Computing Approximate GCRDs of Differential Polynomials

by

Joseph Haraldson

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2015

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

We generalize the approximate greatest common divisor problem to the non-commutative, approximate Greatest Common Right Divisor (GCRD) problem of differential polynomials. Algorithms for performing arithmetic on approximate differential polynomials are presented along with certification results and the corresponding number of flops required. Under reasonable assumptions the approximate GCRD problem is well posed. In particular, we show that an approximate GCRD exists under these assumptions and provide counter examples when these assumptions are not satisfied. We introduce algorithms for computing nearby differential polynomials with a GCRD. These differential polynomials are improved through a post-refinement Newton iteration. It is shown that Newton iteration will converge to a unique, optimal solution when the residual is sufficiently small. Furthermore, if our computed solution is not optimal, it is shown that this solution is reasonably close to the optimal solution.

Acknowledgements

I would like to thank my supervisors Mark Giesbrecht and George Labahn for their support, patience and mentorship. I would also like to thank the other members of my examination committee for their time and valuable feedback.

I would also like to thank Erich Kaltofen for his helpful discussions about the approximate greatest common divisor problem and how to generalize important results to the approximate GCRD problem.

Dedication

I dedicate this work to the advancement of mathematics and computer science.

Table of Contents

Chapter 1

Introduction

Computing the exact Greatest Common Right Divisor (GCRD) of differential polynomials is a well studied problem when working over exact input. It corresponds to computing the common solutions of linear ordinary differential equations (ODEs) or indicating that no common solution exists. Differential polynomials with coefficients from $\mathbb{Q}(t)$ provide a theoretical frame work for working with linear ODEs without errors in their coefficients. However, models obtained from measured data or experimental analysis contain machine roundoff errors and noise in the underlying data. These differential polynomials are represented with floating point real numbers, making them *approximate* differential polynomials.

This thesis is concerned with numeric computation of a GCRD of two differential polynomials while working in a numeric environment. Computing a GCRD numerically is an ill-posed problem. Arbitrarily small perturbations in the input will usually indicate that two (or more) differential polynomials are co-prime, when in fact there is an existing exact solution. This problem is exacerbated further by the fact that even if there is a solution, existing algorithms for working with exact input are numerically unstable when used with floating point arithmetic. To overcome these issues we formulate the approximate GCRD problem.

In our contributions to the approximate GCRD problem we:

- provide algorithms for computing an exact primitive GCRD and corresponding cofactors numerically, that can be certified by linear least squares,
- prove that the approximate GCRD problem has a solution and provide conditions for which the solution is locally unique,

- compute a solution to the approximate GCRD problem that is a suitable initial guess for post-refinement Newton iteration,
- demonstrate that the approximate GCRD problem is well-posed under structured perturbations, and
- provide an upper bound on the distance between a computed solution and the optimal solution.

This work uses techniques from convex optimization and polynomial approximate Greatest Common Divisor (GCD). Our contributions may be extended to the more general Ore polynomials. This thesis is a first step on how to approach non-commutative GCD problems in a Euclidean domain with numerical errors.

Some of these results appear in Giesbrecht and Haraldson [9] and a journal publication Giesbrecht, Haraldson and Kaltofen [10] is being prepared.

Chapter 2

Background

This chapter provides an overview of the background necessary to understand the results of this thesis. We first introduce differential polynomials in the exact setting and some essential results [4, 21]. These results are in the form of the more general Ore polynomials. We adapt these results to our particular instance of differential polynomials. We approach differential subresultants based on the work of Li [19], who considers the case of Ore polynomials in an exact setting.

Our approach to computing a GCRD of differential polynomials is to solve a Diophantine equation over the ring of differential polynomials. This corresponds to computing the Bézout coefficients. This is done by formulating a highly structured linear algebra problem with matrices whose entries belong to $\mathbb{R}(t)$ based on the subresultant techniques of Li [20]. We linearize the corresponding Diophantine equation and transform the problem into an equivalent real linear algebra problem. We then use various numerical linear algebra techniques to overcome roundoff errors and coefficient growth that lead to arbitrarily high degrees. The tools from numerical linear algebra that we make extensive use of include the SVD [2, 5, 9, 11], (linear) least squares [3, 11, 14], and other results from convex optimization [3]. These techniques draw upon results from polynomial approximate GCD [5, 14, 16, 23, 28]. A standard technique for computing polynomial approximate GCDs is to linearize polynomial systems. Solutions to these systems are computed using numerical linear algebra and optimization tools.

2.1 Differential Operators in an Exact Setting

This section presents some well known results [4, 21] on differential polynomials. Many important concepts relating to the non-commutativity of differential operators are covered, which are required in later chapters.

Definition 2.1.1. *Let $\mathbb{R}(t)[\partial;']$ define the ring of differential polynomials. $\mathbb{R}(t)[\partial;']$ is the ring of polynomials in ∂ with coefficients from the commutative field of rational functions, under the usual polynomial addition rule along with the non-commutative multiplication rule*

$$\partial y(t) = y(t)\partial + y'(t) \text{ for } y(t) \in \mathbb{R}(t).$$

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 $\mathcal{C}^\infty[\mathbb{R}]$ of infinitely differentiable functions $y(t) : \mathbb{R} \to \mathbb{R}$. In particular, for any $y(t) \in \mathcal{C}^\infty[\mathbb{R}]$,

$$f(\partial) = \sum_{0 \le i \le M} f_i(t)\partial^i \text{ acts on } y(t) \text{ as } \sum_{0 \le i \le M} f_i(t)\frac{d^i}{dt^i}y(t).$$

Remark 2.1.2. *We will maintain a canonical form for all $f \in \mathbb{R}(t)[\partial;']$ by writing*

$$f = \frac{1}{f_{-1}} \sum_{0 \le i \le M} f_i \partial^i,$$

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 ∂.

It will also be necessary to define a partial ordering on differential polynomials. In later chapters we will need to make use of this partial ordering to preserve structure.

Definition 2.1.3. *Let $\delta : \mathbb{R}[t][\partial;'] \to \mathbb{Z}$ be the degree vector function defined as*

$$\delta(f) = (\deg_t f_0, \deg_t f_1, \ldots, \deg_t f_M), \text{ for } f_0, \ldots, f_M \in \mathbb{R}[t].$$

For $f, g \in \mathbb{R}[t][\partial;']$ with $\deg_\partial f = \deg_\partial g = M$ we write

$$\delta(f) < \delta(g) \text{ if } \deg_t f_i \le \deg_t g_i \text{ for } 0 \le i \le M.$$

We define $\delta(f) = \delta(g), \delta(f) < \delta(g), \delta(f) \ge \delta(g)$ and $\delta(f) > \delta(g)$ analogously.

Example 2.1.4. *Let* $f = t^2\partial^3 + (t-3)\partial + 1 \in \mathbb{R}[t][\partial;']$, *then we write*

$$\delta(f) = (0, 1, -\infty, 2).$$

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.

Definition 2.1.5. *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.*

Example 2.1.6. *Let* $f = t\partial - 1$ *and* $g = \partial$. *Both* f *and* g *are primitive. Also,*

$$\begin{aligned} fg &= (t\partial - 1)\partial \\ &= t\partial^2 - \partial \end{aligned}$$

and

$$\begin{aligned} gf &= \partial(t\partial) - \partial \\ &= t\partial^2 + \partial - \partial \\ &= t\partial^2 \end{aligned}$$

This example illustrates that the multiplication between elements from $\mathbb{R}(t)[\partial;']$ *is non-commutative. It also demonstrates that the product of primitive differential polynomials needs not be primitive. Thus we do not have an equivalent of Gauss' Lemma [25, Chapter 6.2].*

Proposition 2.1.7. *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 [21].*

1. $\deg_\partial(fg) = \deg_\partial f + \deg_\partial g$, $\deg_\partial(f+g) \leq \max\{\deg_\partial f, \deg_\partial g\}$.
2. *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).*
3. *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* $\operatorname{gcrd}(f, g) = h$. 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* $\mathbb{R}(t)$.

4. *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* $\operatorname{lclm}(f,g) = \ell$*. The LCLM is unique up to multiplication from a unit belonging to* $\mathbb{R}(t)$*.*

5. $\deg_\partial \operatorname{lclm}(f,g) = \deg_\partial f + \deg_\partial g - \deg_\partial \operatorname{gcrd}(f,g)$*.*

In an algebraic context we can clear denominators and assume without loss of generality that our GCRD belongs to $\mathbb{R}[t][\partial;']$. It is also worth mentioning 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 $\operatorname{gcrd}(f,g) = h$.

Example 2.1.8. *Let* $f = \partial^2, g = t\partial - 1 \in \mathbb{R}[t][\partial;']$*. Then* $\operatorname{gcrd}(f,g) = t\partial - 1$*. We observe that* f *has at least two factorizations,*

$$f = (\partial)(\partial) \text{ and } f = \left(\frac{1}{t}\partial\right)(t\partial - 1),$$

while $g = (1)(t\partial - 1)$*. The co-factors of* f *and* g *are* $\frac{1}{t}\partial$ *and* 1 *respectively.*

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 will make extensive use of the following mapping.

Definition 2.1.9. *For* $f \in \mathbb{R}(t)[\partial;']$ *and* $K > \deg_\partial f$*, 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* $\mathbb{R}(t)^{1\times K}$*.*

Later we find it useful to linearize (differential) polynomials, that is, express them as an element of Euclidean space.

Definition 2.1.10. *For* $p \in \mathbb{R}[t]$ *with degree* d *and* $f \in \mathbb{R}[t][\partial;']$ *with degree* M *we define*

$$\mathbf{p} = (p_0, p_1, \ldots, p_d) \in \mathbb{R}^{1\times(d+1)} \text{ and } \mathbf{f} = (\mathbf{f_0}, \ldots, \mathbf{f_M}) \in \mathbb{R}^{1\times L} \subseteq \mathbb{R}^{1\times(M+1)(d+1)}.$$

The vector $\mathbf{f}$ *with* L *components is the combined coefficient vector of* f*. We will sometimes pad* $\mathbf{f}$ *with zeros so that it belongs to* $\mathbb{R}^{1\times(M+1)(d+1)}$*, however we will not do this unless specifically stated.*

Example 2.1.11. *Consider*

$$f = (f_{2_2}t^2 + f_{2_1}t + f_{2_0})\partial^2 + (f_{1_2}t^2 + f_{1_1}t + f_{1_0})\partial + f_{0_2}t^2 + f_{0_1}t + f_{0_0} \in \mathbb{R}[t][\partial;']$$

then we write

$$\mathbf{f} = (f_{0_0}, f_{0_1}, f_{0_2}, f_{1_0}, f_{1_1}, f_{1_2}, f_{2_0}, f_{2_1}, f_{2_2}).$$

If we consider

$$f = \partial^2 + (t+2)\partial + t^2 \in \mathbb{R}[t][\partial;'],$$

then

$$\mathbf{f} = (\underbrace{0,0,1}_{f_0}, \underbrace{2,1}_{f_1}, \underbrace{1}_{f_2}).$$

If we pad the latter representation with zeros, then

$$\mathbf{f} = (\underbrace{0,0,1}_{f_0}, \underbrace{2,1,0}_{f_1}, \underbrace{1,0,0}_{f_2}).$$

2.1.1 Linear Algebra over $\mathbb{R}(t)$

Since $\mathbb{R}(t)[\partial;']$ is a right (and left) Euclidean domain [21] a GCRD may be computed by solving a Diophantine equation corresponding to the Bézout coefficients. Using the subresultant techniques of Li [19] 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 2.1.12. *Suppose $f, g \in \mathbb{R}(t)[\partial;']$ with $\deg_\partial f = N$ and $\deg_\partial g = M$. Then $\deg_\partial \text{gcrd}(f,g) \geq 1$ 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 form Proposition 2.1.7. □

Using Lemma 2.1.12 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 subresultant method of Li and Nemes [20].

Let $f, g \in \mathbb{R}(t)[\partial;']$ with $\deg_\partial f = M$ and $\deg_\partial g = N$. Then by Lemma 2.1.12 we have that $\deg_\partial \text{gcrd}(f,g) \geq 1$ 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.

Recall that the standard form of $f, g, u, v \in \mathbb{R}[t][\partial;']$ is

$$f = \sum_{0\leq i\leq M} f_i\partial^i,\ g = \sum_{0\leq i\leq N} g_i\partial^i,\ u = \sum_{0\leq i\leq N-1} u_i\partial^i,\ v = \sum_{0\leq i\leq M-1} v_i\partial^i.$$

We now linearize $uf + vg = 0$ over $\mathbb{R}(t)$. First we write

$$\left(\sum_{0\leq i\leq N-1} u_i\partial^i\right)\left(\sum_{0\leq i\leq M} f_i\partial^i\right) + \left(\sum_{0\leq i\leq M-1} v_i\partial^i\right)\left(\sum_{0\leq i\leq N} g_i\partial^i\right) = 0.$$

We expand this expression in terms of the coefficients of u and v as

$$(u_0 f + u_1\partial f + \cdots + u_{N-1}\partial^{N-1} f) + (v_0 g + v_1\partial g + \cdots + v_{M-1}\partial^{M-1} g) = 0.$$

This can be written in vector-matrix product form as

$$(u_0, u_1, \ldots, u_{N-1}) \begin{pmatrix} f \\ \partial f \\ \vdots \\ \partial^{N-1} f \end{pmatrix} + (v_0, v_1, \ldots, v_{M-1}) \begin{pmatrix} g \\ \partial g \\ \vdots \\ \partial^{M-1} f \end{pmatrix} = 0.$$

Combining the vectors and matrices produces

$$(u_0, u_1, \ldots, u_{N-1}, v_0, v_1, \ldots, v_{M-1}) \left(\begin{array}{c} f \\ \partial f \\ \vdots \\ \partial^{N-1} f \\ \hline g \\ \partial g \\ \vdots \\ \partial^{M-1} g \end{array}\right) = 0.$$

Working with the coefficients, we can write

$$(u_0, u_1, \ldots, u_{N-1}, v_0, v_1, \ldots, v_{M-1}) \begin{pmatrix} \Psi_{M+N}(f) \\ \Psi_{M+N}(\partial f) \\ \vdots \\ \Psi_{M+N}(\partial^{N-1} f) \\ \hline \Psi_{M+N}(g) \\ \Psi_{M+N}(\partial g) \\ \vdots \\ \Psi_{M+N}(\partial^{M-1} g) \end{pmatrix} = 0.$$

We note that this is now an entirely commutative problem over $\mathbb{R}(t)$; there are no more ∂'s.

Definition 2.1.13. *The matrix*

$$S = S(f, g) = \begin{pmatrix} \Psi_{M+N}(f) \\ \Psi_{M+N}(\partial f) \\ \vdots \\ \Psi_{M+N}(\partial^{N-1} f) \\ \Psi_{M+N}(g) \\ \Psi_{M+N}(\partial g) \\ \vdots \\ \Psi_{M+N}(\partial^{M-1} g) \end{pmatrix} \in \mathbb{R}(t)^{(M+N)\times(M+N)}$$

is the differential Sylvester matrix of f and g.

Example 2.1.14. *Let*

$$f = \left(f_{1_2} t^2 + f_{1_1} t + f_{1_0}\right) \partial + \left(f_{0_2} t^2 + f_{0_1} t + f_{0_0}\right)$$

and

$$g = \left(g_{2_2} t^2 + g_{2_1} t + g_{2_0}\right) \partial^2 + \left(g_{1_2} t^2 + g_{1_1} t + g_{1_0}\right) \partial + \left(g_{0_2} t^2 + g_{0_1} t + g_{0_0}\right).$$

The corresponding differential Sylvester matrix S is given by

$$\begin{pmatrix} f_{0_2} t^2 + f_{0_1} t + f_{0_0} & f_{1_2} t^2 + f_{1_1} t + f_{1_0} & 0 \\ 2t f_{0_2} + f_{0_1} & f_{0_2} t^2 + f_{0_1} t + 2t f_{1_2} + f_{0_0} + f_{1_1} & f_{1_2} t^2 + f_{1_1} t + f_{1_0} \\ \hline g_{0_2} t^2 + g_{0_1} t + g_{0_0} & g_{1_2} t^2 + g_{1_1} t + g_{1_0} & g_{2_2} t^2 + g_{2_1} t + g_{2_0} \end{pmatrix}.$$

Example 2.1.15. *Let*

$$f = (5t^3 + 2t^2 - 3t)\partial^3 + (-5t^3 + 13t^2 + 9t - 9)\partial^2 + (-9t^2 - 6t + 6)\partial + (-t^2 + 3t - 2)$$

and

$$g = (9t^2 + 4t)\partial^2 + (-8t^2 + 11t + 8)\partial + (-t^2 - 5t - 7).$$

The corresponding differential Sylvester matrix S is given by

$$\left(\begin{array}{ccccc} -t^2+3t-2 & -9t^2-6t+6 & -5t^3+13t^2+9t-9 & 5t^3+2t^2-3t & 0 \\ -2t+3 & -t^2-15t-8 & -24t^2+20t+15 & -5t^3+28t^2+13t-12 & 5t^3+2t^2-3t \\ \hline -t^2-5t-7 & -8t^2+11t+8 & 9t^2+4t & 0 & 0 \\ -2t-5 & -t^2-21t+4 & -8t^2+29t+12 & 9t^2+4t & 0 \\ -2 & -4t-26 & -t^2-37t+33 & -8t^2+47t+16 & 9t^2+4t \end{array}\right).$$

S has rank 4 with the corresponding (left) nullspace vector

$$\begin{pmatrix} -5t^5+52t^4+931t^3+145t^2-830t-1126 \\ -45t^5-497t^4+1723t^3+2246t^2+310t-136 \\ 5t^5+48t^4+109t^3+473t^2+156t+574 \\ -50t^5-295t^4-152t^3+137t^2-950t-394 \\ 25t^6+275t^5-984t^4-1359t^3+507t^2+530t-102 \end{pmatrix}^T.$$

This matrix is analogous to the Sylvester matrix of real polynomials (for example, see Gathen and Gerhard [25, 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 $wS = 0$, hence w is a non-trivial vector in the (left) nullspace of S. In particular, this solution is equivalent to saying that 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 $S \in \mathbb{R}[t]^{(M+N)\times(M+N)}$. Moreover, for $f, g \in \mathbb{R}[t][\partial;']$ with $\deg_t f \leq d$ and $\deg_t g \leq d$ then $\deg_t S_{ij} \leq d$.

