_0^2 + 4h_0 + 5}{h_0^4 + h_0^2 + 1}$$

is minimized for $h_0 \in \mathbb{R}$. The minimum value (if it exists) of this function corresponds to the approximate GCRD $h = \partial - h_0$. The infimum is 2, which is unattainable. There is no attainable global minimum.

The non-monic real Karmarkar-Lakshman distance is 2, which is achieved if and only if the leading coefficient vanishes. The minimum occurs when the rational function

$$\frac{5h_1^4 - 4h_1^3 + 14h_1^2 + 2}{h_1^4 + h_1^2 + 1}$$

is minimized. The minimum value of this function corresponds to the approximate GCRD $h = h_1\partial + 1$.

In particular, if we consider $\widetilde{f} = (-2\partial + 1)(\varepsilon\partial + 1)$ and $\widetilde{g} = (2\partial + 2)(\varepsilon\partial + 1)$, then $\|f - \widetilde{f}\|_2^2 + \|g - \widetilde{g}\|_2^2$ becomes arbitrarily near 2 as $\varepsilon \to 0$.

There is no real degree 1 approximate GCRD, as

$$\min_{\left\{(\widetilde{f},\widetilde{g})\in\mathbb{R}[t][\partial;']^2 \;|\; \deg_\partial(\mathrm{gcrd}(\widetilde{f},\widetilde{g}))=1\right\}} \|f - \widetilde{f}\|_2^2 + \|g - \widetilde{g}\|_2^2$$

is not defined in the monic case. In the non-monic case, if a minimum exists then it occurs when $\mathrm{lcoeff}_\partial(h)$ vanishes, so the minimum value is not defined either.

This example illustrates that not all $f, g \in \mathbb{R}[t][\partial;']$ have an approximate GCRD. Furthermore, we see that the requirement that $\mathrm{lcoeff}_t(\mathrm{lcoeff}_\partial(h)) = 1$ and $\|h\|$ is bounded, from Theorem 6.4.3 are required, even if there are no additional constraints imposed.

Now it remains to show that it is possible to obtain a (locally) unique solution to $\mathbf{\Phi}$. One of many equivalent conditions for uniqueness of an exact GCRD, is to require it to be primitive and have a monic leading coefficient. Numerically, to obtain a unique solution of the approximate GCRD problem, we impose the same constraints, making solutions locally unique.

6.4.2 Convergence of Newton Iteration and Conditioning

From Theorem 6.4.3 and Corollary 6.4.4 we know a solution to the approximate GCRD problem exists. We now show that a standard Newton iteration will converge quadratically when starting with an estimate sufficiently close to an approximate GCRD. We first describe the Jacobian of the residuals and show that the Jacobian has full rank. This leads to a first-order approximation of the Hessian matrix showing that it is locally positive definite around a global minimum when the residual is sufficiently small. The implication is that Newton's method will converge quadratically. If we consider structured perturbations, then we are able to obtain results similar to that of [112] to the overall conditioning of the system.

In this section we assume without loss of generality that $f^*, g^* \in \mathbb{R}[t][\partial;']$ are primitive, and that f and g may no longer be primitive to simplify computations. We need to clear fractions of rational functions to apply our coefficient norms, and to linearize h, f^* and g^* as vectors of real numbers.

The residual of the approximate GCRD is

$$\begin{aligned} r &= r(h, f^*, g^*) \\ &= (\operatorname{vec}(f^*h) - \operatorname{vec}(f), \operatorname{vec}(g^*h) - \operatorname{vec}(g))^T \in \mathbb{R}^{\eta\times 1}, \end{aligned}$$

where

$$\begin{aligned} \eta &= \sum_{0\le i\le M} \max\{\deg_t(f_i), -1\} + \sum_{0\le i\le N} \max\{\deg_t(g_i), -1\} + (M+1) + (N+1) \\ &\le (M+N+2)(d+1). \end{aligned}$$

Intuitively, η represents the number of components of $(\operatorname{vec}(f), \operatorname{vec}(g)) \in \mathbb{R}^{1\times\eta}$. Let ν be the number of variables needed to represent the coefficients of h, f^* and g^*, i.e. $(\operatorname{vec}(h), \operatorname{vec}(f^*), \operatorname{vec}(g^*)) \in \mathbb{R}^{1\times\nu}$.

Recall that when $f = f^*h$, we can linearize this relationship with differential convolution matrices, by writing

$$f = (f_0^*, \ldots, f_{M-D}^*) C_{M-D}^{\mathbf{R}}(h).$$

If f_i is a coefficient of f with $\deg_t(f_i) = d$, then we may write

$$f_i = \sum_{0 \leq j \leq M-D} f_j^* (\mathcal{C}^{\mathbf{R}}_{M-D}(h)[j,i]).$$

This relationship may be linearized over $\mathbb{R}$ through the use of convolution matrices. Writing

$$\text{vec}(f_i) = \sum_{0 \leq j \leq M-D} \text{vec}(f_j^*) \cdot C_d \left(\mathcal{C}^{\mathbf{R}}_{M-D}(h)[j,i]\right)^T,$$

we now have a direct method of computing $\text{vec}(f_i)$ in terms of the coefficients of f^* and h.

If we differentiate $\text{vec}(f^*h)$ with respect to an entry from $\text{vec}(f^*)$, then we will obtain the corresponding (linearized) row of $\mathcal{C}^{\mathbf{R}}_{M-D}(h)$. Similarly, differentiating $\text{vec}(f^*h)$ with respect to an entry of $\text{vec}(h)$ will give us a (linearized) column of $\mathcal{C}^{\mathbf{L}}_{D}(f^*)$. This relationship becomes clear when we observe that

$$(f_0^*, \ldots, f_{M-D}^*)\mathcal{C}^{\mathbf{R}}_{M-D}(h) = \left(\mathcal{C}^{\mathbf{L}}_{D}(f^*) \begin{pmatrix} h_0 \\ \vdots \\ h_D \end{pmatrix} \right)^T.$$

Differentiating $\text{vec}(g^*h)$ with respect to variables from $\text{vec}(g^*)$ and $\text{vec}(h)$ will produce similar results.

The Jacobian of $r(h, f^*, g^*)$ for arbitrary h, f^* and g^* may be expressed (up to column permutation) in block matrix form as

$$J = \begin{pmatrix} \mathcal{C}^{\mathbf{R}}_{M-D}(h)^T & 0 & \mathcal{C}^{\mathbf{L}}_{D}(f^*) \\ 0 & \mathcal{C}^{\mathbf{R}}_{N-D}(h)^T & \mathcal{C}^{\mathbf{L}}_{D}(g^*) \end{pmatrix} \in \mathbb{R}^{\eta \times \nu},$$

where the block matrices are linearized accordingly. In our formulation of the approximate GCRD problem we normalize $\text{lcoeff}_t(\text{lcoeff}_\partial(h))$ so that it is a predetermined constant, which results in essentially the same Jacobian as described above.

The only difference in the Jacobians, is that the ν^{th} column would become the zero column if differentiated with respect to $\text{lcoeff}_t(\text{lcoeff}_\partial(h))$, since $\text{lcoeff}_t(\text{lcoeff}_\partial(h))$ is constant. When normalized for computational purposes, the Jacobian belongs to $\mathbb{R}^{\eta \times \nu - 1}$ instead (the last column is deleted). In the general case when $\text{gcrd}(f^*, g^*) = 1$, J is rank deficient by 1 and the ν^{th} column is a linear combination of the other columns. The following lemma, similar to [111], formalizes this statement.

Theorem 6.4.6. *Let r be the residual described earlier with Jacobian J. Suppose that* $\operatorname{lcoeff}_t(\operatorname{lcoeff}_\partial(h))$ *is a fixed non-zero constant. If* $\operatorname{gcrd}(f^*, g^*) = 1$*, then all non-zero columns of J are linearly independent.*

Proof. Let $\operatorname{vec}(e_\nu) \in \mathbb{R}^{1\times\nu}$ be a unit vector whose last component is 1. We write

$$\operatorname{vec}(e_\nu)(0,\dots,0,\operatorname{vec}(h))^T = \operatorname{lcoeff}_t(\operatorname{lcoeff}_\partial(h)) \neq 0.$$

We shall prove the equivalent statement that the matrix

$$\begin{pmatrix} J \\ \operatorname{vec}(e_\nu) \end{pmatrix} = \begin{pmatrix} \mathcal{C}^{\mathbf{R}}_{M-D}(h)^T & 0 & \mathcal{C}^{\mathbf{L}}_D(f^*) \\ 0 & \mathcal{C}^{\mathbf{R}}_{N-D}(h)^T & \mathcal{C}^{\mathbf{L}}_D(g^*) \\ & & \operatorname{vec}(e_\nu) \end{pmatrix} \in \mathbb{R}^{(\eta+1)\times\nu}$$

has full rank.

