Structured Total Least Squares for Approximate Polynomial Operations

by

Brad Botting

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, 2004

Brad Botting

Acknowledgements

Thanks to the many members of the Ontario Research Centre for Computer Algebra, not only those at the University of Waterloo, but also at the University of Western Ontario and Wilfrid Laurier University. Their continued cooperation and discussion has been invaluable. Particular thanks to Robert Corless, Ilias Kotsireas and George Labahn for insightful discussions in this area during ORCCA meetings.

Recent developments in approximate polynomial operations, including contributions from those mentioned above, as well as Erich Kaltofen, John May, Shuhong Gao and many others, provided the motivation for this work.

Finally, a special thanks to Mark Giesbrecht, for patience, guidance and endless insight during the development of this thesis.

Abstract

This thesis presents techniques for accurately computing a number of fundamental operations on approximate polynomials. The general goal is to determine nearby polynomials which have a non-trivial result for the operation.

We proceed by first translating each of the polynomial operations to a particular structured matrix system, constructed to represent dependencies in the polynomial coefficients. Perturbing this matrix system to a nearby system of reduced rank yields the nearby polynomials that have a non-trivial result.

The translation from polynomial operation to matrix system permits the use of emerging methods for solving sophisticated least squares problems. These methods introduce the required dependencies in the system in a structured way, ensuring a certain minimization is met. This minimization ensures the determined polynomials are close to the original input.

We present translations for the following operations on approximate polynomials:

- Division
- Greatest Common Divisor (GCD)
- Bivariate Factorization
- Decomposition

The Least Squares problems considered include classical Least Squares (LS), Total Least Squares (TLS) and Structured Total Least Squares (STLS). In particular, we make use of some recent developments in formulation of STLS, to perturb the matrix system, while maintaining the structure of the original matrix. This allows reconstruction of the resulting polynomials without applying any heuristics or iterative refinements, and guarantees a result for the operation with zero residual.

Underlying the methods for the LS, TLS and STLS problems are varying uses of the

Singular Value Decomposition (SVD). This decomposition is also a vital tool for determining appropriate matrix rank, and we spend some time establishing the accuracy of the SVD. We present an algorithm for *relatively accurate* SVD recently introduced in [8], then used to solve LS and TLS problems. The result is confidence in the use of LS and TLS for the polynomial operations, to provide a fair contrast with STLS. The SVD is also used to provide the starting point for our STLS algorithm, with the prescribed guaranteed accuracy.

Finally, we present a generalized implementation of the Riemannian SVD (RiSVD), which can be applied on any structured matrix to determine the result for STLS. This has the advantage of being applicable to all of our polynomial operations, with the penalty of decreased efficiency. We also include a novel, yet naive, improvement that relies on randomization to increase the efficiency, by converting a rectangular system to one that is square.

The results for each of the polynomial operations are presented in detail, and the benefits of each of the Least Squares solutions are considered. We also present distance bounds that confirm our solutions are within an acceptable tolerance.

Contents

Chapter 1

Introduction

When working with polynomials in practical applications, it is often the case that the coefficients are only known to some prescribed accuracy. This restriction may be due to measurement limitations, previous computation, or even the limits of physical storage. We call polynomials with this restriction *approximate polynomials* .

In order to work sensibly with approximate polynomials, even basic polynomial operations must be carefully considered. For example, dividing two approximate polynomials will most likely be impossible using traditional methods. The polynomials

$$\begin{aligned} p &= 37.1336x^2 + 5.6102x - 67.9573, \\ q &= -5.32x - 7.61, \end{aligned}$$

are divisible, with $p/q = -6.98x + 8.93$. However, if we are limited in our measurement of p to 2 decimal places, then

$$\begin{aligned} p &= 37.13x^2 + 5.61x - 67.95, \\ q &= -5.32x - 7.61, \end{aligned}$$

are not divisible. In this case, we seek "nearby" polynomials that are divisible.

This thesis focuses on approximate polynomial inputs, however, solving this problem is not limited to this case. The methods presented here can be used whenever a polynomial operation yields a trivial result, and we seek a nearby "more interesting" case. For example, if a polynomial does not factor, it may be instructive to find the nearest polynomial that does. A simple extension determines a radius of inapplicability of a polynomial operation, with benefits as in [16].

1.1 Approximate Polynomials

We use the term *approximate polynomial* to refer to a polynomial intuitively known to have some error (from measurement, computational roundoff, etc.) in it's coefficients. For example, the approximate univariate polynomial $p \in \mathbb{R}[x]$:

$$p = p_n x^n + p_{n-1}x^{n-1} + \cdots + p_1 x + p_0,$$

having errors in it's coefficients p_i of Δp_i, is actually a perturbed version of some polynomial $\hat{p} \in \mathbb{R}[x]$, where

$$\begin{aligned} \hat{p} &= p + \Delta p \\ &= (p_n + \Delta p_n)x^n + \cdots + (p_0 + \Delta p_0). \end{aligned}$$

Hence, for a general approximate polynomial $p \in \mathbb{R}[x_1 \ldots x_n]$, there is an (implicit) implication that there exists (or that there is suspected to exist) a polynomial $\hat{p} \in \mathbb{R}[x_1 \ldots x_n]$ such that

$$\hat{p} = p + \Delta p,$$

for some $\Delta p \in \mathbb{R}[x_1 \ldots x_n]$, having "small" coefficients. Δp shall be called the *perturbation* of the polynomial $\hat{p}$, that resulted in the approximate polynomial p[1]. The goal of applying operations to approximate polynomials shall be to make the coefficients of Δp as small as possible, while ensuring that $\hat{p}$, the determined polynomial has some desired property, such as factorization.

[1]The sign of Δp is interchanged here from the previous formulae. This is acceptable since the size of Δp is the important thing.

Hence, for each input polynomial p, we recover **a** polynomial $\hat{p}$ that is close to p. There is no guarantee that the recovered polynomial will be **the** polynomial $\hat{p}$ from which p is intuitively formed. We strive to find the closest polynomial to p that gives a non trivial result, which may turn out to be closer to p than the full-accuracy polynomial $\hat{p}$.

In order to formalize the concepts of "nearby", "close", and "small" as mentioned, measures of distance between polynomials must first be discussed.

1.2 Polynomial Norms

As with many mathematical objects that contain numeric values, there are a variety of norms defined for polynomials. The most obvious of this is the *coefficient 2-norm* , defined on an input $f \in \mathbb{R}[x]$:

$$f = f_n x^n + f_{n-1}x^{n-1} + \dots f_1 x + f_0$$

to be the square root of the sum of the squares of the absolute values of the coefficients, i.e.

$$||f||_2 = \sqrt{\sum_{i=1}^{n} |f_i|^2}.$$

This norm is, in fact, a specific instance of the more general coefficient l_p-norm,

$$||f||_p = \left(\sum_{i=1}^{n} |f_i|^p \right)^{1/p} .$$

One additional norm, sometimes called the *polynomial height*, occurs when $p \to \infty$. This corresponds to

$$||f||_\infty = \max_i |f_i|.$$

These norms satisfy the properties consistent with defining a norm (non-negativity, scaling invariance, and the triangle inequality on a vector containing the polynomial coefficients).

They can thus be used to reasonably define the distance between two polynomials g, h as $||g - h||_p$ for a given norm l_p. Minimizing this distance will be the goal throughout this thesis, whenever "nearby" polynomials are sought.

We shall further use small polynomial norms to classify a polynomial as having desired small coefficients. Here it is important to note the effect of different choices of norm.

Consider the polynomials

$$\begin{aligned} f(x) &= \sqrt{98.0}x^{10} + \sqrt{0.2}x^9 + \sqrt{0.2}x^8 + \cdots + \sqrt{0.2}x^1 + \sqrt{0.2}x^0, \\ g(x) &= \sqrt{10.0}x^{10} + \sqrt{10.0}x^9 + \sqrt{10.0}x^8 + \cdots + \sqrt{10.0}x^1 + 0x^0. \end{aligned}$$

Both of these polynomials will have l_2 norm equal to $\sqrt{100} = 10$, however, the l_1 and l_∞ norms are:

$$\begin{aligned} ||f||_1 = \sqrt{98} + 10\sqrt{10} \approx 14.37, \quad ||f||_\infty = 98.0, \\ ||g||_1 = 10\sqrt{10} \approx 31.62, \quad ||g||_\infty = 10.0. \end{aligned}$$

Clearly, the application must be considered before a polynomial norm is chosen. This thesis uses the coefficient 2-norm, l_2, unless otherwise stated.

In order to better demonstrate the effects of each algorithm, independent of the size of the polynomial coefficients, we measure *relative error* of the polynomials. For one input polynomial and a perturbation $f, \Delta f \in \mathbb{R}[x_1, \ldots, x_n]$, we measure

$$\frac{||f + \Delta f||}{||f||}.$$

For the case of two input polynomials f, g, we have

$$\left(\frac{||f + \Delta f||}{||f||}\right)^2 + \left(\frac{||g + \Delta g||}{||g||}\right)^2$$

as a measure of the combined relative error of each polynomial. We chose this metric[2] since it is essentially equivalent to starting the algorithm with polynomials f, g with $||f|| = ||g|| = 1$.

[2]alternative choice would be $\frac{||f+\Delta f||^2 + ||g+\Delta g||^2}{||f||^2 + ||g||^2}$

1.3 Polynomial Operations

Polynomial operations can often be performed by translating the dependencies that would yield a result into a matrix system, where the coefficients of the polynomial(s) are mapped to predefined positions in a matrix. This defines a linear basis for generating matrices that correspond to the polynomial operations.

The basis defines a *structure* on the matrix system, and from any matrix exhibiting this structure we can extract the coefficients of the polynomial(s) it represents. A simple example would be for the polynomial $f(x) = 3x + 2$, we can define the *structured matrix system* $\begin{bmatrix} 3 & 2 \\ 2 & 0 \end{bmatrix}$. Any structured matrix of the form $\begin{bmatrix} a & b \\ b & 0 \end{bmatrix}$ then corresponds to a polynomial, $f(x) = ax + b$. The basis for this particular translation would be

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \text{so that}$$

$$a \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + b \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} a & b \\ b & 0 \end{bmatrix}.$$

The translation of the polynomials involved in a polynomial operation to a structured matrix system permits the use of matrix algorithms to evaluate potential perturbations on the system. We carefully choose the structure of the matrix system, so that linear dependency in the matrix corresponds to dependencies in the coefficients of the polynomial that will yield a non-trivial result.

We determine results for the following polynomial operations defined on approximate polynomials:

- *Division*: Find the nearest polynomials $\hat{p}, \hat{q}$ to approximate polynomial inputs p, q such that $\hat{q}|\hat{p}$
- *GCD*: Find the nearest polynomials $\hat{f}, \hat{g}$ to approximate polynomial inputs f, g such that $\hat{f}, \hat{g}$ have a non-trivial GCD.

- *Bivariate Factorization*: Find the nearest polynomial $\hat{f}$ to approximate polynomial input f such that $\hat{f}$ is reducible to a product of non-trivial factors.
- *Decomposition*: Find the nearest polynomial $\hat{f}$ to approximate polynomial input f such that $\hat{f}$ is the composition of two non-trivial polynomials, i.e. $\hat{f} = g(h(x))$ for some g, h.

The structured matrix system chosen for each operation is linear in the coefficients of the polynomials, and typically of full rank. The goal is to transform this matrix system into one that is rank-deficient, while maintaining the same structure. This is accomplished by solving one of several least squares problems.

By reducing the rank, we are establishing dependencies in the matrix entries (and hence the polynomial coefficients). This perturbed matrix, along with a vector in its null space, is used to construct the perturbed polynomial(s) that yield the non-trivial result, and may provide that result as well.

1.4 Matrix Norms

As with polynomials, there are many norms that can be used to measure the size of a matrix. Since we will be translating our polynomial operations to matrix systems, we first present briefly some standard matrix norms.

Often it is useful to measure such things as the maximum absolute sum of a matrix row ($||M||_\infty$), a column ($||M||_1$), or some measure of the eigenvalues of the matrix ($||M||_2$). However, the matrix norm that most naturally follows the intuitive idea of Euclidean "closeness" is the Frobenius Norm ($||M||_F$).

$$||M||_F = \sqrt{\sum_{i}^{m}\sum_{j}^{n} M[i,j]^2}$$

This norm measures the magnitude of each entry of the matrix. We utilize the convention that $||M|| = ||M||_F$ unless otherwise stated.

1.5 Least Squares Problems

In order to introduce the required dependencies into these structured matrix systems, several variations of the classical Least Squares (LS) problem are solved. The classical LS problem dates back to Gauss in the early 19th century, originally formulated to fit observations to the predicted data.

LS can formulated in terms of matrices $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^{m\times 1}$. Historically, the matrix A would contain the set of observed data, with the goal of fitting it to an expected result b through a set of parameters x.

The determination of the parameter vector x is then thought of as solving a system of linear equations:

$$Ax \approx b.$$

The Least Squares problem is to determine x such that the following minimization holds:

$$min_{||\Delta b||}\big(\exists x \in \mathbb{R}^{n\times 1} : Ax = b + \Delta b\big). \tag{1.1}$$

Methods to solve the LS problem determine such an x by perturbing the vector b to introduce a deficiency. However, by allowing perturbation in the matrix A as well, we can increase the overall proximity of the matrix system to the original. This natural extension to the LS problem is called *Total Least Squares* (TLS).

The TLS problem is to find a vector x with the minimization:

$$min_{||\Delta A||,||\Delta b||}\Bigg(\exists x \in \mathbb{R}^{n\times 1} : (A + \Delta A)x = b + \Delta b\Bigg). \tag{1.2}$$

Finally, the Structured Total Least Squares (STLS) problem involves the same minimization as TLS, however the additional constraint that $A + \Delta A$ has the same linear structure as A is imposed. A solution to this problem will allow us to form the output polynomials without the concern of determining which matrix entry should be used for a certain polynomial coefficient.

1.6 Accurate Matrix Decompositions

Solutions to the Least Squares problems rely heavily on variations of the regular Singular Value Decomposition (SVD). This matrix decomposition has received significant interest [14] for numerical and computational reasons [3].

The SVD of a real matrix A is of the form $U\Sigma V^T$, where U, V are orthogonal ($UU^T = VV^T = 1$) matrices and Σ is a diagonal matrix whose entries are sorted in decreasing order from left to right. The i^{th} diagonal entry of Σ gives the minimal 2-norm distance from the matrix A to a matrix of rank $i - 1$.

Since the desired application involves small relative perturbations of the exact polynomials, we require algorithms that compute the SVD (and it's more sophisticated variations) accurately for the corresponding matrix systems.

In particular, attempts are made to determine the various decompositions to as much accuracy as the input data merits. Traditional algorithms for the SVD attain accuracy dependent on the conditioning of the input matrix [14]. However, by analyzing the perturbation of a matrix and it's effect on the SVD, it has been shown [8] that the entries of the input matrix may determine the decomposition to a much stronger accuracy.

This increased accuracy is used to guarantee precise results where the use of traditional algorithms do not, in particular for the application to the Least Squares problems.

1.7 Overview of Chapters

The remainder of this thesis is structured as follows:

Chapter 2 introduces the transformations of polynomial operations on approximately specified input polynomials into structured linear systems. This includes a brief outline of the

[3]In particular, it allows us to assign a value for the conceptual numeric rank of the matrix

algorithm to be used and the desired result.

Chapter 3 presents the least squares problems to be solved for our linear systems. The motivation behind each problem is illustrated by example, and algorithms that yield a solution are given. Results for matrices of various (linear) structures are presented.

Chapter 4 discusses the accuracy of the least squares solutions, based on the matrix algorithm used to derive them. The matrix algorithms themselves are presented and analyzed when permisible, and an extensive set of matrices are used to show their effectiveness.

Chapter 5 contains the resulting operations on approximate polynomials. This includes numerical evidence of the success of each of the presented techniques for different sets of data. The data is contrasted to yield a good estimate for the success and failure of the techniques under certain conditions and interpretations. The solutions and their distances from the input for each operation and technique are presented and discussed.

Chapter 6 concludes the work by presenting the achievements, and indicating the areas that could benefit from further study.

1.8 Conclusion

This thesis presents a set of polynomial operations and their translation to a corresponding structured matrix system. By applying techniques for least squares problems, implemented with careful matrix decompositions, meaningful results to these operations are obtained for nearby polynomials.

Chapter 2

Approximate Polynomial Operations

This chapter introduces the approximate polynomial operations that motivate the translation to the various least squares problems presented in this thesis. As mentioned in the introduction, these techniques are applied to a structured system that is derived from the coefficients of the input polynomials. The following sections detail the transformation of four polynomial operations to such (linear) systems, and where possible, the recovery of the resulting polynomials..

2.1 Division

Given two polynomials $p, q \in \mathbb{R}[x]$, polynomial division seeks to find a third polynomial, $r \in \mathbb{R}[x]$, such that $qr = p$.

When applied to approximate polynomials p, q, it is expected that there will be no such r, even if the perturbed polynomials $\hat{p}, \hat{q}$ are divisible. For example, consider the polynomials:

$$\begin{aligned} p &= 3.02x^2 + 6.98x + 2, \\ q &= 2.78x + 0.96. \end{aligned}$$

Inspecting these polynomials naturally leads one to consider $\Delta p = 0.02x^2 - 0.02x$, $\Delta q = -0.22x + -0.04$, resulting in

$$\begin{aligned} \hat{p} = p - \Delta p = 3x^2 + 7x + 2, \quad ||\Delta p|| = 0.02828427125, \\ \hat{q} = q - \Delta q = 3x + 1, \quad ||\Delta q|| = 0.2236067977. \end{aligned} \tag{2.1}$$

This is simply the perturbation we would expect upon correcting the polynomials p, q in the natural way, that being rounding the near-integer values to integers.

For this example, this rounding results in a pair of polynomials p, q that are divisible, since $p = q(x + 2)$. However it is clearly naive to simply round all real coefficients to integers and proceed from there. While this particular perturbation did yield a non trivial result, there may be choices for p, q that also give such a result, but are closer in norm to $\hat{p}, \hat{q}$. This could also be viewed as an optimization problem:

$$\min_{||\Delta p||^2 + ||\Delta q||^2} \left(\exists r \in \mathbb{R}[x] : \hat{p} + \Delta p = (\hat{q} + \Delta q)\, r \right).$$

This search for the minimal polynomial is the motivation for transforming the polynomial equation $\hat{p} = \hat{q}r$ into a matrix system, and solving least squares problems to find the corresponding *best* polynomials p, q, r.

For example (2.1), we actually determine the polynomials:

$$\begin{aligned} \hat{p} &= 3.021239746x^2 + 6.97632779x + 2.010877327, \\ \hat{q} &= 2.786505424x + 0.9407305076, \quad \text{and} \end{aligned}$$

$$\frac{\hat{p}}{\hat{q}} = 1.084239679x + 2.137570017.$$

The corresponding polynomial norms show significant improvement over (2.1):

$$\begin{aligned} ||\Delta p|| = 0.01154722214, \\ ||\Delta q|| = 0.02033799102. \end{aligned} \tag{2.2}$$

2.1.1 Matrix Representation

Solving the polynomial equation $p = qr$ can be viewed as a collection of constraints on the coefficients of p. For example, the coefficient of degree 2 of p must be the sum of all degree 2 terms resulting from multiplying q, r.