We can formally summarize these results in the following lemma.

Lemma 2.1.16. *Suppose $f, g \in \mathbb{R}[t][\partial;']$, where $\deg_\partial f = M$, $\deg_\partial g = N, \deg_t f \leq d$ and $\deg_t g \leq d$.*

1. $S = S(f,g)$ *is singular if and only* $\deg_\partial \mathrm{gcrd}(f,g) \geq 1$.

2. $\deg_\partial \mathrm{gcrd}(f,g) = \dim \mathrm{null}_\ell(S)$, *where* $\mathrm{null}_\ell(S)$ *is the left nullspace of* S.

3. *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* $wS = 0$, *we have* $uf + vg = 0$, *where* $u = \sum_{0\leq i<N} u_i \partial^i$ *and* $v = \sum_{0\leq i<M} v_i \partial^i$.

4. *Suppose that* $\deg_\partial \mathrm{gcrd}(f,g) \geq 1$. *Then there exists* $w \in \mathbb{R}[t]^{1\times(M+N)}$ *such that* $wS = 0$ *and* $\deg_t w \leq \mu = 2(M+N)d$.

Proof. Part (1) – (3) follow from Lemma 2.1.12 and the discussion above. Part (4) follows from an application of Cramer's rule and a bound on the degree of the determinants of a polynomial matrix. □

2.1.2 Linear Algebra over $\mathbb{R}$

The basic tools for performing linear algebra over $\mathbb{R}[t]$ have been met with limited success. Round-off errors in the data lead to coefficient growth over $\mathbb{R}$ and considerable degree growth in t. We can encode the existence of a GCRD as a linear algebra problem over $\mathbb{R}$, as opposed to $\mathbb{R}(t)$.

Let $S \in \mathbb{R}[t]^{(M+N)\times(M+N)}$ 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 2.1.16 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 $wS = 0$, with $\deg_t w \leq \mu = 2(M+N)d$.

Definition 2.1.17. *Let* $b = b_0 + b_1 t + \cdots + b_{\mu+d} t^{\mu+d} \in \mathbb{R}[t]$ *and define*

$$\xi(b) = \xi_{\mu+d+1} = (b_0, b_1, \ldots, b_{\mu+d}) \in \mathbb{R}^{1\times(\mu+d+1)}.$$

For any polynomial $a \in \mathbb{R}[t]$ *of degree at most* d *let*

$$\Gamma(a) = \begin{pmatrix} \xi_{\mu+d+1}(a) \\ \xi_{\mu+d+1}(ta) \\ \vdots \\ \xi_{\mu+d+1}(t^\mu a) \end{pmatrix} \in \mathbb{R}^{(\mu+1)\times(\mu+d+1)},$$

be the left multiplier matrix of a *with respect to the basis* $\langle 1, t, \ldots, t^{\mu+d}\rangle$.

Definition 2.1.18. *Given the* $(M+N)\times(M+N)$ *differential Sylvester matrix* S*, we apply* Γ *entry-wise to* S *to obtain* $\widehat{S}\in\mathbb{R}^{(\mu+1)(M+N)\times(M+N)(\mu+d+1)}$*; each entry of* S *in* $\mathbb{R}[t]$ *is mapped to a block entry in* $\mathbb{R}^{(\mu+1)\times(\mu+d+1)}$ *of* $\widehat{S}$*. We refer to* $\widehat{S}$ *as the* inflated differential Sylvester matrix *of* f *and* g*.*

Example 2.1.19. *Consider* $f=(t+3)\partial+(-2t+1)$ *and* $g=\partial+3t$*. The matrix* $\widehat{S}(f,g)$ *is given by*

$$\left(\begin{array}{cccccc|cccccc}
1 & -2 & 0 & 0 & 0 & 0 & 3 & 1 & 0 & 0 & 0 & 0\\
0 & 1 & -2 & 0 & 0 & 0 & 0 & 3 & 1 & 0 & 0 & 0\\
0 & 0 & 1 & -2 & 0 & 0 & 0 & 0 & 3 & 1 & 0 & 0\\
0 & 0 & 0 & 1 & -2 & 0 & 0 & 0 & 0 & 3 & 1 & 0\\
0 & 0 & 0 & 0 & 1 & -2 & 0 & 0 & 0 & 0 & 3 & 1\\
\hline
0 & 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 1 & 0
\end{array}\right)\in\mathbb{R}^{10\times 12}.$$

We note that the dimensions of $\widehat{S}$ *grow quadratically in terms of* $(M+N)$ *and linearly in* d*.*

Lemma 2.1.20. *Let* $f,g\in\mathbb{R}[t][\partial;']$ *have differential Sylvester matrix* $S\in\mathbb{R}[t]^{(M+N)\times(M+N)}$ *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* $wS=0$*, if and only if there exists a* $\widehat{w}\in\mathbb{R}^{(\mu+d+1)\times(M+N)(\mu+1)}$ *such that* $\widehat{w}\widehat{S}=0$*. More generally,*

$$\deg_\partial\mathrm{gcrd}(f,g)=\frac{\dim\mathrm{null}_\ell(\widehat{S})}{\mu+d+1}.$$

Proof. This follows directly from the definition of Γ and Lemma 2.1.16. □

We note that $\widehat{S}$ is no longer a square matrix. This will not pose too many problems as we will see in the following chapters.

2.1.3 Distance and Norms of Differential Polynomials

Approximations require a norm, so we need a proper definition of the *norm* of a differential polynomial.

Definition 2.1.21. *We use the Euclidean norm for polynomials and a distributed coefficient norm for differential polynomials.*

1. *For* $p = \sum_{0 \le i \le d} p_i t^i \in \mathbb{R}[t]$, *define*

$$\|p\| = \|p\|_2 = \left(\sum_{0 \le i \le d} p_i^2 \right)^{1/2}.$$

2. *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. 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 converting back to the representation over $\mathbb{R}(t)$.

Definition 2.1.22. *For any matrix* $S \in \mathbb{R}[t]^{(M+N)\times(M+N)}$, *we define the Frobenius norm* $\|S\|_F$ *by*

$$\|S\|_F^2 = \sum_{ij} \|S_{ij}\|^2.$$

In the instance of real valued entries, the Frobenius norm has an important relationship with the singular values of the matrix S [11].

Lemma 2.1.23. *Let* $f, g \in \mathbb{R}[t][\partial;']$ *have* $\deg_\partial f \le M$, $\deg_\partial g \le N$, *and let them both have degree in* t *at most,* d. *Let* $S = S(f,g)$ *be the differential Sylvester matrix of* f *and* g. *Then* $\|S\|_F^2 \le d^{2N}\|f\|^2 + d^{2M}\|g\|^2$.

Proof. First note that $\|\partial f\|^2 = \|f\partial + f'\|^2 \le (d^2+1)\|f\|^2$, and hence $\|\partial^k f\|^2 \le (d^2+1)^k\|f\|^2$. Thus

$$\begin{aligned}
\|S\|_F^2 &= \sum_{0 \le i < N} \|\partial^i f\|^2 + \sum_{0 \le i < M} \|\partial^i g\|^2 \\
&\le \sum_{0 \le i < N} (d^2+1)^i \|f\|^2 + \sum_{0 \le i < M} (d^2+1)^i \|g\|^2 \\
&\le d^{2N}\|f\|^2 + d^{2M}\|g\|^2.
\end{aligned}$$

□

Note that, although the exponentials of d (or, more precisely, the falling factorials) are intrinsic in the resultant formulation, they will cause considerable numerical instability for large degrees in ∂. As is typical with differential polynomials, we generally restrict ourselves to modest degrees in ∂.

Lemma 2.1.24. *Let $f, g \in \mathbb{R}[t][\partial;']$ have $\deg_\partial f \leq M$, $\deg_\partial g \leq N$ and both have degree at most d in t. Let $\widehat{S} \in \mathbb{R}^{(M+N)(\mu+1)\times(M+N)(\mu+d+1)}$ be the inflated differential Sylvester matrix of f and g, where $\mu = 2(M+N)d$. Then $\|\widehat{S}\|_2 \leq (\mu+1)(d^{2N}\|f\|^2 + d^{2M}\|g\|^2)$.*

Proof. Each row of $\widehat{S}$ consists precisely of entries of $S = S(f,g)$, shifted in position with respect to the previous row. Thus

$$\|\widehat{S}\|_F^2 \leq (\mu+1)\|S\|_F^2 \leq (\mu+1)(d^{2N}\|f\|^2 + d^{2M}\|g\|^2), \text{ and}$$
$$\|\widehat{S}\|_2^2 \leq \|\widehat{S}\|_F^2.$$

□

2.2 Optimization Review

The resultant formulation of Section 2.1.1 leads to asymptotically fast modular algorithms [20] over $\mathbb{Q}(t)[\partial;']$ that control interim coefficient growth. Working over $\mathbb{R}(t)[\partial;']$, these tools are no longer available to us (as the modular reductions are numerically unstable). Naively using algorithms designed for $\mathbb{Q}(t)[\partial;']$ lead to substantial degree growth in t, from round-off errors. The degree growth in t is especially problematic when considered in the context of Lemma 2.1.24. We resolve these problems through the use of optimization techniques. We provide a brief survey of optimization tools used to approach the approximate GCRD problem. We make use of quadratic minimization, linear least squares, the Singular Value Decomposition (SVD) and other tools from convex optimization [3, 11].

Definition 2.2.1. *The symmetric matrix $A \in \mathbb{R}^{n\times n}$ is positive definite if $x^T Ax > 0$ for all $x \in \mathbb{R}^{n\times 1}\backslash\{0\}$. We say that A is positive semidefinite if $x^T Ax \geq 0$.*

Positive definite matrices have a close relationship with the local convexity of functions from $\mathcal{C}^2[\mathbb{R}]$ around a local minimum.

Definition 2.2.2. *Let $X \subseteq \mathbb{R}[t][\partial;'] \times \mathbb{R}(t)[\partial;']^2 \subseteq \mathbb{R}^{n\times 1}$, where $\times$ denotes the Cartesian product, and $\mathbb{R}(t)[\partial;'] \times \mathbb{R}(t)[\partial;'] = \mathbb{R}(t)[\partial;']^2$. Precisely, we treat $\mathbb{R}[t][\partial;'] \times \mathbb{R}(t)[\partial;']^2$ as*

a subset of Euclidean space. The set X is convex if for all $x_1, x_2 \in X$ and $a \in [0,1]$ we have that

$$ax_1 + (1-a)x_2 \in X.$$

A function $\Phi : X \to \mathbb{R}$ is convex if for all $x_1, x_2 \in X$ and $a \in [0,1]$ that

$$\Phi(ax_1 + (1-a)x_2) \leq a\Phi(x_1) + (1-a)\Phi(x_2).$$

We say that Φ is strictly convex if the above inequality is strict.

We will treat X as a closed convex subset of $\mathbb{R}[t][\partial;'] \times \mathbb{R}(t)[\partial;']^2$ for the rest of this section. We note that open balls, often referred to as set neighborhoods in the literature, are convex sets. For $x \in X$, if the function $\Phi(x)$ is convex, then $\Phi(x)$ has a minimum value over X. That is, there exists (not necessarily unique) $x^* \in X$ such that $\Phi(x^*) \leq \Phi(x)$ for all $x \in X$. If $\Phi(x)$ is strictly convex, then x^* is a unique minimum.

Definition 2.2.3. *Let $\Phi : X \to \mathbb{R}$ belong to $\mathcal{C}^2[\mathbb{R}]$. For $x \in X$, the gradient of Φ is a column vector, written as $\nabla\Phi(x)_i = (\partial\Phi/\partial x_i)(x)$, while the Hessian matrix of Φ is the symmetric matrix of second-order partial derivatives defined by $\nabla^2\Phi(x)_{ij} = (\partial^2\Phi/\partial x_i \partial x_j)(x)$.*

Later we will observe that the local properties of the Hessian matrix are very important around minima. For $x \in X$, we have that $\Phi(x)$ is strictly convex if and only if $\nabla^2\Phi(x)$ is positive definite [3, Chapter 3.1]. This second-order condition ensures that Φ will have a unique (local) minimum over X.

2.2.1 The Singular Value Decomposition (SVD)

The SVD is a rank revealing factorization, that when implemented with the QR decomposition is robust and numerically stable [11]. The SVD is used extensively to solve approximate polynomial GCD problems [5, 14, 16, 26]. We use the SVD to compute solutions to linear least squares problems and to infer the numerical rank of real matrices.

Theorem 2.2.4 (The SVD [11, Theorem 2.4.1])**.** *If $A \in \mathbb{R}^{m\times n}$, then there exist orthogonal matrices $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ such that $U^T AV = \Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_p) \in \mathbb{R}^{m\times n}$, $p = \min\{m, n\}$ where $\sigma_1, \ldots, \sigma_p \geq 0$.*

Corollary 2.2.5 ([11, Corollary 2.4.3])**.** *If $A \in \mathbb{R}^{m\times n}$ then*

$$\|A\|_2 = \sigma_1 \text{ and } \|A\|_F = \sqrt{\sigma_1^2 + \cdots + \sigma_p^2}.$$

The relationship with the SVD and the Frobenius norm proves useful when we analyze perturbations in the inflated differential sylvester matrix.

Furthermore, the SVD tends to a natural pseudo-inverse that can be used for solving least squares problems. We can write $A^+ = V\Sigma^+U^T$ where Σ^+ is a pseudo-inverse of Σ given by $\Sigma^+ = \text{diag}(\sigma_1^{-1}, \ldots, \sigma_p^{-1}, 0, \ldots, 0)^T$. We implicitly make use of this pseudo-inverse formulation when solving least squares problems.

2.2.2 Quadratic Minimization

Quadratic minimization is computing the minimum value of a quadratic function, provided that one exists. Although not all quadratic functions have an extreme value, they do have one if they are convex. We consider unconstrained quadratic minimization, in the specific application of least squares.

Definition 2.2.6. *A quadratic function $\Phi(x)$ with $x = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^{n\times 1}$ may be written in general form as*

$$\Phi(x) = \frac{1}{2}x^TAx - x^Tb + c$$

for a symmetric $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^{n\times 1}$ and $c \in \mathbb{R}$.

We are interested in unconstrained quadratic minimization, that is, computing the minimum value of $\Phi(x)$ over all $x \in X$. If A is positive semidefinite, then there is a minimum. The minimum value is unique if A is positive definite. This is characterized precisely when Φ is convex and strictly convex respectively [11, Chapter 5]. Note that we add the factor of $1/2$ because $A = \nabla^2\Phi$, is the Hessian matrix of Φ (the Hessian is constant for quadratic functions).

In the case when A is positive definite the minimum is given by $x = A^{-1}b$ and in the event that A is semidefinite the minimum is given by $x = A^+b$. This particular method of solving the quadratic minimization problem analytically is known as the normal equation method. In practical applications the normal equations can be ill conditioned, so we typically use the pseudo-inverse arising from the SVD to compute a solution.

2.2.3 The Method of Linear Least Squares

The method of linear least squares consists of computing a minimum value of the function $\Phi(x) = \|Ax - b\|_2^2$. This corresponds to the quadratic minimization problem

$$\|Ax - b\|_2^2 = x^TA^TAx - 2b^Tx + b^Tb.$$

This problem always has a solution, since A^TA is positive semidefinite. As a consequence, if A has full rank then the minimization problem has a unique solution. The linear least squares problem with linear constraints is completely solved. Golub and Van Loan [11] describe how to solve this problem by means of the SVD, Lagrange multipliers and other matrix factorization techniques.

2.2.4 Post-Refinement and Newton's Method

Newton's method of minimization is a second-order optimization technique that is well suited for computing local extrema. With a good initial guess the Hessian matrix is locally positive definite. Newton iteration will converge (usually quadratically) to the unique local minimum. In the approximate GCRD problem, we consider a quartic objective function Φ, which is not convex in general.

For any $\Phi \in \mathcal{C}^2[\mathbb{R}]$ and a point $x^* \in X$ we can write the second-order approximation about x^* as

$$\Phi(x) = \Phi(x^*) + (x - x^*)^T \nabla\Phi(x) + (x - x^*)^T \left[\frac{1}{2}\nabla^2\Phi(x^*)\right](x - x^*) + O(\|x - x^*\|_2^2).$$

Lemma 2.2.7. *Let $\Phi \in \mathcal{C}^2[\mathbb{R}]$ be such that $\Phi : X \to \mathbb{R}$ and $x^* \in X$.*

1. *If x^* is a local minimum of $\Phi(x)$ then $\nabla\Phi(x^*) = 0$ and $\nabla^2\Phi(x^*)$ is positive semidefinite.*
2. *If $\nabla\Phi(x^*) = 0$ and $\nabla^2\Phi(x^*)$ is positive definite, then x^* is a local minimum of Φ.*

Proof.

1. If x^* is a local minimum of Φ, then we consider the vector $x^* + tz$ for $t \in \mathbb{R}$ and $z \in X$. The function $\Phi(x^*+tz)$ has a local minimum when $t = 0$, so $\frac{d}{dt}\Phi(x^*+tz)|_{t=0} = z^T\nabla\Phi(x^*) = 0$ and $\frac{d^2}{dt^2}\Phi(x^* + tz)|_{t=0} = z^T\left[\nabla^2\Phi(x^*)\right]z \geq 0$. Since z is arbitrary, we must have that $\nabla\Phi(x^*) = 0$ and $\nabla^2\Phi(x^*)$ is positive semidefinite.
2. Let $A = \nabla^2\Phi(x^*)$. Then we can write

$$\Phi(x) = \Phi(x^*) + \frac{1}{2}(x - x^*)^T A(x - x^*) + O(\|x - x^*\|_2^2).$$

Thus for all $\varepsilon_X > 0$ there exists a $\delta_X > 0$ such that

$$\frac{\|\Phi(x) - \Phi(x^*) - \frac{1}{2}(x - x^*)^T A(x - x^*)\|_2^2}{\|x - x^*\|_2^2} < \varepsilon_X$$

when $0 < \|x - x^*\|_2 < \delta_X$. We can re-write this as

$$\Phi(x) - \Phi(x^*) > \frac{1}{2}(x - x^*)^T A(x - x^*) - \varepsilon_X \|x - x^*\|_2^2.$$

By hypothesis, A is positive definite, so $\frac{1}{2}(x - x^*)^T A(x - x^*) > 0$ for all $x \neq x^*$. This is a continuous function in the entries of $(x - x^*)$, so on a closed and bounded (compact) subset of X, it will obtain a minimum value, $\alpha > 0$. For any $t \in \mathbb{R}$ we have

$$\frac{1}{2}\frac{t(x - x^*)^T}{\|x - x^*\|_2} A \frac{t(x - x^*)}{\|x - x^*\|_2} \geq t^2 \alpha.$$

We can choose $t = \|x - x^*\|_2$ giving us

$$\frac{1}{2}(x - x^*)^T A(x - x^*) \geq \alpha \|x - x^*\|_2^2,$$

so our second-order approximation gives us $\Phi(x) - \Phi(x^*) > (\alpha - \varepsilon_X)\|x - x^*\|_2^2$ when $0 < \|x - x^*\|_2 < \delta_X$. Since ε_X is arbitrary, we consider $\varepsilon_X \in (0, \alpha)$, so there exists $\delta_X > 0$ such that $\Phi(x) > \Phi(x^*)$ for $0 < \|x - x^*\|_2 < \delta_X$. It follows that Φ has a local minimum at x^*. □

An important observation is that if another matrix is sufficiently close to a positive definite matrix, then it too will be positive definite. In the case of Hessians, if the Hessian is positive definite at a point, then it will be positive definite in some neighborhood around it. If the Hessian matrix is (locally) strictly convex, then we can use descent methods to compute the (local) minimum.