Suppose the converse holds, then there exists $q_1, q_2, p \in \mathbb{R}[t][\partial;']$ with $\deg_\partial(q_1) \le M - D, \deg_\partial(q_2) \le N - D$ and $\deg_\partial(p) \le D$ such that their combined coefficient vector satisfies

$$\begin{pmatrix} J \\ \operatorname{vec}(e_\nu) \end{pmatrix} \begin{pmatrix} \operatorname{vec}(q_1)^T \\ \operatorname{vec}(q_2)^T \\ -\operatorname{vec}(p)^T \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

Expressing this as multiplication over $\mathbb{R}[t][\partial;']$, we have that

$$f^*p = q_1h,$$
$$g^*p = q_2h.$$

We conclude that $\operatorname{gcrd}(f^*p, g^*p) = p$, as $\operatorname{gcrd}(f^*, g^*) = 1$. If $p = 0$ or $q_1 = 0$ or $q_2 = 0$, then we are done (as $\mathbb{R}[t][\partial;']$ is a domain). Suppose that $p \neq 0$ and $q_1 \neq 0$ and $q_2 \neq 0$. Accordingly we must also have that $\operatorname{gcrd}(q_1h, q_2h) = \operatorname{gcrd}(q_1, q_2)\alpha h = p$ for some $\alpha \neq 0$. Since $\deg_\partial(p) \le \deg_\partial(h)$ it follows that $\operatorname{gcrd}(q_1, q_2) = 1$ so $p = \alpha h$.

Since $p = \alpha h$ we must have that $\alpha f^* = q_1$ and $\alpha g^* = q_2$. Now,

$$\operatorname{vec}(e_\nu)(0,\dots,0,\operatorname{vec}(h))^T = \operatorname{lcoeff}_t(\operatorname{lcoeff}_\partial(h)) \neq 0.$$

On the other hand,

$$\operatorname{vec}(e_\nu)(0,\dots,0,\alpha\operatorname{vec}(h))^T = 0.$$

This occurs if and only if $\alpha = 0$. But in this case $p = 0$ as well, so

$$\begin{pmatrix} \operatorname{vec}(q_1)^T \\ \operatorname{vec}(q_2)^T \\ -\operatorname{vec}(p)^T \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

It follows that the only vector in the nullspace is the zero vector, hence $\begin{pmatrix} J \\ \operatorname{vec}(e_\nu) \end{pmatrix}$ has full rank. Since any subset of linearly independent vectors is also linearly independent, we have that when $\operatorname{lcoeff}_t(\operatorname{lcoeff}_\partial(h))$ is a fixed non-zero constant that J has rank $\nu - 1$. □

Note that from the proof we see that if $\operatorname{lcoeff}_t(\operatorname{lcoeff}_\partial(h))$ were not fixed, then the vector $(\operatorname{vec}(f^*), \operatorname{vec}(g^*), \operatorname{vec}(h))^T$ forms a basis for the nullspace of J. Intuitively, if we did not fix $\operatorname{lcoeff}_t \operatorname{lcoeff}_\partial$ in advance, then there would be infinitely many tuples of (h, f^*, g^*) with the same degree structure over $\mathbb{R}[t]$ that minimized $\mathbf{\Phi}$, since for any $\alpha \neq 0$ we have

$$\|f - f^*h\|_2^2 + \|g - g^*h\|_2^2 = \|f - (\alpha f^*)(\alpha^{-1}h)\|_2^2 + \|g - (\alpha g^*)(\alpha^{-1}h)\|_2^2.$$

In other words, we need to normalize h in advance to obtain a unique solution.

Corollary 6.4.7. *Let r be the residual defined earlier in this section with* $\operatorname{lcoeff}_t(\operatorname{lcoeff}_\partial(h))$ *a non-zero constant. If $r = 0$, then the Hessian matrix $\nabla^2\mathbf{\Phi}(h, f^*, g^*)$ is positive definite.*

Proof. Let J be the Jacobian of r. J has full rank, so J^TJ has full rank and is positive semidefinite. If $r = 0$, at the global minimum we have that $2J^TJ = \nabla^2\mathbf{\Phi}$, and $\nabla^2\mathbf{\Phi}(h, f^*, g^*)$ is positive definite. □

When there is no residual, the Hessian $\nabla^2\mathbf{\Phi}(h, f^*, g^*)$ is positive definite. It follows that if f and g are perturbed by a sufficiently small amount, then $\nabla^2\mathbf{\Phi}$ remains locally positive definite, and a Newton iteration will converge to the (local) global minimum with an initial guess that is sufficiently close.

We are able to obtain a condition number for a structured perturbation through the Jacobian of the residuals. Since J has full rank, the smallest singular value $\sigma_{\nu-1}$ of $J(r(h, f^*, g^*))$ is strictly positive. If we consider structured perturbations, then we are able to show that the approximate GCRD problem is (locally) well-posed.

In the next lemma, we make use of the fact that for any $f \in \mathbb{R}[t][\partial;']$, we have that $\|f\|_2 = \|\operatorname{vec}(f)\|_2$.

Lemma 6.4.8. *Let $f, g, h, f^*, g^* \in \mathbb{R}[t][\partial;']$ be such that $\mathbf{\Phi}(h, f^*, g^*) < \varepsilon$ for some $\varepsilon > 0$, with* $\operatorname{lcoeff}_t(\operatorname{lcoeff}_\partial(h))$ *a fixed non-zero constant. Suppose $\widehat{f}, \widehat{g}, \widehat{h}, \widehat{f^*}, \widehat{g^*} \in \mathbb{R}[t][\partial;']$ possess the same degree structures as f, g, h, f^* and g^* and that*

$$\widehat{\mathbf{\Phi}}(\widehat{h}, \widehat{f^*}, \widehat{g^*}) = \|\widehat{f} - \widehat{f^*}\widehat{h}\|_2^2 + \|\widehat{g} - \widehat{g^*}\widehat{h}\|_2^2 < \varepsilon.$$

Then,

$$\left\|(h - \widehat{h}, f^* - \widehat{f^*}, g^* - \widehat{g^*})\right\|_2^2 \leq \frac{1}{\sigma_{\nu-1}^2}\left(2\varepsilon + \left\|(f - \widehat{f}, g - \widehat{g})\right\|_2^2\right) + \text{\textit{higher order terms.}}$$

Proof. Let $J = J(r(h, f^*, g^*))$ be the Jacobian of the residuals from earlier in this section. We have that

$$\begin{aligned}&(\text{vec}(f^*h) - \text{vec}(\widehat{f^*}\widehat{h}), \text{vec}(g^*h) - \text{vec}(\widehat{g^*}\widehat{h}))^T\\ \approx &J(\text{vec}(h - \widehat{h}), \text{vec}(f^*) - \text{vec}(\widehat{f^*}), \text{vec}(g) - \text{vec}(\widehat{g^*}))^T.\end{aligned}$$

Ignoring high order terms and using the well known fact that for a (left) pseudo inverse J^+ of J, that $\|J^+\|_2 = \frac{1}{\sigma_{\nu-1}}$ gives us

$$\begin{aligned}\left\|(\text{vec}(f^*h - \widehat{f^*}\widehat{h}, g^*h - \widehat{g^*}\widehat{h}))^T\right\|_2^2 &\approx \left\|J(\text{vec}(h - \widehat{h}, f^* - \widehat{f^*}, g^* - \widehat{g^*}))^T\right\|_2^2\\ &\geq \sigma_{\nu-1}^2\left\|(\text{vec}(h - \widehat{h}, f^* - \widehat{f^*}, g^* - \widehat{g^*}))\right\|_2^2.\end{aligned}$$

A straightforward application of the triangle inequality gives

$$\begin{aligned}&\left\|(h - \widehat{h}, f^* - \widehat{f^*}, g^* - \widehat{g^*})\right\|_2^2\\ &\quad\leq \frac{1}{\sigma_{\nu-1}^2}\left\|(f^*h - \widehat{f^*}\widehat{h}, g^*h - \widehat{g^*}\widehat{h})\right\|_2^2\\ &\quad\leq \frac{1}{\sigma_{\nu-1}^2}\left(\mathbf{\Phi}(h, f^*, g^*) + \widehat{\mathbf{\Phi}}(\widehat{h}, \widehat{f^*}, \widehat{g^*}) + \left\|(f - \widehat{f}, g - \widehat{g})\right\|_2^2\right)\\ &\quad\leq \frac{1}{\sigma_{\nu-1}^2}\left(2\varepsilon + \left\|(f - \widehat{f}, g - \widehat{g})\right\|_2^2\right) + \text{higher order terms.}\end{aligned}$$ □