In general, the degree of q must also be considered, since it will be less than that of p, or else the division is trivial. Let $m = deg_x(q), n = deg_x(p) - m$, so that $deg_x p = m + n$. If $dq = min(d, n)$, then the degree d term be of form:

$$p_d = \sum_{i=0}^{dq} q_i r_{d-i}.$$

This can be written as a vector product equation as

$$\begin{bmatrix} q_0 & q_1 & \dots & q_{dq-1} & q_{dq} \end{bmatrix} \begin{bmatrix} r_d \\ r_{d-1} \\ \vdots \\ r_{d-dq+1} \\ r_{d-dq} \end{bmatrix} = p_d.$$

There will be one such equation for every possible coefficient of p. The system shall then be solved for the coefficients of r, which are contained in the vector $\vec{r}$. To get an idea for this structure, consider the equations of lowest degree.

$$\begin{aligned} q_0 r_0 &= p_0 \\ q_1 r_0 + q_0 r_1 &= p_1 \\ q_2 r_0 + q_1 r_1 + q_0 r_2 &= p_2 \\ &\vdots \end{aligned}$$

The matrix representation ($Q\vec{r} = \vec{p}$) becomes clear

$$\begin{bmatrix} q_0 & 0 & 0 & \dots & 0 & 0 \\ q_1 & q_0 & 0 & \dots & 0 & 0 \\ q_2 & q_1 & q_0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ q_n & q_{n-1} & q_{n-2} & \dots & q_{n-m+1} & q_{n-m} \\ 0 & q_n & q_{n-1} & \dots & q_{n-m+2} & q_{n-m+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & q_n \end{bmatrix} \begin{bmatrix} r_0 \\ r_1 \\ r_2 \\ \vdots \\ r_m \end{bmatrix} = \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ \vdots \\ p_{m+n} \end{bmatrix}$$

where $Q \in \mathbb{R}^{m+n \times n}$ is formed from the coefficients of q as indicated. Consider the rounded example (2.1):

$$\begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 0 & 3 & 1 \end{bmatrix} \begin{bmatrix} r_0 \\ r_1 \\ r_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 7 \\ 3 \end{bmatrix}.$$

A solution to this matrix equation is $r_0 = 2, r_1 = 1, r_2 = 0$, or $r = 2 + x$ as expected.

2.1.2 Determining Solutions

If there is no immediate solution to the matrix equation $Q\vec{r} = p$, then we could solve the Total Least Squares problem for minimal $\Delta Q, \Delta\vec{p}$ such that

$$(Q + \Delta Q)\vec{r} = \vec{p} + \Delta\vec{p} \tag{2.3}$$

for some $\vec{r}$. The polynomials $r, \hat{p} = p + \Delta p$ can be immediately extracted from the vectors $\vec{r}, \vec{p} + \Delta\vec{p}$. However, we have applied a perturbation ΔQ to the matrix representation of the polynomial q. If the perturbation does not maintain the structure of Q, then the solution that has been determined may not be meaningful, since the system will no longer correspond to a polynomial. The values of (2.3) may not yield divisible polynomials at all.

Since we can verify that $\hat{q}r = p$ relatively easily, a collection of heuristics may be applied to $Q + \Delta Q$ in an attempt to recover a suitable $\hat{q}$. It is difficult to assert that any polynomial $\hat{q}$ can be found in this way, let alone that such a polynomial is in any way optimal.

A second option is to apply such an algorithm iteratively, refining the input polynomials at each iteration. We have the relationship that for $p/q = r$, then $p/r = q$, so take the polynomials $r, p + \Delta p$ discovered in (2.3) and use them as input to the next iteration. This may eventually converge to a solution for (2.3) that has $Q + \Delta Q$ having the same structure as Q, at which point the polynomial $q + \Delta q$ can be extracted from $Q + \Delta Q$.

Instead of either of these options, we shall apply a method of solving (2.3) that preserves the structure of the matrix Q. With this condition, $\hat{q}$ can be extracted trivially along with $\hat{p}, r$ to yield the desired solution to $\hat{p} = \hat{q}r$.

2.1.3 Multivariate Polynomials

The approach outline for univariate polynomials can be logically extended to multivariate polynomial inputs. The idea remains the same, that being enforcing constraints on the coefficients of certain degrees. One caveat is that the matrix system generated will grow quickly, as can be imagined if one considers all of the contributing terms to the coefficient of f in $x^3y^2z^4$.

To deal with the expanding array of coefficients, we introduce some notation. Denote the coefficient of the polynomial $f \in \mathbb{R}[x_1 \dots x_n]$, in the variables $x_1^{e_1} \dots x_n^{e_n}$ as $f_{e_1,\dots,e_n}$.

Consider the following polynomials $p, q \in \mathbb{R}[x, y]$:

$$\begin{aligned} p &= 3x^2y + 2x^2 + 6xy^2 + 4xy, \\ q &= 3y + 2. \end{aligned} \tag{2.4}$$

One can verify that

$$\begin{aligned} p_{1,1} &= q_{0,0}r_{1,1} + q_{1,0}r_{0,1} + q_{1,1}r_{0,0} + q_{0,1}r_{1,0} \\ &= 2r_{1,1} + 3r_{1,0}. \end{aligned}$$

Before proceeding to the matrix system, a decision must be made on an ordering for the terms of a general multivariate polynomial.

Term Ordering

For a polynomial p in n variables, i.e., $p \in \mathbb{R}[x_1 \ldots x_n]$, the coefficients must be ordered in some way to allow consistent application of the matrix equations. This will, in fact, fix a basis for the matrix structure, as we shall see.

We have chosen lexicographical ordering, best illustrated by the following example of 3 variables x, y, z, with degree 2:

$$1 \quad x \quad x^2 \quad xy \quad xz \quad y \quad y^2 \quad yz \quad z \quad z^2$$

This choice directly affects the structure of the generating matrix system, so it may be instructive to consider other orderings, such as the Chebyshev basis. This will not change the operation of the algorithms, however it may lead to matrix systems that have an easier to determine solution to the Least Squares problems under particular matrix norms. Further, a different basis may make it easier to apply heuristics to recover the perturbed polynomials from the TLS solution.

Matrix Structure

Under this ordering, our input polynomial p from (2.4) can be represented by the array

$$\overset{1}{0}\ \ \overset{x}{0}\ \ \overset{x^2}{2}\ \ \overset{x^3}{0}\ \ \overset{x^2y}{3}\ \ \overset{xy}{4}\ \ \overset{xy^2}{6}\ \ \overset{y}{0}\ \ \overset{y^2}{0}\ \ \overset{y^3}{0}\ ,$$

and similarly q by

$$\overset{1}{2}\ \ \overset{x}{0}\ \ \overset{y}{3}\ .$$

Matching up coefficients leads to the following matrix system

$$\left[\begin{array}{cccccc} {\scriptstyle 1} & {\scriptstyle x} & {\scriptstyle x^2} & {\scriptstyle xy} & {\scriptstyle y} & {\scriptstyle y^2} \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 3 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 3 & 2 \end{array}\right] \begin{bmatrix} r_{0,0} \\ r_{1,0} \\ r_{2,0} \\ r_{1,1} \\ r_{0,1} \\ r_{0,2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 2 \\ 0 \\ 3 \\ 4 \\ 6 \\ 0 \\ 0 \end{bmatrix},$$

where each column of the matrix is the coefficient vector corresponding to our input polynomial q multiplied by the monomial indicated at the top of the column.

This system has the solution vector

$$\begin{array}{cccccc} {\scriptstyle 1} & {\scriptstyle x} & {\scriptstyle x^2} & {\scriptstyle xy} & {\scriptstyle y} & {\scriptstyle y^2} \\ 0 & 0 & 1 & 2 & 0 & 0 \end{array}.$$

This corresponds to the output polynomial $h = x^2 + 2xy$, since for (2.4) there is an exact solution. If there is no such solution, then we solve one of the the least squares problems to find a solution to (2.3). This allows us to recover $\hat{p}, \hat{q}, r$ with $\hat{p} = \hat{q}r$ as desired.

Hence the methods applied in the univariate case can be directly applied to multivariate inputs, for suitably formed structured matrix Q.

2.2 Greatest Common Divisor

Given two polynomials $f, g \in \mathbb{R}[x]$, a common divisor $d \in \mathbb{R}[x]$ is a polynomial that satisfies $d \mid f$ and $d \mid g$. A *greatest common divisor* (GCD) ensures that, for all polynomials $d_0 \in \mathbb{R}[x]$, $d_0 \mid f$ and $d_0 \mid g$ implies $d_0 \mid d$.

We can write d as a linear combination of f, g, i.e.

$$uf + vg = d. \tag{2.5}$$

Along with GCD, we have the usual *least common multiple* (LCM) $l \in \mathbb{R}[x]$ of the polynomials f, g. If $f/d = r_f$ and $g/d = r_g$, then $l = fr_g = gr_f$.

Almost all pairs of polynomials have a GCD of 1. Particularly, for approximate polynomial GCD, we expect any perturbation Δf or Δg to destroy coefficient dependencies that would lead to a GCD of degree > 0.

We thus regard this as an optimization problem, to determine the nearest polynomials to f, g that have a non-trivial GCD.

2.2.1 Matrix Representation

Determination of a greatest common divisor is equivalent to finding a particular linear combination of f, g, since this combination produces the GCD. The Sylvester matrix of two polynomials provides a convenient way of representing linear combinations of two polynomials. Consider

$$\begin{aligned} f &= x^4 + 4x^2 + 3, \\ g &= 2x^3 - x^2 + 2x - 1, \end{aligned} \tag{2.6}$$

which has GCD $(1 + x^2)$ by construction. The Sylvester matrix of f, g can be written as:

$$S(f,g) \quad = \quad \left[\begin{array}{ccccccc} f & xf & x^2f & g & xg & x^2g & x^3g \\ 3 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 3 & 0 & 2 & -1 & 0 & 0 \\ 4 & 0 & 3 & -1 & 2 & -1 & 0 \\ 0 & 4 & 0 & 2 & -1 & 2 & -1 \\ 1 & 0 & 4 & 0 & 2 & -1 & 2 \\ 0 & 1 & 0 & 0 & 0 & 2 & -1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 2 \end{array}\right].$$

Multiplying this matrix by a partitioned vector $x = \begin{bmatrix} u \\ v \end{bmatrix}$ gives the linear combination $uf + vg$. This is precisely the form of the desired GCD.

By solving the Least Squares problems on the Sylvester matrix of the input polynomials, we can reduce the rank of $S(f, g)$. This determines the nearest Sylvester matrix (because of the minimization in the least squares problem) of perturbed polynomials that have a non-trivial GCD.

2.2.2 Determining Solutions

Once the Sylvester matrix of the perturbed polynomials has been determined, the computation of the GCD is done by determining the linear combination of f, g that gives the monic GCD. The degree of the GCD can be determined by examining the rank of the Sylvester matrix.

In [4], the GCD is determined by examining the Singular Value Decomposition of the Sylvester matrix. This method determines a nearby matrix that is rank deficient, but does not maintain the structure of a Sylvester matrix.

We shall develop a solution to the STLS problem that avoids the need to reconstruct a polynomial, since the matrix structure will remain constant.

The construction of $S(f, g)$ is done to ensure that it has column dimension equal to the maximum possible degree of the LCM of f, g. We also know that the degree of the actual LCM of f, g will be the rank of $S(f, g)$. Combining these two ideas yields:

Fact 1 *The rank deficiency of the Sylvester matrix $S(f, g)$ is equal to the degree of the GCD of polynomials f, g, where the rank deficiency is defined as the column dimension of $S(f, g)$ minus the rank of $S(f, g)$.*

Thus the rank of the GCD for the perturbed polynomials is known, which will then allow us to compute the polynomials u, v that result in the *monic* GCD of f, g. This is accomplished by solving a subsystem of

$$S(f,g)\begin{bmatrix}u\\v\end{bmatrix} = d$$

that corresponds to the entries of d with degree greater than or equal to the degree of the GCD. We can do this since we know our desired result d will have the form

$$\begin{bmatrix}*\\\vdots\\*\\1\\0\\\vdots\\0\end{bmatrix}$$

with the 1 representing the leading coefficient of the GCD. The system (2.2.2) is then solved just for this leading coefficient and the zeros at higher degree. If the degree of the GCD is γ, then we take the sub-matrix of $S(f,g)$ consisting of the rows $\gamma \ldots m+n$. (2.2.2) becomes:

$$\bar{S}(f,g)\begin{bmatrix}u\\v\end{bmatrix} = \begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix}$$

which can be easily solved for u, v. Then this u, v are multiplied by f, g to give the GCD $d = uf + vg$.

2.3 Bivariate Factorization

A polynomial f is said to be irreducible if there does not exist two polynomials g, h such that $f = gh$. There have been several recent methods developed for factoring bivariate polynomials. In [5], a method based on integrating a local solution along the curve is used to reconstruct components or factors of the original bivariate polynomial.

In [3], a subsequent method using numerical computation to determine a candidate for factorization is presented. The candidate with high probability is a correct factorization. This method relies on 4 tools: zero-sum relations at triplets, partial information on monodromy action, Newton interpolation on a structured grid, and finally a homotopy method.

A third method, that closely fits the structured matrix construction methods for division and GCD, was developed in [11]. This method relies on a new and useful criteria for determining when a bivariate polynomial is irreducible [21] [16]:

Fact 2 *A bivariate polynomial $f(x, y) \in \mathbb{R}[x, y]$ is irreducible if and only if there are no non-trivial solutions $g, h \in \mathbb{R}[x, y]$ to the equation*

$$\frac{\partial}{\partial y}\frac{g}{f} = \frac{\partial}{\partial x}\frac{h}{f}. \tag{2.7}$$

Such a solution g, h can be used to construct [11] the factors of f . Applying the quotient rule to (2.7) we get

$$f\frac{\partial g}{\partial y} - g\frac{\partial f}{\partial y} - f\frac{\partial h}{\partial x} + h\frac{\partial f}{\partial x} = 0, \tag{2.8}$$

which corresponds to a linear system of equations in the coefficients of g, h.

By solving the least squares problems for the linear system (2.8), we can ensure that there *are* non-trivial solutions to the equation (2.7), for a nearby polynomial $\hat{f}$.

2.3.1 Degree Bounds

Before proceeding to the linear system, degree bounds for the polynomials should be considered.

Let $\deg_{x,y}(f) = (m, n)$ denote the maximal degree of x, y respectively in $f(x, y) \in \mathbb{R}[x, y]$. So the total degree of f is at most $m + n$, and minimally $max(m, n)$. Ruppert [21] further demonstrated that degree bounds should be placed on g, h to avoid trivial factors as follows:

$$\deg_x(g) \leq m-1, \quad \deg_y(g) \leq n,$$
$$\deg_x(h) \leq m, \quad \deg_y(h) \leq n-2.$$

Now since $\deg_{x,y}(g) \leq (m-1, n)$, we have $\deg_{x,y}(\frac{\partial g}{\partial y}) \leq (m-1, n-1)$, so the term $f\frac{\partial g}{\partial y}$ has bounds

$$\deg_{x,y}(f\frac{\partial g}{\partial y}) \leq (2m-1, 2n-1).$$

Checking all four terms of (2.8) confirms that the maximal degree of the terms in (2.8) is $(2m-1, 2n-1)$, so there are a total of $(2m-1+1)(2n-1+1) = 4mn$ coefficients to be set to zero.

Fact 3 *Of the* $4mn$ *coefficients of* (2.8)*,* $2n$ *correspond to terms that will always be zero.*

This can be observed by considering the terms of (2.8) of maximal degree $(n-1)$ in y. The last two terms, $f\frac{\partial h}{\partial x}+h\frac{\partial f}{\partial x}$, clearly have coefficient 0 at $y = 2n-1$, since $deg_y(h) = n-2$.

In $f\frac{\partial g}{\partial y}$, we require y degrees of $n, n-1$ in $f, \frac{\partial g}{\partial y}$ respectively. The terms of $\frac{\partial g}{\partial y}$ with y degree $n-1$ are the terms of g with y degree n multiplied by the terms of f with y degree n.

The second term $g\frac{\partial f}{\partial y}$ results in the same coefficients, hence the opposite signs yield a cancelation of terms. So any term of (2.8) with degree in y of $2n-1$ is zero.

2.3.2 Matrix Representation

From the degree bounds above, it is clear that (2.8) can be expressed as a linear system of dimension $(4mn-2m) \times (2mn+n-1)$. Denote this matrix $R(f)$, the Ruppert matrix of the input polynomial f.

The rows of the matrix shall correspond to the coefficients of (2.8) at each of the possible degree pairs (i, j), following the same ordering described for polynomial division.