Definition 2.2.8. *A descent method produces a sequence of points*

$$\{x^0, x^1, \ldots, x^i, \ldots | x^i \in X\}$$

satisfying

$$\Phi(x^i) > \Phi(x^{i+1}) \text{ for all } i,$$

where x^0 is a special starting point. If $\Phi(x^i) = \Phi(x^{i+1})$, then x^i is a stationary point and $\nabla\Phi(x^i) = 0$.

Typically we will have an infinite sequence when $\Phi(x)$ is strictly convex in a neighborhood around x^0. This sequence will converge to a unique limit point that is the minimum, as the only stationary point in this neighborhood is the (local) minimum. We note that while all minima are stationary points, not all stationary points are minima.

Newton's Method and Gauss-Newton Iteration

The Newton step of Φ with respect to x is defined as

$$x^{Newton} = -[\nabla^2\Phi(x)]^{-1}\nabla\Phi(x).$$

When $\nabla^2\Phi(x)$ is (locally) positive definite, this gives us

$$\nabla\Phi(x)^T x^{Newton} = -\nabla\Phi(x)^T[\nabla^2\Phi(x)]^{-1}\nabla\Phi(x) < 0,$$

except when $\nabla\Phi(x) = 0$. The implication is the Newton step is a descent direction (except when we start at a minimum). Newton's method is then characterized by successively computing x^i from

$$x^{i+1} = x^i - [\nabla^2\Phi(x^i)]^{-1}\nabla\Phi(x^i) \text{ until } \|\Phi(x^i) - \Phi(x^{i+1})\| \text{ is sufficiently small.}$$

Under the assumption that $\Phi(x^0)$ is strictly convex and that $\nabla^2\Phi(x^0)$ is (locally) Lipschitz continuous (there exists $L > 0$ such that $\|\nabla^2\Phi(x) - \nabla^2\Phi(x^*)\|_2 \leq L\|x - x^*\|_2$), it can be shown that Newton's method converges quadratically [3, Chapter 9.5]. In our instance we are minimizing a quartic polynomial, so all of these conditions are satisfied around a minimum. Directly inverting the Hessian matrix can be unstable and expensive. Instead, we compute x^{i+1} implicitly by solving an equivalent linear system,

$$\left[\nabla^2\Phi(x^i)\right] x^{i+1} = \left[\nabla^2\Phi(x^i)\right] x^i - \nabla\Phi(x^i).$$

Definition 2.2.9. *The residual $r = r(x) \in \mathcal{C}^2[\mathbb{R}^n]$ for $x \in X$ is a vector valued function that satisfies $\|r(x)\|_2^2 = \Phi(x)$. We define the Jacobian of $r(x)$ as $J = [J_{ij}(x)] = [\frac{\partial r_i}{\partial x_j}(x)]$.*

A first-order approximation of $r(x)$ about x^* is given as

$$r(x) = r(x^*) + J(x^*)(x - x^*) + O(\|x - x^*\|_2^2).$$

The Jacobian and Hessian are related by the gradient mapping, with $\nabla\Phi : x \to \nabla\Phi(x)$ having Jacobian $\nabla(\nabla\Phi(x^*)) = \nabla^2\Phi(x^*)$ at x^*. Using this approximation instead of the Hessian, we obtain the Gauss-Newton iteration.

In Gauss-Newton we replace the Newton step of Φ with respect to x by

$$x^{GN} = -[J^T J]^{-1}\nabla\Phi(x),$$

where $\nabla^2\Phi(x) \approx 2J^T J$ when the residuals are small. The matrix $J^T J$ is always positive semidefinite (even if $\nabla^2\Phi$ is indefinite). In the approximate GCRD problem, J has full

rank, so J^TJ is always positive definite. The convergence of Gauss-Newton can approach quadratic in some instances. However, it is typically linear, as we are using a first-order approximation.

Gauss-Newton iteration is not always guaranteed to converge. When applicable, it has advantages over Newton's method. The matrix J^TJ can be computed without computing any second-order derivatives. In some situations this is preferable, especially when using computationally expensive numerical approximations for second-order derivatives. When the matrix J^TJ is positive definite, we can compute the next step in the iteration in a fast and stable manner using the Cholesky decomposition [11, Chapter 4.2]. Furthermore, when J^TJ is positive definite and $\nabla^2\Phi$ is semidefinite, we are still able to continue the iteration, as the next value in the iteration is unique (and matrix inversion is still well defined).

2.3 Approximate Polynomial Arithmetic

This section provides a brief overview on arithmetic for real polynomials with floating point coefficients known only to machine precision, i.e. approximate polynomials. Addition and subtraction are straightforward so we draw our attention to multiplication, division and computing approximate polynomial GCDs. The standard techniques for multiplication and division involve Fourier transforms. In the context of approximate polynomial GCD it is advantageous to linearize the corresponding system instead. Approximate polynomial GCD is a very rich subject with many different approaches. We restrict our discussion on polynomial approximate GCD to widely used techniques and important results. We will assume that the polynomials $a, b \in \mathbb{R}[t]$ satisfy $\deg a = m$, $\deg b = n$ and $m \geq n$ unless otherwise stated throughout the rest of this section.

Multiplication

Multiplication of two usual polynomials may be performed efficiently using a Fast Fourier Transform (FFT) and Inverse FFT (IFFT). We assume that multiplication of two polynomials may be computed in $O(m \log m)$ floating point operations if the degrees of the input are bounded by m. FFT based multiplication becomes less useful when we want to create a system of undetermined coefficients for an optimization problem. Instead we consider convolution matrices and vector-matrix product approaches.

Definition 2.3.1. *The k^{th} convolution matrix of a is defined as*

$$C_k(a) = \begin{pmatrix} a_0 & & \\ a_1 & \ddots & \\ \vdots & \ddots & a_0 \\ a_m & & a_1 \\ & \ddots & \vdots \\ & & a_m \end{pmatrix} \in \mathbb{R}^{(m+k+1)\times(k+1)}.$$

We may compute $\mathbf{ab}^T = C_n(a)\mathbf{b}^T = C_m(b)\mathbf{a}^T$ using $O(m^2)$ floating point operations. Despite being relatively slow, this formulation of multiplication may be reversed to perform division numerically.

Division Without Remainder

Performing long division of two polynomials with floating point coefficients is known to be unstable [27]. Instead we introduce interpolation and least squares based division algorithms.

Division without remainder between a and b may be performed quickly using the FFT and IFFT in $O(m \log m)$ floating point operations if special care is taken to prevent division by zero. Performing division numerically, we can expect very small remainders, as the division is usually not exact due to machine error. These remainders will perturb the coefficients of the resulting quotient. Typically FFT based division is used to produce an initial guess for an optimization procedure, such as Newton's method.

Linearizing multiplication through convolution matrices allows us to perform this procedure in reverse. Division is equivalent to solving a linear system over the real numbers. Input is typically known to machine precision, so solving a least squares problem is preferable. Given ab and a we can write $b \approx \frac{ab}{a}$ by computing a linear least squares solution to $\mathbf{ab}^T = C_n(a)\mathbf{b}^T$.

2.3.1 Approximate Polynomial GCD

Univariate polynomial approximate GCD is deeply connected to the approximate GCRD problem, as univariate polynomial GCD is a special instance of computing a GCRD. This

relationship provides many insights towards the approximate GCRD problem. Approximate polynomial GCD is also an important tool in obtaining primitive differential polynomials. Polynomial GCD computations with floating point arithmetic are no longer straight forward, as the Euclidean algorithm is numerically unstable. The standard techniques involve linearizing a polynomial Diophantine equation in terms of the Bézout coefficients. Numerical linear algebra techniques and optimization tools are used to solve this system.

Definition 2.3.2. *The (Exact) Greatest Common Divisor (GCD) of* $a, b \in \mathbb{R}[t]$ *is the polynomial* $p \in \mathbb{R}[t]$ *of largest degree in* t *that divides both* a *and* b*. We write* $\gcd(a, b) = p$*. When* $\deg p = 0$*, we say that* a *and* b *are relatively prime.*

In general the GCD is not unique; it differs by a unit over $\mathbb{R}$. We will adopt the convention that the GCD is normalized so that it is unique. Algebraically we make p monic ($\operatorname{lcoeff} p = 1$) and numerically we require that p has unit norm, i.e. $\|p\| = 1$.

The Euclidean algorithm for univariate polynomials is summarized as follows [25, Chapter 3]. Given a and b we can write $a = bq + r$ where $\deg r < \deg b$. Using this relationship recursively, one may compute a GCD by computing the quotient q and the remainder r successively on each step. The first attempts at approximate polynomial GCD consisted of using the Euclidean algorithm in a clever manner [23], however the underlying divisions made this algorithm unstable.

Almost all polynomials over $\mathbb{R}[t]$ are relatively prime, i.e. $\gcd(a, b) = 1$ for almost all a and b. Furthermore, if two polynomials have an exact GCD, then introducing arbitrarily small perturbations to their coefficients will usually make them relatively prime. Instead we look at a slightly different problem; find a nearby a pair of polynomials with an exact GCD, or what the literature refers to as an ε-GCD [5, 7, 16, 23].

Definition 2.3.3. *An* ε-GCD *of* a, b *is* $p \in \mathbb{R}[t]$ *if* p *satisfies the following conditions:*

1. p *is an exact GCD of* $a + \Delta a$ *and* $b + \Delta b$ *for structured perturbations* $\Delta a, \Delta b \in \mathbb{R}[t]$*,*
2. $\|a\| = \|b\| = 1$ *and* $\|\Delta a\| + \|\Delta b\| < \varepsilon$*,*
3. $\deg p$ *is the largest possible for* ε*.*

When this definition is generalized to multivariate polynomials, the last condition is typically omitted or relaxed as certification becomes difficult. Most algorithms receive a and b as input, and output nearby polynomials $a + \Delta a$ and $b + \Delta b$ that have an exact GCD, p. The distance $\|\Delta a\| + \|\Delta b\|$ is an output of the algorithm, i.e. ε isn't specified

in advance. ε-GCDs are not in general unique. Multiple ε-GCD's are not typically scalar multiples of each other either. To guarantee existence of an ε-GCD whose ε is minimal, it is sufficient to verify that $\operatorname{lcoeff} p$ does not vanish and $\|p\|$ is bounded [15]. We restrict ourselves to ε-GCDs whose coefficients and underlying perturbations are real; there are often nearer ε-GCDs with complex coefficients [14, 16].

Definition 2.3.4. *The Sylvester matrix [25, Chapter 6] of $a, b \in \mathbb{R}[t]$ is given as*

$$\operatorname{Syl} = \operatorname{Syl}(a,b) = \underbrace{\begin{pmatrix} a_0 & & & \\ a_1 & a_0 & & \\ \vdots & \ddots & \ddots & \\ a_m & & \ddots & a_0 \\ & a_m & & a_1 \\ & & \ddots & \vdots \\ & & & a_m \end{pmatrix}}_{n} \underbrace{\begin{pmatrix} b_0 & & & \\ b_1 & b_0 & & \\ \vdots & \ddots & \ddots & \\ b_n & & \ddots & b_0 \\ & b_n & & b_1 \\ & & \ddots & \vdots \\ & & & b_n \end{pmatrix}}_{m} \in \mathbb{R}^{(m+n)\times(m+n)}.$$

Many authors will also define the Sylvester matrix described above to be the transpose, as we do with our notion of differential Sylvester matrix. Intuitively the Sylvester matrix linearizes a system of Diophantine equations. Accordingly, the Sylvester matrix plays an important role in determining the existence and computation of a polynomial GCD.

Theorem 2.3.5 ([25])**.** *The polynomials a and b are co-prime if and only if their Sylvester matrix,* Syl *has full rank. If $\gcd(a,b)$ is non-trivial then* $\operatorname{rank} \operatorname{Syl} = \deg \gcd(a,b)$.

Theorem 2.3.6 ([8, 18])**.** *If the Sylvester matrix is triangularized to row echelon form using only row operations, then the last nonzero row gives the coefficients of the polynomial GCD over $\mathbb{R}$.*

In other words, the Sylvester matrix encodes enough information to obtain a GCD via nullspace basis computation. Numerically this is exploited through the SVD [5] and other linear algebra techniques, such as the QR decomposition [6].

Corless, Gianni, Trager and Watt [5] use the SVD on the Sylvester matrix to infer the degree of possible ε-GCDs. They discuss how to compute an ε-GCD using techniques from numerical linear algebra and optimization. We adapt their rank computation technique to the instance of differential polynomials.

Karmakar and Lakshman [16, 17] improve upon Corless et al. [5] by using the SVD to determine the degree of a likely ε-GCD and formulate the minimum perturbations of a and

b as a positive definite quadratic minimization problem. The general idea is to express the possible ε-GCD as a common root of $a + \Delta a$ and $b + \Delta b$. In the real degree 1 case when $p = t - p_0$, we can write

$$a + \Delta a = (t - p_0) \sum_{0 \le i \le m-1} a_i^* t^i \text{ and } b + \Delta b = (t - p_0) \sum_{0 \le i \le n-1} b_i^* t^i,$$

for $p_0, a_i^*, b_i^* \in \mathbb{R}$. The objective function $\|\Delta a\|_2^2 + \|\Delta b\|_2^2$ is a positive definite quadratic function in the coefficients of a_i^* and b_i^*. The minimum value of the quadratic form is a rational function in the coefficients of p. If the objective function has a minimum value, then it occurs when this rational function is minimized. Karmakar and Lakshman handle the cases of real and complex perturbations.

Emiris, Galligo and Lombardi [7] provide certification results that improve upon the SVD technique of Corless et. al. [6]. They certify the degree of an ε-GCD through a gap theorem on the singular values of subresultants; however there is an exponential factor present in the gap that can limit the direct usefulness of the theorem. We state their theorem below without proof.

Let $\|a\| = \|b\| = 1$ and let $m \le n$. Let $\tau_0, \tau_1, \ldots, \tau_m$ be the increasing sequence of minimal singular values of the subresultant mappings Syl_r of a and b for $r \ge 1$.

Theorem 2.3.7 ([7, Theorem 6]). *Suppose that $\tau_{r-1} \le \varepsilon \le \tau_r/\sqrt{2}$ for $\varepsilon \le 1$, $1 \le r \le m \le n$ and*

$$\tau_r > \tau_{r-1} 2^{2n+m-2r}.$$

If, moreover,

$$\left(1 + \frac{2 + \tau_r^2}{\tau_r - \tau_{r-1}}\right)^{n+1} \left(1 + \frac{2^{2n+m-2r}}{\tau_r - \tau_{r-1} 2^{2n+m-2r}}\right) \tau_{r-1} \le \varepsilon,$$

then the degree of the ε-GCD of a and b is equal to r.

Corless, Watt and Zhi [6] developed a method of computing an ε-GCD through QR factorization that handles the case when common roots are near to or outside the unit circle. The algorithm, known as `QRGCD` has been included in the Symbolic-Numeric Algorithms for Polynomials (SNAP) package included in Maple. We make use of the `QRGCD` algorithm (among others) in our implementation of approximate GCRD, as it is better than the other GCD tools present in the SNAP package.

Zeng [26] uses QR decomposition and post-refinement Gauss-Newton iteration to compute univariate GCDs. This result is generalized to multivariate polynomials by Zeng

and Dayton [28]. We adapt their optimization technique to the instance of differential polynomials.

Conversely, we now consider the notion of relative primality of approximate polynomials. One may naively consider the conditioning of the Sylvester matrix to infer relative primality. Beckerman and Labahn [1] consider a structured approach to relative primality by looking at the first column of the inverse of the Sylvester matrix.

Definition 2.3.8 ([1]). *We define*

$$\varepsilon(a,b) = \inf\{\|a-\widetilde{a}, b-\widetilde{b}\|_1 : (\widetilde{a},\widetilde{b}) \text{ have a common root}, \deg_t \widetilde{a} \leq m, \deg_t \widetilde{b} \leq n\}.$$

Under this definition we say that a and b are $\varepsilon-$prime.

Lemma 2.3.9 ([1, Lemma 2.1]). *For any two polynomials $a, b \in \mathbb{R}[t]$ we have*

$$\varepsilon(a,b) > \frac{1}{\|\mathrm{Syl}(a,b)\|_1},$$

where $\|\cdot\|_1$ is the corresponding matrix 1-norm.

The polynomials a and b, which are relatively prime if and only if there exist $u, v \in \mathbb{R}[t]$ with $\deg u < n$ and $\deg v < m$ such that $au + bv = 1$. In matrix form this can be expressed as $\mathrm{Syl}(a,b) \cdot \begin{pmatrix} v \\ u \end{pmatrix} = (1, 0, \ldots, 0)^T$, which gives us two polynomials are relatively prime if and only if we can determine the first column of the inverse of their corresponding Sylvester matrix [1].