Corollary 6.4.9. *Suppose that $h_{opt}, f^*_{opt}, g^*_{opt} \in \mathbb{R}[t][\partial;']$ are a locally unique global minimum of $\mathbf{\Phi}$ in some neighborhood around h, f^* and g^*. If*

$$\mathbf{\Phi}(h, f^*, g^*) < \varepsilon \textit{ and } \mathbf{\Phi}(h_{opt}, f^*_{opt}, g^*_{opt}) < \varepsilon$$

for $\varepsilon > 0$, then

$$\left\|(h - h_{opt}, f^* - f^*_{opt}, g^* - g^*_{opt})\right\|_2^2 \leq \frac{2\varepsilon}{\sigma_{\nu-1}^2} + \textit{higher order terms.}$$

If we compute different approximate GCRD pairs of f and g (using different optimization techniques or initial guesses), then we are able to bound the size of the perturbations of f^*, g^* and h based on how near they are. Furthermore, this corollary allows us to certify an upper bound on the distance between our computed approximate GCRD tuple and the actual global minimum.

6.5 Approximate GCRD via Nearest Singular Matrix Polynomial

Another way to approach the approximate GCRD problem is through finding the nearest matrix polynomial with a linear structure. The differential Sylvester matrix is a matrix polynomial that is endowed with a linear structure.

Problem 6.5.1 (Approximate GCRD via Nearest Singular Differential Sylvester Matrix)**.** *Given* $f, g \in \mathbb{R}[t][\partial;']$ *with* $\deg_\partial(f) \leq M, \deg_\partial(g) \leq N$ *and* $\max\{\deg_t(f), \deg_t(g)\} \leq d$ *then we can formulate the approximate GCRD problem as the following minimization problem:*

$$\min_{\Delta f, \Delta g} \|\Delta f\|_2^2 + \|\Delta g\|_2^2 \quad \textit{subject to} \quad \operatorname{rank}(\operatorname{Syl}_\partial(f+\Delta f, g+\Delta g)) \leq M+N-1, \tag{6.5}$$

for some reasonable perturbation structures Δf *and* Δg *imposed upon* f *and* g *respectively.*

Some examples of reasonable perturbation structures are preserving low or high-order zero coefficients, or allowing perturbations in t of at most degree $\deg_t(f)$ or $\deg_t(g)$.

It is important to note that it is not generic that the nearest rank deficient differential Sylvester matrix has rank $M+N-1$ (despite being a matrix polynomial), which is the generic instance of the nearest singular matrix polynomial problem. This occurs because of the particular structure differential Sylvester structure encountered and the factorization properties of $\mathbb{R}(t)[\partial;']$.

6.5.1 Why Consider Approximate GCRD via Nearest Rank Deficient Matrix Polynomial?

At a first glance the unconstrained formulation appears superior; obviously there are no constraints, when the residual is small the problem is locally convex (strictly convex if normalized properly) and the size of the problem is typically small. A natural question is what is gained by studying the problem in the context of the nearest rank deficient matrix polynomial (with a linear structure)?

First, we note that the unconstrained version of the problem has irregular solutions. That is, Theorem 6.4.3 only characterizes regular solutions. It is entirely possible that at a solution $(\Delta f^\star, \Delta g^\star)$ that $\operatorname{lcoeff}_\partial(f+\Delta f) \to 0$ and $\operatorname{lcoeff}_\partial(g+\Delta g) \to 0$. In such a scenario, the differential Sylvester matrix $\operatorname{Syl}_\partial(f+\Delta f^\star, g+\Delta g^\star)$ will have a column of zero entries because the leading coefficients of f and g were both perturbed to zero. Naturally, this leads to an unattainable or irregular solution.

Theorem 6.5.2 (Characterization of Solutions)**.** *A nearest rank deficient differential Sylvester matrix exists for perturbations Δf and Δg that are linearly structured.*

Proof. This is essentially a corollary of Theorem 3.3.4. □

Theorem 6.5.2 characterizes both regular and irregular solutions to the approximate GCRD problem. Indeed, if the leading coefficient with respect to ∂ of both f and g vanish, then there is a GCRD at infinity, which is entirely analogous to the usual commutative approximate GCD theory. To regularize approximate GCD problems with solutions at infinity (i.e. an irregular or unattainable solution), then we typically reverse the coefficients as was done in Chapter 5. In the instance of differential operators, reversing the coefficients no longer regularizes the problem. Fortunately a close inspection of Proposition 6.2.5 reveals that a GCRD at infinity is theoretically permissible. Analogously, an alternative regularization to approximate GCD problems is to approach them through their (generalized) Sylvester matrices, which is considered in Chapter 5.

6.5.2 Equivalence of Problem Formulations

Next, we observe that (6.5) permits restricting the perturbations to f and g to have some reasonable structure. The unconstrained formulation does not have such fine control over the coefficient perturbations, and tends to increase the degrees of the input. Perturbing higher-order zero coefficients in t may not make sense in the context of several problems.

Note that Theorem 6.4.6 implies that if the unconstrained problem were transformed into a constrained problem, Lagrange multipliers would exist via a combination of linearly independent constraint qualification and linear constraint qualification for *regular* solutions. Indeed, the optimization problem

$$\min \|\Delta f\|_2^2+\|\Delta g\|_2^2 \text{ subject to } \begin{cases} \Delta f = f - f^* h, \\ \Delta g = g - g^* h, \\ f^* \in \mathbb{R}(t)[\partial;'], \\ g^* \in \mathbb{R}(t)[\partial;'], \\ \Delta f \text{ has a prescribed linear structure,} \\ \Delta g \text{ has a prescribed linear structure and} \\ h \text{ is primitive and has a monic leading coefficient in } t. \end{cases} \tag{6.6}$$

is easily seen to be equivalent to (6.5) in several instances. The primitivity and monic leading coefficient assumption is analogous to the minimal degree embedding encountered in Chapter 3 to ensure that solutions are locally unique. If the primitivity assumption in (6.6) is relaxed, then (6.6) and (6.5) are only equivalent if all possible degrees in ∂ of h are considered. In theory, (6.5) and (6.6) are functionally equivalent for *regular or attainable solutions*, although in practical instances there is some care required to ensure that different implementations solve the same problem.

Furthermore in the matrix polynomial version of the problem, the co-factors f^* and g^* are no longer required. Since $f^*, g^* \in \mathbb{R}(t)[\partial;']$ there was size growth in the original problem, and the headache of possibly being required to solve an associate problem. In the nearest singular matrix polynomial formulation of the problem there are no associates instances to consider, as there will always be a kernel vector that is polynomial and a GCRD that belongs to $\mathbb{R}[t][\partial;']$ that is primitive. If desired, the co-factors may be extracted from the computed solution via a least squares division.

6.5.3 A Hybrid Algorithmic Approach

The unconstrained formulation of the problem is highly desirable as one may apply descent methods, alternating minimizations or some other reasonable block coordinate descent. Initial guesses are easily computed via the SVD by performing a round of lift and project. Unfortunately, the unconstrained formulation does not encapsulate structure in the coefficients of f and g very well. To overcome this limitation a hybrid approach can be used.

One can first compute a local minimizer to Problem 6.4.1, then fit via least squares a guess for Δf and Δg from $\Delta f^{(0)} \approx f - f^* h$ and $\Delta g^{(0)} \approx g - g^* h$. The initial guesses $\Delta f^{(0)}$ and $\Delta g^{(0)}$ can be improved by projecting to the feasible set by using Newton's method to compute a solution to

$$\begin{pmatrix} \mathrm{Syl}(f+\Delta f, g+\Delta g)b \\ \|b\|_2^2 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

From this feasible point we can then use the machinery of Chapter 3 to compute a nearby rank deficient differential Sylvester matrix to solve Problem 6.5.1.