Working with a general degree $(2,2)$ polynomial $f \in \mathbb{R}[x,y]$, we expect $4mn - 2n = 12$ such rows. We have:

$$f(x,y) = \sum_{0\le i\le 2} \sum_{0\le j\le 2} f_{i,j} x^i y^j.$$

The first row of $R(f)$ corresponds to the degree 0 term of (2.8), so equals

$$f_{0,0}g_{0,1} - g_{0,0}f_{0,1} + h_{0,0}f_{1,0} - f_{0,0}h_{1,0},$$

resulting in the row:

$$\begin{array}{ccccccccc} {\scriptstyle g_{0,0}} & {\scriptstyle g_{1,0}} & {\scriptstyle g_{1,1}} & {\scriptstyle g_{1,2}} & {\scriptstyle g_{0,1}} & {\scriptstyle g_{0,2}} & {\scriptstyle h_{0,0}} & {\scriptstyle h_{1,0}} & {\scriptstyle h_{2,0}} \\ -f_{0,1} & 0 & 0 & 0 & f_{0,0} & 0 & f_{1,0} & -f_{0,0} & 0 \end{array}.$$

The full matrix system (including rows normally ignored due to Fact 3, indicated by **bold** row labels) for this example is as follows:

$$\begin{array}{c|ccccccccc|} & {\scriptstyle g_{0,0}} & {\scriptstyle g_{1,0}} & {\scriptstyle g_{1,1}} & {\scriptstyle g_{1,2}} & {\scriptstyle g_{0,1}} & {\scriptstyle g_{0,2}} & {\scriptstyle h_{0,0}} & {\scriptstyle h_{1,0}} & {\scriptstyle h_{2,0}} \\ 1 & -f_{0,1} & 0 & 0 & 0 & f_{0,0} & 0 & f_{1,0} & -f_{0,0} & 0 \\ x & -f_{1,1} & -f_{0,1} & f_{0,0} & 0 & f_{1,0} & 0 & 2f_{2,0} & 0 & -2f_{0,0} \\ x^2 & -f_{2,1} & -f_{1,1} & f_{1,0} & 0 & f_{2,0} & 0 & 0 & f_{2,0} & -f1,0 \\ x^3 & 0 & -f_{2,1} & f_{2,0} & 0 & 0 & 0 & 0 & 0 & 0 \\ x^3y & 0 & -2f_{2,2} & 0 & 2f_{2,0} & 0 & 0 & 0 & 0 & 0 \\ x^3y^2 & 0 & 0 & -f_{2,2} & f_{2,1} & 0 & 0 & 0 & 0 & 0 \\ \mathbf{x^3y^3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ x^2y & -2f_{2,2} & -2f_{1,2} & 0 & 2f_{1,0} & 0 & 2f_{2,0} & 0 & f_{2,1} & -f_{1,1} \\ x^2y^2 & 0 & 0 & -f_{1,2} & f_{1,1} & -f_{2,2} & f_{2,1} & 0 & f_{2,2} & -f_{1,2} \\ \mathbf{x^2y^3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ xy & -2f_{1,2} & -2f_{0,2} & 0 & 2f_{0,0} & 0 & 2f_{1,0} & 2f_{2,1} & 0 & -2f_{0,1} \\ xy^2 & 0 & 0 & -f_{0,2} & f_{0,1} & -f_{1,2} & f_{1,1} & 2f_{2,2} & 0 & -2f_{0,2} \\ \mathbf{xy^3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ y & -2f_{0,2} & 0 & 0 & 0 & 0 & 2f_{0,0} & f_{1,1} & -f_{0,1} & 0 \\ y^2 & 0 & 0 & 0 & 0 & -f_{0,2} & f_{0,1} & f_{1,2} & -f_{0,2} & 0 \\ \mathbf{y^3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}.$$

2.3.3 Determining Solutions

$R(f)$ will have full rank if and only if there are no non-trivial solutions to our equation (2.8), hence if and only if f is irreducible due to Fact (2).

In solving the Least Squares problems, we compute a matrix $\hat{R} = R(f) + \Delta R$ that is rank-deficient. At the same time, a vector y is computed such that $\hat{R}y = 0$. This vector will contain the coefficients of the polynomials g, h in the prescribed coefficient order.

As with the other operations, if the matrix $\hat{R}$ does not have the same structure as $R(f)$, then recovering the polynomial $\hat{f} = f + \Delta f$ will be difficult. Once again, this problem is avoiding by solving the STLS problem, as opposed to LS or TLS. STLS ensures that $\hat{R} = R(\hat{f})$ for some polynomial $\hat{f}$.

2.4 Decomposition

A polynomial $f \in \mathbb{R}[x]$ is said to be decomposable if there exists polynomials $g, h \in \mathbb{R}[x]$ such that $f(x) = g(h(x))$. Applying the restrictions that $\deg_x(g) < \deg_x(f)$,$\deg_x(h) < \deg_x(f)$ eliminates trivial solutions.

Decomposition of such a polynomial f is the search for suitable polynomials g, h. As with the previous polynomial operations, if no such g, h can be found, then the closest polynomial $\hat{f} = f + \Delta f$ is the desired solution.

A method proposed by [12] verifies decomposability by transforming the problem to that of irreducibility of the bivariate polynomial $\phi_f(x, y) = \frac{(f(x)-f(y))}{(x-y)}$.

Fact 4 *A polynomial $f(x) \in \mathbb{R}[x]$ of composite degree*[1] *is indecomposable if and only if $\phi_f(x, y)$ is irreducible.*

One can then use the methods described in the previous section on bivariate factorization to attain a solution. For a general polynomial of degree 3, $f = f_0 + f_1x + f_2x^2 + f_3x^3$, we

[1]A polynomial of prime degree will trivially be indecomposable.

have:

$$\begin{aligned}\phi_f(x,y) &= f_1(1)+\\ &\quad f_2(x+y)+\\ &\quad f_3(x^2+xy+y^2).\end{aligned}$$

2.4.1 Degree Bounds

The method of the previous section can be refined for the particular polynomial $\phi_f x, y$, since the Ruppert matrix $R(\phi)$ for this polynomial will exhibit a certain structure.

Let $f = \sum_{i=0}^{n} f_i x^i$, then the terms of degree i in $\phi_f(x,y)$ (combined in x and y) must have coefficient f_{i+1} (as was seen in the degree 4 example). The polynomial ϕ has form

$$\phi_f(x,y) = \sum_{i=0}^{n-1} f_{i+1} \sum_{j=0}^{i} x^{i-j} y^j.$$

The degree bounds of $\phi_f(x,y)$ are thus less than of $f(x)$, namely

$$\deg_y(\phi_f(x,y)) \le n-1,$$
$$\deg_x(\phi_f(x,y)) \le n-1.$$

Therefore the required g, h will have bounds of

$$\deg_x(g) \le n-2, \quad \deg_y(g) \le n-1,$$
$$\deg_x(h) \le n-1, \quad \deg_y(h) \le n-3.$$

2.4.2 Matrix Representation

From the degree bounds above, the equation (2.8), applied to $\phi_f(x,y)$, can be expressed as a linear system. The number of equations shall be

$$(4(n-1)(n-1) - 2(n-1)) = 4n^2 - 10n + 6.$$

There are $(2(n-1)(n-1)+(n-1)-1) = 2n^2-3n$ variables, so the linear system has dimension $(4n^2-10n+6)\times(2n^2-3n)$.

Denote the matrix representation of the system $R(\phi_f(x,y))$, the Ruppert matrix of the input polynomial $\phi_f(x,y)$.

For the degree 3 example, the matrix would be of dimension 12×9:

$$R_{\phi_f(x,y)} = \begin{bmatrix} -f_2 & f_1 & 0 & 0 & 0 & 0 & f_2 & -f_1 & 0 \\ -f_3 & f_2 & 0 & -f_2 & f_1 & 0 & 2f_3 & 0 & -2f_1 \\ 0 & f_3 & 0 & -f_3 & f_2 & 0 & 0 & f_3 & -f_2 \\ 0 & 0 & 0 & 0 & f_3 & 0 & 0 & 0 & 0 \\ -2f_3 & 0 & 2f_1 & 0 & 0 & 0 & f_3 & -f_2 & 0 \\ 0 & 0 & 2f_2 & -2f_3 & 0 & 2f_1 & 0 & 0 & -2f_2 \\ 0 & 0 & 2f_3 & 0 & 0 & 2f_2 & 0 & 0 & -f_3 \\ 0 & 0 & 0 & 0 & 0 & 2f_3 & 0 & 0 & 0 \\ 0 & -f_3 & f_2 & 0 & 0 & 0 & 0 & -f_3 & 0 \\ 0 & 0 & f_3 & 0 & -f_3 & f_2 & 0 & 0 & -2f_3 \\ 0 & 0 & 0 & 0 & 0 & f_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

which again will be full rank if the polynomial f is indecomposable.

2.4.3 Determining Solutions

Solving the STLS problem preserves the structure of the computed $\hat{R}$ giving $\hat{R}, y$ as before. Then the algorithm of [11] can be used to recover the factors of $\phi_f(x,y)$. Of course, once f has been determined a regular decomposition algorithm can be run to determine g, h.

2.5 Conclusion

This section has shown a reduction of the approximate polynomial operations division, GCD, bivariate factorization, and decomposition, to a corresponding structured matrix system. The structure of each system has been presented, and a method of manipulating this system has been briefly discussed, yielding the result of the polynomial operation. .

Chapter 3

Least Squares Problems

Solving linear systems of equations is one of the most fundamental numerical computations. In its most basic form, we seek a solution vector $x \in \mathbb{R}^n$ that maps each of the input equations $a_i \in \mathbb{R}^n$ to their desired value $b_i \in \mathbb{R}$, using the standard dot product vector multiplication. Placing the inputs a_i into the rows of a matrix $A \in \mathbb{R}^{m \times n}$ yields the typical matrix equation

$$Ax = b. \tag{3.1}$$

The vector x can be generally thought of as a collection of parameters x_i, whose values are determined so that they satisfy equation (3.1). Depending on the number of equations (m), it may be the case that there are no such values. In such circumstances, a *best* value for x would conceptually be one that makes $Ax \approx b$. For the classical LS problem, this vector satisfies $Ax = b + \Delta b$, and *best* x minimizes $||\Delta b||_2$. The more sophisticated Least Squares problems involve a more complicated definition, given with their introduction in this chapter.

One can reformulate (3.1) as a rank reduction problem on the transformed system

$$Cy = 0, \tag{3.2}$$

where $C \in \mathbb{R}^{m\times n+1}$, $y \in \mathbb{R}^{n+1}$, and are generated as follows:

$$C = \begin{bmatrix} a_{11} & \dots & a_{1n} & b_1 \\ a_{21} & \dots & a_{2n} & b_2 \\ \vdots & \ddots & \vdots & \vdots \\ a_{m1} & \dots & a_{mn} & b_m \end{bmatrix} \qquad y = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ -1 \end{bmatrix}. \tag{3.3}$$

Setting $Cy = 0$ ensures the $a_i x = b_i$ as required, but now we are looking for vectors y in the nullspace of C. Reducing the rank of C can then guarantee the existence of such a non-trivial solution y. Any computed vector y, scaled by $-\frac{1}{y_{n+1}}$ (provided y_{n+1} is non-zero) to have a last entry $y_{n+1} = -1$, is of the same required form in (3.3).

There are many techniques for determining such x (or y) for a linear system, but we must first specify the exact problem to be solved. We consider three *Least Squares* problems, applicable not when an exact solution x or y exists for the system, but when the input A, b or C must be perturbed to find such solutions.

3.1 Singular Value Decomposition

As we shall see, the singular value decomposition is a valuable tool for computing solutions to the minimization of the least squares problems. It's computation is left to the next chapter, however we formally introduce the concept here.

Theorem 3.1.1 *For a matrix $A \in \mathbb{R}^{m\times n}$, there exist orthogonal matrices $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$, and a diagonal matrix $\Sigma \in \mathbb{R}^{m\times n}$ such that*

$$A = U\Sigma V^T. \tag{3.4}$$

Further, the entries in the matrix Σ are called the singular values of A, and are in decreasing order from left to right of the matrix, i.e:

$$\sigma_1 \geq \cdots \geq \sigma_r > 0.$$

The value $r \leq n$ is commonly known as the *rank* of the matrix A. If $r < n$, then A is said to be singular, or rank deficient.

3.2 Least Squares

We have the definition of the LS minimization from (1.1), which minimizes the *residual* $||Ax - b||_2$, so that as $||Ax - b||_2 \to 0$, $Ax \to b$. (1.1) can thus be restated as:

$$\min_{||x||} \big(||Ax - b||_2\big). \tag{3.5}$$

Essentially, the problem is to determine a new vector $\hat{b} = b + \Delta b$ for which there exists an x having residual of zero, and minimizes ΔB. In terms of (3.1), we seek a solution vector x to

$$Ax = b + \Delta b.$$

3.2.1 Solution

For a matrix $A \in \mathbb{R}^{m\times n}$ with rank r, we can rewrite the SVD of A, in terms of the columns of U, V from (3.4) as:

$$A = \sum_{i=1}^{n} u_i \sigma_i v_i^T, \tag{3.6}$$

where u_i, v_i are the i^{th} column of U, V respectively. The solution to (3.5) can be determined from the following theorem:

Theorem 3.2.1 *For $A \in \mathbb{R}^{m\times n}$ with rank r, and $b \in \mathbb{R}^m$, the vector of smallest 2-norm that minimizes $||Ax - b||_2$ is:*

$$x = \left(\sum_{i=1}^{r} v_i \sigma_i^{-1} u_i^T\right) b. \tag{3.7}$$

The matrix in (3.7) is known as the (Moore-Penrose) *pseudo-inverse* of A, denoted $A^\dagger$, so that

$$A^\dagger = \left(\sum_{i=1}^{r} v_i \sigma_i^{-1} u_i^T\right),$$

and (3.7) becomes

$$x = A^\dagger b.$$

3.2.2 Discussion

It is clear that in our applications, a solution to the Least Squares problem will only assume coefficients of certain polynomials (i.e. those that appear in b) are approximate. The structure of the matrix A will clearly be preserved, since it remains unchanged for the solution to (3.5).

A natural extension of the LS problem must also be considered, namely Total Least Squares. By allowing perturbations in the matrix A, the space in which we search for solutions x is significantly expanded. This will most often yield a solution that better approximates (3.1).

3.3 Total Least Squares

The Total Least Squares (TLS) problem incorporates perturbations in the input matrix A. To formulate this, the minimization and the equation are separated as:

$$\begin{gathered}(A + \Delta A)x = b + \Delta b, \\ \min_{||\Delta A||^2 + ||\Delta b||} || \, [\Delta A | \Delta b] \, ||.\end{gathered} \tag{3.8}$$

3.3.1 Solution

The solution to the total least squares problem can also be determined from the singular value decomposition of the matrix C, formed by joining A, b as in (3.2). If the rank of $C < n + 1$, then there exists a non-trivial vector in it's null space, and the vector x can be determined.

If the rank of $C = n + 1$, then the Eckhart-Young-Mirsky Theorem [17] is used to compute the *best* rank n approximation of C. Write the SVD of C, as in (3.4)

$$C = U\Sigma V^T, \quad \Sigma = \begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \dots & \sigma_{n+1} \end{bmatrix}.$$

Then form the matrix:

$$\hat{\Sigma} = \begin{bmatrix} \sigma_1 & 0 & \dots & 0 & 0 \\ 0 & \sigma_2 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & \dots & \sigma_n & 0 \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix}$$

by forcing the smallest singular value σ_{n+1} to be zero.

Theorem 3.3.1 *The matrix $\hat{C} = U\hat{\Sigma}V^T$ is of rank n, such that $||C - \hat{C}||$ is minimized, and the solution*[1] *to* (3.8) *is*

$$x = \frac{-1}{v_{n+1,n+1}}[v_{1,n+1}, \dots, v_{n,n+1}]^T. \tag{3.9}$$

3.3.2 Discussion

Application of the TLS technique to the input system yields an x that better approximates (3.1). However, the result is a solution

$$(A + \Delta A)x = (b + \Delta b),$$

which does not necessarily preserve the structure of A as described in the introduction. For the application to approximate polynomial operations, we shall see this can have undesirable consequences.

We also must be wary of matrices whose SVD yields a value of $v_{n+1,n+1} \to 0$. This

[1]Subject to the assertion that $v_{n+1,n+1} \neq 0$.

should not be the case for full rank matrices C, however roundoff error in computation of the SVD can introduce this possibility.

The alternative is to solve a modified form of TLS, with an additional constraint that the structure of the input matrix remains constant. This is known as the Structured Total Least Squares (STLS) problem.

3.4 Structured Total Least Squares

Structured Total Least Squares (STLS) incorporates an additional constraint to the TLS problem. The purpose of the constraint is to ensure that the matrices considered for solution have the same structure as the matrices used for input.

To illustrate, consider an input matrix $C \in \mathbb{R}^{2\times 3} = \begin{bmatrix} \alpha & \beta & \psi \\ \beta & \gamma & \omega \end{bmatrix}$

If the entries of C are derived from input polynomials[2] , then the matrix used to generate a solution should be of the same form, namely:

$$\hat{C} = \begin{bmatrix} \alpha + \Delta\alpha & \boldsymbol{\beta + \Delta\beta} & \psi + \Delta\psi \\ \boldsymbol{\beta + \Delta\beta} & \gamma + \Delta\gamma & \omega + \Delta\omega \end{bmatrix}$$

This would ensure that the [2,1] and [1,2] entries of $\hat{C}$ are the same, since presumably they are derived from the same polynomial coefficient.

A natural way of formalizing this constraint is to define a collection of *basis matrices* for the matrix structure in question. For this example, we define the *entry vector* $c \in \mathbb{R}^k = (c_1, \ldots, c_k)$ containing the distinct entries of C:

$$c = \begin{bmatrix} \alpha & \beta & \gamma & \psi & \omega \end{bmatrix}.$$

[2]or from are formed for other application that requires preserving structure of the matrix

The matrix C can then be expressed as:

$$C = \sum_{i=1}^{k} c_i \, (T_i),$$

for a set of $k = 5$ basis matrices $T_i \in \mathbb{R}^{2\times 3}$:

$$T_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad T_2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \quad T_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix},$$
$$T_4 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad T_5 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Using this general definition, the STLS problem becomes the determination of a vector $y \in \mathbb{R}^{m\times n+1}$ and perturbed entry vector $\hat{c} = c + \Delta c$ with:

$$\begin{gathered} \left(\sum_{i=1}^{k} \hat{c}_i \, (T_i) \right) y = 0, \\ \min_{\Delta c} ||\hat{c} - c||. \end{gathered} \tag{3.10}$$

This choice of basis matrices and entry vector corresponds to a particular method of viewing the STLS problem, the RiSVD.

3.4.1 RiSVD

The Riemannian SVD (RiSVD) approach (proposed in [19]) to formalizing the STLS problem relies on the *entry vector* $\hat{c} \in \mathbb{R}^{k+1}$ and structure-dependent basis matrices T_i as above.

Structure Representation

The structure relation for $C, \hat{C}$ of (3.2) is then simply

$$\hat{C} = C + \Delta C = T_0 + \sum_{i=1}^{k} T_i \hat{c}_i,$$
$$C = T_0 + \sum_{i=1}^{k} T_i c_i.$$

For completeness we have introduced the matrix T_0 as a constant matrix that can be added to the structured matrix. This will not impact our application in any way. Consider the structure matrix $H \in \mathbb{R}^{3\times 2}$ and vector $b \in \mathbb{R}^3$

$$H = \begin{bmatrix} h_1 & h_2 \\ h_2 & h_3 \\ h_3 & h_4 \end{bmatrix} \quad b = \begin{bmatrix} c_1 \\ h_4 \\ c_2 \end{bmatrix}$$

with $k = 6$, and

$$c = \begin{bmatrix} h_1 & h_2 & h_3 & h_4 & c_1 & c_2 \end{bmatrix}^T .$$

The structure dependent matrices are then

$$T_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad T_2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad T_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix},$$
$$T_4 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad T_5 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad T_6 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Problem Formulation

We have the formalization of the STLS problem in (3.10), further add the constraint $||y|| = 1$ (simple normalization). Then, in [19], it is show that (3.10) is equivalent to the non-linear, generalized SVD:

Find the triplet (u, τ, v) corresponding to the smallest τ such that

$$\begin{aligned} Cv = D_v u\tau, &\quad u^T D_v u = 1, \\ C^T u = D_u v\tau, &\quad v^t D_u v = 1, \\ &\quad v^T v = 1. \end{aligned} \tag{3.11}$$

with D_u defined as

$$\begin{aligned} D_u v &= \sum_{i=1}^{m} T_i^T (u^T T_i v) u, \\ D_u &= \sum_{i=1}^{m} (T_i^T u)(u^T T_i), \end{aligned}$$

and D_v similarly

$$D_v = \sum_{i=1}^{m} (T_i v)(v^T T_i^T).$$

Then, for $y = v$ and $\hat{c}_i = c_i - u^T T_i v \tau$, a solution for (3.10) is found. We note at this stage that the solutions appear to be a good approximation to (3.10), but they are not necessarily exact solutions, as was the case with the LS and TLS problems.