Lemma 2.3.10 ([1, Lemma 2.4]). *Let $f(z) = f_{-1}z^{-1} + \cdots + f_{1-m-n}z^{1-m-n} = \frac{u(z)}{a(z)} + O(z^{-m-n})_{z\to\infty}$ for $z \in \mathbb{C}$. Then $\mathrm{Syl}(a,b)$ is invertible with inverse given by*

$$\begin{pmatrix} v_0 & 0 & \cdots & \cdots & \cdots & 0 & b_0 & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & & & \vdots & \vdots & \ddots & \ddots & & & \vdots \\ v_{n-1} & \cdots & v_0 & 0 & \cdots & 0 & b_{n-1} & \cdots & b_0 & 0 & \cdots & 0 \\ u_0 & 0 & \cdots & \cdots & \cdots & 0 & -a_0 & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & & & \vdots & \vdots & \ddots & \ddots & & & \vdots \\ u_{m-1} & \cdots & u_0 & 0 & \cdots & 0 & -a_{m-1} & \cdots & -a_0 & 0 & \cdots & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \\ 0 & f_{-1} & \cdots & f_{1-m-n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & f_{-1} \end{pmatrix}.$$

Theorem 2.3.11 ([1, Theorem 3.1]). *Let u, v be polynomials of degrees at most $m-1$ and $n-1$ solving $au+bv=1$. Then*

$$\left\| \begin{pmatrix} v \\ u \end{pmatrix} \right\|_1 \leq \left\| \mathrm{Syl}(a,b)^{-1} \right\|_1 \leq \left\| \begin{pmatrix} v \\ u \end{pmatrix} \right\|_1 + 2 \left\| f \right\|_1 \cdot \left\| (a,b) \right\|_1 .$$

Corollary 2.3.12 ([1, Corollary 3.2]). *Let u, v be polynomials of degrees at most $m-1$ and $n-1$ solving $au+bv=1$ and $\underline{\cdot}$ be the reverse operator for polynomials over $\mathbb{R}[t]$. Then*

$$\kappa = \left\| \begin{pmatrix} v & \underline{v} \\ u & \underline{u} \end{pmatrix} \right\|_1 = \max \left\{ \left\| \begin{pmatrix} v \\ u \end{pmatrix} \right\|_1 \left\| \begin{pmatrix} \underline{v} \\ \underline{u} \end{pmatrix} \right\|_1 \right\},$$

we have that

$$\kappa \leq \left\| \mathrm{Syl}(a,b)^{-1} \right\|_1 \leq \kappa + 2 \left\| f \right\|_1 \left\| (a,b) \right\|_1 ,$$

where $\left\| f \right\|_1 = \left\| \underline{v} \cdot u - \underline{u} \cdot v \right\|_1$. Furthermore, $\left\| f \right\|_1 \leq \kappa^2$.

Corollary 2.3.13 ([1, Corollary 4.4]). $\varepsilon(a,b) \geq \kappa^{-1}$.

For appropriately scaled a and b, numerical experiments indicate that $\left\| \mathrm{Syl}(a,b)^{-1} \right\|_1$ is proportional to κ and not κ^2 [1]. In the context of differential operators, the results of Beckerman and Labahn can be used for a fast test to determine if an answer is likely to be primitive.

Chapter 3

The Approximate GCRD Problem

This chapter formalizes the approximate GCRD problem. We describe our methodology and prove important results. First we standardize some notation and assumptions, with the assumptions holding unless otherwise stated. The polynomials $f, g, h \in \mathbb{R}[t][\partial;']$ will be primitive and have norm 1. The co-factors $f^*, g^* \in \mathbb{R}(t)[\partial;']$ will have rational function coefficients in lowest terms. Furthermore, we assume the degrees $\deg_\partial f = M, \deg_t f \leq d, \deg_\partial g = N, \deg_t g \leq d, \deg_\partial h = D, \deg_\partial f^* = M - D$ and $\deg_\partial g^* = N - D$. The matrices $S = S(f, g) \in \mathbb{R}[t]^{(M+N)\times(M+N)}$ will be the differential Sylvester matrix of f and g, and $\widehat{S} = \widehat{S}(f, g) \in \mathbb{R}^{(M+N)(\mu+1)\times(M+N)(\mu+d+1)}$ will be the inflated differential Sylvester matrix of f and g, where $\mu = 2(M + N)d$.

Problem. *Given f and g such that $\operatorname{gcrd}(f, g) = 1$ we wish to compute $\widetilde{f}, \widetilde{g} \in \mathbb{R}[t][\partial;']$ such that $\operatorname{gcrd}(\widetilde{f}, \widetilde{g}) = h$ with $D > 1$ such that:*

1. *$\|f - \widetilde{f}\|_2 + \|g - \widetilde{g}\|_2 \leq \varepsilon$ for a (hopefully small) $\varepsilon \in \mathbb{R}^{\geq 0}$ and*
2. *the degree of h, D, is the largest possible for the computed ε.*

The differential polynomial h is said to be an approximate GCRD of f and g if these conditions are satisfied.

The approximate GCRD problem is a generalization of computing an ε-GCD from Section 2.3.1. The requirement that the GCRD has maximal degree is difficult to certify outside the exact setting, however this usually isn't 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.

3.1 Approximate Differential Polynomial Arithmetic

This section is an overview of approximate differential polynomial arithmetic. Addition and subtraction are straightforward, while multiplication and division requires some care in specifying whether a left or right canonical form is used to represent the coefficients. We also need to pay special attention on whether multiplications and divisions are performed on the right or left.

Division requires even more care. Long division for Ore polynomials consists of performing several divisions of many polynomials on each step. Errors and perturbations from the underlying polynomial divisions accumulate on each step, making this process unstable.

Using resultant-like matrices, we express multiplication and division without remainder as a linear algebra problem with rational functions. Using tools from polynomial approximate GCD, we linearize the corresponding linear algebra problem over $\mathbb{R}[t]$ and compute a primitive solution. When rational functions appear, we formulate a real linear least squares problem so that our results are in lowest terms.

3.1.1 Multiplication

Definition 3.1.1. *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 appear to the left).*

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 appear to the right).*

Both right and left differential convolution matrices can be used to perform multiplication. Recall that

$$f = \sum_{0\leq i\leq M} f_i\partial^i,\ f^* = \sum_{0\leq i\leq M-D} f_i^*\partial^i \text{ and } h = \sum_{0\leq i\leq D} h_i\partial^i$$

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.

Using differential convolution matrices, we are able to perform multiplication between two differential polynomials whose degrees in ∂ are bounded by M using $O(M^2)$ operations over $\mathbb{R}(t)$. If the degrees of the input in t are bounded above by d, then this corresponds to roughly $O(M^2d^2)$ floating point operations using standard techniques.

3.1.2 Division Without Remainder

While multiplication of approximate differential polynomials is straightforward, division is more difficult. The Euclidean algorithm is unstable so we take a direct approach. In the exact case, if $\text{gcrd}(f, g) = h$ we can write $f = f^*h$ and $g = g^*h$. If we know h then we

would compute f^* and g^* by performing a division. Since h is a right factor of f and g, the remainder of the division will be zero. In the approximate setting we can expect the remainder to be negligible, so we choose to ignore it.

Given f and h, we can express $f = f^*h$ as a linear system. This equation can be linearized over $\mathbb{R}(t)$ by writing

$$(f_0, f_1, \ldots, f_M) = (f_0^*, \ldots, f_{M-D}^*)\mathcal{C}_{M-D}^{\mathbf{R}}(h). \tag{3.1}$$

This is a linear algebra problem whose solution belongs to $\mathbb{R}(t)^{1\times(M-D+1)}$ (as f and h are primitive). Naively attempting to solve this system will yield no solution, as $\mathcal{C}_{M-D}^{\mathbf{R}}(h)$ is an $(M-D+1)\times(M+1)$ matrix; the matrix is not square. Arbitrarily small perturbations from round off errors will prevent the computation of an exact solution, provided that one exists. In the event that the factorization is approximate, there is no solution. We overcome both cases by formulating a linear least squares problem. The idea is that the first D columns of $\mathcal{C}_{M-D}^{\mathbf{R}}(h)$ in (3.1) are redundant, so we can ignore them during computations.

Definition 3.1.2. *The matrix $\mathcal{M}(h) \in \mathbb{R}[t]^{(M-D+1)\times(M-D+1)}$ is the sub-matrix of $\mathcal{C}_{M-D}^{\mathbf{R}}(h)$ with the first D columns removed.*

Lemma 3.1.3. *The columns of the matrix $\mathcal{M}(h)$ are a triangular basis for the column space of (3.1).*

Proof. We note that

$$\begin{pmatrix} \Psi_{M+1}(h) \\ \Psi_{M+1}(\partial h) \\ \vdots \\ \Psi_{M+1}(\partial^{M-D}h) \end{pmatrix} = \underbrace{\begin{pmatrix} * & * & \ldots & * & h_D & & & \\ * & * & \ldots & * & * & h_D & & \\ \vdots & \vdots & \ldots & * & * & * & \ddots & \\ * & * & \ldots & * & * & * & * & h_D \end{pmatrix}}_{D \qquad\qquad M-D+1} \in \mathbb{R}[t]^{(M-D+1)\times(M+1)},$$

so we can write

$$\mathcal{M}(h) = \begin{pmatrix} h_D & & & \\ * & h_D & & \\ * & * & \ddots & \\ * & * & * & h_D \end{pmatrix} \in \mathbb{R}[t]^{(M-D+1)\times(M-D+1)}.$$

By inspection we see that each column is linearly independent and $\mathcal{M}(h)$ is triangular. $\mathcal{M}(h)$ is a rank $M-D+1$ sub-matrix of $\mathcal{C}_{M-D}^{\mathbf{R}}(h)$ which has rank $M-D+1$. The columns of $\mathcal{M}(h)$ span the columns of $\mathcal{C}_{M-D}^{\mathbf{R}}(h)$. □

Corollary 3.1.4. *h_D^{M-D+1} is a common denominator of the coefficients of f^*. Precisely, $h_D^{M-D+1} f^* \in \mathbb{R}[t][\partial;']$.*

Proof. This follows by computing the Cramer solution of

$$(f_D, f_{D+1}, \ldots, f_M) = (f_0^*, f_1^*, \ldots, f_{M-D}^*)\mathcal{M}(h),$$

for the coefficients of f^*. □

The first D columns of $\mathcal{C}_{M-D}^{\mathbf{R}}(h)$ contain the most derivatives. These entries are typically the largest magnitude and introduce considerable instability. This is noticeable when $\deg_t h > M - D$ and $M - D$ is large. Furthermore, $\mathcal{M}(h)$ has full rank so we can always solve for f^* using linear algebra over $\mathbb{R}(t)$. Round-off errors and other noise in the coefficients no longer prevent computation of a solution (note that we have a least squares solution instead).

In the special case when h is monic, i.e. $h_D = 1$, naive linear algebra over $\mathbb{R}(t)$ will yield the desired least squares solution. There are no fractions by Corollary 3.1.4, so the output is in lowest terms.

In general h is not monic, so we formulate a real linear least squares problem. First we compute a solution naively using backwards substitution over $\mathbb{R}(t)$, using $\mathcal{M}(h)$. Next we use approximate polynomial GCD to infer the degrees of numerators and denominators of the coefficients of f^*. This information is used to establish an undetermined coefficient representation of $f_0^*, \ldots, f_{M-D}^*$ that are in lowest terms. Finally, we solve for the coefficients of f^* with real linear least squares.

The linear least squares formulation of the problem is necessary. Approximate polynomial GCD operations do not obtain an exact divisor. Dividing the coefficients of f^* by the approximate polynomial GCDs will perturb the coefficients of f^*. If a least squares division is used, this is still highly unstable as the approximate polynomial GCDs obtained can be poorly conditioned. We note that the same solution obtained by linear least squares can also be obtained using weighted total least squares subject to quadratic constraints (although this isn't asymptotically faster). The following example illustrates why a least squares division is preferred over computing approximate GCDs.

Example 3.1.5. *Consider $f^* = 3t\partial + (2t - \frac{1}{7})$ and $h = (2t-1)\partial^2 + 3t\partial + (t - \frac{1}{2})$.*

If we compute $f = f^*h$ numerically (to 10 digits), we obtain

$$\begin{aligned} f = &\left(6.0\,t^2 - 3.0\,t\right)\partial^3 + \left(13.0\,t^2 + 3.714285714\,t + .1428571429\right)\partial^2 \\ &+ \left(9.0\,t^2 + 7.071428571\,t\right)\partial + \left(2.0\,t^2 + 1.857142857\,t + .07142857145\right). \end{aligned}$$

Performing division using $\mathcal{M}(h)$ yields

$$f^*_{naive} = \frac{t\,(6.0\,t - 3.0)\,\partial}{2.0\,t - 1.0} + \frac{8.0\,t^3 - 8.571428572\,t^2 + 2.571428572\,t - .1428571429}{(2.0\,t - 1.0)^2},$$

If we use approximate GCD to make f^*_{naive} in lowest terms, we have the error

$$\left\| \frac{f^*_{naive}}{\operatorname{lcoeff}_t \operatorname{lcoeff}_\partial f^*_{naive}} - \frac{f^*}{\operatorname{lcoeff}_t \operatorname{lcoeff}_\partial f^*} \right\|_2 \approx .0281.$$

Using linear least squares, we compute

$$f^*_{LS} = .612142080t\partial + (.4080947208t - .0291496228).$$

Note that f and h are normalized when computing f^*_{LS}, i.e. $\|f\| = \|h\| = 1$. The error from the least squares division is

$$\left\| \frac{f^*_{LS}}{\operatorname{lcoeff}_t \operatorname{lcoeff}_\partial f^*_{LS}} - \frac{f^*}{\operatorname{lcoeff}_t \operatorname{lcoeff}_\partial f^*} \right\|_2 \approx 5.0179 \times 10^{-10},$$

in other words, least squares computes f^* to our specified precision.

This example illustrates how small perturbations from round-off errors can prevent computation of the desired solution. Furthermore, we observe the effect of the perturbations from approximate GCD on the coefficients, and why it is necessary to perform a least squares division over $\mathbb{R}(t)[\partial;']$.

3.1.3 Numerically Computing an Exact GCRD

Computing an exact GCRD with exact coefficients is a well studied problem. Fast modular algorithms are available [20] based on the theory of subresultants. Tools based on modular algorithms are unstable, so we work with a linearization of the differential Sylvester matrix. We describe how to obtain an exact GCRD numerically (or an approximation to machine error) as a real linear least squares problem. We accomplish this by linearizing a Diophantine equation corresponding to the Bézout coefficients.

Let $u, v \in \mathbb{R}[t][\partial;']$ and $w \in \mathbb{R}[t]^{1\times(M+N)}$ be from Lemma 2.1.16. Recall that $S = S(f,g)$ is the differential Sylvester matrix of f and g.

If $\operatorname{gcrd}(f,g) = h$ then we can write $uf+vg = h$ if and only if $wS = (h_0, \ldots, h_D, 0, \ldots, 0)$. We know that $\deg_t w \le \mu$ and $\deg_t h \le \mu + d$, so we have an upper bound on the number

of variables needed for the linearization. The only information we are missing is D, the degree of the GCRD. Using the SVD [5] we can infer the rank of $\widehat{S} = \widehat{S}(f, g)$ and scale it by a factor of $\mu + d + 1$, according to Lemma 2.1.20.

Now that D is known, in exact arithmetic we would solve for $w \in \mathbb{R}(t)^{1\times(M+N)}$ from

$$wS(f, g) = (\underbrace{*, *, \ldots, 1}_{D+1}, 0, \ldots, 0), \tag{3.2}$$

and clear fractions. Using exact polynomial GCD we can make this answer primitive. Numerically this is no longer straightforward. The differential Sylvester matrix can be ill-conditioned for large degree inputs. Furthermore, using approximate polynomial GCD to make the answer primitive will perturb the coefficients, so this isn't a good idea either.

Observe that (3.2) has an exact solution, so the least squares problem

$$\min_{w,h} \|wS - (h_0, h_1, \ldots, h_D, 0, \ldots, 0)\|_2 \text{ subject to } \operatorname{lcoeff}_t h_D = 1$$

will only be minimized at an exact solution. In other words, a linear least squares solution is also an exact solution (to machine precision). Now the problem is to determine which solution to compute.

An upper bound on the degree in t of h is $\mu + d$, so we are guaranteed a solution if we use $\deg_t \operatorname{lcoeff}_\partial h = \mu + d$. This solution is most likely an associate, that is not primitive. We will make this answer primitive by using tools from approximate polynomial GCD [14]. Using a generalized Sylvester matrix of many polynomials and the SVD [14], we can infer the degree of the content of h. Now that the degree in t of the leading coefficient of h is known, we can formulate a linear least squares problem with a primitive solution.

Another option is to perform a binary search on $\deg_t \operatorname{lcoeff}_\partial h$ [12] until a primitive solution is computed. The constraint that $\operatorname{lcoeff}_t \operatorname{lcoeff}_\partial h = 1$ can be changed to $\|h\|_2^2 = 1$. However this requires a total linear least squares solution with weighting subject to quadratic constraints. The weighting is necessary for some instances of the approximate GCRD problem, as the solution with the least norm is not necessarily primitive according to some approximate GCD algorithms or notions of relative primality [1]. Furthermore, this solution can have serious problems if $\operatorname{lcoeff}_t \operatorname{lcoeff}_\partial h$ vanishes or is very small.

3.2 Approximate GCRD via Unstructured Least Squares

Our first approach to the approximate GCRD problem is to compute the nearest (with respect to $\|\cdot\|_F$) singular *unstructured* matrix to the inflated differential Sylvester matrix

via the SVD. Using the SVD we can find the matrix $\Delta\widehat{S}$ of minimal 2-norm such that $\widehat{S} + \Delta\widehat{S}$ has a prescribed rank. First, we compute the SVD of $\widehat{S}$ as

$$\widehat{S} = P\Sigma Q,$$

where

$$P \in \mathbb{R}^{(M+N)(\mu+1)\times(M+N)(\mu+1)}, \quad \text{and} \quad Q \in \mathbb{R}^{(M+N)(\mu+d+1)\times(M+N)(\mu+d+1)}$$

are orthogonal and

$$\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_{(M+N)(\mu+1)}) \in \mathbb{R}^{(M+N)(\mu+1)\times(M+N)(\mu+d+1)},$$

satisfies $\sigma_1 \geq \sigma_2 \ldots \geq \sigma \geq \sigma_{(M+N)(\mu+1)}$. Note that Σ is not square (it has more columns than rows), and we simply pad it with zeros to obtain the desired shape.