6.6 Implementation of Approximate GCRD

In this section we discuss an implementation of computing a pair of nearby differential polynomials with a non-trivial GCRD. We use the concepts presented in Chapter 3 to compute a nearby differential Sylvester matrix that is singular. The implementation is in Maple.

```
Algorithm 6 : Nearest Singular Differential Sylvester Matrix
Input:
  • $f, g \in \mathbb{R}[t][\partial;']$.
  • Initial guesses $\Delta f^{init}$ and $\Delta g^{init}$.
  • Displacement structure $\Delta f$ and $\Delta g$ to optimize over.
Output:
  • A local minimizer to (6.5) or an indication of failure.
1: Extract a (minimally embedded) kernel vector from $\ker(\mathrm{Syl}_\partial(f + \Delta f^{init}, g + \Delta g^{init}))$.
2: Apply Algorithm 3 with displacement structure $\mathrm{Syl}_\partial(\Delta f, \Delta g)$, $\mathrm{Syl}_\partial(f + \Delta f^{init}, g + \Delta g^{init})$ and the extracted kernel vector as an initial guess.
3: Return the locally optimal $\Delta f$ and $\Delta g$ or an indication of failure.
```

Note that one can also use Algorithm 1 or Algorithm 2 in lieu of Algorithm 3. This technique does not compute the GCRD, however the techniques summarized earlier can be used. See [51] or [36] for a detailed discussion.

Example 6.6.1. *Consider a pair of differential polynomials*

$$\begin{aligned}f = {} & \left(0.11329\,t^6 + 0.23414\,t^5 + 0.12840\,t^4 + 0.00755\,t^3 + 0.00005\right)\partial^3 \\ & + \left(0.00001\,t^6 + 0.23414\,t^5 + 0.59667\,t^4 + 0.02269\,t^3 - 0.04528\,t^2 - 0.02266\,t + 0.000000367436\right)\partial^2 \\ & + \left(-0.11329\,t^6 + 0.33231\,t^5 - 0.43054\,t^4 - 0.00754\,t^3 - 0.00003\,t^2 - 0.06798\,t + 0.00003\right)\partial \\ & + \left(-0.00001\,t^6 - 0.23414\,t^5 + 0.34741\,t^4 + 0.01510\,t^3 - 0.06799\,t^2 + 0.09064\,t + 0.00004\right)\end{aligned}$$

and

$$\begin{aligned}g = {} & \left(0.01938\,t^4 - 0.03876\,t^3 - 0.07752\,t^2 + 0.03876\,t + 0.05819\right)\partial^3 \\ & + \left(0.13567\,t^4 + 0.23252\,t^3 - 0.07750\,t^2 - 0.34879\,t + 0.29066\right)\partial^2 \\ & + \left(-0.01938\,t^4 + 0.13563\,t^3 + 0.03873\,t^2 + 0.25195\,t - 0.23257\right)\partial \\ & + \left(-0.13562\,t^4 + 0.44570\,t^3 - 0.56198\,t^2 - 0.03874\,t + 0.17439\right).\end{aligned}$$

A nearby pair of differential polynomials with an exact GCRD computed using an unconstrained variant of Newton's method are given as

$$\begin{aligned} f^{init} = {} & \big(0.000000333124\, t^8 + 0.11329\, t^6 + 0.23414\, t^5 + 0.12842\, t^4 + 0.00757\, t^3 + 0.00001\, t^2 + 0.000000833124\big)\, \partial^3 \\ & + \big(0.000000781664\, t^8 + 0.00001\, t^6 + 0.23415\, t^5 + 0.59666\, t^4 + 0.02268\, t^3 - 0.04529\, t^2 - 0.02267\, t\big)\, \partial^2 \\ & + \big(-0.000000333148\, t^8 - 0.000000566435\, t^7 - 0.11329\, t^6 + 0.33232\, t^5 - 0.43054\, t^4 - 0.00753\, t^3 - 0.00001\, t^2 - 0.06797\, t - 0.00001\big) \partial \\ & + \big(-0.00000078172\, t^8 - 0.00001\, t^6 - 0.23413\, t^5 + 0.34741\, t^4 + 0.01510\, t^3 - 0.06798\, t^2 + 0.09064\, t + 0.00001\big) \end{aligned}$$

and

$$\begin{aligned} g^{init} = {} & \big(0.00001\, t^5 + 0.01938\, t^4 - 0.03876\, t^3 - 0.07751\, t^2 + 0.03875\, t + 0.05814\big)\, \partial^3 \\ & + \big(0.13566\, t^4 + 0.23253\, t^3 - 0.07751\, t^2 - 0.34879\, t + 0.29067\big)\, \partial^2 \\ & + \big(0.00001\, t^5 - 0.01939\, t^4 + 0.13563\, t^3 + 0.03873\, t^2 + 0.25195\, t - 0.23256\big)\, \partial \\ & + \big(-0.00001\, t^5 - 0.13563\, t^4 + 0.44569\, t^3 - 0.56198\, t^2 + 0.17441 - 0.03874\, t\big)\, . \end{aligned}$$

We seek to minimize the objective function

$$\min \|\Delta f\|_2^2 + \|\Delta g\|_2^2 \quad \textit{subject to} \quad \operatorname{rank}(\operatorname{Syl}_\partial(f+\Delta f, g+\Delta g)) \leq \deg_\partial(f) + \deg_\partial(g) - 1,$$

where Δf and Δg do not increase the entries of f and g. This problem is the same as computing a nearby pair of differential polynomials with an exact GCRD, except this is done by computing a nearby rank deficient matrix polynomial with a linear structure[2]*. This structure is different than previous ones considered, in that $\|\operatorname{Syl}_\partial(\Delta f, \Delta g)\|_F \neq \|\Delta f\|_F^2 + \|\Delta g\|_F^2$, as some coefficients of f and g are weighted.*

Since f^{init} and g^{init} do not have the same degree structure as f and g we fit initial guesses for Δf and Δg using linear-least squares, then project to a rank deficient differential Sylvester matrix using a variant of Newton's method. The initial guess satisfies $\|\Delta f^{init}\|_2^2 + \|\Delta g^{init}\|_2^2 \approx 0.118542385435895 \times 10^{-7}$.

Using Algorithm 3 we solve $\nabla L = 0$ to roughly 16 digits of precision after five iterations. The quality of the initial guess is $\|\nabla L(x^{init}, \lambda^{init})\|_2 \approx 2.17754343614046737 \times 10^{-4}$. The computed point is a local minimizer satisfying second-order necessary conditions[3]*.*

The computed solution $\operatorname{Syl}_\partial(f + \Delta f^{opt}, g + \Delta g^{opt})$ is rank deficient by two, so the approximate GCRD has degree two. In particular, the nearest rank deficient differential Sylvester

[2]This structure is special because of the falling factorials, some low-order terms have a higher-weight to being perturbed

[3]The computed kernel vector was not minimally degree embedded. The kernel was normalized so that all kernel vectors were $\mathbb{R}(t)$ multiples of the same vector, which is why quadratic convergence was observed.

matrix did not have rank $\deg_\partial f + \deg_\partial g - 1$*, which differs from generic instances of matrix polynomials.*

The computed Δf^{opt} *and* Δg^{opt} *satisfy* $\|\Delta f^{opt}\|_2^2 + \|\Delta g^{opt}\|_2^2 \approx 0.112995220147126 \times 10^{-7}$*, which is a minor improvement over the unconstrained variant. The main difference here is that* $f + \Delta f^{opt}$ *and* $g + \Delta g^{opt}$ *do not perturb non-zero coefficients of* f *and* g*, where the previously computed solution did.*

We note that $10^4 \times \Delta f^{opt}$ *is approximately*

$$\begin{aligned}
&\big(0.085059\,t^6 - 0.035717\,t^5 + 0.15180\,t^4 + 0.12435\,t^3 - 0.48265\big)\,\partial^3 \\
&+ \big(-0.073094\,t^6 + 0.018895\,t^5 - 0.043636\,t^4 - 0.028647\,t^3 - 0.19383\,t^2 - 0.16530\,t + 0.090054\big)\,\partial^2 \\
&+ \big(-0.016301\,t^6 + 0.052931\,t^5 + 0.036604\,t^4 + 0.057169\,t^3 + 0.15617\,t^2 + 0.16275\,t - 0.34846\big)\,\partial \\
&+ \big(0.074632\,t^6 + 0.076493\,t^5 + 0.046422\,t^4 + 0.034308\,t^3 + 0.057546\,t^2 + 0.011222\,t - 0.37588\big)\,,
\end{aligned}$$

and that $10^4 \times \Delta g^{opt}$ *is approximately*

$$\begin{aligned}
&\big(-0.0043637\,t^4 - 0.011309\,t^3 + 0.15654\,t^2 - 0.070927\,t - 0.49168\big)\,\partial^3 \\
&+ \big(-0.16195\,t^4 + 0.060975\,t^3 - 0.036591\,t^2 - 0.043150\,t + 0.11584\big)\,\partial^2 \\
&+ \big(0.017964\,t^4 - 0.037126\,t^3 + 0.024188\,t^2 + 0.054643\,t + 0.10390\big)\,\partial \\
&+ \big(-0.25546\,t^4 - 0.12419\,t^3 - 0.030284\,t^2 + 0.035474\,t + 0.11033\big)\,.
\end{aligned}$$

6.7 Conclusion

In this chapter we have formally defined an approximate GCRD problem for differential polynomials, and given an approach to a robust numerical solution. We have seen that, under reasonable assumptions the approximate GCRD problem is well posed. In particular, we show that Newton iteration will converge to an optimal solution if the residual is sufficiently small. We employ the earlier results in [35], analogous to SVD-based approximate GCD methods like [21], to compute a reasonable initial estimate for the Newton iteration. The refinements of the technique of [35] in [36, 51] is discussed in the light of some technical improvements to the theory. We conclude with an application of the theory of Chapter 3 to provide a factor-free technique to compute nearby differential polynomials with a non-trivial GCRD.