Algorithm

In order to find the triplet (u, τ, v), the QR Decomposition of C is used to create a block triangular representation of the constraints. Since the constraints reduce to a restricted singular value decomposition (RSVD) if D_u, D_v were constant [19], an iterative method can be used to refine u, v with these matrices constant at each iteration.

The QR Decomposition of $C \in \mathbb{R}^{m \times n}$ can be partitioned:

$$C = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix},$$

where $Q_1 \in \mathbb{R}^{m \times n}$,$Q_2 \in \mathbb{R}^{m \times (m-n)}$, $R \in \mathbb{R}^{n \times n}$. The vector $u \in \mathbb{R}^m$ can be written as

$$u = Q_1 z + Q_2 w,$$

where $z \in \mathbb{R}^n$, $w \in \mathbb{R}^{m-n}$. The constraint $C^T u = D_u \tau$ is thus:

$$\begin{aligned}
D_u v \tau &= C^T u \\
&= (\begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix})^T (Q_1 z + Q_2 w) \\
&= (\begin{bmatrix} R^T & 0 \end{bmatrix} \begin{bmatrix} Q_1^T \\ Q_2^T \end{bmatrix})(Q_1 z + Q_2 w) \\
&= (R^T Q_1^T)(Q_1 z + Q_2 w) \\
&= R^T Q_1^T Q_1 z \\
&= R^T z.
\end{aligned}$$

$Cv = D_v u \tau$ is similarly partitioned:

$$\begin{aligned}
Cv &= D_v u \tau \\
(\begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix}) v &= D_v (Q_1 z + Q_2 w) \tau
\end{aligned}$$

which yields the $(m+n) \times (m+n)$ linear system:

$$\begin{bmatrix} R^T & 0 & 0 \\ Q_2^T D_v Q_1 & Q_2^T D_v Q_2 & 0 \\ Q_1^T D_v Q_1 \tau & Q_1^T D_v Q_2 \tau & -R \end{bmatrix} \begin{bmatrix} z \\ w \\ v \end{bmatrix} = \begin{bmatrix} D_u v \tau \\ 0 \\ 0 \end{bmatrix}. \tag{3.12}$$

Each step of the iteration will solve (3.12) for fixed D_u, D_v using the values of u, v from the previous iteration. This can be easily accomplished since the linear system is block triangular in structure.

The solution yields the refined v, and z, w such that the refined u can be constructed as $Q_1 z + Q_2 w$. At the highest level, the algorithm can be described

1. $z_{i+1} = R^{-T} D_{u_i} v_i \tau_i$;
2. $w_{i+1} = -(Q_2^T D_{v_i} Q_2)^{-1} (Q_2^T D_{v_i} Q_1) z_{i+1}$;
3. $u_{i+1} = Q_1 z_{i+1} + Q_2 w_{i+1}$;

4. $v_{i+1} = R^{-1}Q_1^T D_{v_i} u_{i+1}$.

Our implementation of the QR iteration is left to the following chapter, which describes the accuracy constraints on the computation. A novel approach to increase the efficiency of the simple implementation, by working with only square matrix systems, is also presented.

3.4.2 STLN

The choice of basis matrices to represent the structure of an input matrix for RiSVD is certainly not unique (see [17] for a detailed treatment). A second way to approach the STLS problem is called Structured Total Least Norm (STLN).

STLN formalizes the structure constraint by first defining vectors α, β that contain the distinct perturbations of A, b in (3.1). We then force the perturbation $\begin{bmatrix}\Delta A & \Delta b\end{bmatrix}$ to follow the same structure as the input. This clearly ensures the correct structure for $A + \Delta A, b + \Delta b$.

The minimization in (3.10) is then rewritten in terms of the vectors α, β and two *weighting matrices* W_α, W_β.

Structure Representation

For the input $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, let $\alpha \in \mathbb{R}^p$ be a vector containing each of the *distinct* entries of ΔA. Further, define $\beta \in \mathbb{R}^q$ to contain all of the distinct entries of Δb that are *not* in α (hence not in ΔA.

STLN requires the formation of three structure-dependent basis matrices, $P_1, P_2 \in \mathbb{R}^{m \times p}$,$Q \in \mathbb{R}^{m \times q}$. These matrices are constructed such that

$$\Delta A x = P_1 \alpha,$$
$$\Delta b = P_2 \alpha + Q\beta.$$

providing the means of mapping the vectors α, β to the desired perturbation matrices. Consider once again the structure matrix $H \in \mathbb{R}^{3\times 2}$ and vector $b \in \mathbb{R}^3$

$$H = \begin{bmatrix} h_1 & h_2 \\ h_2 & h_3 \\ h_3 & h_4 \end{bmatrix}, \quad b = \begin{bmatrix} c_1 \\ h_4 \\ c_2 \end{bmatrix}.$$

Hence $p = 4$, $q = 2$, and

$$\alpha = \begin{bmatrix} \Delta h_1 & \Delta h_2 & \Delta h_3 & \Delta h_4 \end{bmatrix}^T,$$
$$\beta = \begin{bmatrix} \Delta c_1 & \Delta c_2 \end{bmatrix}.$$

The basis matrices for this example begin with:

$$P_1 = \begin{bmatrix} x_1 & x_2 & 0 & 0 \\ 0 & x_1 & x_2 & 0 \\ 0 & 0 & x_1 & x_2 \end{bmatrix},$$

so that

$$\begin{aligned} P_1\alpha &= \begin{bmatrix} x_1 & x_2 & 0 & 0 \\ 0 & x_1 & x_2 & 0 \\ 0 & 0 & x_1 & x_2 \end{bmatrix} \begin{bmatrix} \Delta h_1 \\ \Delta h_2 \\ \Delta h_3 \\ \Delta h_4 \end{bmatrix} \\ &= \begin{bmatrix} \Delta h_1 x_1 + \Delta h_2 x_2 \\ \Delta h_2 x_1 + \Delta h_3 x_2 \\ \Delta h_3 x_1 + \Delta h_4 x_2 \end{bmatrix} \\ &= \Delta A x. \end{aligned}$$

Similarly we have

$$P_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad Q = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

Problem Formulation

The system $(A + \Delta A)x = b + \Delta b$ is thus transformed to $Ax + P_1\alpha = b + (P_2\alpha + Q\beta)$. The minimization in (3.10), derived from

$$\min_{||\Delta A||^2 + ||\Delta b||} ||\, [\Delta A|\Delta b]\, ||,$$

is reformulated in terms of α, β:

$$\min_{\alpha,\beta,x} \alpha^T W_\alpha^2 \alpha + \beta^T W_\beta^2 \beta.$$

with weighting matrices $W_\alpha \in \mathbb{R}^{p\times p}$,$W_\beta \in \mathbb{R}^{q\times q}$. These are diagonal matrices, defined such that $W_\alpha^2[i,i] = d$ if $\alpha[i]$ appears d times in the matrix A, similarly for β in b. So (3.10) is now:

$$\begin{aligned} Ax + P_1\alpha &= b + P_2\alpha + Q\beta, \\ \min_{\alpha,\beta,x} \alpha^T W_\alpha^2 \alpha &+ \beta^T W_\beta^2 \beta. \end{aligned} \tag{3.13}$$

Methods

Several methods for solving the STLS problem under the Structured Total Least Norm framework are suggested in [15]. We outline two such algorithms, and motivate further exploration of these in our applications:

1. *The weighted residual method*

   This methods adds a weighted term of a form of the residual ($r = Ax - b$) to the minimization in (3.13). The additional term is defined as $\omega \hat{r}^T \hat{r}$, where $\hat{r} = r - Q\beta$, essentially providing a corrected residual $\hat{r}$ as the elements of β are refined.

   By doing so, the constrained minimization problem is transformed into an unconstrained minimization, which can be solved in a variety of ways. However, this will only yield an approximation of the constrained problem.

   The weight ω appears to be a limiting factor for this method. The choice of ω

must be adequately large to ensure that the approximation is sufficiently close to a solution of the constrained problem. However, large ω can lead to numeric instability (see e.g. [18]).

2. *The iterative quadratic programming method*

   IQP solves the minimization of (3.13) iteratively, and is studied in detail in [17] In each iteration, the objective function is solved by first linearizing the constraints around the current solution point.

3.4.3 CTLS

The Constrained Total Least Squares (CTLS) approach formalizes the structure constraint by putting the *distinct* entries of C in (3.2) into a vector v, and defining the set of allowable matrices as the result of a mapping Fv on this vector v.

The matrix $\hat{C} = C + \Delta C$ is formed by mapping the combined vector $v + \Delta v$ under the mapping rule, where Δv is denoted the *noise vector* .

Structure Representation

Given a matrix $A \in \mathbb{R}^{m\times n}$ with k distinct entries, form the vector $v \in \mathbb{R}^k$ from these k entries. CTLS defines n structure matrices $F_i \in \mathbb{R}^{m\times k}$, one for each column of A, such that $F_i v$ yields the i^{th} column of A.

Consider once again the structure matrix $H \in \mathbb{R}^{3\times 2}$ and the vector $b \in \mathbb{R}^3$, in the form of the full matrix C as (3.2)

$$C = \begin{bmatrix} h_1 & h_2 & c_1 \\ h_2 & h_3 & h_4 \\ h_3 & h_4 & c_2 \end{bmatrix}.$$

There are 6 distinct elements of the matrix, so $k = 6$, and

$$v = \begin{bmatrix} h_1 & h_2 & h_3 & h_4 & c_1 & c_2 \end{bmatrix}^T.$$

The second structure matrix and the result of mapping v by it are:

$$F_2 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \quad F_2 v = \begin{bmatrix} h_2 & h_3 & h_4 \end{bmatrix}^T.$$

$F_2 v$ is the required second column of C. The other structure matrices are formed to have the same effect.

Problem Formulation

The formulation of the STLS problem under the CTLS framework differs from (3.10) in the specification of the function to be minimized. Define $W \in \mathbb{R}^{k \times k}$, as before, a *weighting matrix* of the vector $v \in \mathbb{R}^k$, where $W[i,i] = w_i$ is the number of times the element v_i appears in the matrix A. The weighting matrix for the example is:

$$W = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

If $\Delta v \in \mathbb{R}^k$ is the *noise vector* , then the minimization becomes

$$\min_{\Delta v} \Delta v^T W \Delta v. \tag{3.14}$$

The equation $(A + \Delta A)x = (b + \Delta b)$ must still hold, but is reformulated to include the structure constraint as follows

$$\left(\sum_{i=1}^{n} F_i (v + \Delta v) \right) x = F_{n+1}(v + \Delta v).$$

In [1], the following problem formulation is derived from (3.14)

$$\min_x y^T C^T \left(H(x) W^{-1} H(x)^T \right)^{-1} C y, \tag{3.15}$$

with $H(x) = (\sum_{i=1}^{n} x(i) F_i) - F_{n+1}$.

Methods

There are several methods to solve the minimization of (3.15), see e.g. [17]. We discuss three such methods here:

1. *Newton's Method*

   Newton's method can be used to calculate (analytically) the first and second order (gradient and Hessian respectively) information for (3.15). The convergence rate is quite high, however there are a few points of concern.

   First, pure Newton's method does not necessarily descend to a minimum, and may in fact result in a maximum value for the function. Initial criteria must be carefully chosen, and the Hessian can also become ill-conditioned.

   Further, the initial values are crucial for Newton's method, since it will converge to a global minimum or maximum if they are near the starting criteria.

   Finally, the computational cost of Newton's method makes it inappropriate for this problem. There are well-known alternatives that maintain an adequately high convergence rate, and are more computationally efficient.

2. *Conjugate Gradient*

   The most obvious alternative, conjugate gradient can be used to find the minimum of the function (3.15) (see [14]).

3. *Quasi-Newton Method*

   A second alternative, Quasi-Newton (see [13] [10] for a detailed treatment) is noted to be less sensitive to accuracy issues (see [17]) than CG. Since the reduction of STLS often involves relatively small quantities, Quasi-Newton may be better suited for solving the minimization problem.

3.4.4 Summary

The main focus of contrasting the three formulations of the STLS problem is that each one formulates the minimization in a different way. The respective minimizations have been shown in [17] to be equivalent, however they are quite different, and required very different algorithms to be optimized. Table 3.1 summarizes the formulations, including the minimization functions.

Method	Form	Storage	Mapped Solution	Minimization	Algorithm(s)
CTLS	$(A+\Delta A)x$ $= b+\Delta b$	v	$\left(\sum F_i\hat{v}\right)x$ $= F_{n+1}\hat{v}$	$\|\|\hat{v}-v\|\|$	CG Quasi-Newton
STLN	$(A+\Delta A)x$ $= b+\Delta b$	α, β	$Ax + P_1\alpha =$ $b + P_2\alpha + Q\beta$	$\alpha^T W_\alpha^2 \alpha +$ $\beta^T W_\beta^2 \beta$	Weighted Residual Iterative QP
RiSVD	Cy $= 0$	c_i	$\left(\sum T_i\hat{c}_i\right)y = 0$	$\sum\left(c[i]-\hat{c}[i]\right)^2$	Inverse Iteration

Table 3.1: Summary of the three formulations of STLS presented in this chapter.

We once again emphasize that the computed solutions to the minimizations of the various formulations appear to only be approximate solutions of the STLS problem (3.10).

3.5 Geometric Interpretation

To graphically illustrate the differences between the three Least Squares problems (LS, TLS, STLS), consider the simple example:

$$\begin{bmatrix} 1.04 \\ 3.48 \end{bmatrix} \begin{bmatrix} x \end{bmatrix} = \begin{bmatrix} 3.48 \\ 7.88 \end{bmatrix}.$$

So that $C = \begin{bmatrix} A & b \end{bmatrix} = \begin{bmatrix} 1.04 & \mathbf{3.48} \\ \mathbf{3.48} & 7.88 \end{bmatrix}$ is a structured (Hankel) matrix.

We have one variable, x, which should yield

$$\begin{aligned} 1.04x &= 3.48, \\ 3.48x &= 7.88. \end{aligned}$$

These are derived from the satisfiable values of:

$$1.04 - 0.04 = 1, \quad 3.48 + 0.02 = 3.50, \quad 7.88 - 0.13 = 7.50,$$

which hold for $x = 2.50$. We can assign the values in the two equations to points in the plane, $(1.04, 3.48)$ and $(3.48, 7.88)$ to provide a geometric interpretation of the methods.

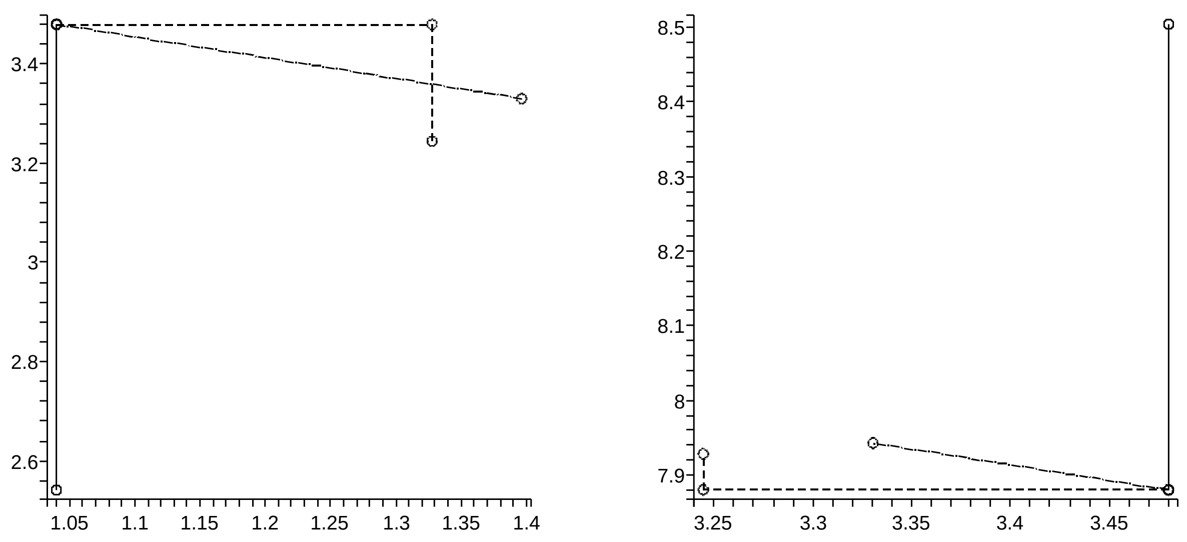
LS perturbs only the vector b, so the points will only be moved along the vertical axis. TLS minimizes the sum of the distances between the original points and their perturbed value.

STLS requires constant structure, in this case, that means the horizontal perturbation of the second point must equal the horizontal perturbation of the first.

Figure 3.1 demonstrates this geometric interpretation for the three methods, with a solid line for the LS perturbation, a dotted line for the TLS perturbation, and a dashed line for the STLS perturbation. STLS has been split into the change in each of the coordinates, to emphasize the point that the vertical coordinate change of Point 1 must equal the horizontal coordinate change of point 2.

It is interesting to note that, for this small example, the values of x determined by LS,TLS, and STLS are $2.353, 2.385, 2.444$ respectively. Having STLS closer to our expected value than TLS is unexpected, as we shall see [3].

[3] In this case, it is caused by a matrix closer than the matrix of satisfiable values used for the construction, that also happens to be satisfiable (without the structure constraint).



(a) Point 1, $(1.04, 3.48)$

(b) Point 2, $(3.48, 7.88)$

Figure 3.1: A geometric representation of the perturbations from LS,TLS, and STLS

3.6 Results

We contrast the solutions to the three Least Squares problems here. In particular, we identify the matrix norms of the perturbation that led to the derived systems. It is clear that the Least Squares solution should have higher matrix norm than that of the Total Least Squares solution, since TLS generalizes to LS if only perturbing the last row of the matrix corresponds to the minimum.

The STLS solution however, must also maintain the structure of the vector b (as well as A), a restriction not imposed by the TLS and LS problem. This will make it possible (although very unlikely) that the perturbation for the LS solution will actually be smaller (in matrix norm) than that of the STLS solution.

We consider three common type of structured matrices, but first we must define the metrics used to measure the size of the perturbation.

3.6.1 Metrics

We focus on the second formulation of the input system, $Cy = 0$:

$$\begin{bmatrix} A + \Delta A & b + \Delta b \end{bmatrix} \begin{bmatrix} x \\ -1 \end{bmatrix} = 0,$$

of which all three methods determine a solution x for a particular $\Delta A, \Delta b$. In the case of LS, $\Delta A = 0$. Each of the least squares problems is attempting to find a solution to $Ax \approx b$ in some sense, so we measure the perturbation $\begin{bmatrix} \Delta A & \Delta b \end{bmatrix}$.

