Multi-Vehicle Function Tracking by Moment Matching

Trevor Avant

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science in Aeronautics & Astronautics

University of Washington

2016

Committee:

Kristi Morgansen

Christopher Lum

Program Authorized to Offer Degree:
Aeronautics & Astronautics

University of Washington

Abstract

Multi-Vehicle Function Tracking by Moment Matching

Trevor Avant

Chair of the Supervisory Committee:
Professor Kristi Morgansen
Aeronautics & Astronautics

The evolution of many natural and man-made environmental events can be represented as scalar functions of time and space. Examples include the boundary and intensity of wildfires, of waste spills in bodies of water, and of natural emissions of methane from the earth. The difficult task of understanding and monitoring these processes can be accomplished through the use of coordinated groups of vehicles. This thesis devises a method to determine positions of the members of a group of vehicles in the domain of a scalar function which lead to effective sensing of the function. This method involves equating the moments of a scalar function to the moments of a group of positions, which results in a system of polynomial equations to be solved. This methodology also allows for other explicit geometric constraints, in the form of polynomial equations, to be imposed on the vehicles. Several example simulations are shown to demonstrate the advantages and challenges associated with the moment matching technique.

TABLE OF CONTENTS

LIST OF FIGURES

NOTATION

$\mathbb{C}$: complex numbers

$\mathbb{R}$: real numbers

$\mathbb{R}_+$: nonnegative real numbers

$\mathbb{N}$: natural numbers, defined as the nonnegative integers 0,1,2,...

ACKNOWLEDGMENTS

I would like to thank and acknowledge Prof. Morgansen for coming up with the moment matching idea as well as the idea of bounding vehicles by their moments. I would also like to generally thank Prof. Morgansen for helping to guide me through this project. Additionally, I would like to thank Dr. Lum for serving as my committee member and helping edit my thesis. I would like to thank all members of the Nonlinear Dynamics and Control Lab for their camaraderie. I especially would like to thank Jake Quenzer who provided many helpful edits to my thesis and Nathan Powel who helped out with many IT problems that affected my computer. Also, I would like to thank Travis Scholl and Iman Datta, both of whom I had helpful conversations with regarding this project. Finally I would like to thank those involved with the ARCS Foundation Fellowship as well as the NDSEG Fellowship for supporting me through the creation of this thesis.

DEDICATION

To all of my teachers throughout my life

Chapter 1
INTRODUCTION

1.1 Motivation

Many natural and man-made environmental events take place over a large area and change with time. Examples in the terrestrial, aquatic and aerial domains are, respectively, wildfires [6], waste spills in bodies of water [16], and seepage of methane from eruptions in the earth [14]. Gathering information about events like these has been limited due to the fact that it would require deploying many sensors, which is a complicated and expensive task. Furthermore, these sensors would need to be mobile and rapidly deployable. Due to these constraints, many environmental events are often poorly understood or cannot be monitored.

Developments in coordinating groups of autonomous vehicles will soon provide a means of overcoming these limitations. Advances in computation, sensing and control have led to the theoretical and experimental demonstration of a variety of multi-vehicle systems.

There are a wide range of environmental events that can be studied with multi-vehicle systems. The approach to studying such an event will depend on which properties of that event are of interest. However, in many cases our interests can be reduced to trying to characterize a scalar function of space and time. Returning to the examples of monitoring wildfires, aquatic waste spills, and methane seepage, we could consider the scalar functions of interest to be fire intensity, waste concentration and methane concentration.

An effective way to characterize a scalar function is through its moments. Moments represent certain properties of a function and are commonly used in fields such as statistics and mechanics. In statistics, moments are used to describe properties of a probability density function such as the mean, variance, skewness and kurtosis. In mechanics, moments are used

to describe properties of a mass distribution such as the center of mass and rotational inertia. Describing a function by its moments is both powerful and helpful as it allows a continuous function to be described by a representative, finite set of values.