Now, by Lemma 2.1.20, we want to find a nearby matrix whose left nullspace has reduced dimension by multiples of $(\mu + d + 1)$. This matrix is

$$P\,\overline{\Sigma}\,Q = \widehat{S} + \Delta\widehat{S},$$

where

$$\begin{aligned}\overline{\Sigma} &= \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_{(M+N-\varrho)(\mu+d+1)}, 0, \ldots, 0) \\ &\in \mathbb{R}^{(M+N)(\mu+1)\times(M+N)(\mu+d+1)},\end{aligned}$$

and $\varrho = \frac{\dim \operatorname{null}_\ell(\widehat{S}+\Delta\widehat{S})}{\mu+d+1}$.

Then $\widehat{S}$ will be nearby the singular matrix $\widehat{S} + \Delta\widehat{S}$ of prescribed rank. Of course, $\widehat{S} + \Delta\widehat{S}$ is probably an unstructured matrix, and in particular, not an inflated differential Sylvester matrix.

Lemma 3.2.1. *The matrix $\widehat{S} + \Delta\widehat{S} = P\overline{\Sigma}Q$ of rank $(M + N - \varrho)(\mu + d + 1)$ is nearby $\widehat{S}$. In particular if $\sigma_{(M+N-\varrho)(\mu+d+1)+1} < \varepsilon$ for some $\varepsilon > 0$ then*

$$\|P\overline{\Sigma}Q - \widehat{S}\|_F^2 \leq \varepsilon^2\varrho(\mu + d + 1).$$

Proof. First we note that $\|\widehat{S} - P\overline{\Sigma}Q\|_F^2 = \|\Sigma - \overline{\Sigma}\|_F^2$ [11, Corollary 2.4.3], because $P\overline{\Sigma}Q - \widehat{S} = P(\overline{\Sigma} - \Sigma)Q$, whose singular values are easily computed by inspection. It follows that

$$\begin{aligned}\|\Sigma - \overline{\Sigma}\|_F^2 &= \sum_{i=(M+N-\varrho)(\mu+d+1)+1}^{(\mu+1)(M+N)} \sigma_i^2 \\ &\leq \varepsilon^2\left[(\mu + 1)(M + N) - (M + N - \varrho)(\mu + d + 1)\right] \\ &\leq \varepsilon^2\varrho(\mu + d + 1).\end{aligned}$$

□

Next we show that in general, a matrix of the desired rank deficiency and inflated differential structure exists within a relatively small radius of $\widehat{S}$.

Lemma 3.2.2. *Suppose there are $\widetilde{f}, \widetilde{g} \in \mathbb{R}[t][\partial;']$, with $\|\widetilde{f} - f\| \leq \varepsilon$ and $\|\widetilde{g} - g\| \leq \varepsilon$, such that $\deg_\partial \operatorname{gcrd}(\widetilde{f}, \widetilde{g}) = \varrho \geq 1$. Let $\Delta f = f - \widetilde{f}$ and $\Delta g = g - \widetilde{g}$. The differential Sylvester matrix $S(\Delta f, \Delta g) \in \mathbb{R}[t][\partial;']^{(M+N)\times(M+N)}$ satisfies*

$$\|S(\Delta f, \Delta g)\|_F^2 < (d^{2N} + d^{2M})\varepsilon^2$$

and

$$\|\widehat{S}(\Delta f, \Delta g)\|_F^2 < (\mu + 1)(d^{2N} + d^{2M})\varepsilon^2.$$

Proof. The first inequality follows by Lemma 2.1.23. The second inequality follows from Lemma 2.1.24. □

Moreover, $\dim \operatorname{null}_\ell(\widehat{S} + \widehat{S}(\Delta f, \Delta g)) = \varrho(\mu + d + 1)$. Thus, for sufficiently small ε there exists a perturbation $\Delta f, \Delta g$ such that $\|\widehat{S}(\Delta f, \Delta g)\|$ is small (at least assuming small M, N) and $\widehat{S} + \widehat{S}(\Delta f, \Delta g)$ has appropriate rank and structure.

3.3 Theory of Approximate GCRD via Optimization

The method of Section 3.2 is not ideal, but it does provide a suitable initial guess for an optimization problem. We describe how to formulate an objective function Φ, that when minimized corresponds to a solution to the approximate GCRD problem. Given f and g such that $\operatorname{gcrd}(f, g) = h$ we can write

$$f = f^* h,$$
$$g = g^* h.$$

Recall that f^* and g^* are called the left co-factors of f and g respectively. The co-factors typically have rational function coefficients. Later we will find it useful to clear fractions and work with a primitive associate.

We define the objective function $\Phi : \mathbb{R}[t][\partial;'] \times \mathbb{R}(t)[\partial;']^2 \to \mathbb{R}$ as

$$\Phi(h, f^*, g^*) = \|f - f^* h\|_2^2 + \|g - g^* h\|_2^2.$$

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 Φ, as f and g will typically be relatively prime. To

compute guesses for the co-factors given h, we will perform a least squares division using the method of Section 3.1.2. We only require an initial guess for f^* and g^* to minimize Φ, so this factorization doesn't need to be exact, in the event that $\gcrd(f, g) = h$.

We show that Φ 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.

3.3.1 Existence of Solutions to the Approximate GCRD Problem

Lemma 3.3.1. *Let $f, h \in \mathbb{R}[t][\partial;']$ with monic leading coefficients, but 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 Cramer solution to

$$(f_D, f_{D+1}, \ldots, f_M) = (f_0^*, f_1^*, \ldots, f_{M-D}^*)\mathcal{M}(h). \qquad \square$$

As an observation, we relax the assumption that f is primitive 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.

Theorem 3.3.2. *Suppose that Φ is a continuous real function on a compact metric space X. Then there exist points p and q in X such that*

$$\Phi(p) \leq \Phi(x) \leq \Phi(q),$$

for all $x \in X$. Precisely, Φ attains its minimum and maximum values at p and q respectively.

Proof. See Theorem 4.16 from Rudin [22] for a proof. $\square$

We state a general version of the theorem where a logic predicate Ξ can be chosen to impose additional constraints on the problem. We prove a particular instance of the theorem that interests us, in a less general setting. 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 $\phi(f, g) = (\mathbf{f}, \mathbf{g})$, where $\mathbf{f}$ and $\mathbf{g}$ are padded with zeros to have the desired dimensions.

Theorem 3.3.3 (Existence of Global Minima [15])**.** *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*

$$\mathcal{F}_\Omega = \left\{ \begin{array}{ll} (\widetilde{f}, \widetilde{g}) \in \mathbb{R}[t][\partial;']^2 \text{ such that} & \deg_\partial \widetilde{f} = M, \\ & \deg_\partial \widetilde{g} = N, \\ & \deg_\partial \widetilde{h} \geq D, \\ & \widetilde{h} = \operatorname{gcrd}(\widetilde{f}, \widetilde{g}), \\ & \|\widetilde{h}\| \leq \Omega, \\ & \operatorname{lcoeff}_t(\operatorname{lcoeff}_\partial \widetilde{h}) = 1, \\ & \text{and } \Xi(\phi(\widetilde{f}, \widetilde{g})) = \text{true}. \end{array} \right\}$$

Suppose that $\mathcal{F}_\Omega \neq \emptyset$. Then the minimization problem

$$\min_{(\widetilde{f}, \widetilde{g}) \in \mathcal{F}_\Omega} \|f - \widetilde{f}\|_2^2 + \|g - \widetilde{g}\|_2^2$$

has an attainable global minimum.

For the less general version of the theorem, given arbitrary $f, g \in \mathbb{R}[t][\partial;']$ we define

$$\mathcal{S} = \mathcal{S}(f, g) = \left\{ \phi(\widetilde{f}, \widetilde{g}) \mid \widetilde{f}, \widetilde{g} \in \mathbb{R}[t][\partial;'] \text{ such that } \delta(\widetilde{f}) \leq \delta(f) \text{ and } \delta(\widetilde{g}) \leq \delta(g) \right\}.$$

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.

Theorem 3.3.4. *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*

$$\mathcal{F}_\Omega = \left\{ \begin{array}{ll} (\widetilde{f}, \widetilde{g}) \in \mathbb{R}[t][\partial;'] \times \mathbb{R}[t][\partial;'] \text{ such that} & \deg_\partial \widetilde{f} = M, \\ & \deg_\partial \widetilde{g} = N, \\ & \phi(\widetilde{f}, \widetilde{g}) \in \mathcal{S}, \\ & \deg_\partial \widetilde{h} \geq D, \\ & \widetilde{h} = \operatorname{gcrd}(\widetilde{f}, \widetilde{g}), \\ & \|\widetilde{h}\| \leq \Omega, \\ & \text{and } \operatorname{lcoeff}_t(\operatorname{lcoeff}_\partial(\widetilde{h})) = 1. \end{array} \right\}$$

Suppose that $\mathcal{F}_\Omega \neq \emptyset$. *Then the minimization problem*

$$\min_{(\widetilde{f},\widetilde{g})\in\mathcal{F}_\Omega} \|f - \widetilde{f}\|_2^2 + \|g - \widetilde{g}\|_2^2 \tag{3.3}$$

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 ρ in $\mathbb{R}[t]^\ell$. Let $\mathcal{H}_{\ell,\rho}$ denote the set of all differential polynomials over $\mathbb{R}[t][\partial;']$ of degree ℓ with coefficients ρ. We optimize over the continuous real objective function

$$\Phi(h, f^*, g^*) = \|f - f^*h\|_2^2 + \|g - g^*h\|_2^2,$$

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 Φ 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 (3.3). 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 Φ has a value on a closed and bounded set (i.e., compact with respect to the Euclidean metric) that is smaller than elsewhere. Hence Φ attains a global minimum by Theorem 3.3.2.

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 $\Phi(\widetilde{h}, \widetilde{f}^*, \widetilde{g}^*) > \Phi(h, f^*, g^*)$ 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 3.3.1 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 3.3.1.

Thus we can restrict the domain of Φ 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 $\zeta^{-1}(\mathcal{S} \cap \zeta(B))$, which is a compact set. □

We note that Theorem 3.3.3 does not guarantee a unique minimum of Φ, merely that Φ has an attainable minimum (opposed to 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 is ill-conditioned, then f^* and g^* can be ill-conditioned. Furthermore, choosing a poorly structured h can result in a Φ that cannot be minimized for the specified structure, but would otherwise have a minimum for a different choice of h. The following example illustrates this.

Example 3.3.5 ([15]). *Consider $f = \partial^2$ and $g = \partial^2 + 1$. We are looking for $\widetilde{f}, \widetilde{g} \in \mathbb{R}[t][\partial;']$ such that $\deg_\partial \operatorname{gcrd}(\widetilde{f}, \widetilde{g}) = 1$. In this instance $\operatorname{gcrd}(f, g) = \gcd(f, g)$ as differential polynomials with constant coefficients are isomorphic to $\mathbb{R}[t]$. We can write*

$$\widetilde{f} = \widetilde{f}_0 + \widetilde{f}_1\partial + \widetilde{f}_2\partial^2 \text{ and } \widetilde{g} = \widetilde{g}_0 + \widetilde{g}_1\partial + \widetilde{g}_2\partial^2.$$

We impose the constraints that $\widetilde{f}_0 = \widetilde{f}_2 = \widetilde{g}_2 = 0$ and $\widetilde{g}_0 = 1$. Thus a family of solutions is given by $\widetilde{f} = 0$ and $\widetilde{g} = \varepsilon\partial + 1$. This family corresponds to $f^* = 0$, $g^* = \varepsilon$ and $h = \partial + 1/\varepsilon$. We now have that

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

If it were possible to have

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

then we would have that under our constraints $\widetilde{f} = 0$ and $\widetilde{g} = 1$, which are relatively prime. In this case, the minimum distance of 2 is not attainable. This happens because $\|\operatorname{lcoeff}_t h\|$ can become arbitrarily small or $\|h\|$ can become arbitrarily large depending on our choice of g^*.

In this example we see that under the given constraints, the objective function does not have an attainable minimum. In order to guarantee a minimum, we need to make sure that h is well conditioned.

Example 3.3.6. *Consider $f = \partial^2 - 2\partial + 1$ and $g = \partial^2 + 2\partial + 2$. 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 $\deg_\partial \operatorname{gcrd}(\widetilde{f}, \widetilde{g}) = 1$ and $\|f - \widetilde{f}\|_2^2 + \|g - \widetilde{g}\|_2^2$ is minimized.*

The real monic Karmakar-Lakshman distance [16, 17] of

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

occurs when the rational function

$$\frac{2\gamma_0^4 + 14\gamma_0^2 + 4\gamma_0 + 5}{\gamma_0^4 + \gamma_0^2 + 1} \tag{3.4}$$

is minimized for $\gamma_0 \in \mathbb{R}$. The minimum value (if it exists) of this function corresponds to the approximate GCRD $h = \partial - \gamma_0$. There are 3 critical points. Two critical points correspond to a local and global maximum value. The other critical point corresponds to a local minimum (that is larger than 2). The infimum is 2, which is unattainable. There is no attainable global minimum.

The non-monic real Karmakar-Lakshman distance is 2, which is achieved if and only if the leading coefficient vanishes. The minimum distance occurs when the rational function

$$\frac{5\gamma_1^4 - 4\gamma_1^3 + 14\gamma_1^2 + 2}{\gamma_1^4 + \gamma_1^2 + 1}$$

is minimized. The minimum value of this function corresponds to the approximate GCRD $h = \gamma_1\partial + 1$.

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

$$\min_{\left\{(\widetilde{f},\widetilde{g}) \in \mathbb{R}[t][\partial;']^2 \mid \deg_\partial \operatorname{gcrd}(\widetilde{f},\widetilde{g}) = 1\right\}} \|f - \widetilde{f}\|_2^2 + \|g - \widetilde{g}\|_2^2$$

is not defined in the monic case. In the non-monic case, if a minimum exists then it occurs when $\operatorname{lcoeff}_\partial$ 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 $\operatorname{lcoeff}_t \operatorname{lcoeff}_\partial h = 1$ and $\|h\|$ is bounded, from Theorem 3.3.3 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 Φ. 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.

3.3.2 Convergence of Newton Iteration and Conditioning

Now we know a solution to the approximate GCRD problem exists. We show that $\nabla^2\Phi$, the Hessian matrix of Φ is positive definite around a global minimum, when the residual is sufficiently small. If we consider structured perturbations, then we are able to obtain results similar to that of Zeng and Dayton [28] 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

$$\begin{aligned} r &= r(h, f^*, g^*) \\ &= (\mathbf{f}^*\mathbf{h} - \mathbf{f}, \mathbf{g}^*\mathbf{h} - \mathbf{g})^T \in \mathbb{R}^{\eta\times 1} \end{aligned}$$

where

$$\eta = \sum_{0\le i\le M} \max\{-1, \deg_t f_i\} + \sum_{0\le i\le N} \max\{-1, \deg_t g_i\} + (M+1)+(N+1) \le (M+N+2)(d+1).$$

Intuitively, η represents the number of components of $(\mathbf{f}, \mathbf{g}) \in \mathbb{R}^{1\times\eta}$. Let ν be the number of variables needed to represent the coefficients of h, f^* and g^*, i.e. $(\mathbf{h}, \mathbf{f}^*, \mathbf{g}^*) \in \mathbb{R}^{1\times\nu}$.

Recall that when $f = f^*h$, we can linearize this relationship with differential convolution matrices, by writing

$$f = (f_0^*, \ldots, f_{M-D}^*)\mathcal{C}_{M-D}^{\mathbf{R}}(h).$$

If f_i is a coefficient of f with $\deg_t f_i = d$, then we may write

$$f_i = \sum_{0\le j\le M-D} f_j^*(\mathcal{C}_{M-D}^{\mathbf{R}}(h)[j, i])$$

This relationship may be linearized over $\mathbb{R}$ through the use of convolution matrices. Writing

$$\mathbf{f_i} = \sum_{0 \le j \le M-D} \mathbf{f_j^*} \cdot C_d \left(\mathcal{C}^{\mathbf{R}}_{M-D}(h)[j,i] \right)^T,$$

we now have a direct method of computing $\mathbf{f_i}$ in terms of the coefficients of f^* and h.

If we differentiate $\mathbf{f^*h}$ with respect to an entry from $\mathbf{f^*}$, then we will obtain the corresponding (linearized) row of $\mathcal{C}^{\mathbf{R}}_{M-D}(h)$. Similarly, differentiating $\mathbf{f^*h}$ with respect to an entry of $\mathbf{h}$ 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 $\mathbf{g^*h}$ with respect to variables from $\mathbf{g^*}$ and $\mathbf{h}$ 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 $\operatorname{lcoeff}_t \operatorname{lcoeff}_\partial h$ 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 $\operatorname{lcoeff}_t \operatorname{lcoeff}_\partial h$, since $\operatorname{lcoeff}_t \operatorname{lcoeff}_\partial h$ 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 $\gcrd(f^*, g^*) = 1$, J is rank deficient by 1 and the ν^{th} column is a linear combination of the other columns. The following lemma formalizes this statement.

Lemma 3.3.7 ([26]). *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 $\gcrd(f^*, g^*) = 1$, then all non-zero columns of J are linearly independent.*

Proof. Let $\vec{e}_\nu \in \mathbb{R}^{1 \times \nu}$ be a unit vector whose last component is 1. We write

$$\vec{e}_\nu (0, \ldots, 0, \mathbf{h})^T = \operatorname{lcoeff}_t \operatorname{lcoeff}_\partial h \neq 0.$$

We shall prove the equivalent statement that the matrix

$$\begin{pmatrix} J \\ \vec{e}_\nu \end{pmatrix} = \begin{pmatrix} \mathcal{C}^{\mathbf{R}}_{M-D}(h)^T & 0 & \mathcal{C}^{\mathbf{L}}_{D}(f^*) \\ 0 & \mathcal{C}^{\mathbf{R}}_{N-D}(h)^T & \mathcal{C}^{\mathbf{L}}_{D}(g^*) \\ & & \vec{e}_\nu \end{pmatrix} \in \mathbb{R}^{(\eta+1)\times\nu}$$

has full rank.