3.6.2 Experimental Evidence

The size of the matrix $\begin{bmatrix} \Delta A & \Delta b \end{bmatrix}$ has been measured for a varying set of (linearly) structured matrix inputs. These structured matrices have been chosen from some of the most common in computer algebra. Each result is outlined here, and comparisons are made to show the required perturbation for each problem.

Toeplitz Matrices

A Toeplitz matrix $T \in \mathbb{R}^{n \times n}$ has the form

$$T = \begin{bmatrix} t_1 & t_2 & t_3 & \dots & t_n \\ t_{n+1} & t_1 & t_2 & \ddots & \vdots \\ t_{n+2} & t_{n+1} & t_1 & \ddots & t_3 \\ \vdots & \ddots & \ddots & \ddots & t_2 \\ t_{2n-1} & \dots & t_{n+2} & t_{n+1} & t_1 \end{bmatrix},$$

which contains $2n - 1$ distinct elements. The result of the three least squares problems for Toeplitz matrices of dimension $n = 5$, with varying size of entries t_1 are:

Entry Range	LS	TLS	STLS
0..10	1.106834972	.5854754222	.8051267476
-10..10	1.714899221	.8582976909	1.173944129
0..100	39.50966527	22.13895275	29.80559918
-100..100	59.52824294	31.20138288	42.58801001
$-10^{10}..10^{10}$	$6.867239734\ 10^{10}$	$3.750138140\ 10^{10}$	$5.071373274\ 10^{10}$
$-10^{-10}..10^{-10}$	$6.832120886\ 10^{-10}$	$3.836338463\ 10^{-10}$	$4.843752406\ 10^{-10}$

Table 3.2: Average $||\Delta T||$ for 100 random matrices with entries t_i in the range indicated

The computed norm appears to increase linearly with the size of the matrix entries. This is unfortunate, but clearly was expected, since the distance between entries becomes greater. This results in a larger perturbation being required to introduce a dependency.

Next we consider increasing the dimension of the matrix from $n = 5$ to $n = 100$.

Distinct Entries	Matrix Dimension	LS	TLS	STLS
9	5	6.855527289	3.617996045	4.810891056
39	20	7.457438422	2.285583544	4.846676928
99	50	10.55219407	1.792919462	3.679065444
199	100	11.146653176	1.537119066	4.595730262

Table 3.3: Average $||\Delta T||$ for 100 random matrices with the number of distinct entries indicated

With LS, the norm sees a small increase. However TLS actually sees an improvement in the computed matrix ΔT. This is not altogether surprising, since the TLS solution is free to perturb any entries of the input matrix. Increasing the number of entries gives the TLS solution more freedom to vary the structure of the matrix.

The STLS results appear to be a bit more cryptic. The matrix norm seems to decrease with growing dimension, but not in all cases. We shall defer discussion until different structured matrices have been considered.

It is expected that TLS perturbation will be smaller than STLS, which will in turn be smaller than LS. The percent reduction in the matrix norm is listed below, averaged out over the variable dimension and input size.

	TLS v LS	STLS v LS	TLS v STLS
Variable Entry Size	46.31 %	27.85 %	25.58 %
Variable Dimension	71.45 %	47.18 %	48.86 %

Table 3.4: Perturbation Norm Comparison

The difference between the STLS perturbation and LS perturbation is almost the same as that of TLS and STLS. This is quite an encouraging result for STLS.

Hankel Matrices

A Hankel matrix $H \in \mathbb{R}^{n \times n}$ is similar to the Toeplitz matrix, but the diagonal bands are in the opposite direction. The Hankel matrix of dimension 5 is

$$H = \begin{bmatrix} h_1 & h_2 & h_3 & h_4 & h_5 \\ h_2 & h_3 & h_4 & h_5 & h_6 \\ h_3 & h_4 & h_5 & h_6 & h_7 \\ h_4 & h_5 & h_6 & h_7 & h_8 \\ h_5 & h_6 & h_7 & h_8 & h_9 \end{bmatrix},$$

which contains $2n-1$ distinct elements. The result of the three least squares problems for Hankel matrices of dimension $n = 5$, with varying size of entries t_1 are shown below.

Entry Range	LS	TLS	STLS
0..10	3.736294816	2.067872052	2.847205556
-10..10	6.279127286	3.683435474	4.900647284
0..100	39.50966527	22.13895275	29.80559918
-100..100	59.52824294	31.20138288	42.58801001
$-10^{10}..10^{10}$	$6.867239734\ 10^{10}$	$3.750138140\ 10^{10}$	$5.071373274\ 10^{10}$
$-10^{-10}..10^{-10}$	$6.832120886\ 10^{-10}$	$2.836338463\ 10^{-10}$	$3.443752406\ 10^{-10}$

Table 3.5: Average $||\Delta H||$ for 100 random matrices with entries t_i in the range indicated

The results are similar to that of the Toeplitz matrix, with the norm growing with the size of the entries. Varying the dimension also appears to have similar results, as can be seen below:

Distinct Entries	Matrix Dimension	LS	TLS	STLS
9	5	6.279127286	3.683435474	4.900647284
39	20	7.546744004	2.217896928	4.664149136
99	50	10.36000905	1.781211831	3.620477956
199	100	9.384907377	1.476655292	4.598787336

Table 3.6: Average $||\Delta H||$ for 100 random matrices with the number of distinct entries indicated

Once again we see the peculiarity with STLS, which does not see a linear improvement. We conjecture that this is a consequence of the increased dimension both increasing the number of values to be perturbed, but also increases the difficulty of maintaining the structure.

Dimension	TLS v LS	STLS v LS	TLS v STLS
Variable Entry Size	65.76 %	58.72 %	27.22 %
Variable Dimension	70.58 %	47.01 %	48.66 %

Table 3.7: Perturbation Norm Comparison

This result closely parallels that of Toeplitz matrices, except that for fixed dimension, the STLS perturbation seems to be much closer to TLS perturbation.

Banded Matrices

Finally, we shall look at a particular class of banded matrices, namely a five-diagonal matrix B. These matrices have non-zero entries on and surrounding the matrix diagonal. The general five-diagonal matrix looks like:

$$B = \begin{bmatrix} a_1 & b_1 & c_1 & 0 & \dots & 0 \\ d_1 & a_2 & b_2 & \ddots & \ddots & \vdots \\ e_1 & d_2 & a_3 & \ddots & \ddots & 0 \\ 0 & e_2 & d_3 & \ddots & \ddots & c_{n-2} \\ \vdots & \ddots & \ddots & \ddots & \ddots & b_{n-1} \\ 0 & \dots & 0 & e_{n-2} & d_{n-1} & a_n \end{bmatrix},$$

which contains at most $5n - 6$ non-zero elements. The result of the three least squares problems for banded matrices of dimension $n = 5$, with varying size of entries t_1 are shown here:

Entry Range	LS	TLS	STLS
0..10	2.223362237	.8705973794	1.015604533
-10..10	4.139204088	1.699833434	1.986179678
0..100	38.06424199	22.13895275	29.80559921
-100..100	61.63919317	31.09812152	42.47299864
$-10^{10}..10^{10}$	5.316521541 10^{9}	2.987158552 10^{9}	3.922439387 10^{9}
$-10^{-10}..10^{-10}$	3.461578640 10^{-9}	6.700901544 10^{-10}	8.002919634 10^{-10}

Table 3.8: Average $||\Delta B||$ for 100 random matrices with entries t_i in the range indicated

The computed norm appears to vary linearly with the matrix entries as expected. Next we consider increasing the dimension of the matrix from $n = 5$ to $n = 100$.

Distinct Entries	Matrix Dimension	LS	TLS	STLS
19	5	3.673034334	1.656436934	1.894319779
94	20	2.916759876	.3015408709	.4481092128
244	50	2.451905580	0.08840127934	0.1984047521
494	100	1.666472430	0.03756954195	0.2270315949

Table 3.9: Average $||\Delta B||$ for 100 random matrices with the number of distinct entries indicated

With five-diagonal matrices, we see that all of the problems have results with better matrix norms for increased dimension. We conjecture this is due to increased number of zeroes in the matrix as the dimension increases, yielding greater flexibility of perturbation.

Dimension	TLS v LS	STLS v LS	TLS v STLS
Variable Entry Size	55.94 %	43.70 %	20.22 %
Variable Dimension	84.68 %	77.84 %	46.04 %

Table 3.10: Perturbation Norm Comparison

For this structured matrix, STLS seems to be much closer to TLS than it was for Toeplitz and Hankel matrices. This is likely due to the increased number of structure entries. The five-diagonal matrix makes no assertion on two entries being the same (except for the zeroes), so TLS and STLS will be acting the same way on the band around the diagonal. TLS will see improvement by perturbing the zeros, something not permitted for STLS.

3.6.3 Discussion

Given a matrix C, the three Least Squares problems all require a matrix ΔC and a vector y such that $(C + \Delta C)y = 0$. The results shown indicate that the Total Least Squares solution achieves this for the smallest possible ΔC.

However, the solution found for STLS appears to be a substantial improvement over LS as well, and guarantees the structure of the matrix ΔC. For applications that need the structure of $C + \Delta C$ to remain consistent, the STLS solution can be used to find this y. Of course, STLS algorithms do not guarantee minimal solutions, where as LS and TLS do. However, the distance from the smallest structured matrix seems to also be fairly close to the smallest unstructured matrix, dependent of course on the structure chosen.

3.7 Conclusion

In this chapter, a pair of techniques deriving from the basic Least Squares problem have been introduced. For these problems, given input data $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$ (or similarly $C \in \mathbb{R}^{m\times n+1}$), determine a solution x that in some manner *best* solves $Ax = b$ ($Cy = 0$)

for an interpretation of the input data.

The solution to the Structured Total Least Squares problem is particularly useful for polynomial operations in computer algebra, since the structure of the input matrix is preserved. This in turn shall allow reconstruction of the perturbed polynomials that yield an interesting result.

Chapter 4

Singular Value Decompositions

The Least Squares problems of the previous chapter rely heavily on various forms of the classical singular value decomposition (SVD). In particular, the presented solution for TLS required the determination of the smallest singular value(s) of the regular SVD, while STLS requires a solution to the RiSVD.

In this chapter we present algorithms for the computation of the SVD, paying close attention to the accuracy of the singular values that they generate. These algorithms can then be used to effectively solve the minimization equations of each of the Least Squares problems.

4.1 Accuracy

The accuracy of the SVD is quite easily stated if the goal is simply the singular values in the decomposition. We state here the error bound that is attained with a traditional algorithm for the singular value decomposition, as well as a more modern error bound that we shall employ. But first, the error bounds are motivated by the introduction of *numeric rank* .

4.1.1 Numeric Rank

The singular value decomposition is a very valuable computational tool, since it provides a means to determine an estimate for the rank of a matrix $A \in \mathbb{R}^{m \times n}$.

Given the singular value decomposition of A as in (3.4), the rank of the matrix A is the number r with, for all $\sigma_i \neq 0$, $\sigma_i \geq \sigma_r$ and $\sigma_r > 0$. In other words, σ_r is the smallest non-zero singular value.

When computing the SVD numerically, roundoff error can lead to the singular value σ_{r+1} (in some cases, many of $\sigma_i, i > r$) to become non-zero. These values are quite small, usually much less than one. Consider the matrix $A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$, with perturbed version:

$$\begin{aligned} \hat{A} &= A + \Delta A \\ &= \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} -0.0003 & 0.0000 & 0.0003 \\ -0.0010 & 0.0006 & 0.0003 \\ -0.0008 & 0.0003 & 0.0010 \end{bmatrix} \\ &= \begin{bmatrix} -0.0003 & 1.0000 & 0.0003 \\ 0.9990 & 0.0006 & 0.0003 \\ -0.0008 & 0.0003 & 0.0010 \end{bmatrix}. \end{aligned}$$

Clearly A is constructed to have rank two, with $\sigma_1 = \sigma_2 = 1$, and $\sigma_3 = 0$. Now ΔA represents roundoff error in A, meaning the computed SVD will actually be the SVD for the perturbed matrix $\hat{A}$. These (rounded) singular values are $(1.0000, 0.9999, 0.0010)$.

So if the computed SVD is used to decide the rank of A, the rank will be three. This is often called the *numeric rank* of A, and is usually computed based on some ϵ, with $0 < \epsilon \ll 1$, that is used to indicate a level at which the singular values should be interpreted as zero. For this example, if ϵ is chosen to be 0.01, then the last singular value would be interpreted as zero, and the numeric rank would (correctly) be determined to be two.

We wish to avoid this complication, since incorrect rank determination will cause the Least Squares methods to discover a minimum that may not satisfy the primary constraints, i.e.

$$Ax \neq b \quad (\text{or } Cy \neq 0).$$

This would in turn lead to polynomials that do not provide a result for the desired operation. By applying the more modern (relative) accuracy bounds, we can ensure correct rank determination, and avoid these concerns. We first discuss traditional error bounds and note why they are inadequate in some cases.

Traditional Error Bounds

With both input error and roundoff, a traditional algorithm computes the SVD for a perturbed version of an exact matrix $A \in \mathbb{R}^{m\times n}$, i.e.

$$\hat{A} = A + \Delta A.$$

ΔA is kept small by applying the restriction $||\Delta A|| \leq \eta||A||$ for $0 \leq \eta \ll 1$. The error bounds that one can achieve is:

> Let $\hat{U} \in \mathbb{R}^{m\times m}, \hat{\Sigma} \in \mathbb{R}^{m\times n}, \hat{V}^T \in \mathbb{R}^{n\times n}$ be the SVD of input matrix $\hat{A}$, where $\hat{\Sigma} = diag(\hat{\sigma_1}, \ldots, \hat{\sigma_n})$. Then the traditional error bound, for algorithms that provide *absolute accuracy* is:
>
> $$|\sigma_i - \hat{\sigma_i}| \leq \eta \cdot ||G|| = \eta \cdot \sigma_1. \tag{4.1}$$

The values σ_i are the singular values of the exact matrix A. From (4.1) it is clear that the error in all of the singular values is bounded in terms of the largest singular value, σ_1. In particular, the smallest singular value σ_r may not be determined to any accuracy at all if $\sigma_1 \gg \sigma_r$.

Relative Accuracy Bounds

As for traditional bounds, the input shall be a perturbed version of the exact matrix A, however, in [7], it is shown that $\hat{A}$ can be expressed as a multiplicative perturbation of A, i.e.,

$$\hat{A} = \hat{E}A\hat{F}.$$

$\hat{E}, \hat{F}$ are defined as

$$\hat{E} = I + E,$$
$$\hat{F} = I + F.$$

The error in the input is bounded by constraining the perturbations E, F as $||E|| \leq \eta_E, ||F|| \leq \eta_F$, and defining $\eta = max(\eta_E, \eta_F)$. Then we have:

> Let $\hat{U}, \hat{\Sigma}, \hat{V}^T$ be the SVD of input matrix $\hat{A}$, where $\hat{\Sigma} = diag(\hat{\sigma_1}, \ldots, \hat{\sigma_n})$. Then the bound for algorithms that provide guaranteed *relative accuracy* is:
>
> $$\frac{|\sigma_i - \hat{\sigma_i}|}{\sigma_i} \leq \eta_E + \eta_F + \eta_E\eta_F \leq 2\eta + \eta^2. \quad (4.2)$$

Any algorithm that meets this relative accuracy bound can thus guarantee the smallest singular value(s) are computed accurately. We can thus use this SVD computation to confidently determine the solutions for LS and TLS. Note that this does *not* mean that algorithms that can only provide the traditional bound will not be able to determine solutions for LS and TLS. Relative error bounds simply give confidence that the computed values are correct for the input data.

4.2 Singular Value Decomposition

Algorithms for computing the singular value decomposition have a long history, see e.g. [14]. With [8], algorithms that provide this guaranteed relative accuracy have been developed. Most such algorithms follow the same steps:

1. Reduce the input matrix A to bidiagonal form B using orthogonal matrices U_1, V_1, so $A = U_1BV_1$.

2. Determine the SVD of B, i.e. U_2, Σ, V_2 such that $B = U_2\Sigma V_2$

3. Combine these to get the SVD of $A = (U_1U_2)\Sigma(V_1V_2)^T$

In [9], the practical algorithms for the SVD are described:

1. *QR Iteration and its variations*
   These algorithms are generally the fastest for determining the singular values of a bidiagonal matrix. The method of [8] is of this form, and shall be explored in detail. It has the advantage that is also provides the required singular vectors.

2. *Divide and Conquer*
   The fastest algorithm for large dimension matrices, but does not provide relative accuracy of singular values.

3. *Bisection and Inverse Iteration*
   This method works well for determining the singular values within an interval, and can provide guaranteed relative accuracy of singular values by narrowing the interval. However, the singular vectors may suffer from a loss of orthogonality, making it impossible to recover the perturbed matrix as required in our applications.

4. *Jacobi's Method*
   By implicitly applying Jacobi's Method to AA^T, some classes of matrices A can yield accurate singular values. Attempts to fit the matrices derived from the polynomial operations could yield a better SVD-based algorithm.

All but Jacobi's method follow the 3 steps outlined above, involving first reducing the input matrix to bidiagonal form, meaning a matrix that is zero except for the diagonal and first super-diagonal.

4.2.1 Bidiagonal SVD

For an input matrix A, orthogonal matrices $U_1, V_1 \in \mathbb{R}^{n\times n}$ can be constructed such that $A = U_1 B V_1$ (see [2]), with B a matrix of bidiagonal form. The question is then to determine the Singular Value Decomposition of B with guaranteed relative accuracy.

Perturbation Bounds

Consider a symmetric tridiagonal matrix J, with zero diagonal in $\mathbb{R}^{n\times n}$, i.e.:

$$J = \begin{bmatrix} 0 & b_1 & 0 & \dots & 0 \\ b_1 & 0 & b_2 & \ddots & \vdots \\ 0 & b_2 & 0 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & b_{n-1} \\ 0 & \dots & 0 & b_{n-1} & 0 \end{bmatrix}. \tag{4.3}$$

Construct $J + \Delta J$, by multiplying one offdiagonal entry of J by $\alpha \neq 0$. Then, for $\bar{\alpha} = \max(|\alpha|, |\alpha|^{-1})$, Theorem 2 of [8] established that if λ_i are the eigenvalues of J, then the eigenvalues $\bar{\lambda_i}$ of $J + \Delta J$ are bounded by:

$$\frac{\lambda_i}{\bar{\alpha}} \leq \bar{\lambda_i} \leq \bar{\alpha}\lambda_i.$$

So a change in one of the entries of the matrix can only change the eigenvalues by that same amount. This can clearly be applied for perturbations to all the entries, taking $\bar{\alpha} = \prod_{i=1}^{n} \max|\alpha_i|, |\alpha_i|^{-1}$. No eigenvalue will change more than this product.