1.2 Problem Formulation

The goal of this thesis is to determine positions of a group of sensing vehicles which lead to effective characterization and/or monitoring of a scalar function of space and time. The method we propose to accomplish this goal is to calculate moments of the function, calculate moments of a group of positions, then set a certain number of these moments equal to each other. Doing so results in a system of polynomial equations in which the vehicle positions are the independent variables. To solve this system of polynomials, techniques from the field of computational algebraic geometry are used.

1.3 Previous Work

Multiple vehicle coordination has become a major area of research. Groups of autonomous ground, aerial and underwater vehicles are now common tools for engineering research. The long list of applications which have been proposed for multi-vehicle systems includes mapping, pollution monitoring, structural inspection, surveillance, biological surveying and search and rescue [5, 24].

Several methods have been proposed for groups of vehicles to perform characterization or investigation of a scalar function. The properties of interest of a function are what guide the formulation of these methods. For example, Mathaler and Bertozzi used environmental boundary tracking, which is positioning vehicles on a level set of a scalar function [17]. Similarly, Kalantar and Zimmer developed a technique to position vehicles on a specific isocline of a scalar function [13]. Another technique developed by Ögren, Fiorelli and Leonard used gradient climbing to find local maxima and minima of a function [19].

The idea of positioning vehicles based on the values of their moments (for definitions of moments as they are referred to here, see Section 3.1.2) has been done by at least two

groups in the past. Belta and Kumar developed a method to map vehicle positions to a lower dimensional manifold [2]. They give an example of a manifold which is defined by parameters which are similar (but not identical) to first and second order moments. Additionally, Yang, Freeman and Lynch developed a decentralized control scheme for a group of vehicles such that the first and second order moments of the group converge to given values [23]. This formulation requires that in general, a group of positions which satisfy these moments is known beforehand.

Computational algebraic geometry is a tool which has been used to solve a wide range of problems in fields including cryptography, biology, and theoretical physics [3, 11, 18]. In the robotics and controls community, computational algebraic geometry has been a popular tool for solving problems involving systems with linkages or manipulators [20].

1.4 Contributions

This thesis presents a new method to determine the positions of sensing vehicles in the domain of a scalar function. This method can be applied to a wide variety of scalar functions and can incorporate different numbers of vehicles. This method also allows for other explicit geometric constraints to be imposed on vehicle positions using polynomial equations.

1.5 Thesis Organization

Chapter 2 provides some background for the field of algebraic geometry as well as the concept of moments in statistics. Chapter 3 describes the moment matching formulation, which is the process of determining the positions for a group of vehicles given a scalar function. Chapter 4 discusses several challenges to the formulation and then extends the formulation to address these challenges. Chapter 5 describes various polynomials with geometric interpretations which can be applied as alternatives to moment polynomials. Finally, Chapter 6 provides a thesis summary and describes future work.

Chapter 2
MATHEMATICAL BACKGROUND

The approach used in this thesis requires solving a system of multivariate polynomials. It will become evident in later chapters that solving this system is the mathematical backbone of this approach. For this reason, our approach is heavily tied to the field of algebraic geometry, which is the field of math concerned with solving these systems. Additionally, this thesis relies on moment matching, which is the idea of equating the moments of a function to the moments of a finite group of points. The intuition behind using moment matching is greatly enhanced by drawing an analogy to statistical moments. Since polynomial systems and moment matching are key elements of this thesis, this chapter provides a background in algebraic geometry and statistical moments.

2.1 Algebraic Geometry Background

A simple definition of algebraic geometry is that it is the study of the solutions of a system of polynomial equations. *Real* algebraic geometry is the study of only the real solutions to systems of polynomial equations. Most of the information in this section can be found in [8].

2.1.1 Polynomials, Ideals and Varieties

Define x as a vector with m variables, written as $x = [x_1 \;\; x_2 \;\; \cdots \;\; x_m]$.