It follows that there exists $q_1, q_2, p \in \mathbb{R}[t][\partial;']$ such that their combined coefficient vector satisfies

$$\begin{pmatrix} J \\ \vec{e}_\nu \end{pmatrix} \begin{pmatrix} \mathbf{q_1}^T \\ \mathbf{q_2}^T \\ -\mathbf{p}^T \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

Expressing this as multiplication over $\mathbb{R}[t][\partial;']$, we have that

$$f^*p = q_1 h,$$
$$g^*p = q_2 h.$$

We conclude that $\operatorname{gcrd}(f^*p, g^*p) = p$, as $\operatorname{gcrd}(f^*, g^*) = 1$. If $p = 0$, then we are done (as $\mathbb{R}[t][\partial;']$ is a domain). We suppose that $p \neq 0$. It follows that $\operatorname{lclm}(f^*p, g^*p) = \operatorname{lclm}(f^*, g^*)$ and so we must have that $\operatorname{lclm}(q_1 h, q_2 h) = \operatorname{lclm}(q_1, q_2)$, hence $\operatorname{lclm}(q_1, q_2) = \operatorname{lclm}(f^*, g^*)$. There exist $\sigma, \tau, \widehat{\sigma}, \widehat{\tau} \in \mathbb{R}[t][\partial;']$ such that $\sigma f^* = \tau g^* = \widehat{\sigma} q_1 = \widehat{\tau} q_2$. Since $\mathbb{R}[t][\partial;']$ is a domain, we have $\sigma f^* p = \widehat{\sigma} q_1 h$ which implies $p = \alpha h$ for $\alpha \in \mathbb{R}$. Since $p = \alpha h$ we must have that $\alpha f^* = q_1$ and $\alpha g^* = q_2$. Now,

$$\vec{e}_\nu (0, \ldots, 0, \mathbf{h})^T = \operatorname{lcoeff}_t \operatorname{lcoeff}_\partial h \neq 0.$$

On the other hand,

$$\vec{e}_\nu (0, \ldots, 0, \alpha\mathbf{h})^T = 0.$$

This occurs if and only if $\alpha = 0$. But in this case $p = 0$ as well, so

$$\begin{pmatrix} \mathbf{q_1}^T \\ \mathbf{q_2}^T \\ -\mathbf{p}^T \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

It follows that the only vector in the null space is the zero vector, hence $\begin{pmatrix} J \\ \vec{e}_\nu \end{pmatrix}$ 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 $(\mathbf{f}^*, \mathbf{g}^*, \mathbf{h})^T$ 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 Φ, 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 3.3.8. *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 $\nabla^2\Phi(h, f^*, g^*)$ is positive definite.*

Proof. Let J be the Jacobian of r. J has full rank rank, so J^TJ has full rank and is positive semidefinite. If $r = 0$, at the global minimum we have that $2J^TJ = \nabla^2\Phi$, and $\nabla^2\Phi(h, f^*, g^*)$ is positive definite. □

When there is no residual, the Hessian $\nabla^2\Phi(h, f^*, g^*)$ is positive definite. It follows that if f and g are perturbed by a sufficiently small amount, then $\nabla^2\Phi$ remains locally positive definite, and 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 results, we make use of the fact that for any $f \in \mathbb{R}[t][\partial;']$, we have that $\|f\|_2 = \|\mathbf{f}\|_2$.

Lemma 3.3.9 ([26])**.** *Let $f, g, h, f^*, g^* \in \mathbb{R}[t][\partial;']$ be such that $\Phi(h, f^*, g^*) < \varepsilon$ 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*

$$\widehat{\Phi}(\widehat{h}, \widehat{f^*}, \widehat{g^*}) = \|\widehat{f} - \widehat{f^*}\widehat{h}\|_2^2 + \|\widehat{g} - \widehat{g^*}\widehat{h}\|_2^2 < \varepsilon.$$

Then,

$$\left\|(h - \widehat{h}, f^* - \widehat{f^*}, g^* - \widehat{g^*})\right\|_2^2 \leq \frac{1}{\sigma_{\nu-1}^2}\left(2\varepsilon + \left\|(f - \widehat{f}, g - \widehat{g})\right\|_2^2\right) + \textit{higher order terms.}$$

Proof. Let $J = J(r(h, f^*, g^*))$ be the Jacobian of the residuals from earlier in this section. We have that

$$(\mathbf{f}^*\mathbf{h} - \widehat{\mathbf{f}^*}\widehat{\mathbf{h}}, \mathbf{g}^*\mathbf{h} - \widehat{\mathbf{g}^*}\widehat{\mathbf{h}})^T \approx J(\mathbf{h} - \widehat{\mathbf{h}}, \mathbf{f}^* - \widehat{\mathbf{f}^*}, \mathbf{g} - \widehat{\mathbf{g}^*})^T.$$

Ignoring high order terms gives

$$\begin{aligned}\left\|(\mathbf{f}^*\mathbf{h} - \widehat{\mathbf{f}^*}\widehat{\mathbf{h}}, \mathbf{g}^*\mathbf{h} - \widehat{\mathbf{g}^*}\widehat{\mathbf{h}})^T\right\|_2^2 &\approx \left\|J(\mathbf{h} - \widehat{\mathbf{h}}, \mathbf{f}^* - \widehat{\mathbf{f}^*}, \mathbf{g}^* - \widehat{\mathbf{g}^*})^T\right\|_2^2 \\ &\geq \sigma_{\nu-1}^2 \left\|(\mathbf{h} - \widehat{\mathbf{h}}, \mathbf{f}^* - \widehat{\mathbf{f}^*}, \mathbf{g}^* - \widehat{\mathbf{g}^*})\right\|_2^2.\end{aligned}$$

A straightforward application of the triangle inequality gives

$$\begin{aligned}\left\|(h - \widehat{h}, f^* - \widehat{f^*}, g^* - \widehat{g^*})\right\|_2^2 &\leq \frac{1}{\sigma_{\nu-1}^2} \left\|(f^*h - \widehat{f^*}\widehat{h}, g^*h - \widehat{g^*}\widehat{h})\right\|_2^2 \\ &\leq \frac{1}{\sigma_{\nu-1}^2} \left(\Phi(h, f^*, g^*) + \widehat{\Phi}(\widehat{h}, \widehat{f^*}, \widehat{g^*}) + \left\|(f - \widehat{f}, g - \widehat{g})\right\|_2^2\right) \\ &\leq \frac{1}{\sigma_{\nu-1}^2} \left(2\varepsilon + \left\|(f - \widehat{f}, g - \widehat{g})\right\|_2^2\right) + \text{higher order terms.}\end{aligned}$$

□

Corollary 3.3.10. *Suppose that $h_{opt}, f_{opt}^*, g_{opt}^* \in \mathbb{R}[t][\partial;']$ are a locally unique global minimum of Φ in some neighborhood around h, f^* and g^*. If*

$$\Phi(h, f^*, g^*) < \varepsilon \text{ and } \Phi(h_{opt}, f_{opt}^*, g_{opt}^*) < \varepsilon$$

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.

We have now shown that locally, the approximate GCRD problem is well-posed. we have shown that a solution exists, the solution is (locally) unique, and that structured perturbations in the input are bounded in terms of the output. Furthermore, around this minimum we have that $\nabla^2\Phi$ is positive definite. As a consequence, Newton iteration will converge quadratically.

Chapter 4

Implementation of Approximate GCRD

This chapter discusses the particulars and implementation of the algorithms in this thesis. The algorithms are described in a Maple-like pseudo code, with Matlab style matrix indexing. All of the algorithms have been implemented in the Maple programming language. For convenience, the notation and assumptions introduced at the start of Chapter 3 will hold, unless otherwise stated. Additionally, we will assume that content can be removed numerically, as computed quantities are typically not primitive.

The presentation and theoretical analysis of the algorithms is presented in a bottom-up manner, reflecting their dependencies. Asymptotic upper bounds on the number of floating point operations required are provided. Furthermore, we discuss if the output of the algorithm can be certified in some manner, when applicable.

We demonstrate the robustness of our algorithms in practice. Specific examples are provided to thoroughly demonstrate the steps of the algorithms. We investigate interesting families of input. In particular, we investigate exact inputs with an exact GCRD, and perturbed differential polynomials with varying errors and noise introduced. The test cases of differential polynomials of interest to us have

- low degree in t and high degree in ∂ (unbalanced in ∂),
- high degree in t and low degree in ∂ (unbalanced in t), and
- proportional degrees in t and ∂ (balanced degrees).

4.1 Algorithms for Approximate GCRD

We provide an algorithmic description of the algorithms introduced in Section 3.1. The algorithms of Section 3.1.3 are required to compute an exact GCRD numerically. These algorithms compute the rank of the differential Sylvester matrix and a least squares solution to a polynomial linear system, corresponding to the Bézout coefficients. We describe the algorithm for finding nearby differential polynomials introduced in Section 3.2, whose (inflated) differential Sylvester matrix is nearly singular [9]. Using the least squares numeric GCRD algorithm, we can compute an approximate GCRD candidate from the nearly singular differential Sylvester matrix.

This approximate GCRD candidate will be used in a post-refinement Newton iteration. The initial guess for the co-factors are computed using a least squares division. Our initial guess is the tuple $(h_{init}, f^*_{init}, g^*_{init}) \in \mathbb{R}[t][\partial;']\times\mathbb{R}(t)[\partial;']^2$. We describe how to perform Newton iteration when the coefficients of the co-factors are polynomial, then we generalize this to co-factors with rational function coefficients.

Numerical Computation of a GCRD

Before we can compute a GCRD numerically, the rank of the differential Sylvester matrix needs to be determined. Our numeric rank algorithm is an adaptation of the rank algorithm used by Corless, Gianni, Trager and Watt [5]. There are

$$(M+N)(\mu+d+1)-(M+N)(\mu+1)=(M+N)d=\mu/2$$

trivial singular values, and $\mu/2<\mu+d+1$, the column block size. These trivial singular values need to be accounted for when annihilating small singular values. In the full rank case, we should not underestimate the rank of S by inferring from $\widehat{S}$, as there are strictly fewer trivial singular values than the column block size.

Algorithm 1 computes a reasonable guess for the degree in ∂ of an approximate GCRD. This algorithm is not certified to produce the degree of an approximate GCRD in general. However, this does not appear to be a problem in most of our experiments. When $\gcrd(f,g)$ is non-trivial (no errors present in the input coefficients), we compute the degree of the GCRD of f and g. In the exact setting, we can now formulate a linear algebra problem over $\mathbb{R}[t]$ to compute a GCRD. We present two solutions to this problem. Algorithm 2 solves this problem using linear algebra over $\mathbb{R}(t)$. Algorithm 3 linearizes the problem over $\mathbb{R}$ and computes a least squares solution.

Algorithm 1 : Deflated Rank

Input:

- An inflated differential Sylvester matrix $\widehat{S} \in \mathbb{R}^{(M+N)(\mu+1)\times(M+N)(\mu+d+1)}$;
- A search radius $\varepsilon_{rank} > 0$ for comparing singular values.

Output:

- The the numeric rank ϱ of the (non-inflated) differential Sylvester matrix S.

1: Compute the singular values $\sigma_1, \sigma_2, \ldots, \sigma_{(M+N)(\mu+d+1)}$ of $\widehat{S}$ in descending order.
2: Find the maximum k such that $\sigma_k > \varepsilon \frac{\sqrt{(M+N)(2\mu+d+2)}}{\mu+d+1}$ and $\sigma_{k+1} < \varepsilon$.
3: if $\sigma_k > \varepsilon$ for all k then $\widehat{S}$ has full rank.
4: If there is no significant change (there is no maximum k) between σ_k and σ_{k+1} for all k, as determined by step 2 then return failure.
5: Set $\varrho = \left\lceil \frac{k}{\mu+d+1} \right\rceil$, the scaled rank of S.

In the implementation of Algorithm 2, we take special care to ensure that $\operatorname{lcoeff}_t \operatorname{lcoeff}_\partial h$ does not vanish when h is normalized. If $\operatorname{lcoeff}_t \operatorname{lcoeff}_\partial h$ vanishes, then this could be an indication that the input is ill conditioned or content removal failed. In either case, it is possible that this instance of the approximate GCRD problem will not have an attainable global minimum in accordance with Theorem 3.3.3.

We can avoid the need for content removal if we introduce weights on the coefficients of h and seek a total least squares solution. Penalizing coefficients that correspond to higher degrees ensures that the solution of minimal norm is the primitive solution.

Algorithm 2 : NumericGCRD

Input:

- $f, g \in \mathbb{R}[t][\partial;']$ with $\|f\| = \|g\| = 1$;
- A search radius $\varepsilon_{rank} > 0$.

Output:

- $h = \operatorname{gcrd}(f, g) \in \mathbb{R}[t][\partial;']$ with $\deg_\partial h \geq 1$,
- or an indication that f and g are co-prime within search radius ε_{rank}.

1: $M \leftarrow \deg_\partial f$, $N \leftarrow \deg_\partial g$, $d \leftarrow \max\{\deg_t f, \deg_t g\}$, $\mu \leftarrow 2(M+N)d$.
2: $S \leftarrow S(f, g) \in \mathbb{R}[t]^{(M+N)\times(M+N)}$.
3: Form the inflated differential Sylvester matrix $\widehat{S} = \widehat{S}(f, g) \in \mathbb{R}^{(M+N)(\mu+1)\times(M+N)(\mu+d+1)}$ of S.
4: Compute the numerical rank ϱ of S using Algorithm 1 on $\widehat{S}$ with search radius ε_{rank}.
5: If $\varrho > 0$, then set $\deg_\partial h = D = M + N - \varrho$. Otherwise indicate that f and g are co-prime with respect to ε_{rank} and return.
6: Solve for $w \in \mathbb{R}[t][\partial;']^{1\times(M+N)}$ from

$$wS = (\underbrace{*, *, \ldots, *}_{D+1 \text{ entries}}, 0, \ldots, 0),$$

ensuring that $\|\operatorname{lcoeff}_t((wS)[D+1])\|_2 \gg 0$.
7: Set $(h_0, h_1, \ldots, h_D, 0, \ldots, 0) = wS$.
8: **return** $\operatorname{cont}(h)^{-1}h$.

Nearby Differential Polynomials with GCRD Algorithm

We use the method of Giesbrecht and Haraldson [9] to compute nearby differential polynomials with a GCRD. We recover nearby differential polynomials $\widetilde{f}$ and $\widetilde{g}$ whose corresponding differential Sylvester matrix is *almost singular*, i.e. it is reasonably close to a singular matrix with respect to $\|\cdot\|_F$. The output polynomials $\widetilde{f}$ and $\widetilde{g}$ are by no means certified to have an exact GCRD, or be close to a global minimum of the approximate GCRD problem. However, they are a suitable starting point for iterative post-refinement methods, such as Newton's method.

The matrix $\widehat{S}$ is highly structured, as it is composed of block Toeplitz matrices. When we consider the matrix $\widehat{S} + \Delta\widehat{S}$, the nearest inflated differential Sylvester matrix of prescribed rank deficiency, we have considerable flexibility in how we recover the coefficients of $\widetilde{f}$ and $\widetilde{g}$. The matrix $\widehat{S} + \Delta\widehat{S}$ is probably not an inflated differential Sylvester ma-

Algorithm 3 : NumericGCRD via LS

Input:

- $f, g \in \mathbb{R}[t][\partial;']$ with $\|f\| = \|g\| = 1$;
- $\varepsilon_{rank} > 0$ used to compute the degree of the GCRD.

Output:

- $h \in \mathbb{R}[t][\partial;']$ that is primitive with a fixed leading coefficient such that $\|wS(f,g) - h\|_2^2$ is minimized.

1: $M \leftarrow \deg_\partial f$, $N \leftarrow \deg_\partial g$, $d \leftarrow \max\{\deg_t f, \deg_t g\}$, $\mu \leftarrow 2(M+N)d$.
2: Compute D using Algorithm 1 with ε_{rank}.
3: $\delta \leftarrow (\underbrace{\mu + d, \ldots, \mu + d}_{D+1}, 0, \ldots 0)$ (or another valid initial guess).
4: Compute a least squares solution of h from $\|wS(f,g) - h\|_2$ with $\delta(h) = \delta$ and $\operatorname{lcoeff}_t \operatorname{lcoeff}_\partial h = 1$.
5: $\delta \leftarrow \delta(\operatorname{cont}(h)^{-1}h)$.
6: Compute a new least squares solution of h from $\|wS(f,g) - h\|_2$ with $\delta(h) = \delta$ and $\operatorname{lcoeff}_t \operatorname{lcoeff}_\partial h = 1$.
7: **return** h.

trix, however according to Lemma 3.2.2, it is reasonably close to one. We recall that the mapping $\Gamma : \mathbb{R}[t] \to \mathbb{R}^{(\mu+1)\times(\mu+d+1)}$ generates the (non-square) Toeplitz blocks of $\widehat{S}$. To recover the coefficients of $\widetilde{f}$ and $\widetilde{g}$, we look at possible definitions for the mapping $\Gamma^{-1} : \mathbb{R}^{(\mu+1)\times(\mu+d+1)} \to \mathbb{R}[t]$. We apply this mapping on the appropriate blocks of $\widehat{S} + \Delta\widehat{S}$. More formally, we use Γ^{-1} to find $\widetilde{f}, \widetilde{g} \in \mathbb{R}[t][\partial;']$ such that $\widehat{S}(\widetilde{f}, \widetilde{g}) \approx \widehat{S}(f,g) + \Delta\widehat{S}(f,g)$.

One implementation of Γ^{-1} is to use a single row from the blocks corresponding to the coefficients of $\widetilde{f}$ and $\widetilde{g}$. We also consider weighted averages of each row block. In the weighted averages, we apply Γ^{-1} on rows 1 through $\mu + 1$ of $\widehat{S} + \Delta\widehat{S}$ to compute the coefficients of $\widetilde{f}$. Likewise, we apply Γ^{-1} on rows $N(\mu+1)+1$ through $(N+1)(\mu+1)$ to compute the coefficients of $\widetilde{g}$. In our implementation, we weight each row equally, leading to a uniformly weighted block average to compute $\widetilde{f}$ and $\widetilde{g}$. In both instances, we require that the partial orderings satisfy $\delta(\widetilde{f}) \leq \delta(f)$ and $\delta(\widetilde{g}) \leq \delta(g)$. There is further redundancy in the coefficients in other rows, but we choose not formulate another possible mapping of Γ^{-1} to exploit this.

Regardless of our choice of Γ^{-1}, this method of recovering $\widetilde{f}$ and $\widetilde{g}$ can lead to a differential Sylvester matrix that does not have the desired numeric rank, as determined by Algorithm 1. The perturbation $\Delta\widehat{S}$ is unstructured while $\Gamma(\widetilde{f}_i)$ and $\Gamma(\widetilde{g}_j)$ are (highly structured) Toeplitz matrices. Consequently, some non-zero terms of $\Delta\widehat{S}$ are ignored.