The results were presented for real differential polynomials, however the results generalize in a very straight forward way to the instance of complex differential polynomials. This holds more generally than differential polynomials, and a particular example to consider is the shift operator, commonly associated with linear difference equations.

The differential polynomials defined in this chapter are a special case of more general Ore polynomials, which have broader application in the solution of differential and difference equations. In particular, we could potentially apply our methods in the context of q-differentiation (Jackson differentiation) or derivations on exponential polynomials. Ultimately, any Ore structure will have a well-defined Sylvester-like matrix (see, e.g., [42]). However, the numerical properties of different derivations may well be quite difficult or even problematic, and may well introduce poles or other significant sources of numerical instability.

We also hope, the results of this chapter are a foundation for extending the approximate polynomial toolbox to other problems with differential polynomials and more general linear differential operators. Much like approximate GCD, the approximate GCRD is both a stepping stone and a key tool towards operations like approximate factorization and (functional) solution of differential polynomials. More immediately, computation of an approximate GCRD enables computation of a corresponding approximate LCLM, and multiple GCRD's, and to multiple differential variables (i.e., iterated Ore polynomials), which provide an effective method for dealing with linear PDEs.

Chapter 7

Conclusion

In this thesis we have studied several symbolic-numeric problems with a strong emphasis on optimization problems concerning matrix polynomials and their properties in various linear algebra problems. The problems of computing a matrix polynomial with a prescribed spectral structure and kernel are studied in detail revealing insights about local stability properties of optimization algorithms and the existence and uniqueness of solutions. These insights into the local geometric properties of these problems are leveraged to develop algorithms with second-order (quadratic) local convergence.

Important problems that appear as auxiliary problems are computing the determinant and the adjoint of a matrix polynomial. These two operators are studied in detail in the context of matrix polynomials in a floating point enviroment with closed form expressions for their derivatives and condition number results for the problem. We use the determinant and adjoint matrix explicitly to study the spectral structure of matrix polynomials and generalize the theory via the computation of several minors.

An application of the theory developed is in computing a nearby pair of differential polynomials with a non-trivial GC(R)D. The non-commutative algebra problem is turned into a commutative symbolic-numeric optimization problem that consists of finding a nearby matrix polynomial with a special structure that is rank deficient.

7.1 Why the Forms of Smith and Kronecker?

A natural question is why we chose to study the Canonical forms of Smith and Kronecker. Kronecker's canonical form is an obvious choice in that it describes all of the invariants of a matrix polynomial. Smith's canonical form is useful for describing the finite spectral structure and the kernel, and can be used to obtain the infinite spectral structure as well via reversion. For practical purposes, linearizing a pencil and computing Kronecker's canonical form or computing the Smith form yield the same information.

When factoring matrix polynomials, changing the spectral structure or eigenvalues in the factors is undesirable because each factor needs to be analyzed separately. This is why factorizations based on unimodular transformations are desirable, as the unimodular factor is largely irrelevant in the computation. The other factor has the same invariants as the input. For example, Hermite's canonical form is desirable because it is triangular. Unfortunately, computing it amounts to some form of Gaussian-like elimination without pivoting. Hermite's canonical form requires that $\mathcal{UA} = \mathcal{H} \in \mathsf{R}[t]^{m\times n}$ where $\mathcal{U}$ is unimodular and $\mathcal{H}$ is triangular and the pivot entries are monic and have degree larger than the other entries in their column.

If we consider

$$\mathcal{A} = \begin{pmatrix} 0 & t^2-t & 0 \\ -1 & 0 & 1 \\ 0 & -t^2+t & 1 \end{pmatrix} \text{ then the Hermite form of } \mathcal{A} \text{ is } \mathcal{H} = \begin{pmatrix} 1 & & \\ & t^2-t & \\ & & 1 \end{pmatrix}.$$

If we want the Hermite form to be "non-trivial" in the meaning that the diagonal elements are not of the form $\operatorname{diag}(1,\dots,1,\det(\mathcal{A})/\operatorname{lcoeff}(\det(\mathcal{A})),1,\dots,1)$ then a natural question to ask is "how far away is $\mathcal{A}$ from a matrix polynomial $\mathcal{A}+\Delta\mathcal{A}$ such that $\mathcal{A}+\Delta\mathcal{A}$ has a non-trivial Hermite form"? The answer to this question is zero in this instance. If we consider

$$\Delta\mathcal{A} = \begin{pmatrix} 0 & -\varepsilon t & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \text{ then the Hermite form of } \mathcal{A}+\Delta\mathcal{A} \text{ is } \begin{pmatrix} 1 & 0 & -1 \\ & t & \dfrac{-1}{\varepsilon} \\ & & t-1-\varepsilon \end{pmatrix},$$

which is non-trivial for any $\varepsilon \neq 0$. We note that the perturbation did not change the coefficients in any significant way. This example is peculiar, because $\mathcal{A}$ has eigenvalues at infinity with a non-trivial spectral structure, as

$$\operatorname{SNF}(\operatorname{rev}_2(\mathcal{A})) = \begin{pmatrix} 1 & & \\ & t^2 & \\ & & t^2(t-1) \end{pmatrix}.$$

Since the SNF of $\operatorname{rev}_2(\mathcal{A})$ is non-trivial, $\mathcal{A}$ has a non-trivial spectral structure for the eigenvalue at infinity. Indeed, the Hermite form is ill-equipped for such instances of problems and not generally suitable for numerical analysis. This is not surprising, as the requirements imposed upon the Hermite form mean that computing it amounts to some type of Gaussian elimination without pivoting. While Hermite's form is useful in exact arithmetic and in theory, it is not quite as practical from the perspective of a numerical analyst.

Now one could consider other canonical forms as well, but at some level if they are not the forms of Smith or Kronecker, they should be unimodularily equivalent to them to retain useful information about the problem at hand.

The Smith form is structured as an approximate GCD type problem with respect to the minors, and approximate GCD is known to have a radius of stability. As we have seen, the Smith form inherits this stability. Importantly, other canonical forms do not inherit these numerical stability properties. For practical purposes, computing approximations to the unimodular multipliers is not that hard once the Smith form is known. The Smith form can be determined to relatively high accuracy by inferring several blocks of Kronecker's canonical form.

7.2 Faster Algorithms

A possible point of contention in this work is that the algorithms implemented may seem relatively slow, especially when compared to their exact variants. In this work second-order optimization techniques were considered, meaning we ignored first-order gradient descent type techniques for the most part. For many problems, first-order methods will have a cheaper per-iteration cost in exchange for a reduced rate of convergence. Given that we proved several second-order methods will converge quadratically on several instances of problems, this implies that first-order methods will converge linearly.

In general, we treated the KKT matrix as a dense matrix, which may not always be the case. For example, the KKT matrix appearing in our proposed formulation of the rank factorization in Chapter 3.4 is highly structured and closed-form expressions for the quantities exist using the Kronecker product and the Commutation matrix. The abysmal worst-case runtime of $O(n^{12}d^6)$ FLOPs can be improved by several orders of magnitude by exploiting structure via computing entries as needed in conjunction with an iterative linear system solver, since matrix-vector products may be computed efficiently.