Definition 2.1. Using vector notation, a **monomial** in $x_1 ..., x_m$ is a product

$$x^{\alpha} = x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_m^{\alpha_m} \tag{2.1}$$

where $\alpha = [\alpha_1 \;\; \alpha_2 \;\; \cdots \;\; \alpha_m] \in \mathbb{N}^m$.

Definition 2.2. The **degree** of a monomial is the sum of the exponents i.e. $\deg(x^\alpha) = \sum_{i=1}^m \alpha_i$.

It can be extremely helpful to order all possible monomials in x. This leads to the concept of a monomial ordering. There are many different monomial orderings which are discussed in the next paragraph. Ordering is denoted by applying the $<$ and $>$ operators to monomials. So, given a certain ordering, the notation $x^\alpha > x^\beta$ means that x^α is in increasing order of x^β. We can make this notation a bit simpler by using only the exponent vectors; thus, we can equivalently write $\alpha > \beta$. Some important monomial orderings are discussed below.

Definition 2.3. Lexicographic order orders monomials such that $x^\alpha > x^\beta$ if the leftmost nonzero entry of $\alpha - \beta$ is positive.

Definition 2.4. Graded lexicographic order orders monomials such that $x^\alpha > x^\beta$ if $\sum_{i=1}^m \alpha_i > \sum_{i=1}^m \beta_i$, or, if $\sum_{i=1}^n \alpha_i = \sum_{i=1}^n \beta_i$ and the leftmost nonzero entry of $\alpha - \beta$ is positive.

As an example, the graded lexicographic ordering of x in 2-d is

$$1 < x_1 < x_2 < x_1^2 < x_1x_2 < x_2^2 < x_1^3 < x_1^2x_2 < \cdots . \tag{2.2}$$

Definition 2.5. Graded reverse lexicographic order orders monomials such that $x^\alpha > x^\beta$ if $\sum_{i=1}^m \alpha_i > \sum_{i=1}^m \beta_i$, or, if $\sum_{i=1}^n \alpha_i = \sum_{i=1}^n \beta_i$ and the rightmost nonzero entry of $\alpha - \beta$ is positive.

Graded reverse lexicographic order is a somewhat unintuitive ordering but is well-suited for fast computation of Gröbner bases (see Section 2.1.3).

Definition 2.6. A **polynomial** is a finite linear combination of monomials with coefficients in the field K. A polynomial in the variable x, $h(x)$, can be written as

$$h(x) = \sum_\alpha a_\alpha x^\alpha, \quad a_\alpha \in K. \tag{2.3}$$

Let $K[x]$ denote the ring of polynomials in x with coefficients in the field K.

Definition 2.7. A field is **algebraically closed** if every nonconstant polynomial in $K[x]$ has a root in K.

As an example, the field of real numbers is not algebraically closed since not every polynomial with real coefficients has real roots. For this reason, the complex numbers are the algebraic closure of the real numbers.

Definition 2.8. The **ideal** generated by a set of polynomials $h_1, h_2, ..., h_n \in K[x]$, denoted as $\langle h_1, h_2, ..., h_n \rangle$, is the set

$$\langle h_1, ..., h_n \rangle = \left\{ \sum_{i=1}^{n} g_i h_i \quad | \quad g_1, ..., g_n \in K[x] \right\}. \tag{2.4}$$

Definition 2.9. The **affine variety**, or simply variety, defined by $h_1, h_2, ..., h_n$ is the set of all x which lie in K^m and drive all of the polynomials to zero i.e.

$$\mathcal{V}(h_1, ..., h_n) = \{x \in K^m \quad | \quad h_i(x) = 0 \quad \forall i = 1, ..., n\}. \tag{2.5}$$

Note that the variety is not the same as the roots of a system of polynomials since the roots do not necessarily have to lie in K^m.