Algorithm 4 : Deflated Perturbation

Input:

- $f, g \in \mathbb{R}[t][\partial;']$ with $\|f\| = \|g\| = 1$;
- Perturbed inflated differential Sylvester matrix $\widehat{S} + \Delta\widehat{S} \in \mathbb{R}^{(M+N)(\mu+1)\times(M+N)(\mu+d+1)}$;
- $\Gamma^{-1} : \mathbb{R}^{(\mu+1)\times(\mu+d+1)} \to \mathbb{R}[t]$.

Output:

- $\widetilde{f}, \widetilde{g} \in \mathbb{R}[t][\partial;']$ where $\delta(\widetilde{f}) \leq \delta(f)$ and $\delta(\widetilde{g}) \leq \delta(g)$.

1: $M \leftarrow \deg_\partial f$, $N \leftarrow \deg_\partial g$, $d \leftarrow \max\{\deg_t f, \deg_t g\}$, $\mu \leftarrow 2(M+N)d$ and $N \leftarrow \mu+d+1$.
2: **for** $0 \leq i \leq \deg_\partial f$ **do**
3: $\quad [I, J] \leftarrow [1 : \mu + 1][(i + 1) + (i - 1)N(\mu + d + 1) : (i + 1)N(\mu + d + 1)]$
4: $\quad \widetilde{f}_i \leftarrow \Gamma^{-1}\left((\widehat{S} + \Delta\widehat{S})[I, J]\right)$
5: **end for**
6: **for** $0 \leq i \leq \deg_\partial g$ **do**
7: $\quad [I, J] \leftarrow [N(\mu+1)+1 : (N+1)(\mu+1)][(i+1)+(i-1)M(\mu+d+1) : (i+1)M(\mu+d+1)]$
8: $\quad \widetilde{g}_i \leftarrow \Gamma^{-1}\left((\widehat{S} + \Delta\widehat{S})[I, J])\right)$
9: **end for**
10: **return** $\widetilde{f}$ and $\widetilde{g}$.

Numeric Right Division

Numeric right division without remainder between two differential polynomials is a rational function linear algebra problem. The (approximate) quotient is a solution to a triangular linear system, in a least squares sense. The solution to this system may not be in (approximate) lowest terms. We use approximate GCD and real linear least squares to resolve this.

In the special case when $\operatorname{lcoeff}_\partial h = 1$, the output of this algorithm is also in lowest terms. Even if the co-factors in lowest terms have polynomial coefficients, small round-off errors prevent this algorithm from obtaining them. Performing division of computed polynomial approximate GCDs does not typically produce a least squares solution. The division will perturb the coefficients of the co-factor. Instead, we use real linear least squares once we know the degree structure of the coefficients of f^*. The next algorithm, Algorithm 7 computes $f^* \in \mathbb{R}(t)[\partial;']$ in lowest terms such that $\|f - f^*h\|_2$ is minimized.

Algorithm 5 : Nearby Differential Polynomials with GCRD

Input:

- $f, g \in \mathbb{R}[t][\partial;']$ with $\|f\| = \|g\| = 1$;
- A search radius $\varepsilon_{rank} > 0$, used to validate the degree of h.

Output:

- $\widetilde{f}, \widetilde{g} \in \mathbb{R}[t][\partial;']$ where $\delta(\widetilde{f}) \leq \delta(f)$, ($\delta(\widetilde{g}) \leq \delta(g)$ and $h \approx \text{gcrd}(\widetilde{f}, \widetilde{f}) \in \mathbb{R}[t][\partial;']$ with $\deg_\partial h \geq 1$, or ;
- An indication that f and g are co-prime within search radius ε_{rank}.

1: $M \leftarrow \deg_\partial f$, $N \leftarrow \deg_\partial g$, $d \leftarrow \max\{\deg_t f, \deg_t g\}$ and $\mu \leftarrow 2(M+N)d$.
2: $S \leftarrow S(f,g) \in \mathbb{R}[t]^{(M+N)\times(M+N)}$.
3: $\widehat{S} \leftarrow \widehat{S}(f,g) \in \mathbb{R}^{(M+N)(\mu+1)\times(M+N)(\mu+d+1)}$.
4: Compute the SVD of $\widehat{S}$, where $\widehat{S} = P\Sigma Q$ with P, Σ and Q as discussed in Section 3.2.
5: Compute the numerical rank ϱ of S using Algorithm 1 on $\widehat{S}$ with search radius ε_{rank}.
6: If $\varrho > 0$ set the last $\varrho(\mu + d + 1)$ singular values to 0 and compute $\overline{\Sigma}$ as discussed in Section 3.2. Otherwise indicate that f and g are co-prime with respect to ε_{rank}.
7: Compute $\widehat{S} + \Delta\widehat{S} = P\overline{\Sigma}Q$.
8: Compute $\widetilde{f}$ and $\widetilde{g}$ from $\widehat{S} + \Delta\widehat{S}$ using Algorithm 4.
9: Compute $h = \texttt{NumericGCRD}(\widetilde{f}, \widetilde{g})$ using Algorithm 3, with ε_{rank} used to validate the degree of h using Algorithm 1.
10: **return** $\widetilde{f}, \widetilde{g}$ and h.

GCRD via Optimization: Newton's Method

Using Algorithm 5, we can compute an initial guess for an approximate GCRD, h_{init}. We can perform right division without remainder numerically to compute initial guesses for the co-factors, f^*_{init} and g^*_{init}. We now have enough information to set up a post-refinement Newton iteration, to hopefully compute an approximate GCRD. When the co-factors have polynomial coefficients, the products f^*h and g^*h are always polynomial. This makes Newton iteration a very straightforward procedure, as the objective function

$$\Phi(h, f^*, g^*) = \|f - f^*h\|_2^2 + \|g - g^*h\|_2^2$$

is easily computed. However, when the co-factors have rational function coefficients, the quantities f^*h and g^*h usually have rational function coefficients due to round-off error. The rational function coefficients typically arise due to round-off errors. Using Corollary 3.1.4, we can clear fractions and compute the least squares solution of an equivalent associate problem.

Algorithm 6 : Numeric Right Division

Input:

- $f, h \in \mathbb{R}[t][\partial;']$ with $\|f\| = \|h\| = 1$.

Output:

- $f^* \in \mathbb{R}(t)[\partial;']$ satisfying $f = f^*h$.

1: $M \leftarrow \deg_\partial f$, $D \leftarrow \deg_\partial h$.
2: Form the matrix $\mathcal{M}(h)$ from Lemma 3.1.3.
3: Solve

$$(f_D, f_{D+1}, \ldots, f_M) = (f_0^*, f_1^*, \ldots, f_{M-D}^*)\mathcal{M}(h)$$

by backwards substitution for the coefficients of f^*.
4: **for** $0 \leq i \leq M - D$ **do**
5: $\quad f_i^* \leftarrow$ Approximate f_i^* in rational function least terms
6: **end for**
7: **return** f^*.

Algorithm 7 : Numeric Right Division via LS

Input:

- $f, h \in \mathbb{R}[t][\partial;']$ with $\|f\| = \|h\| = 1$.

Output:

- $f^* \in \mathbb{R}(t)[\partial;']$ in lowest terms satisfying $f = f^*h$.

1: $M \leftarrow \deg_\partial f$, $D \leftarrow \deg_\partial h$.
2: Form the matrix $\mathcal{M}(h)$ from Lemma 3.1.3.
3: Use Algorithm 6 to compute the degrees in t of the numerators and denominators of the coefficients belonging to f^*.
4: Solve

$$(f_D, f_{D+1}, \ldots, f_M) = (f_0^*, f_1^*, \ldots, f_{M-D}^*)\mathcal{M}(h)$$

by linear least squares for the coefficients of f^*.
5: **return** f^*.

The normalization we impose, that $\operatorname{lcoeff}_t \operatorname{lcoeff}_\partial h$ is fixed, ensures the solution is (locally) unique, in accordance with Corollary 3.3.8. We note that this normalization can be changed. However one must ensure that the normalization vector is not orthogonal to $(\mathbf{f}^*, \mathbf{g}^*, \mathbf{h})$. We now generalize the Newton iteration for the instance when the co-factors have rational function coefficients.

In the implementation we clear fractions. Without loss of generality, we restrict our

Algorithm 8 : Newton Iteration

Input:

- $f, g, h_{init} \in \mathbb{R}[t][\partial;']$ with $\|f\| = \|g\| = \|h_{init}\| = 1$;
- $k \in \mathbb{N}$, the number of iterations.

Output:

- $f^*, g^* \in \mathbb{R}[t][\partial;']$ and $h \in \mathbb{R}[t][\partial;']$ such that $\Phi(f^*h, g^*h)$ is locally minimized and
- $\delta(f^*h) \leq \delta(f)$ and $\delta(g^*h) \leq \delta(g)$.

1: $M \leftarrow \deg_\partial f$, $N \leftarrow \deg_\partial g$ and $D \leftarrow \deg_\partial h_{init}$.
2: Compute initial guesses of f^* and g^* using Algorithm 7.
3: $\operatorname{lcoeff}_t \operatorname{lcoeff}_\partial h \leftarrow \operatorname{lcoeff}_t \operatorname{lcoeff}_\partial h_{init}$.
4: $x^0 \leftarrow (\mathbf{f^*_{init}}, \mathbf{g^*_{init}}, \mathbf{h_{init}})^T$.
5: **for** $1 \leq i \leq k$ **do**
6: Solve $\nabla^2\Phi(x^i) \cdot x^{i+1} = \nabla^2\Phi(x^i) \cdot x^i - \nabla\Phi(x^i)$ for x^{i+1}.
7: **end for**
8: **return** f^*, g^* and h computed from x^k.

Algorithm 9 : Modified Newton Iteration

Input:

- $f, g, h_{init} \in \mathbb{R}[t][\partial;']$ with $\|f\| = \|g\| = \|h_{init}\| = 1$;
- $k \in \mathbb{N}$, the number of iterations.

Output:

- $f^*, g^* \in \mathbb{R}(t)[\partial;']$ and $h \in \mathbb{R}[t][\partial;']$ such that $\Phi(f^*h, g^*h)$ is locally minimized and
- $\delta(f^*h) \leq \delta(f)$ and $\delta(g^*h) \leq \delta(g)$.

1: $M \leftarrow \deg_\partial f$, $N \leftarrow \deg_\partial g$ and $D \leftarrow \deg_\partial h_{init}$.
2: Compute initial guesses of f^* and g^* using Algorithm 7.
3: $\operatorname{lcoeff}_t \operatorname{lcoeff}_\partial h \leftarrow \operatorname{lcoeff}_t \operatorname{lcoeff}_\partial h_{init}$.
4: $f \leftarrow f^*_{-1} f, g \leftarrow g^*_{-1} g, f^* \leftarrow h_D^{M-D+1} f^*$ and $g^* \leftarrow h_D^{N-D+1}$.
5: $x^0 \leftarrow (\mathbf{f^*_{init}}, \mathbf{g^*_{init}}, \mathbf{h_{init}})^T$.
6: **for** $1 \leq i \leq k$ **do**
7: Solve $\nabla^2\Phi(x^{i+1}) \cdot x^i = \nabla^2\Phi(x^i) \cdot x^i - \nabla\Phi(x^i)$ for x^{i+1}.
8: **end for**
9: $f^* \leftarrow \frac{1}{f^*_{-1}} f^*$ and $g^* \leftarrow \frac{1}{g^*_{-1}} g^*$.
10: **return** f^*, g^* and h computed from x^k.

discussion on how to clear fractions for the term $f^* \in \mathbb{R}(t)[\partial;']$. Clearing fractions is not entirely straightforward. We know that h_D^{M-D+1} is a common denominator of the

coefficients of f^*, however using $f^*_{-1} = h_D^{M-D+1}$ is not efficient. In practice we choose f^*_{-1} that satisfies $\deg_t f^*_{-1} = \deg_t f^*_{init-1}$, that is we use degree of our initial guess. The function $\|f^*_{-1}(f - f^*h)\|_2^2$ is still quartic. The problem encountered with this method is that f^* may no longer be in lowest terms if f^*_{-1} is chosen poorly. Using a different common denominator we can resolve this.

Recall that we can write

$$f^* = \sum_{0 \le i \le M-D} \frac{f^*_i}{f^*_{-1_i}} \partial^i, \text{ for } f^*_i, f^*_{-1_i} \in \mathbb{R}[t].$$

We can use

$$f^*_{-1} = \prod_{0 \le i \le M-D} f^*_{-1_i} \text{ for } f^*_{-1_i} = \sum_{0 \le j \le \deg_t f^*_{-1_i}} f^*_{-1_i,j} t^j$$

as a common denominator for the coefficients of f^*. This choice of f^*_{-1} preserves the structure of each coefficient of f^*, ensuring that f^* will be in lowest terms. There are two concerns that warrant further discussion. The first concern is that there are more variables needed, resulting in a larger Hessian matrix. The other concern is that the term $\|f^*_{-1}(f - f^*h)\|_2^2$ is not necessarily quartic. As a multivariate polynomial, the total degree depends on $\deg f^*_{-1}$, which can be quite large. Specifically, when $\deg_t h_D > 1$, we have

$$\deg \left(\prod_{0 \le i \le M-D} f^*_{-1_i} \right) = O(\deg_t h_D^{M-D+1}).$$

4.2 Performance Analysis

In this section we address the computational complexity in terms of the number of floating point operations or *flops*. Where applicable, we discuss the numerical stability of the algorithms and whether or not their output can be certified. The algorithms are analyzed in the order they were presented in the previous section.

Analysis of Algorithm 1 `Deflated Rank`

The number of flops Algorithm 1 requires is dominated by the cost of performing the SVD on $\widehat{S}$. The SVD requires $O((M+N)^3(\mu+d+1)^3) = O((M+N)^6 d^3)$ flops, using standard arithmetic.

Analysis of Algorithm 2 `Numeric GCRD`

The number of flops Algorithm 2 requires is ultimately bounded by the cost of computing the rank of S using Algorithm 1. The cost of Algorithm 1 is $O((M+N)^6 d^3)$ flops. The cost of computing a GCRD given the degree in ∂ is $O((M+N)^3)$ operations over $\mathbb{R}(t)$ which corresponds to $O((M+N)^3 d^2)$ flops. The cost of the approximate GCD and division to remove content depends on the specific method used, but is usually negligible when compared to the rank computation.

This algorithm is not numerically stable for large degree inputs in t and ∂. Performing linear algebra over $\mathbb{R}(t)$ leads to considerable degree growth in t, and removing (approximate) content with a division further perturbs the coefficients of the GCRD. The output of this algorithm is not certified to be correct in most instances.

Analysis of Algorithm 3 `Numeric GCRD via LS`

There are $(M+N)(\mu+d+1)$ equations and $(M+N)(\mu+1)+(D+1)(\mu+d+1) = O((M+N)^2 d)$ unknowns. The cost of computing the least squares solution is $O((M+N)^6 d^3)$ flops. The cost of inferring the content by looking at singular values of the (generalized) Sylvester matrix is bounded by $O((D)^3(\mu+d+1)^3) = O((M+N)^6 d^3)$ flops. The total number of flops required for this algorithm is $O((M+N)^6 d^3)$.

This algorithm relies on solving a real linear least squares problem. As such, this algorithm is numerically stable, provided that the underlying least squares problem is reasonably conditioned, and solved in a reasonable way. One such method of solving the least squares problem is the SVD and arising pseudo-inverse. As linear least squares is an instance of positive semidefinite quadratic minimization, we are able to certify the correctness of the answer obtained, provided that the underlying approximate GCD algorithm computes the degree of the content correctly.

Analysis of Algorithm 4 `Deflated Perturbation`

The number of flops Algorithm 4 requires is is $O((M+N)^2 d^2)$, assuming that Γ^{-1} uses the weighted block average. The mean minimizes the sum of squares, so this is not a bad approximation to the nearest structured matrix. However it is not certified to provide meaningful output.

Analysis of Algorithm 5 `Nearby With GCRD`

The number of flops Algorithm 5 requires is is dominated by the cost of computing the singular values of $\widehat{S}$, which is $O((M+N)^6 d^3)$ flops.

This algorithm is exactly the same as Algorithm 2 when $\widehat{S}$ has the desired rank deficiency. In the event that the input is approximate, the quality of our answer depends on the largest singular value of $\widehat{S}$ that we annihilate. Lemma 3.2.2 provides an upper bound on how much we perturb $\widehat{S}$ to obtain the desired singular matrix. This algorithm is not certified to provide meaningful output, but if used in conjunction with Algorithm 3, the output can be certified as a least squares approximation to the solution of the Bézout coefficients.

Analysis of Algorithm 6 `Numeric Right Division`

The number of flops Algorithm 6 requires depends on the method used to solve the linear system. The particular system is highly structured so we can solve it by backwards substitution directly, which costs $O((M-D)^2)$ operations over $\mathbb{R}(t)$. This corresponds to $O((M-D)^2 d^2)$ flops. An upper bound on the degree required for approximate GCD computations is $\deg_t h_D^{M-D+1}$. The total cost of each approximate GCD computation is $O((\deg_t h_D^{M-D+1})^3)$ flops. There are at most $M-D+1$ approximate GCD computations performed, so the total cost of the algorithm is

$$\max\{O((M-D)^2(\max\{d, \deg_t h\})^2), O((\deg_t h_D^{M-D+1})^3)\}$$

flops.

If approximate GCD is not used, then the output of this algorithm is certified to be a least squares solution, but not a total least squares solution. If we assume that $\operatorname{lcoeff}_t \operatorname{lcoeff}_\partial h = 1$ and $\|h\|$ is not arbitrarily large, then the backwards substitution is well conditioned. The approximate GCD computations and following divisions can perturb the coefficients, so the algorithm can be unstable for poorly conditioned inputs. This is especially problematic when $\operatorname{lcoeff}_\partial h$ is poorly conditioned.

Analysis of Algorithm 7 `Numeric Right Division via LS`

If f^* has polynomial coefficients, then $\deg_t f^* \leq \deg_t f \leq d$ as f and h have polynomial coefficients as well. If f^* has rational function coefficients, we know that h_D^{M-D+1} is a

common denominator of f^*, so we need at most $(M-D+1)(2\deg_t h_D^{M-D+1}+d+1)$ unknowns. Clearing fractions, there are $(M-D+1)(\deg_t h_D^{M-D+1}+d+1)$ equations, as we ignore equations corresponding to $f_0,\ldots,f_{D-1}$ (assuming we use $\mathcal{M}(h)$). The cost of solving this linear least squares problem is $O((M-D)^3(\deg_t h_D^{M-D+1}+d)^3)$ flops.