Alternatively, the rank factorization can be improved by working on several images evaluated at suitable complex roots of unity. The run time is $O(n^9d^3)$ FLOPs in exchange for some careful considerations in an implementation. In general, the Hessian of the evaluated rank factorization has $O(n^4d)$ non-zero entries (the Jacobian of the constraints has $O(n^4d)$

non-zero entries while the projected Hessian has $O(n^3d)$ non-zero entries) but dimension $O(n^3d) \times O(n^3d)$. The constraints from the rank factorization are highly structured, and if a matrix-vector product can be performed faster, then this suggests a fast iterative solver could compute a Newton step in $O(n^6d^2)$ FLOPs, which is quadratic in the output size of the problem. This should be possible given the Kronecker-product structure of the constraints.

Several of these problems can also be approached via matrix-free methods to solve them. Conjugate gradient methods are well suited as a solver for a regularized Gauss-Newton type technique (by solving normal equations). Exploiting the sparsity in the Hessian of the Lagrangian and using automatic differentiation, several problems can have their per-iteration cost reduced by a factor of $O(d)$ or $O(n)$ or more. For example, the linearization technique for lower McCoy rank approximations in Chapter 5.5 can be improved almost trivially by a factor of $O(d)$ FLOPs by exploiting the sparsity of the companion linearization.

7.3 Truly Global Convergence

This work views the problems considered from a perspective that is almost entirely local. Despite Lagrange multipliers existing for most formulations of the problems encountered, the proofs relied on using local information at some level. In the context of each problem, global convergence may be interpreted in several different ways.

When considering the instance of lower rank approximations of matrix polynomials, an algorithm with global convergence to a point satisfying second-order conditions is highly desirable and makes sense given Theorem 3.3.4. While we did not investigate global convergence explicitly, the rank factorization algorithm can be iterated (with possible re-initialization if the factors are found to be rank deficient) to a point satisfying the second-order necessary condition (recall that the projected Hessian $\nabla^2_{xx}L$ is always rank deficient, but the rank is locally constant after the factors are compressed). In the context of affinely structured matrices, feasible points may not even exist (which is a decidable, but NP-hard problem), of which there is no solution to compute.

In the context of computing a nearby matrix polynomial with a prescribed spectral structure, global convergence may not even make sense, given that there are often irregular solutions. Furthermore, the discontinuity between the eigenvalues of a matrix polynomial and its entries make this problem incredibly challenging from a global perspective. The eigenvalues are typically *locally continuous* in the entries of the matrix polynomial. Global convergence for a problem that is only well-behaved locally may not always be reasonable.

7.4 NP Hardness of Problems

A natural question to ask is whether the nearest singular matrix polynomial problem and approximate Smith form problems are intractable when we consider unstructured perturbations (each coefficient matrix is completely unstructured and may be perturbed in any manner). Indeed, all of the hardness results mentioned assume that the perturbation structure is affine, and this hardness comes from determining if a feasible point exists. Both of these problems are non-convex, but this alone is not enough to show that they are difficult to solve. For example, the singular value decomposition solves a non-convex problem. In this work, we have only stated that very general versions of these problems are NP hard to approximate solutions to.

7.4.1 Nearest Singular Matrix Polynomial

Not all problems involving Toeplitz-block matrices are intractable. The approximate greatest common divisor problem of two polynomials can be solved in a polynomial amount of time [72] using a variant of variable projection [45, 46]. The technique of Karmarkar and Lakshman was successful because the solution was known to exist inside of a "box" and the fact that a degree one solution always exists to the complex valued approximate greatest common divisor problem, which reduces the number of variables required to a finite number for problems of arbitrary degree. Such a technique is unlikely to find success, as the variable projection for the nearest singular matrix polynomial problem is unable to reduce the number of variables by exploiting special low dimension properties of solutions. This results in minimizing a rational function with a large number of variables and of large degree, which is not necessarily easier than the original problem.

The work of Lasserre and Henrion [54] show that the sum of squares hierarchy often solves (approximates to sufficiently high precision) structured lower rank approximation problems when feasible points exist. Indeed, when feasible points to the problems considered exist, most of the hardness results are no longer applicable. This is a sizable gap between the theory and practical results. It could very well be that most instances of computing the nearest singular matrix polynomial are easy, and that only some instances are difficult.

We know by Theorem 3.6.2 that rank one perturbations are usually not optimal, since they will not satisfy the KKT conditions in most instances. This is in contrast to unstructured scalar matrices. While this result is negative, we do know that regularity conditions and second-order sufficiency hold in some formulation of the nearest singular matrix polynomial problem. These local conditions are part of the reason why the hierarchy of Lasserre [76] is able to compute solutions to several problems [88].

Investigating the gap between the theory and implementations is a problem left as future work, although the author conjectures that the problem is intractable.

7.4.2 Approximate Smith Normal Form and Related Problems

The approximate Smith normal form problem can be formulated in a manner similar to structured lower rank approximation (see Section 5.5) via companion (or other) linearizations. Naturally, one would assume that the difficulty would be comparable or harder than computing the nearest singular matrix polynomial. In particular, there exist instances where the nearest matrix polynomial with an interesting Smith form is singular.

These similarities aside, the approximate Smith form problem has the issue of solutions at infinity. If the iterates converge towards an eigenvalue of large magnitude, it is difficult to ascertain if the eigenvalue is infinite in the algorithms implemented. We know that the problem is decidable by Theorem 5.3.2. This is accomplished by verifying that leading coefficients of sufficiently many minors of sufficiently high order vanish at a critical point.

In some variations of the approximate Smith form where we prescribe entries, we know that it is at least as hard as the nearest singular matrix polynomial problem, because we can prescribe the last invariant factor to be zero.

Investigating the computational hardness of these problems is left as future work, although the author conjectures that the problem is intractable given that several instances are at least as hard as computing the nearest singular matrix polynomial.

References

[1] T. Abatzoglou, J. Mendel, and G. Harada. The constrained total least squares technique and its applications to harmonic superresolution. *IEEE Transactions on Signal Processing*, 39(5):1070–1087, 1991.

[2] S. Abramov, H. Le, and Z. Li. Univariate ore polynomial rings in computer algebra. *J. Math. Sci.*, 131(5):58855903, 2005.

[3] P-A Absil, R. Mahony, and R. Sepulchre. *Optimization algorithms on matrix manifolds.* Princeton University Press, 2009.

[4] S. Ahmad and R. Alam. Pseudospectra, critical points and multiple eigenvalues of matrix polynomials. *Linear Algebra and its Applications*, 430(4):1171–1195, 2009.

[5] B. Beckermann and G. Labahn. A fast and numerically stable euclidean-like algorithm for detecting relatively prime numerical polynomials. *Journal of Symbolic Computation*, 26(6):691–714, 1998.

[6] B. Beckermann and G. Labahn. When are two numerical polynomials relatively prime? *Journal of Symbolic Computation*, 26:677–689, 1998.

[7] Th. Beelen and P. Van Dooren. An improved algorithm for the computation of Kronecker's canonical form of a singular pencil. *Linear Algebra and its Applications*, 105:9–65, 1988.

[8] Th. Beelen and G.W. Veltkamp. Numerical computation of a coprime factorization of a transfer function matrix. *Systems & Control Letters*, 9(4):281–288, 1987.

[9] J. Bell, A. Heinle, and V. Levandovskyy. On noncommutative finite factorization domains. *Trans. AMS*, 369:2675–2695, 2017.

[10] D. Bertsekas. *Nonlinear programming.* Athena Scientific, USA, 1999.

[11] S. Beslin. Cofactor matrices. *Linear algebra and its applications*, 165:45–52, 1992.

[12] R. Bhatia and T. Jain. Higher order derivatives and perturbation bounds for determinants. *Linear Algebra and its Applications*, 431(11):2102–2108, 2009.

[13] B. Botting, M. Giesbrecht, and J.P. May. Using the Riemannian SVD for problems in approximate algebra. In *Proc. Workshop on Symbolic-Numeric Comp.*, pages 209–219, 2005.

[14] R. P. Braatz, P. M. Young, J. C. Doyle, and M. Morati. Computational complexity of mu calculation. *IEEE Transactions on Automatic Control*, 39(5):1000–1002, 1994.

[15] M. Bronstein and M. Petkovšek. On Ore rings, linear operators and factorisation. *Programmirovanie*, 20:27–45, 1994.

[16] M. Bronstein and M. Petkovšek. An introduction to pseudo-linear algebra. *Theoretical Computer Science*, 1996.

[17] R. Byers, C. He, and V. Mehrmann. Where is the nearest non-regular pencil? *Linear algebra and its applications*, 285(1):81–105, 1998.