Note that for any matrix B, the eigenvalues of

$$\bar{B} \equiv \begin{bmatrix} 0 & B^T \\ B & 0 \end{bmatrix}$$

are equal to the singular values of B (and possibly their negatives). So perform a permutation on the rows and columns of $\bar{B}$ formed when B is bidiagonal, to yield a suitable matrix J of the form (4.3).

Then for

$$B + \Delta B = \begin{bmatrix} \alpha_1 b_1 & \alpha_2 b_2 & 0 & \dots & 0 \\ 0 & \alpha_3 b_3 & \alpha_4 b_4 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \alpha_{2n-3} b_{2n-3} & \alpha_{2n-2} b_{2n-2} \\ 0 & \dots & 0 & 0 & \alpha_{2n-1} b_{2n-1} \end{bmatrix},$$

$$\bar{\alpha} = \prod_{i=1}^{2n-1} \max\left(|\alpha_i|, |\alpha_i|^{-1}\right),$$

we have

$$\frac{\sigma_i}{\bar{\alpha}} \le \bar{\sigma_i} \le \bar{\alpha}\sigma_i.$$

σ_i are the singular values of B and $\bar{\sigma_i}$ are the singular values of $B + \Delta B$. This result provides the vehicle for relatively accurate singular values, as opposed to the traditional absolute accuracy.

Zero-Shift QR Iteration

Traditional QR Iteration on $B^T B$ can be used to compute the singular values of a bidiagonal matrix B. This algorithm iteratively refines B to converge on a diagonal matrix whose entries are the singular values of B.

Each iteration proceeds by first computing a shift σ^2, and (implicitly) forming the QR factorization of the matrix $QR = B^T B - \sigma^2 I$. The next B is generated from

$$B^T B = RQ + \sigma^2 I.$$

However, the roundoff errors can be order ϵB, which can destroy the relative accuracy of the singular values.

The standard method of computing the QR decomposition of a matrix involves applying Given's rotations about a certain angle, using Wilkinsons's shift. By setting the shift to zero, and carefully choosing the angles of rotation, the offdiagonal elements can be zeroed *without* the possibility for cancellation, i.e. without $-\sigma^2 I$.

Let $J(i,j,\theta)$ denote the Given's rotation of the i,j entries by θ, i.e. for $n=5$:

$$J(3,4,\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \cos\theta & \sin\theta & 0 \\ 0 & 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The choice of θ is based on making the j^{th} entry of $(J(i,j,\theta))\, x$ equal to zero, for a given vector x. This can be accomplished by choosing $\tan\theta = x_j/x_i$ for a particular x.

Both algorithms begin by applying the Given's rotation $J_1 \equiv J(1,2,\theta_1)$. They differ in their choice of θ_1, as will be discussed in a moment. B is post-multiplied by J_1, introducing a non-zero value in the (2,1) entry of BJ_1. This non-zero is then "chased" off the matrix by a series of further rotations.

$J_2 \equiv J(1,2,\theta_2)$ is chosen to introduce a zero in this (2,1) entry of J_2BJ_1, but will then introduce a non-zero in the (1,3) entry. $J_3 \equiv J(2,3,\theta_3)$ is then chosen to "chase" that non-zero down to the (3,2) position, and the process continues for the duration of the iteration.

In total, $2(n-1)$ rotations will be required to chase the non-zero entry off of the matrix.

Now, standard QR Iteration chooses the original angle, θ_1, to introduce a zero in the (2,1) entry of $J_1^T(B^TB - \sigma^2 I)$, so

$$\tan\theta_1 = \frac{(B^TB)_{1,2}}{\sigma^2 - (B^TB)_{1,1}}.$$

Using Wilkinson's shift guarantees linear convergence to the desired diagonal matrix [8].

However, for zero-shift QR, we simply need to zero the (2,1) entry of $J_1^T(B)$, so

$$\tan\theta_1 = -\frac{(B^TB)_{1,2}}{(B^TB)_{1,1}} = -\frac{b_{1,2}}{b_{1,1}}.$$

Thus when the matrix J_1 is applied to B, the (1,2) entry of BJ_1 will be zero. Again the non-zero in the (2,1) will have to be "chased" off the matrix with further rotation, but this zero in (1,2) will propagate through the rotation. This propagation can be incorporated into an algorithm, yielding improved efficiency over standard QR iteration.

Implementation

The zero-shift QR algorithm is fairly simple to implement. First we set up a procedure, *GivensRotation* , which computes the values of $\cos, \sin$ (and r) for a given vector $[f, g]$, i.e.

$$\begin{bmatrix} \cos & \sin \\ -\sin & \cos \end{bmatrix} \begin{bmatrix} f \\ g \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}.$$

The diagonal elements of the matrix are stored in a vector $s = [s_1 \ldots s_n]$, and the off-diagonal entries in a vector $e = [e_1 \ldots e_{n-1}]$. Then one iteration can be summarized as:

Initialization Phase:

```
1. $\cos_0 = 1$;
2. $f = s_1 \quad g = e_1$;
```

for i from 1 to n-1

```
    1. $\cos, \sin, r = GivensRotation(f, g)$;
    2. $e_{i-1} = r\sin_0$;
    3. $f = r\cos_0$;
    4. $g = s_{i+1}\sin \quad h = s_{i+1}\cos$;
    5. $cos, sin, r = GivensRotation(f, g)$;
    6. $s_i = r$;
    7. $f = h \quad g = e_{i+1}$;
    8. $\sin_0 = \sin \quad \cos_0 = \cos$;
```

end for

Cleanup Phase:

1. $e_{n-1} = h \sin$;
2. $s_n = h \cos$.

There is clearly an absence of possible cancellation, so the algorithm shall proceed at nearly machine precision.

Convergence

The convergence rate of the zero shift QR on a symmetric matrix is known to be essentially the same as inverse iteration [8]. The last off-diagonal element will linearly converge to zero with a constant factor of $\sigma_{n-1}^2/\sigma_n^2$.

4.2.2 Experimental Results

We present here results for the bidiagonal SVD algorithm. First we present results for increased QR iterations:

Iterations	Roundoff error in σ_n	Relative error in σ_n	Roundoff error for all σ_i
10	0.1125208366	0.05356465432	12.75890948
20	0.1978281924	0.02912351033	6.325897846
50	0.004771707879	0.0006027857690	2.266760745
100	$2.187994035 \times 10^{-15}$	$1.544765955 \times 10^{-17}$	1.1737740818

Table 4.1: Error in the computed singular values for different iterations

The relative errors in σ_n are made quite small in early iterations. Varying dimension has no negative effect on the relative accuracy, as can be seen from the following two tables. The first presents the error for random matrices, and the second presents the error when the matrices are constructed to have one very small singular value, of order 10^{-30}.

Dimension	Roundoff error in σ_n	Relative error in σ_n	Roundoff error for all σ_i
3	$5.377902855 \times 10^{-4}$	$7.767028455 \times 10^{-6}$	$1.289983128 \times 10^{-3}$
5	$1.289338934 \times 10^{-6}$	$2.601620526 \times 10^{-8}$	$2.518860262 \times 10^{-3}$
10	$1.081430462 \times 10^{-8}$	$8.618170255 \times 10^{-10}$	$2.589038785 \times 10^{-2}$
20	$6.662715090 \times 10^{-10}$	$5.337934560 \times 10^{-11}$	0.1238719948
50	$2.401918446 \times 10^{-12}$	$8.006394821 \times 10^{-13}$	9.440612729

Table 4.2: Error in the computed singular values for varying dimension

Dimension	Roundoff error in σ_n	Relative error in σ_n	Roundoff error for all σ_i
3	$3.978700646 \times 10^{-30}$	$3.947160142 \times 10^{-2}$	$6.980728515 \times 10^{-4}$
5	$4.958117479 \times 10^{-30}$	$4.925267976 \times 10^{-2}$	$1.206230610 \times 10^{-3}$
10	$4.577435537 \times 10^{-30}$	$4.482349158 \times 10^{-2}$	0.02467295934
20	$5.651057135 \times 10^{-30}$	$5.361805845 \times 10^{-2}$	0.1378697980
50	$1.294070995 \times 10^{-28}$	$8.189721750 \times 10^{-1}$	1.088537792

Table 4.3: Error in the computed singular values for varying dimension of matrices with one small singular value

We lastly contrast the errors for varying coefficient sizes for both random matrices and matrices constructed with small singular values. Again we see that the relative error is not adversely affected by widely varying coefficient size.

Entry Range	Roundoff error in σ_n	Relative error in σ_n	Roundoff error for all σ_i
$0 \ldots 10$	$2.141029366 \times 10^{-4}$	$2.140981382 \times 10^{-4}$	$3.067681214 \times 10^{-2}$
$-10 \ldots 10$	$8212201265 \times 10^{-5}$	$7.261861170 \times 10^{-5}$	$3.640709528 \times 10^{-2}$
$-100 \ldots 100$	$7.399891445 \times 10^{-8}$	$2722134104 \times 10^{-8}$	0.1159135002
$-10^{10} \ldots 10^{10}$	$1.379236650 \times 10^{5}$	$8.338349303 \times 10^{-6}$	$9.825063442 \times 10^{7}$

Table 4.4: Error in the computed singular values for varying size of entries

Entry Range	Roundoff error in σ_n	Relative error in σ_n	Roundoff error for all σ_i
$0 \ldots 10$	$2.598692468 \times 10^{-31}$	$2.476215472 \times 10^{-3}$	$2.817809720 \times 10^{-2}$
$-10 \ldots 10$	$3.277414658 \times 10^{-31}$	$1.145099780 \times 10^{-2}$	$3.033119536 \times 10^{-2}$
$-100 \ldots 100$	$5.749740370 \times 10^{-30}$	$5578860270 \times 10^{-1}$	$9.116097130 \times 10^{-2}$
$-10^{10} \ldots 10^{10}$	$1.215984740 \times 10^{7}$	$1.312539217 \times 10^{-8}$	$8.749806106 \times 10^{7}$

Table 4.5: Error in the computed singular values for varying size of entries, of matrices with one small singular value

Discussion

These results provide the verification that the bidiagonal SVD algorithm determines singular values with low relative error. In particular, the relative error in a very small singular value was shown to be correct.

Once again, we emphasize that the presence of guaranteed relative accuracy merely gives confidence in the computed singular values. We found it extremely rare that traditional algorithms for the singular value decomposition were not able to provide sufficiently accurate singular values, particularly for our applications to Least Squares problems for approximate polynomial operations.

4.3 Riemannian Singular Value Decomposition

From (3.11) for the STLS problem, we have the RiSVD formulation:

Find the triplet (u, τ, v) corresponding to the smallest τ such that

$$\begin{aligned} Cv &= D_v u\tau, \quad u^T D_v u = 1, \\ C^T u &= D_u v\tau, \quad v^t D_u v = 1, \\ & v^T v = 1. \end{aligned}$$

This non-linear SVD corresponds to the restricted singular value decomposition (RSVD) if the matrices D_u, D_v are assumed to be constant (see [20] for an extensive study of the RSVD).

This suggests an iterative algorithm, as noted in [19], that would hold these matrices constant for each iteration, and refine solutions u, v.

From (3.12), we have the translation of (3.11) to the lower triangular linear system:

$$\begin{bmatrix} R^T & 0 & 0 \\ Q_2^T D_v Q_1 & Q_2^T D_v Q_2 & 0 \\ Q_1^T D_v Q_1 \tau & Q_1^T D_v Q_2 \tau & -R \end{bmatrix} \begin{bmatrix} z \\ w \\ v \end{bmatrix} = \begin{bmatrix} D_u v\tau \\ 0 \\ 0 \end{bmatrix}.$$

The iterations will proceed as follows:

1. $z_{i+1} = R^{-T} D_{u_i} v_i \tau_i$;
2. $w_{i+1} = -(Q_2^T D_{v_i} Q_2)^{-1} (Q_2^T D_{v_i} Q_1) z_{i+1}$;
3. $u_{i+1} = Q_1 z_{i+1} + Q2 w_{i+1}$;
4. $v_{i+1} = R^{-1} Q_1^T D_{v_i} u_{i+1}$.

The vectors v_i must then be normalized, or else $||v_i|| \to 0$, even though $||v_i||||u_i||$ remains constant.

A second normalization is required to ensure the following constraint holds:

$$v_i^T D_{u_1} v_i = u_i^T D_{v_1} u_i = 1.$$

We can specialize this algorithm to the case when $m = n$, which simplifies the system to:

$$\begin{bmatrix} R^T & 0 \\ Q^T D_v \tau & -R \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} D_u v \tau \\ 0 \end{bmatrix}.$$

We provide two implementations, one for regular (rectangular) matrices, and one for these special square matrices. We also provide methods of converting rectangular matrices to square ones, in an attempt to use the square algorithm on rectangular matrices.

4.3.1 Implementation

As we require a functioning algorithm for the RiSVD of several different structured matrix classes, we implemented the inverse iteration algorithm of [19]. This algorithm was chosen for it's simplicity and generality over the different matrix classes. Where efficiency is a concern, the algorithm should be tuned for the particular matrix structure in question.

We first describe our simplified square matrix version of the algorithm of [19]. We use placeholders for forming the structured matrix of the algorithm described in 3.4.1, namely D_{u_i}, D_{v_i} and C.

Input: $\hat{c}, B$, where $\hat{c}$ is a list of the entries of the matrix $\hat{C}$, and B is a list of structure basis matrices.
Output: c, v, where c contains the entries of the perturbed matrix C, and v is a vector such that $Cv = 0$.

```
  Initialization Phase:
    1. $Q, R =$ the QR Decomposition of $C$;
    2. $U, \Sigma, Vt =$ the SVD of $C$;
    3. $\tau_0, u_0, v_0 =$ the smallest singular value and it's corresponding singular vectors;
    4. $\gamma_0 = \left(v_0^T D_{u_0} v_0\right)^{1/4}$;
    5. $u_0 = u_0/\gamma_0 \quad v_0 = v_0/\gamma_0$;
Repeat for $i = 1$ until convergence:
    System Solving Phase:
      1. $u_i = Q\left(R^{-T} D_{u_{i-1}} v_{i-1} \tau_{i-1}\right)$;
      2. $v_i = R^{-1} Q^T D_{v_{i-1}} u_i \tau_{i-1}$;
    Normalization Phase:
      1. $v_i = v_i/||v_i||$;
      2. $\gamma_i = \left(v_i^T D_{u_i} v_i\right)^{1/4}$;
      3. $u_i = u_i/\gamma_i \quad v_i = v_i/\gamma_i$;
      4. $\tau_i = u_i^T C v_i$;
End Repeat
```

For rectangular matrices, we present the algorithm in detail as discussed in section 3.4.1. But first, note that you can use the previous algorithm for square matrices on a matrix of dimension $m \times n$. This is accomplished by pre-multiplying the input matrix C by a matrix $E \in \mathbb{R}^{n \times m}$. The new basis matrices are EB_i, where B_i were the original basis matrices.

By choosing random E that does not introduce dependencies into EC, we see that the computed EC, v will often have the desired property that $Cv = 0$.

We have also implemented the full algorithm:

Input: $\hat{c}, B$, where $\hat{c}$ is a list of the entries of the matrix $\hat{C}$, and B is a list of structure basis matrices.
Output: c, v, where c contains the entries of the perturbed matrix C, and v is a vector such that $Cv = 0$.

```
Initialization Phase:
    1. $Q_1, Q_2, R =$ the full QR Decomposition of $C$;
    2. $U, \Sigma, Vt =$ the SVD of $C$;
    3. $\tau_0, u_0, v_0 =$ the smallest singular value and it's corresponding singular vectors;
    4. $\gamma_0 = \left(v_0^T D_{u_0} v_0\right)^{1/4}$;
    5. $u_0 = u_0/\gamma_0 \quad v_0 = v_0/\gamma_0$;
Repeat for $i = 1$ until convergence:
    System Solving Phase:
        1. $z_i = \left(R^{-T} D_{u_{i-1}} v_{i-1} \tau_{i-1}\right)$;
        2. $w_i = -\left(Q_2^T D_{v_{i-1}} Q_2\right)^{-1} \left(Q_2^T D_{v_{i-1}} Q_1\right) z_i$;
        3. $u_i = Q_1 z_i + Q_2 w_i$;
        4. $v_i = R^{-1} Q^T D_{v_{i-1}} u_i \tau_{i-1}$;
    Normalization Phase:
        1. $v_i = v_i / ||v_i||$;
        2. $\gamma_i = \left(v_i^T D_{u_i} v_i\right)^{1/4}$;
        3. $u_i = u_i/\gamma_i \quad v_i = v_i/\gamma_i$;
        4. $\tau_i = u_i^T C v_i$;
End Repeat
```

The major difference here is the extra steps in the system solving phase. First, we require the full QR decomposition of C, partitioned as:

$$C = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix}.$$

Next, we have to solve for z, w at each iteration (where $u = Q_1 z + Q_2 w$). This involves the potentially costly inversion of $\left(Q_2^T D_{v_{i-1}} Q_2\right)$ when determining w. Again, the particular matrix structure is the determining factor for the practicality of determining w, either by the computing the stated inversion, or solving the system $\left(Q_2^T D_{v_{i-1}} Q_2\right) w = \left(Q_2^T D_{v_{i-1}} Q_1 z\right)$.

4.3.2 Experimental Results

The results of the previous section, for STLS, were compiled using the RiSVD algorithm. As such, we simply contrast the speed of the two techniques (squared vs rectangular) here, as well as providing a measure for the failure caused by the randomization. The metrics below are for Toeplitz matrices.

Rows	Columns	Speedup	Success Rate
20	2	29.14%	0.12
20	5	23.67%	0.26
20	10	21.49%	0.54
20	15	18.91%,	0.87
20	20	2.057907268%	1.00

Table 4.6: Speedup and Success Rate of various eccentricities of input matrix

4.4 Conclusion

In this chapter, we have presented accurate algorithms for both the regular singular value decomposition and the Riemaniann singular value decomposition. These algorithms permit the use of the Least Squares methods described in the previous chapter, including on severely ill-conditioned matrices.

Using these, we can now implement the approximate polynomial operations with confidence, using the solutions from the various Least Squares problems.

Chapter 5

Results

This chapter outlines the results of applying the solutions to the Least Squares problems of Chapter 3 to the polynomial operations of Chapter 2. For each operation, we show an example illustrating the steps of the algorithm. Results are then presented for randomly selected input polynomials, as well as for polynomials known to be *close* to others that yield a non-trivial result for the operation in question.

5.1 Metrics

Our approximate polynomial operations take, as input, a set of one or two polynomials, namely $f \in \mathbb{R}[x]$[1] or $f, g \in \mathbb{R}[x]$. The notation introduced here for f should also be applied to g where applicable.