The output of this algorithm is certified as a linear least squares solution. Like the previous algorithm, Algorithm 6, the conditioning of this algorithm is strongly related to the conditioning of h_D.

Analysis of Algorithms 8 and 9 `(Modified) Newton Iteration`

We transform the problem of computing a GCRD to that of optimizing $\Phi : \mathbb{R}[t][\partial;']\times\mathbb{R}(t)[\partial;']^2 \to \mathbb{R}$. Using Corollary 3.1.4, we can assume without loss of generality that f^* and g^* have polynomial coefficients. The dominating cost of the Newton iteration is solving a linear system to get the next value which requires $O(\nu^3)$ operations, where ν is the number of variables needed to represent the coefficients of h, f^* and g^*.

Newton iteration can fail for many reasons, however our Newton iteration usually fails because:

1. $\nabla^2\Phi$ is positive semidefinite at a point in the iteration; the stationary point is a saddle point;
2. The initial guess is poorly chosen and $\nabla^2\Phi$ is indefinite at a point.

In the event that Newton iteration fails we can perform a Gauss-Newton iteration instead. Despite Gauss-Newton iteration having linear convergence, J^TJ is positive definite, so saddle points are no longer a problem. Furthermore, tools such as the Cholesky decomposition are applicable in computing the next step in the iteration. According to Corollary 3.3.8, if the residual is sufficiently small then Newton iteration will converge to a global minimum.

4.3 Examples and Experimental Results

This section contains some examples of our implementation. The (inflated) differential Sylvester matrix is ill-conditioned for large degree inputs in t and ∂. We restrict ourselves to modest examples with minimal coefficient growth.

Example 4.3.1 (No Noise, many factors)**.**

$$\begin{aligned}
f =&.00769\partial^5 + (.00035t^2 + .05386t - .05386)\partial^4 \\
&+(.00140t^3 + .06820t^2 - .16928t + .17313)\partial^3 \\
&+(-.09513t^3 + .22559t^2 + .16928t - .33472)\partial^2 \\
&+(.18607t^3 - .65720t^2 - .04617t + .32702)\partial \\
&+(-.09234t^3 + .36305t^2 - .00769t - .11927). \\
g =&(.01001t - .01001)\partial^5 + (.04019t^2 - .07007t + .03003)\partial^4 \\
&+(.00063t^3 - .01048t^2 + .15014t - .11010)\partial^3 \\
&+(.27901t^3 - .32921t^2 - .09008t + .17016)\partial^2 \\
&+(-.55990t^3 + .52909t^2 - .04004t - .08007)\partial \\
&+(.28026t^3 - .22959t^2 + .04004t).
\end{aligned}$$

We compute initial guesses (removing content numerically where appropriate)

$$\begin{aligned}
h_{guess} =&.09285\partial^3 + (.37139t - .27854)\partial^2 + (-.74278t + .27854)\partial + (.37139t - .09285), \\
f^*_{guess} =&.08287\partial^2 + (.00377t^2 + .24862t - .33150)\partial + \\
&(-2.05844 \times 10^{-10}t^3 - .24862t^2 + .91162t - .04144), \\
g^*_{guess} =&(.10780t - .10780)\partial^2 + (.00168t^2 + 8.67540 \times 10^{-9}t - 2.71283 \times 10^{-9})\partial \\
&+(2.35115 \times 10^{-8}t^3 + .75463t^2 - .43122t + 6.78976 \times 10^{-8}).
\end{aligned}$$

The quality of this initial guess is

$$\|f - f^*_{guess}h_{guess}\|_2^2 + \|g - g^*_{guess}h_{guess}\|_2^2 = 4.04506 \times 10^{-14}.$$

The condition number for the Hessian matrix valuated at our initial guess is 18354.38336 *and our smallest eigenvalue is* .00314*. Since* $\nabla^2\Phi$ *is locally positive definite, we know that we will converge to a unique (local) minimum. The minimum we converge to is* 2.33030×10^{-20}*.*

The exact GCRD in this example is $h = (\partial + 4t - 1)(\partial - 1)(\partial - 1)$*.*

Example 4.3.2 (Noise)**.** *In this example we introduced normalized noise of size* 10^{-5} *to* f

and g.

$$\begin{aligned}
f =&.00583\partial^5 \\
&+(-9.45614\times10^{-7}t^3+.00027t^2+.03498t-.03498)\partial^4 \\
&+(-8.26797\times10^{-7}t^5+.04743t^3+.01113t^2-.05247t+.07287)\partial^3 \\
&+(-9.08565\times10^{-8}t^5+.13885t^4-.21623t^3+.30950t^2-.17781t-.05247)\partial^2 \\
&+(-.18655t^5-.02226t^4-.20166t^3-.41974t^2+.33812t-.10202)\partial \\
&+(.18655t^5-.30315t^4+.43935t^3-.22868t^2-.13117t+.15740). \\
g =&(.00780t-.00779)\partial^5 \\
&+(9.10928\times10^{-7}t^5+6.83196\times10^{-7}t^3+.02351t^2-.07018t+.02729)\partial^4 \\
&+(5.94796\times10^{-8}t^4+.02376t^3-.07822t^2+.12086t-.06238)\partial^3 \\
&+.16326t^4+.04654t^3-.27267t^2+.12476t+.03898)\partial^2 \\
&+(-.21833t^5-.10868t^4-.05617t^3+.63939t^2-.38597t+.14036)\partial \\
&+(.21833t^5-.27291t^4-.01462t^3-.09418t^2+.24952t-.12086).
\end{aligned}$$

We compute initial guesses (removing content numerically where appropriate)

$$\begin{aligned}
h_{guess} =&.11192\partial^3+(.33514t-.22357)\partial^2 \\
&+(-.44667t^2-.22358t-.11327)\partial+.44754t^2-.55869t+.22453, \\
f^*_{guess} =&(5.97992\times10^{-8}t^5-.00001t^4+.05212-8.67362\times10^{-18}t^2+5.20417\times10^{-18}t)\partial^2 \\
&+(-.0001t^5-.00002t^4-.00001t^3+.00238t^2+.15646t-.20842)\partial \\
&+(.00003t^5-+.41663t^3-.15629t^2+.57193t-.02463), \\
g^*_{guess} =&(-2.10091\times10^{-8}t^5+-6.93889\times10^{-18}t^3-1.73472\times10^{-17}t^2+.06967t-.06963)\partial^2 \\
&+(.00002t^4+.00001t^3+.00146t^2-.27937t+.10474)\partial \\
&+(-.00004t^5+.00002t^4+.48596t^3+.00189t^2-.27763t-.00158).
\end{aligned}$$

The quality of this initial guess is

$$\|f-f^*_{guess}h_{guess}\|_2^2+\|g-g^*_{guess}h_{guess}\|_2^2=.00003.$$

The condition number for the Hessian matrix valuated at our initial guess is 21971.20356 *and our smallest eigenvalue is* .00818. *Since $\nabla^2\Phi$ is locally positive definite, we know that we will converge to a unique (local) minimum. The minimum we converge to is* 1.06759×10^{-10}, *which is roughly the amount of noise we added.*

Example 4.3.3 (GCRD via LS)**.** *In this example we added a noise factor of* 10^{-4} *to* f *and* g*. Performing Linear Algebra over* $\mathbb{R}(t)$ *produced completely unacceptable answers, so we used a Least Squares algorithm to compute an approximate GCRD.*

$$\begin{aligned}
f =&(.11329t^6 + .23414t^5 + .12840t^4 + .00755t^3 + .00005)\partial^3 \\
&+(.00001t^6 + .23414t^5 + .59667t^4 + .02269t^3 - .04528t^2 - .02266t + 3.67436 \times 10^{-7})\partial^2 \\
&+(-.11329t^6 + .33231t^5 - .43054t^4 - .00754t^3 - .00003t^2 - .06798t + .00003)\partial \\
&(-.00001t^6 - .23414t^5 + .34741t^4 + .01510t^3 - .06799t^2 + .09064t + .00004). \\
g =&(.01938t^4 - .03876t^3 - .07752t^2 + .03876t + .05819)\partial^3 \\
&+(.13567t^4 + .23252t^3 - .07750t^2 - .34879t + .29066)\partial^2 \\
&+(-.01938t^4 + .13563t^3 + .03873t^2 + .25195t - .23257)\partial \\
&+(-.13562t^4 + .44570t^3 - .56198t^2 - .03874t + .17439).
\end{aligned}$$

Using a Least Squares variant of our Numeric GCRD algorithm, we are able to compute (without removing content)

$$\begin{aligned}
h_{guess} =&(t^2 + 1.94162t + .93768)\partial^2 + 2.87182\partial \\
&+(-.94502t^2 + 2.84696t - 3.82712), \\
f^*_{guess} =&(.00712t^6 - .01655t^5 + .71917t^4 + .05630t^3 - .00048t^2 - .00199t + .00155)\partial \\
&+(-.00061t^6 - .01248t^5 + .02319t^4 + .04309t^3 - .02430t^2 - .12387t - .00820), \\
g^*_{guess} =&(.00041t^4 - .00715t^3 + .13885t^2 - .50330t + .36942)\partial \\
&+(.00381t^4 - .01139t^3 + .86465t^2 - .48856t + .00231).
\end{aligned}$$

The quality of this initial guess is

$$\|f - f^*_{guess}h_{guess}\|_2^2 + \|g - g^*_{guess}h_{guess}\|_2^2 = .00328.$$

The condition number for the Hessian matrix valuated at our initial guess is 148.62547 *and our smallest eigenvalue is* .04615*. Since* $\nabla^2\Phi$ *is locally positive definite, we know that we will converge to a unique (local) minimum. The minimum we converge to is* 9.53931×10^{-9}*.*

General Examples

We provide results that demonstrate the robustness of our algorithms. We consider differential polynomials whose degrees in t and ∂ are balanced and unbalanced. The coefficients

of the inputs were generated using the Maple routine `randpoly()`. The inputs f and g were normalized so that $\|f\| = \|g\| = 1$. We introduced normalized noise to the coefficients of f and g, so that the relative error is size of the perturbation. Precisely, if $f + \Delta f$ and $g + \Delta g$ are perturbed from f and g by the quantities Δf and Δg, then the relative error in the coefficients of f and g is given by $\|\Delta f\|_2 = \|\Delta g\|_2$.

We recall that the Newton iteration optimizes $\|f-\widetilde{f}\|_2^2+\|g-\widetilde{g}\|_2^2$, which is the *sum of the squares* of the errors. The initial error and error from post-refinement are expressed as the sum of square errors accordingly. In all of the examples when there were no perturbations in the coefficients of f and g, our numeric GCRD algorithm and post-refinement procedures were able to compute an exact GCRD to machine precision. When perturbations imposing a relative error of 10^{-8} in the coefficients of f and g were introduced, we were able to compute a solution to the approximate GCRD problem in every example.

Introducing perturbations imposing a relative error of order 10^{-4} and 10^{-2} into the coefficients of f and g prevented computation of an approximate GCRD in some examples. Instead, we provide examples of the largest perturbation in the coefficients of f and g that we were able to compute an approximate GCRD.

Balanced Degrees in t and ∂

The following results of experiments were conducted on differential polynomials whose degrees in t and ∂ were proportional, or balanced.

Example	Input (∂, t)	GCRD (∂, t)	Noise	Initial Error	Newton Error
1	(2,2)	(1,1)	1e-2	2.63579e-3	9.37365e-5
2	(2,2)	(1,1)	1e-2	6.98136e-4	8.96068e-5
3	(3,2)	(2,1)	1e-2	1.69968e-2	1.26257e-4
4	(3,4)	(2,2)	1e-2	3.8269e-3	1.04271e-4
5	(4,4)	(3,2)	1e-2	3.15314e-1	FAIL
5	(4,4)	(3,2)	1e-4	9.29336e-7	8.97294e-9

Unbalanced Degrees in ∂

The following results of experiments were conducted on differential polynomials whose degrees in ∂ were relatively larger than their degree in t.

Example	Input (∂, t)	GCRD (∂, t)	Noise	Initial Error	Newton Error
1	(2,2)	(1,1)	1e-2	1.13109e-3	2.90713-e5
2	(3,2)	(2,1)	1e-2	6.72179e-4	1.13998e-4
3	(4,2)	(3,1)	1e-2	3.00365e-4	1.04038e-4
4	(5,2)	(4,1)	1e-2	9.01982e-4	1.23557e-4
5	(6,2)	(5,1)	1e-2	6.61552e-3	FAIL
5	(6,2)	(5,1)	1e-4	2.74084e-4	1.12566 e-8

Unbalanced Degrees in t

The following results of experiments were conducted on differential polynomials whose degrees in t were relatively larger than their degree in ∂.

Example	Input (∂, t)	GCRD (∂, t)	Noise	Initial Error	Newton Error
1	(2,3)	(1,2)	1e-2	1.27092e-2	1.43153e-4
2	(2,6)	(1,4)	1e-2	5.04286e-1	FAIL
2	(2,6)	(1,4)	1e-4	7.78993e-4	1.31180e-8
3	(2,8)	(1,6)	1e-4	6.9361e-2	FAIL
3	(2,8)	(1,6)	1e-8	3.92268e-10	1.15653e-16
4	(2,11)	(1,8)	1e-8	6.20749e-10	1.26549e-16
5	(2,13)	(1,10)	1e-8	2.23588e-10	1.03136e-16

Chapter 5

Conclusion

In this thesis we have formally presented our approach to the approximate GCRD problem. 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. Furthermore, the results of this thesis are a foundation that can be generalized to other non-commutative polynomials in a Euclidean domain. We believe that this work and related problems can also be approached from a structured low-rank approximation [13, 24]. In particular, most the results of this thesis can be adapted to Ore polynomials, whose coefficient field is $\mathbb{R}[t]$. This is true in general, as we use resultant-like and convolution-like matrices to linearize most operations. A particular example amenable to our results is the shift operator, commonly associated with linear difference equations.

The results of this thesis have many rich applications in working with approximate differential polynomials. Computation of an approximate GCRD enables computation of a corresponding approximate LCLM, by solving a linear least squares problem. The tools developed for approximate GCRD provide an initial approach to approximate factorization of differential polynomials. Furthermore, we believe that these results can be easily extended to computing the approximate GCRD of many differential polynomials, through the use of a generalized differential Sylvester matrix.

Although we do not have certification results on the degree of an approximate GCRD, we can obtain a reasonable guess by enumerating over the degrees of all possible approximate GCRDs, similar to the Structured Total Least Norm approach adopted for multivariate polynomial approximate GCD [14]. A possible direction of research for certification, would be to look at the differential subresultant sequence and the singular values of their

inflated block matrices [7].

References

[1] B. Beckermann and G. Labahn. When Are Two Numerical Polynomials Relatively Prime? *Journal of Symbolic Computation*, 26:677–689, 1998.

[2] B. Botting, M. Giesbrecht, and J.P. May. Using the Riemannian SVD for Problems in Approximate Algebra. In *Proc. Workshop on Symbolic-Numeric Computation (SNC'05)*, pages 209–219, 2005.

[3] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[4] M. Bronstein and M. Petkovšek. An introduction to pseudo-linear algebra. *Theoretical Computer Science*, 1996.

[5] R. M. Corless, P. M. Gianni, B. M. Trager, and S. M. Watt. The Singular Value Decomposition for Polynomial Systems. In *Proc. International Symposium on Symbolic and Algebraic Computation (ISSAC'95)*, pages 189–205, 1995.

[6] 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.

[7] Ioannis Z. E., A. Galligo, and H. Lombardi. Certified approximate univariate . *Journal of Pure and Applied Algebra*, 117–118:229–251, 1997.

[8] K. Geddes, S.R. Czapor, and G. Labahn. *Algorithms for Computer Algebra*. Kluwer Academic Publishers, 1992.

[9] M. Giesbrecht and J. Haraldson. Computing GCRDs of Approximate Differential Polynomials. In *Proc. Symposium on Symbolic-Numeric Computation (SNC '14)*, pages 78–87. ACM, 2014.

[10] M. Giesbrecht, J. Haraldson, and E. Kaltofen. Computing GCRDs of Approximate Differential Polynomials. *manuscript*, 2015.

[11] G. Golub and C. van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, USA, 4th edition, 2013.

[12] E. Kaltofen, M. S. Krishnamoorthy, and D. B. Saunders. Fast parallel computation of Hermite and Smith forms of polynomial matrices. *SIAM Journal on Algebraic Discrete Methods*, 8(4):683–690, 1987.

[13] E. Kaltofen, Z. Yang, and L. Zhi. Structured Low Rank Approximation of a Sylvester Matrix. In *Proc. Workshop on Symbolic-Numeric Computation (SNC'05)*, pages 69–83, 2005.

[14] E. Kaltofen, Z. Yang, and L. Zhi. Approximate greatest common divisors of several polynomials with linearly constrained coefficients and singular polynomials. In *Proc. International Symposium on Symbolic and Algebraic Computation (ISSAC'06)*, pages 169–176, 2006.

[15] E. Kaltofen, Z. Yang, and L. Zhi. Unpublished Manuscript. 2007.

[16] 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, 1996.

[17] N. Karmarkar and Y. N. Lakshman. On approximate GCDs of univariate polynomials. *Journal of Symbolic Computation*, 26(6):653–666, 1998.

[18] M. A. Laidacker. Another Theorem Relating Sylvester's Matrix and the Greatest Common Divisor. *Mathematics Magazine*, 42(3):pp. 126–128, 1969.

[19] 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.

[20] 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.

[21] O. Ore. Theory of non-commutative polynomials. *Annals of Mathematics. Second Series*, 34:480–508, 1933.

[22] W. Rudin. *Principles of Mathematical Analysis*. International series in pure and applied mathematics. McGraw-Hill, 1976.

[23] A. Schönhage. Quasi-GCD computations. *J. Complexity*, 1:118–137, 1985.

[24] E. Schost and P. J. Spaenlehauer. A quadratically convergent algorithm for structured low-rank approximation. *Foundations of Computational Mathematics*, to appear.

[25] J. von zur Gathen, J.Gerhard. *Modern Computer Algebra*. Cambridge University Press, New York, NY, USA, 2 edition, 2003.

[26] Z. Zeng. The Approximate GCD of Inexact Polynomials. Part 1: a univariate algorithm. *preprint*, 2004.

[27] Z. Zeng. Computing multiple roots of inexact polynomials. *Mathematics of Computation*, 74(250):869–903, 2005.

[28] 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, 2004.