[18] R. Byers and N. Nichols. On the stability radius of a generalized state-space system. *Linear Algebra and Its Applications*, 188:113–134, 1993.

[19] J. Cadzow. Signal enhancement – a composite property mapping algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(1):49–62, 1988.

[20] E. J. Candès and T. Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080, 2010.

[21] R. M. Corless, P. Gianni, B. Trager, and S. Watt. The singular value decomposition for polynomial systems. In *Proc. Int. Symp. Symb. Algebraic Computation (ISSAC)*, pages 96–103, 1995.

[22] R. M. Corless, S. M. Watt, and L. Zhi. QR Factoring to Compute the GCD of Univariate Approximate Polynomials. *IEEE Transactions on Signal Processing*, 52(12), 2004.

[23] B. De Moor. Structured total least squares and L_2 approximation problems. *Linear algebra and its applications*, 188:163–205, 1993.

[24] B. De Moor. Total least squares for affinely structured matrices and the noisy realization problem. *IEEE Transactions on Signal Processing*, 42(11):3104–3113, 1994.

[25] B. De Moor. The Riemannian singular value decomposition. In *Signal Processing, III: Algorithms, Architectures and Applications*, pages 61–78. Elsevier, 1995.

[26] J. Demmel and B. Kågström. The generalized Schur decomposition of an arbitrary pencil A-λB – robust software with error bounds and applications. part i: theory and algorithms. *ACM Transactions on Mathematical Software (TOMS)*, 19(2):160–174, 1993.

[27] J. Demmel and B. Kågström. The generalized Schur decomposition of an arbitrary pencil A-λB – robust software with error bounds and applications. part ii: software and applications. *ACM Transactions on Mathematical Software (TOMS)*, 19(2):175–201, 1993.

[28] J. W. Demmel and A. Edelman. The dimension of matrices (matrix pencils) with given jordan (Kronecker) canonical forms. *Linear Algebra and its Applications*, 230:61–87, 1995.

[29] A. Edelman, E. Elmroth, and B. Kågström. A geometric approach to perturbation theory of matrices and matrix pencils. part I: Versal deformations. *SIAM Journal on Matrix Analysis and Applications*, 18(3):653–692, 1997.

[30] A. Edelman, E. Elmroth, and B. Kågström. A geometric approach to perturbation theory of matrices and matrix pencils. part II: A stratification-enhanced staircase algorithm. *SIAM Journal on Matrix Analysis and Applications*, 20(3):667–699, 1999.

[31] I. Z. Emiris, A. Galligo, and H. Lombardi. Certified approximate univariate gcds. *Journal of Pure and Applied Algebra*, 117–118:229–251, 1997.

[32] J-Y. Fan and Y-X. Yuan. On the quadratic convergence of the Levenberg-Marquardt method without nonsingularity assumption. *Computing*, 74(1):23–39, 2005.

[33] S. Fatouros and N. Karcanias. Resultant properties of gcd of many polynomials and a factorization representation of gcd. *International Journal of Control*, 76(16):1666–1683, 2003.

[34] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, New York, NY, USA, 3 edition, 2013.

[35] M. Giesbrecht and J. Haraldson. Computing GCRDs of approximate differential polynomials. In *Proc. Workshop on Symbolic-Numeric Computing (SNC'14)*, pages 78–87, 2014.

[36] M. Giesbrecht, J. Haraldson, and E. Kaltofen. Computing approximate greatest common right divisors of differential polynomials. *To Appear in Foundations of Computational Mathematics*.

[37] M. Giesbrecht, J. Haraldson, and G. Labahn. Computing lower rank approximations of matrix polynomials. *To Appear in Journal of Symbolic Computation*, 2017.

[38] M. Giesbrecht, J. Haraldson, and G. Labahn. Computing the nearest rank-deficient matrix polynomial. In *Proc. ACM on International Symposium on Symbolic and Algebraic Computation (ISSAC'17)*, pages 181–188, Kaiserslautern, Germany, 2017.

[39] M. Giesbrecht, J. Haraldson, and G. Labahn. Computing nearby non-trivial Smith forms. In *Proceedings of International Symposium on Symbolic and Algebraic Computation (ISSAC'18)*, pages 159–166, New York, USA, 2018.

[40] M. Giesbrecht, J. Haraldson, and G. Labahn. Computing nearby non-trivial Smith forms. *Submitted to Journal of Symbolic Computation*, 2018.

[41] M. Giesbrecht, A. Heinle, and V. Levandovskyy. Factoring linear partial differential operators in n variables. *Journal of Symbolic Computation*, 75:127–148, 2016.

[42] M. Giesbrecht and M. Kim. Computing the Hermite form of a matrix of Ore polynomials. *Journal of Algebra*, 376:341–362, 2013.

[43] N. Gillis and F. Glineur. Low-rank matrix approximation with weights or missing data is NP-hard. *SIAM Journal on Matrix Analysis and Applications*, 32(4):1149–1165, 2011.

[44] I. Gohberg, P. Lancaster, and L. Rodman. *Matrix polynomials*. SIAM, USA, 2009.

[45] G. Golub and V. Pereyra. The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM Journal on numerical analysis*, 10(2):413–432, 1973.

[46] G. Golub and V. Pereyra. Separable nonlinear least squares: the variable projection method and its applications. *Inverse problems*, 19(2):R1, 2003.

[47] G. Golub and C. Van Loan. *Matrix Computations*, volume 3. Johns Hopkins University Press, USA, 2012.

[48] D. Grigor'ev. Complexity of factoring and calculating the GCD of linear ordinary differential operators. *Journal of Symbolic Computation*, 10(1):7–37, 1990.

[49] L. Grippo and M. Sciandrone. On the convergence of block nonlinear Gauss-Seidel method under convex constraints. *Operations Research Letters*, (26).

[50] N. Guglielmi, C. Lubich, and V. Mehrmann. On the nearest singular matrix pencil. *Preprint*, 12, 2016.

[51] J. Haraldson. Computing Approximate GCRDs of Differential Polynomials. Master's thesis, University of Waterloo, 2015.

[52] J. Haraldson. Numerical differentiation and computation of matrix polynomial adjoint matrices. 2018.

[53] A. Heinle and V. Levandovskyy. A factorization algorithm for g-algebras and applications. In *Proc. International Symposium on Symbolic and Algebraic Computation (ISSAC 16)*, pages 263–270. ACM Press, 2016.

[54] D. Henrion and J-B. Lasserre. Convergent relaxations of polynomial matrix inequalities and static output feedback. *IEEE Transactions on Automatic Control*, 51(2):192–202, 2006.

[55] D. Henrion and M. Šebek. Numerical methods for polynomial matrix rank evaluation. *IFAC Proceedings Volumes*, 31(18):369–374, 1998.

[56] D. Henrion and M. Šebek. Reliable numerical methods for polynomial matrix triangularization. *IEEE Transactions on Automatic Control*, 44(3):497–508, 1999.

[57] N. J. Higham. *Accuracy and stability of numerical algorithms*, volume 80. SIAM, 2002.

[58] N. J. Higham and S. D. Relton. Higher order Freéchet derivatives of matrix functions and the level-2 condition number. *SIAM Journal on Matrix Analysis and Applications*, 35(3):1019–1037, 2014.

[59] A. Hjørungnes and D. Gesbert. Complex-valued matrix differentiation: Techniques and key results. *IEEE Transactions on Signal Processing*, 55(6):2740–2746, 2007.

[60] A. J. Hoffman. On approximate solutions of systems of linear inequalities. *Journal of Research of the National Bureau of Standards*, 49(4), 1952.

[61] R. A. Horn and C. R. Johnson. *Matrix Analysis.* Cambridge University Press, 1985.

[62] M. Hromcik and M. Sebek. New algorithm for polynomial matrix determinant based on fft. In *Control Conference (ECC), 1999 European*, pages 4173–4177. IEEE, 1999.

[63] M. Hromcik and M. Sebek. Numerical and symbolic computation of polynomial matrix determinant. In *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, volume 2, pages 1887–1888. IEEE, 1999.

[64] I. Ipsen and R. Rehman. Perturbation bounds for determinants and characteristic polynomials. *SIAM Journal on Matrix Analysis and Applications*, 30(2):762–776, 2008.

[65] T. Kailath. *Linear systems*, volume 156. Prentice-Hall, USA, 1980.