We shall denote the computed polynomial $\hat{f}$, which are equal to $f + \Delta f$ for some (hopefully small) polynomial Δf. In order to measure the success of our operations in terms of a (loose) upper bound, we construct input polynomials f from another set of polynomials $\bar{f}$, which are known to give a non-trivial result for the operation in question. We then seek to verify

$$||f - \hat{f}|| \leq ||f - \bar{f}||.$$

[1]For factorization, the polynomial would be $f \in \mathbb{R}[x, y]$

We also present $||f - \hat{f}||$ for randomly generated input polynomials f, of varying degree d and magnitude range of coefficients.

Unless otherwise stated, the results presented in this chapter have been run on a random sampling of 100 inputs. The presented metrics are the average values attained for that sample. In all cases, we have ensured that no individual sample was outside any sought-after bounds.

5.2 Polynomial Division

For input (approximate) polynomials $\hat{p}, \hat{q} \in \mathbb{R}[x]$, we seek nearby polynomials

$$\begin{aligned} p &= \hat{p} + \Delta p, \\ q &= \hat{q} + \Delta q, \end{aligned} \tag{5.1}$$

such that $q \mid p$. The polynomial r is also determined that satisfies $p/q = r$.

5.2.1 Example

Consider example (2.1), which had input

$$\begin{aligned} p &= 3.02x^2 + 6.98x + 2, \\ q &= 2.78x + 0.96. \end{aligned}$$

The multiplication matrix Q for the polynomial q is

$$\begin{bmatrix} 0.96 & 0 \\ 2.78 & 0.96 \\ 0 & 2.78 \end{bmatrix},$$

and we seek an r such that solves the obviously over-constrained system:

$$\begin{bmatrix} 0.96 & 0 \\ 2.78 & 0.96 \\ 0 & 2.78 \end{bmatrix} \begin{bmatrix} r_0 \\ r_1 \end{bmatrix} \approx \begin{bmatrix} 2 \\ 6.98 \\ 3.02 \end{bmatrix}. \tag{5.2}$$

Least Squares

The Least Squares solution is

$$\begin{aligned}\begin{bmatrix}0.96 & 0\\ 2.78 & 0.96\\ 0 & 2.78\end{bmatrix}\begin{bmatrix}2.129494442\\ 1.08823181892465714\end{bmatrix} &= \begin{bmatrix}2\\ 6.98\\ 3.02\end{bmatrix}+\begin{bmatrix}0.044314664\\ 0.015302906\\ 0.00528445661054679\end{bmatrix}\\ &= \begin{bmatrix}2.044314664\\ 6.96469709439862862\\ 3.02528445661054679\end{bmatrix}.\end{aligned}$$

Notice that LS does not perturb the multiplication matrix, only the right-hand side. The result is a perturbed polynomial $\hat{p} = p + \Delta p$ and a polynomial r such that $\hat{p}/q = r$, i.e.

$$\frac{3.02528445661054679x^2 + 6.96469709439862862x + 2.044314664}{2.78x + 0.96} = 1.08823181892465714x + 2.129494442.$$

The distance from the perturbed $\hat{p}$ to p is computed to be 0.04717937964 using the polynomial 2-norm.

Total Least Squares

The TLS result for (5.2), has left hand side:

$$\left(\begin{bmatrix}0.96 & 0\\ 2.78 & 0.96\\ 0 & 2.78\end{bmatrix}+\begin{bmatrix}-0.0140677749386000084 & -0.0071888946710800001 0\\ 0.00477751005300008914 & 0.00244139650460006852\\ -0.169088714121500006 & -0.00086407478199967968 7\end{bmatrix}\right)\begin{bmatrix}r_0\\ r_1\end{bmatrix}$$

$$= \begin{bmatrix}0.9459322250614 & -0.007188894671080\\ 2.784777510053 & 0.9624413965046\\ -0.001690887141215 & 2.779135925218\end{bmatrix}\begin{bmatrix}2.129570368\\ 1.088250070675\end{bmatrix},$$

and right hand side:

$$\begin{aligned}
&\begin{bmatrix} 2 \\ 6.98 \\ 3.02 \end{bmatrix} + \begin{bmatrix} 0.006605922 \\ -0.002243415 \\ 0.000794004 \end{bmatrix} \\
= \quad &\begin{bmatrix} 2.006605922 \\ 6.977756585025 \\ 3.020794003883 \end{bmatrix}.
\end{aligned}$$

The polynomials $r, \hat{p}$ can be read off the system as in the LS case, giving us the values

$$\begin{aligned}
\hat{p} &= 3.020794003883x^2 + 6.977756585025x + 2.006605922, \\
r &= 1.088250070675x + 2.129570368.
\end{aligned}$$

However, TLS also perturbed the matrix Q in arriving at these values. Due to this, there may not be a polynomial $\hat{q}$ such that $\hat{p}/\hat{q} = r$, and this result provides little more than polynomials $\hat{p}, r$ that may be closer to being divisible. To check how far off the solution is, we compute the residual $\hat{p} - qr$ for both the original polynomial q and a somewhat naive attempt to recover $\hat{q}$ from the first column of $Q + \Delta Q$.

This $\hat{q}$ will likely introduce higher degree than q, if the zeroes in the first column of Q were destroyed by TLS. We compute the residual with $\hat{q}_1, \hat{q}_2$, where $\hat{q}_1$ includes the higher degree terms, and $\hat{q}_2$ does not. [2]

The result of using these estimates $\hat{q}_1, \hat{q}_2$ for $\hat{q}$, and the original polynomial q, are:

$$\begin{aligned}
p - qr &= -0.004541193x^2 + 0.012830894x - 0.037781631, \\
p - \hat{q}_1 r &= 0.001840108051x^3 - 0.006139456x^2 + 0.017966107x - 0.007823315, \\
p - \hat{q}_2 r &= 0.001840108051x^3 - 0.006139456x^2 + 0.016125999x - 0.011424178.
\end{aligned}$$

[2] There are all sorts of other heuristics that can be applied to $Q + \Delta Q$ to recover a polynomial q that is hoped to divide p. Instead, we shall use STLS to guarantee a q that divides p.

Clearly any of these options does not have zero residual, so the choices for $\hat{q}$ are not correct. Note that $||\Delta p|| = 0.007021506867$ is quite small, as expected. This p is certainly closer to $\hat{p}$ than the one computed by LS. But it does not result in a pair of divisible polynomials.

Structured Total Least Squares

Solving STLS on the system (5.2) gives a left hand side perturbation of:

$$\left(\begin{bmatrix} 0.96 & 0 \\ 2.78 & 0.96 \\ 0 & 2.78 \end{bmatrix} + \begin{bmatrix} -0.0192694923999999768 & 0 \\ 0.00650542400000020393 & -0.0192694923999999768 \\ 0 & 0.00650542400000020393 \end{bmatrix} \right) \begin{bmatrix} r_0 \\ r_1 \end{bmatrix}$$
$$= \begin{bmatrix} 0.9407305076 & 0 \\ 2.786505424 & 0.9407305076 \\ 0 & 2.786505424 \end{bmatrix} \begin{bmatrix} 2.13757001674005265 \\ 1.08423967866386128 \end{bmatrix},$$

and right hand side:

$$\begin{bmatrix} 2 \\ 6.98 \\ 3.02 \end{bmatrix} + \begin{bmatrix} 0.0108773270000002144 \\ -0.00367221000000039766 \\ 0.00123974599999998603 \end{bmatrix}$$
$$= \begin{bmatrix} 2.010877327 \\ 6.97632779 \\ 3.021239746 \end{bmatrix}.$$

Now the structure of Q was preserved, so we can read $\hat{p}, r$ *and* $\hat{q}$ off of the system. We have:

$$\hat{p} = 3.021239746x^2 + 6.976327794x + 2.010877327,$$
$$\hat{q} = 2.786505424x + .9407305076,$$
$$r = 1.08423967866386128x + 2.13757001674005265.$$

As expected, $\hat{p} - \hat{q}r = -0.0000000001x$, which we shall call zero. [3]

[3]Since it's a negligible difference, presumably due to roundoff error in the computation

The polynomial norms $||\Delta p||, ||\Delta q||$ are 0.01154722214, 0.02033799102 respectively. This is in an improvement over the LS solution, which had $||\Delta p|| = 0.04717937964$. The results of the next section show the differences in norm more thoroughly.

5.2.2 Metrics

Polynomial division takes two input polynomials p, q as input, so the metrics run will be on the polynomials p, q and their corresponding matrix systems.

5.2.3 Results

We begin by illustrating the size of the residual when using a naive choice [4] of coefficients of q from the matrix $Q + \Delta Q$ constructed to solve the TLS problem.

Coefficient Range	$\|\|\hat{p} - \hat{q}r\|\|$	$\frac{\|\|\hat{p}-\hat{q}r\|\|}{\|\|p\|\|^2+\|\|q\|\|^2}$
0..10	9.658454272	0.6118358388
-10..10	45.61990497	1.619070883
-100..100	215.0753496	1.542072386
$-10^{-10}..10^{-10}$	$6.923436617 \times 10^{-9}$	0.9983980614
$-10^{10}..10^{10}$	$1.800493692 \times 10^{10}$	1.051108620

Table 5.1: Size of the residual for computed $\hat{p}, \hat{q}$ for varying coefficient sizes

The results of Table 5.1 are using $\hat{q} = q_2$, which appears to be the best of our naive choices for the coefficients of $\hat{q}$, as discussed in the example.

We now present the proximity of the computed polynomials for each of the three Least Squares solutions. Table 5.2 presents the *relative* error for varying coefficient size:

[4]Recall that the solution to the TLS problem does not maintain the structure of Q in $Q + \Delta Q$, so any choice of coefficients of q will likely not divide p *exactly*.

Coefficient Range	LS $\left(\frac{\|\|\Delta p\|\|}{\|\|p\|\|}\right)^2$	STLS $\left(\frac{\|\|\Delta p\|\|}{\|\|p\|\|}\right)^2 + \left(\frac{\|\|\Delta q\|\|}{\|\|q\|\|}\right)^2$	TLS $\left(\frac{\|\|\Delta p\|\|}{\|\|p\|\|}\right)^2 + \left(\frac{\|\|\Delta q\|\|}{\|\|q\|\|}\right)^2$
0..10	0.2135766876	0.07989683728	0.04585486761
-10..10	0.4623086923	0.1172845136	0.06909769835
-100..100	0.4309319746	0.1102616381	0.07852343945
$-10^{-10}..10^{-10}$	0.4740415579	0.1410581410	0.1081253334
$-10^{10}..10^{10}$	0.4823037453	0.1159398145	0.08518335096

Table 5.2: Relative distance from computed polynomials to input polynomials for varying coefficient sizes

Clearly the coefficient size has no dramatic effect on the computed distance. We next proceed for increasing degrees of the polynomials p, q, keeping the degree of q about 1/2 that of p.

$\deg p$	$\deg q$	LS $\left(\frac{\|\|\Delta p\|\|}{\|\|p\|\|}\right)^2$	STLS $\left(\frac{\|\|\Delta p\|\|}{\|\|p\|\|}\right)^2 + \left(\frac{\|\|\Delta q\|\|}{\|\|q\|\|}\right)^2$
2	1	0.1404706996	0.1120178032
5	3	0.1727037476	0.07683297644
10	5	0.1526386114	0.01681960245
50	25	0.1434667582	0.006326673340

Table 5.3: Relative distance from computed polynomials to input polynomials for varying degrees of p,q.

The results from Table 5.3 are particularly encouraging, as increasing degrees does not adversely affect the relative distances. Further, the STLS solution actually appears to get closer with increased degree. We speculate this is due to the increased number of coefficients that can be perturbed, making dependency easier to achieve via perturbation.

With an efficient algorithm (tailored to the matrix for polynomial division) for solving the STLS problem, approximate division of very large degree polynomials may be possible. We next hold the degree of p constant, to examine the effect of the relationship between the degrees of p, q.

		LS	STLS
$\deg p$	$\deg q$	$\left(\frac{\lvert\lvert\Delta p\rvert\rvert}{\lvert\lvert p\rvert\rvert}\right)^2$	$\left(\frac{\lvert\lvert\Delta p\rvert\rvert}{\lvert\lvert p\rvert\rvert}\right)^2 + \left(\frac{\lvert\lvert\Delta q\rvert\rvert}{\lvert\lvert q\rvert\rvert}\right)^2$
10	1	0.03338328964	0.03006721368
10	2	0.07686260030	0.03611606497
10	5	0.1792974687	0.01908563260
10	9	0.3022687571	0.02172860283

Table 5.4: Distance from computed polynomials to input polynomials for varying degrees of q

It appears from this result that STLS problem significantly outperforms the LS solution when $\deg p \rightarrow \deg q$.

Finally we present the verification that the solution to the STLS problem comes within a loose upper bound for the polynomial perturbation. As described in the first section of this chapter, we have constructed p, q to be near polynomials $\bar{p}, \bar{q}$ that *are* divisible. We vary the perturbation as a percentage of the coefficient range of f. Our methods should give polynomials at least as close as those, and we see that they do:

	Constructed $\bar{p}, \bar{q}$	Computed $\hat{p}, \hat{q}$
Perturbation	$\left(\frac{\lvert\lvert p-\bar{p}\rvert\rvert}{\lvert\lvert p\rvert\rvert}\right)^2 + \left(\frac{\lvert\lvert q-\bar{q}\rvert\rvert}{\lvert\lvert q\rvert\rvert}\right)^2$	$\left(\frac{\lvert\lvert p-\hat{p}\rvert\rvert}{\lvert\lvert p\rvert\rvert}\right)^2 + \left(\frac{\lvert\lvert q-\hat{q}\rvert\rvert}{\lvert\lvert q\rvert\rvert}\right)^2$
0.001	0.001808138829	0.000001467121824
0.05	0.01058177531	0.003097649796
0.5	0.6464343778	0.1112758661
1.0	1.442602022	0.1491508842

Table 5.5: The relative polynomial norms for $\hat{p}, \hat{q}$ from the solutions to the STLS problem for perturbed p, q

5.3 Polynomial GCD

For input (approximate) polynomials f, g, we seek nearby polynomials

$$\begin{aligned} \hat{f} &= f + \Delta f, \\ \hat{g} &= g + \Delta g, \end{aligned} \tag{5.3}$$

such that $\hat{f}, \hat{g}$ have a non-trivial GCD d.

5.3.1 Metrics

The distance between these polynomials is clearly $||\Delta f||, ||\Delta g||$ from (5.3), for a suitable choice of polynomial norm.

For the results of this section, we present the sizes of each of these polynomial norms, as well as their sum. This is intended to illustrate that, depending on the application, different choices for the data fitting algorithm may be appropriate.

5.3.2 Example

To illustrate the Least Squares solutions, consider a perturbed version of example 2.6:

$$\begin{aligned} f &= 1.3x^4 + 3.86x^2 + 2.99, \\ g &= 1.6x^3 - 0.66x^2 + 2.1x - 1. \end{aligned}$$

The Sylvester Matrix of f, g is:

$$\begin{bmatrix} 2.99 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 2.99 & 0 & 2.1 & -1 & 0 & 0 \\ 3.86 & 0 & 2.99 & -0.66 & 2.1 & -1 & 0 \\ 0 & 3.86 & 0 & 1.6 & -0.66 & 2.1 & -1 \\ 1.3 & 0 & 3.86 & 0 & 1.6 & -0.66 & 2.1 \\ 0 & 1.3 & 0 & 0 & 0 & 1.6 & -0.66 \\ 0 & 0 & 1.3 & 0 & 0 & 0 & 1.6 \end{bmatrix}.$$

STLS is used to find the closest rank deficient Sylvester matrix, which will then contain the coefficients of the polynomials f, g that have non-trivial GCD. The entries of f, g can then be read directly off of the matrix.

Using the RiSVD based STLS method, we get:

$$\hat{f} = 1.25824867x^4 - 0.01353770965x^3 + 3.89146970x^2 + 0.008763275821x + 2.966337760,$$
$$\hat{g} = 1.590471846x^3 - 0.6565466692x^2 + 2.106918905x - 1.002835919.$$

This seems odd, since we were expecting the polynomials of example 2.6. We compare the distance of the $\hat{f}, \hat{g}$ that we have computed from f, g to that of $\bar{f}, \bar{g}$ (the f, g from example 2.6).

$$||f - \bar{f}|| = 0.3312099032, \quad ||f - \hat{f}|| = 0.05961115731,$$
$$||g - \bar{g}|| = 7.016387960, \quad ||g - \hat{g}|| = 0.01259463756.$$

This makes it clear that the computed $\hat{f}, \hat{g}$ are much closer to f, g than the example data $\bar{f}, \bar{g}$. Hence for this example, one does not have to move f, g very far to find polynomials with a non-trivial GCD.

We multiply the Sylvester matrix by the vector $\begin{bmatrix} u \\ v \end{bmatrix}$ to find the polynomial combination of $\hat{f}, \hat{g}$ that yields the GCD. Computing the rank of $S(\hat{f}, \hat{g})$ indicates the degree of the GCD is two, so we solve the system

$$S(\hat{f}, \hat{g}) \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

for the polynomials u, v. The result is

$$u = 1.258248657x^2 - 0.08178672133x + 2.197212408,$$
$$v = 1.590471846x - 0.7428157879,$$
$$d = 1.000000000x^2 + 0.05424121083x + 1.350046013,$$

The GCD remains close to the one used in the construction of example (2.6), which was $x^2 + 1$. However, as we have seen, the polynomials $\hat{f}, \hat{g}$ are much closer to f, g than $\bar{f}, \bar{g}$ were.

5.3.3 Results

We first consider varying coefficient ranges for the input polynomials.

Coefficient Range	$\frac{\|\|\Delta f\|\|}{\|\|f\|\|}$	$\frac{\|\|\Delta g\|\|}{\|\|g\|\|}$
0..10	0.1049889030	0.08243073495
-10..10	.1280712135	.1574863840
-100..100	.1362641966	.1156061933
$-10^{-10}..10^{-10}$	0.07231108725	0.1797500075
$-10^{10}..10^{10}$	0.07294338785	0.1002649575

Table 5.6: Normalized distance to nearest polynomial for varying coefficient sizes

The results in Table 5.6 demonstrate that, as was the case for division, the coefficient size does not impact the relative proximity of the computed solutions.

We next alter the degrees of f, g, and find that the increasing degrees of g compared to f does not have the same impact as it did for polynomial division.

Degree f	Degree g	$\frac{\|\|\Delta f\|\|}{\|\|f\|\|}$	$\frac{\|\|\Delta g\|\|}{\|\|g\|\|}$
2	1	0.1462551207	0.2185669912
5	3	0.09102810325	0.1358309873
10	1	0.08718679625	0.2826079907
10	5	0.03892541024	0.1273642670
10	9	0.05592024420	0.1457978732
10	10	0.03676434782	0.08493033090

Table 5.7: Relative distance to nearest polynomial for varying degrees

We present here the average degree of the computed GCD's. It is interesting to note that, although we are only seeking the closest non-trivial GCD, occasionally we find polynomials with a GCD of degree greater than one. This becomes more prevalent as the degree of f increases, and particularly when the degree of g approaches that of f.