[66] E Kaltofen, J. P. May, Z. Yang, and L. Zhi. Approximate factorization of multivariate polynomials using singular value decomposition. *Journal of Symbolic Computation*, 43(5):359–376, 2008.

[67] E. Kaltofen and A. Storjohann. The complexity of computational problems in exact linear algebra. In *Encyclopedia of Applied and Computational Mathematics*, pages 227–233. Springer, Germany, 2015.

[68] E. Kaltofen, Z. Yang, and L. Zhi. Structured low rank approximation of a sylvester matrix. In *Proc. Int. Workshop on Symbolic-Numeric Computation (SNC 2005)*, pages 188–201, 2005.

[69] E. Kaltofen, Z. Yang, and L. Zhi. Approximate greatest common divisors of several polynomials with linearly constrained coefficients and singular polynomials. In *Proc. 2006 Int. Symp. Symbolic Algebraic Comput.*, pages 169–176. ACM Press, 2006.

[70] E. Kaltofen, Z. Yang, and L. Zhi. Approximate greatest common divisors of several polynomials with linearly constrained coefficients and singular polynomials. Unpublished manuscript, 2007.

[71] E. Kaltofen, Z. Yang, and L. Zhi. Structured low rank approximation of a sylvester matrix. In *Symbolic-Numeric Computation*, Trends in Mathematics, pages 69–83, Basel, Switzerland, 2007. Birkhäuser Verlag.

[72] N. Karmarkar and Y. N. Lakshman. Approximate polynomial greatest common divisors and nearest singular polynomials. In *Proc. International Symposium on Symbolic and Algebraic Computation (ISSAC'96)*, pages 35–39, Zurich, Switzerland, 1996. ACM Press.

[73] N. Karmarkar and Y. N. Lakshman. On approximate GCDs of univariate polynomials. *Journal of Symbolic Computation*, 26(6):653–666, 1998.

[74] D. Kressner and M. Voigt. Distance problems for linear dynamical systems. In *Numerical Algebra, Matrix Theory, Differential-Algebraic Equations and Control Theory*, pages 559–583. Springer, 2015.

[75] M. A. Laidacker. Another Theorem Relating Sylvester's Matrix and the Greatest Common Divisor. *Mathematics Magazine*, 42(3):pp. 126–128, 1969.

[76] J-B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001.

[77] B. Li, Z. Yang, and L. Zhi. Fast low rank approximation of a Sylvester matrix by structure total least norm. *J. Japan Soc. Symbolic and Algebraic Computation*, 11:165–174, 2005.

[78] Z. Li. A subresultant theory for Ore polynomials with applications. In *Proc. International Symposium on Symbolic and Algebraic Computation (ISSAC'98)*, pages 132–139. ACM, 1998.

[79] Z. Li and I. Nemes. A modular algorithm for computing greatest common right divisors of Ore polynomials. In *Proc. International Symposium on Symbolic and Algebraic Computation (ISSAC'97)*, pages 282–289, 1997.

[80] O.P. Lossers. Solution to problem 73-17: A Hadamard-type bound on the coefficients of a determinant of polynomials. *SIAM review*, 16(3):394–395, 1974.

[81] D. S. Mackey, N. Mackey, C. Mehl, and V. Mehrmann. Structured polynomial eigenvalue problems: Good vibrations from good linearizations. *SIAM Journal on Matrix Analysis and Applications*, 28(4):1029–1051, 2006.

[82] D. S. Mackey, N. Mackey, C. Mehl, and V. Mehrmann. Vector spaces of linearizations for matrix polynomials. *SIAM Journal on Matrix Analysis and Applications*, 28(4):971–1004, 2006.

[83] J. Magnus and H. Neudecker. The commutation matrix: some properties and applications. *The Annals of Statistics*, pages 381–394, 1979.

[84] J. Magnus and H. Neudecker. *Matrix differential calculus with applications in statistics and econometrics*. Wiley, 1988.

[85] I. Markovsky. Structured low-rank approximation and its applications. *Automatica*, 44(4):891–909, 2008.

[86] I. Markovsky. *Low rank approximation: algorithms, implementation, applications*. Springer Science & Business Media, 2011.

[87] J. Moré. The Levenberg-Marquardt algorithm: implementation and theory. In *Numerical analysis*, pages 105–116. Springer, 1978.

[88] J. Nie. Optimality conditions and finite convergence of lasserre's sos hierarchy. *Mathematical Programming*, 146(1-2):97–121, 2014.

[89] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2006.

[90] O. Ore. Theory of non-commutative polynomials. *Annals of Mathematics. Second Series*, 34:480–508, 1933.

[91] S. Poljak and J. Rohn. Checking robust nonsingularity is NP-hard. *Mathematics of Control, Signals, and Systems (MCSS)*, 6(1):1–9, 1993.

[92] B. Rosen, H. Park, and J. Glick. Total least norm formulation and solution for structured problems. *SIAM Journal on Matrix Analysis and Applications*, 17(1):110–126, 1996.

[93] B. Rosen, H. Park, and J. Glick. Structured total least norm for nonlinear problems. *SIAM Journal on Matrix Analysis and Applications*, 20(1):14–30, 1998.

[94] B. Salvy and P. Zimmermann. Gfun: a Maple package for the manipulation of generating and holonomic functions in one variable. *ACM Trans. Math. Software*, 20(2):163–177, 1994.

[95] T. Sasaki and M. Sasaki. Polynomial remainder sequence and approximate GCD. *ACM SIGSAM Bulletin*, 31:4–10, 1997.

[96] A. Schönhage. Quasi-GCD computations. *J. Complexity*, 1:118–137, 1985.

[97] É. Schost and P.-J. Spaenlehauer. A quadratically convergent algorithm for structured low-rank approximation. *Foundations of Computational Mathematics*, 16(2):457–492, 2016.

[98] N. Srebro and T. Jaakkola. Weighted low-rank approximations. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 720–727, 2003.

[99] G. Stewart. Perturbation theory for rectangular matrix pencils. *Linear algebra and its applications*, 208:297–301, 1994.

[100] G. W. Stewart. On the adjugate matrix. *Linear Algebra and its Applications*, 283(1-3):151–164, 1998.

[101] A. Storjohann. On the complexity of inverting integer and polynomial matrices. *computational complexity*, 24(4):777–821, 2015.

[102] A. Storjohann and G. Villard. Computing the rank and a small nullspace basis of a polynomial matrix. In *Proc. Int. Symp. Symb. Alg. Comp. (ISSAC)*, pages 309–316. ACM Press, 2005.

[103] P. Van Dooren. The computation of Kronecker's canonical form of a singular pencil. *Linear Algebra and its Applications*, 27:103–140, 1979.

[104] P. Van Dooren and P. Dewilde. The eigenstructure of an arbitrary polynomial matrix: computational aspects. *Linear Algebra and its Applications*, 50:545–579, 1983.

[105] P. Van Dooren, P. Dewilde, and J. Vandewalle. On the determination of the Smith-MacMillan form of a rational matrix from its Laurent expansion. *IEEE Transactions on Circuits and systems*, 26(3):180–189, 1979.

[106] A. Vardulakis and P. Stoyle. Generalized resultant theorem. *IMA Journal of Applied Mathematics*, 22(3):331–335, 1978.

[107] S. A. Vavasis. On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization*, 20(3):1364–1377, 2009.

[108] Y-X. Wang and Y-J Zhang. Nonnegative matrix factorization: A comprehensive review. *IEEE Transactions on Knowledge and Data Engineering*, 25(6):1336–1353, 2013.

[109] S. Wright. An algorithm for degenerate nonlinear programming with rapid local convergence. *SIAM Journal on Optimization*, 15(3):673–696, 2005.

[110] N. Yamashita and M. Fukushima. On the rate of convergence of the Levenberg-Marquardt method. In *Topics Num. Analysis*, pages 239–249. Springer, 2001.

[111] Z. Zeng. The approximate gcd of inexact polynomials. part 1: a univariate algorithm. *preprint*, 2004.

[112] Z. Zeng and B. H. Dayton. The approximate GCD of inexact polynomials. In *Proc. International Symposium on Symbolic and Algebraic Computation (ISSAC'04)*, pages 320–327, Santander, Spain, 2004.

[113] L. Zhi. Numerical optimization in hybrid symbolic-numeric computation. In *Proc. 2007 International Workshop on Symbolic-Numeric Computation*, pages 33–35, 2007.

[114] J.C. Zúniga Anaya and D. Henrion. An improved Toeplitz algorithm for polynomial matrix null-space computation. *Applied Mathematics and Computation*, 207(1):256–272, 2009.