Degree f	Degree g	Degree of GCD
2	1	1.00
5	3	1.25
10	1	1.00
10	5	1.31
10	9	1.72
10	10	1.78

Table 5.8: Average Degree of Computed GCD

Finally we present the verification that the solution to the STLS problem comes within the loose upper bound for the polynomial perturbation.

As described in the first section of this chapter, we have constructed f, g to be near polynomials $\bar{f}, \bar{g}$ that *have* a non trivial GCD. Our methods should give polynomials $\hat{f}, \hat{g}$, that are at least as close as those, and we see that they do:

Perturbation	Constructed $\bar{f}, \bar{g}$ $\left(\frac{\|\|f-\bar{f}\|\|}{\|\|f\|\|}\right)^2 + \left(\frac{\|\|g-\bar{g}\|\|}{\|\|g\|\|}\right)^2$	Computed $\hat{f}, \hat{g}$ $\left(\frac{\|\|f-\hat{f}\|\|}{\|\|f\|\|}\right)^2 + \left(\frac{\|\|g-\hat{g}\|\|}{\|\|g\|\|}\right)^2$
0.001	$3.182791951 \times 10^{-6}$	$2.407164294 \times 10^{-7}$
0.05	0.009695806014	0.0006064908956
0.5	0.5453310650	0.02018600801
1.0	1.183334514	0.1780722686

Table 5.9: The polynomial norms for f, g of the solutions from the STLS problem

Comparison

We also compare our result to a current implementation of approximate polynomial GCD in Maple 9.5. The Symbolic-Numeric Algorithms for Polynomials (SNAP) package contains an implementation of an QR-based algorithm from [6], which we call QRGCD.

We begin by noting the efficiency of the QRGCD was far superior than our STLS-based GCD for our trials. This was to be expected, since we are using our generalized RiSVD algorithm. Implementation of an RiSVD algorithm specifically designed for Sylvester matrices would clearly lessen this distance.

The GCD of the polynomials computed by the QRGCD were also generally of higher degree in x, compared to those computed from our GCD algorithm. If this is a consideration of the application, the QRGCD result certainly should be considered.

However, our formulation of approximate polynomial GCD simply requires a non-trivial GCD of the computed polynomials. The following table shows the distance of the computed f, g from the inputs for both the QRGCD and our STLS GCD for varying degrees.

Degree f, g	QRGCD	STLS GCD
2,2	0.8582967314	0.001410190318
4,4	1.710840424	0.0007058442816
10,10	0.6526856978	0.0005886810762
20,20	3.133090590	0.00001634913584

Table 5.10: Distance from input to computed polynomials, i.e. $||f - \hat{f}||^2 + ||g - \hat{g}||^2$, for varying polynomial degrees

The QRGCD algorithm requires a tolerance ϵ, which we provided the values of 10^{-i} for $1 \leq i \leq 10$, and took the best resulting polynomials.

We see a significant improvement in the distance for the computed polynomials using our STLS GCD, which indeed shows our algorithm better minimizes the distance to nearby polynomials with non-trivial GCD.

5.4 Bivariate Polynomial Factorization

For an input (approximate) polynomial $f \in \mathbb{R}[x, y]$, we seek a nearby polynomial

$$\hat{f} = f + \Delta f \tag{5.4}$$

that factors into two polynomials g, h of degree greater than 0 in either x, y, i.e. $\hat{f} = gh$.

5.4.1 Metrics

The success of this algorithm is measured by the closeness of the computed $\hat{f}$ to f. We thus present metrics for this distance as a polynomial norm, as well as the matrix norm for the corresponding Ruppert matrices $\hat{C}, C$.

5.4.2 Example

Consider the polynomial product:

$$\begin{aligned}\hat{f} &= (-9y^2 + 21y - 9)(8x - 10) \\ &= -72xy^2 + 168xy + 90y^2 - 72x - 210y + 90.\end{aligned} \tag{5.5}$$

This would be a satisfactory result for approximate factorization on polynomial near $\hat{f}$, such as:

$$f = -71.9996xy^2 + 168.0001xy + 90.0040y^2 - 72.0004x - 209.9990y + 89.9960,$$

which has $||\hat{f} - f|| = 0.005773214010$.

The Ruppert matrix of $\hat{f}$, which is used to represent equation (2.8), will be rank deficient, since $\hat{f}$ is reducible to gh. However the Ruppert matrix of f, namely:

$$R(f) = \begin{bmatrix} 210.0070 & 90.0040 & 0. & -71.9955 & -90.0040 \\ -167.9937 & -71.9955 & 0. & 0. & 0. \\ 143.9928 & 0. & -143.9910 & 0. & 0. \\ 0. & 71.9964 & 167.9937 & 0. & 0. \\ -180.0080 & 0. & 180.0010 & 167.9937 & 210.0070 \\ 0. & -90.0040 & -210.0070 & -71.9964 & -90.00400 \end{bmatrix},$$

is full rank, indicating the perturbed polynomial is irreducible. We apply the methods to solve the least squares problems and attempt to recover the polynomial $\hat{f}$.

Least Squares

Applying LS to the matrix $R(f)$ results in the solution vector and perturbed last column

$$\begin{bmatrix} .208685443367336242 \\ -.486945588191690038 \\ .208691506720978608 \\ 1.25009058659023832 \\ -1 \end{bmatrix}, \quad \begin{bmatrix} -90.0030305868063891 \\ 0.0000513272355462390806 \\ -0.000497434556258724570 \\ 0.000529026947866384490 \\ 210.008851811640398 \\ -90.0016484407331063 \end{bmatrix}.$$

A problem is evident, the LS solution destroys the structure of $R(f)$, and we would have to turn to heuristics to recover the perturbed $\hat{f}$. Clearly the same would be true for TLS, so we move on to STLS.

Structured Total Least Squares

Running STLS on $R(f)$ results in the perturbed matrix $R(\hat{f})$:

$$\begin{bmatrix} 209.9986210 & 89.99666990 & 0 & -71.99951198 & -89.99666990 \\ -168.0005362 & -71.99951198 & 0 & 0 & 0 \\ 143.99894026 & 0 & -143.99902396 & 0 & 0 \\ 0 & 71.99947013 & 168.0005362 & 0 & 0 \\ -180.00842892 & 0 & 179.9933398 & 168.0005362 & 209.998621 \\ 0 & -90.00421446 & -209.9986210 & -71.99947013 & -90.00421446 \end{bmatrix},$$

which is a rank 4 matrix. The recovered polynomial $\hat{f} = -71.99947013xy^2 + 168.0005362xy + 90.00421446y^2 - 71.99951198x - 209.9986210y + 89.99666990$ is hence reducible, by Fact 2 from Chapter 2.

5.4.3 Results

We present the result for varying coefficient size for the polynomial f. As expected, the norms are proportional to the coefficient size, so the normalized values are near constant.

Coefficient Range	$\frac{\|\|f-\hat{f}\|\|}{\|\|f\|\|}$
0..10	$9.551343893 \times 10^{-9}$
-10..10	$1.051857727 \times 10^{-8}$
-100..100	$7.460063028 \times 10^{-9}$
$-10^{-10}..10^{-10}$	$1.012089194 \times 10^{-8}$
$-10^{10}..10^{10}$	$8.031845900 \times 10^{-9}$

Table 5.11: The polynomial norms for varying polynomial coefficient sizes

Next we examine the result with varied degrees of x, y in f.

$\deg_x f$	$\deg_y f$	$\frac{\|\|f-\hat{f}\|\|}{\|\|f\|\|}$
3	3	$3.650159990 \times 10^{-9}$
5	5	$1.008749778 \times 10^{-8}$
5	2	$7.839490195 \times 10^{-10}$
1	10	$7.192546625 \times 10^{-11}$

Table 5.12: The absolute and relative error in polynomial norm for varying degrees of randomly selected input matrices

We finally present the results illustrating the loose upper bound creating by construction of $f = \bar{f} + \Delta r$, where $\bar{f}$ is reducible and the perturbation Δf is of varying size.

	Constructed $\bar{f}$	Computed $\hat{f}$
Perturbation	$\frac{\|\|f-\bar{f}\|\|}{\|\|f\|\|}$	$\frac{\|\|f-\hat{f}\|\|}{\|\|f\|\|}$
0.001	0.001711663405	0.00007071475000
0.05	0.08202853172	0.002880334318
0.5	0.4655400370	0.009005855892
1.0	0.6970496082	0.01817045542

Table 5.13: The absolute and relative polynomial norms for varying percentage perturbations

5.5 Polynomial Decomposition

For an input (approximate) polynomial $f \in \mathbb{R}[x]$, we seek a nearby polynomial

$$\hat{f} = f + \Delta f, \tag{5.6}$$

that is the result of composing two polynomials g, h of degree greater than 0, i.e $\hat{f} = g(h(x))$.

5.5.1 Metrics

The success of this algorithm is measured by the closeness of the computed $\hat{f}$ to f. We thus present metrics for this distance as a polynomial norm, as well as the matrix norm for the corresponding Ruppert matrices $\hat{C}, C$.

5.5.2 Example

Consider the polynomial:

$$\begin{aligned}\bar{f} &= 5(5x^2 - 4x - 1)^2 + 7(5x^2 - 4x - 1) - 9 \\ &= 125x^4 - 200x^3 + 65x^2 - 11.\end{aligned} \tag{5.7}$$

This would be a satisfactory result for a polynomial near $\bar{f}$, such as:

$$f = 125.004x^4 - 200.001x^3 + 64.992x^2 + 12.010x - 10.997,$$

which has $||\bar{f} - f|| = 0.01378404875$. As with bivariate factorization, we will not be able to directly recover the polynomial using LS or TLS.

Structured Total Least Squares

Applying STLS to the Ruppert matrix of the polynomial

$$\phi_f = \frac{f(x) - f(y)}{x - y}$$

results in the polynomial

$$\hat{f} = 125.0027x^4 - 200.0025x^3 + 64.9903x^2 + 12.0079x - 11.0000,$$

whose Ruppert matrix is rank-deficient. This is not the expected polynomial, however, checking polynomial norms, we see that:

$$||\hat{f} - f|| = 0.013024047933250483519,$$

which was less than the distance from $\bar{f}$ to f.

5.5.3 Results

We begin once more by showing the result for varying coefficient size.

Coefficient Range	$\frac{\|\|f-\hat{f}\|\|}{\|\|f\|\|}$
0..10	0.2298759767
-10..10	0.1088538228
-100..100	0.2017510047
$-10^{-10}..10^{-10}$	0.1183948973
$-10^{10}..10^{10}$	0.05163572271

Table 5.14: The polynomial norms for varying polynomial coefficient sizes

Next we consider varying the degree of f in x.

deg f	$\frac{\|\|f-\hat{f}\|\|}{\|\|f\|\|}$
4	0.1857000690
6	0.5693608685
8	0.3129447854
10	0.5845054782

Table 5.15: The polynomial norms for varying polynomial degrees

Finally we check the upper bound forced by construction.

	Constructed $\bar{f}$	Computed $\hat{f}$
Perturbation	$\frac{\|\|f-\bar{f}\|\|}{\|\|f\|\|}$	$\frac{\|\|f-\hat{f}\|\|}{\|\|f\|\|}$
0.001	0.002713153120	0.0005457163710
0.05	0.1040215824	0.001764036423
0.5	0.6309007380	0.1212705356
1.0	0.7983857975	0.1521307424

Table 5.16: The matrix and polynomial norms for varying percentage perturbations

Again the polynomial norm is within the upper bound, so the solution to the STLS problem is within this bound.

5.6 Conclusion

We have demonstrated the successful computation of four fundamental operations on approximate polynomials. The resulting polynomial norms have been shown to be within a modest upper bound. Furthermore, the computation has been shown to be successful for varying polynomial degree and coefficient sizes, with predicted effects on these norms.

Chapter 6

Conclusions and Further Research

This chapter summarizes the conclusions from this work. It further describes areas that may benefit from further research.

6.1 Conclusions

This thesis has presented the application of the Structured Total Least Squares problem to approximate polynomial operations. We now detail the conclusions that have been made as a result of this work.

- For each of the four basic polynomial operations, it is shown that our methods find a nearby set of polynomials with a non-trivial (and hence interesting) result.
- The classical Least Squares problem, and more sophisticated variations on it have been applied to ensure an improved minimization on the polynomial norms of the derived result. In particular, the Structured Total Least Squares result has been generically applied to determine a result for all of the approximate polynomial operations.
- The Least Squares problems have been thoroughly contrasted, and a novel geometric interpretation of the problem being solved has been illustrated. The Least Squares and Total Least Squares problems have been solved using a high accuracy Singular

Value Decomposition. This has resulted in a guarantee that the solution to the Least Squares and Total Least Squares is found.

- The accuracy of the Singular Value Decomposition algorithm has been established to meet prescribed relative accuracy bounds. The increased accuracy has been exploited in cases where rank detection would have previously been ambiguous. The resulting starting values for the RiSVD algorithm thus have the correct numeric rank for the input matrices.

- An inverse iteration algorithm to solve the Riemannian SVD has been presented. This generalized algorithm has been shown to determine a solution to the Rimeannian SVD for many classes of structured matrices, including those defined by corresponding approximate polynomial operations.

- A naive randomized version of the inverse iteration algorithm has been developed, which increases the efficiency dramatically for skewed ($m \gg n$) matrices.

- The application of the Structured Total Least Squares problem to the systems representing approximate polynomial operations has been shown to give results that meet and exceed modest upper bounds. Furthermore, applying this method to random matrices has resulted in polynomials that are very close to the inputs (in terms of distance relative to the input polynomial norm(s)).

- Certain coefficients of the differential equations used for bivariate factorization of polynomials with real coefficients were established to be zero (Fact 3, Section 2.3.1). This reduces the size of the Ruppert matrix by $2n$, where n is the degree of the second variable of the polynomial. The algorithms for solving the Least Squares problems on this reduced matrix will clearly be more efficient. This result was also used to reduce the size of the Ruppert matrix defined for polynomial decomposition.

6.2 Further Research

The application of Structured Total Least Squares to matrix representations of approximate polynomial operations is a new approach. We summarize here some future possibilities in this area.

- In this thesis we focused on the general application of the problem to several approximate polynomial operations. The consequence of this was the adoption of the very general inverse iteration algorithm, which is quite inefficient in its presented form. This algorithm needs to be adapted to run efficiently for *each* class of structured matrices.

- We further motivate the exploration of an algorithm that would determine an efficient algorithm for the STLS problem. That is, given input of the basis matrices for the class of structured matrices, determine an efficient algorithm for convergence on the RiSVD solution.

- The translation to the Least Squares problem results in a minimization in terms of matrix norm. Minimizing this norm may not guarantee a minimal polynomial is found. The link between polynomial and matrix norm may result in a new Least Squares type problem being formulated to ensure minimal polynomial norm.

- The randomization of the inverse iteration algorithm is quite naive. The choice of a squaring matrix should be done much more cautiously, to ensure that a solution to the derived system will also be correct for the original.

- The choice of term ordering for multivariate division, bivariate factorization and decomposition can be altered to potentially improve the result derived from the Least Squares solutions.

- Proof of convergence of the inverse iteration algorithm for the STLS problem would clearly be of great importance.

Bibliography

[1] T.J. Abatzoglou, J.M. Mendel, and G.A. Harada. The constrained total least squares technique and its application to harmonic superposition. *IEEE Transactions on Signal Processing*, 39:1070–1087, 1991.

[2] J. Barlow and J. Demmel. Computing accurate eigensystems of scaled diagonally dominant matrices. *SIAM Journal on Numerical Analysis*, 27(3):762–791, 1990.

[3] Robert M. Corless, André Galligo, Ilias S. Kotsireas, and Stephen M. Watt. A geometric-numeric algorithm for factoring multivariate polynomials. In Teo Mora, editor, *ACM International Symposium on Symbolic and Algebraic Computation*, pages 37–45, Lille, France, 2002. ACM Press.

[4] Robert M. Corless, Patrizia M. Gianni, Barry M. Trager, and Stephen M. Watt. The singular value decomposition for polynomial systems. In *Proceedings of the 1995 international symposium on Symbolic and algebraic computation*, pages 195–207. ACM Press, 1995.

[5] Robert M. Corless, Mark W. Giesbrecht, Mark van Hoeij, Ilias S. Kotsireas, and Stephen M. Watt. Towards factoring bivariate approximate polynomials. In *ACM International Symposium on Symbolic and Algebraic Computation*, pages 85–92. ACM Press, 2001.

[6] Robert M. Corless, Stephen M. Watt, and Lihong Zhi. *QR* factoring to compute the GCD of univariate approximate polynomials. *IEEE Transactions on Signal Processing*, To appear, 2004.

[7] James Demmel, Ming Gu, Stanley Eisenstat, Ivan Slapnièar, Kresemir Veseli, and Zlatko Drmaè. Computing the singular value decomposition with high relative accuracy. *Linear Algebra and Its Applications*, 299(1-3):21–80, 1999.

[8] James Demmel and W. Kahan. Accurate singular values of bidiagonal matrices. *SIAM Journal on Scientific and Statistical Computing*, 11(5):873–912, 1990.

[9] James W. Demmel. *Applied Numerical Linear Algebra.* The Society for Industrial and Applied Mathematics, 1997.

[10] R. Fletcher. *Practical methods of optimization; (2nd ed.).* Wiley-Interscience, 1987.

[11] S. Gao. Factoring multivariate polynomials via partial differential equations. *Mathematics of Computation*, 72:801–822, 2003.

[12] M. Giesbrecht, E. Kaltofen, and J. May. Exact and approximate polynomial and rational function decomposition. Manuscript.

[13] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization.* Academic Press, London and New York, 1981.

[14] G. H. Golub and C. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, 1983.

[15] S. Van Huffel, H. Park, and J.B. Rosen. Formulation and solution of structured total least norm problems for parameter estimation. *IEEE Transactions on Signal Processing*, 44:2464–2474, 1996.

[16] E. Kaltofen and J. May. On approximate irreducibility of polynomials in several variables. In *ACM International Symposium on Symbolic and Algebraic Computation*, pages 161–168, Philadelphia, PA, USA, 2003. ACM Press.

[17] Phillippe Lemmerling. *Structured Total Least Squares: Analysis, Algorithms and Applications.* PhD thesis, Katholieke Universiteit Lueven, Belgium, 1999.

[18] C.F. Van Loan. On the method of weighting for equality constrained least squares problem. *SIAM Journal on Numerical Analysis*, 22:851–864, 1985.

[19] Bart De Moor. Total least squares for affinely structured matrices and the noisy relaxation problem. *IEEE Transactions on Signal Processing*, 42:3004–3113, 1994.

[20] Bart De Moor and G. H. Golub. The restricted singular value decomposition: Properties and applications. *SIAM Journal on Matrix Analysis and Applications*, 12(3):401–425, 1991.

[21] W. M. Ruppert. Reducibility of polynomials f(x,y) modulo p. *Journal of Number Theory*, 77:62–70, 1999.