Definition 2.10. A **zero-dimensional ideal** is an ideal whose affine variety is a finite set.

If an ideal is zero-dimensional, then the generators of that ideal have a finite number of solutions in K^m.

2.1.2 Methods of Root Solving for Polynomial Systems

There are several distinct techniques for finding the roots of a system of mulitvariate polynomials. These techniques can be categorized into groups which include Gröbner basis techniques, homotopy continuation methods, multipolynomial resultants, and Ritt-Wu's method. The techniques explored in this thesis - Gröbner basis techniques and homotopy coninuation methods - are briefly described below.

2.1.3 Gröbner bases

Many methods to compute the roots of a polynomial system involve first computing a Gröbner basis. We will not formally define Gröbner bases, but we will describe some of their features. A Gröbner basis is a particular set of polynomial equations. One can be found for any set of polynomials $h_1, ..., h_n \in K[x]$. Additionally, by setting a monomial ordering, a *reduced* Gröbner basis can be found which is unique to $h_1, ..., h_n$. Denote the Gröbner basis of $h_1, ..., h_n$ as $\mathcal{G}(h_1, ..., h_n)$. $\mathcal{G}(h_1, ..., h_n)$ generates the same ideal as $h_1, ..., h_n$ and specifies the same variety as the ideal $\langle h_1, ..., h_n \rangle$. A Gröbner basis has many properties which are more convenient than the original polynomial system including that it is more easily solvable. An interesting note is that finding a Gröbner basis generalizes the techniques of Gaussian elimination, the Euclidean algorithm and the simplex algorithm.

2.1.4 Homotopy Continuation Methods

Homotopy continuation methods involve gradually transforming a system whose solutions are known into a system of interest [7]. More precisely, if the system of interest is defined by polynomials $h_1(x) = \cdots = h_n(x) = 0$, then with a system of known solutions $g(x) = 0$, a homotopy or continuation system can be defined as

$$0 = c(1-t)g(x) + th(x), \quad t \in [0, 1]. \tag{2.6}$$

In this equation $c \in \mathbb{C}$ is constant which is chosen for good behavior. By varying t in (2.6) from 0 to 1, the known system $g(x) = 0$ can be transformed into the system $h(x) = 0$. Solving a homotopy system can be done numerically. One numerical method is to use a root finding algorithm such as the Newton-Raphson method. Another method is to note that since the left-hand-side of (2.6) equals zero, the derivative of the right-hand-side with respect to t should equal zero for all t. As a consequence, the homotopy system can be reformulated as an ODE problem.

In comparison with Gröbner basis techniques, which operate algebraically, homotopy continuation methods operate numerically and therefore cannot determine exact roots. However,

they can approximate roots with a high degree of accuracy.

2.2 Statistics Background

2.2.1 Moments

Define X to be a random variable and x to be a realization of X. Consider a probability density function $P(x)$. The k^{th} raw moment of $P(x)$ for $k \in \mathbb{N}$ is calculated by

$$\mu_k = E[X^k] = \int_{\mathbb{R}} x^k P(x) dx \tag{2.7}$$

where $E[X]$ is the expected value. This formula generalizes to the multivariate case where $X = [X_1 \quad X_2 \quad \cdots \quad X_m]$. Redefining k as $k = [k_1 \quad k_2 \quad \cdots \quad k_m] \in \mathbb{N}^m$, the raw moment formula becomes [15]

$$\mu_k = E[X_1^{k_1} X_2^{k_2} \cdots X_n^{k_n}] = \int_{\mathbb{R}^m} x_1^{k_1} x_2^{k_2} \cdots x_n^{k_n} P(x) dx. \tag{2.8}$$

2.2.2 The Method of Moments

Now consider a probability density function of x with h parameters. By expressing the parameters as the variable $\theta = [\theta_1 \quad \cdots \quad \theta_h]$, the probability density function can be defined as $P(x; \theta)$. For a density function with uncertain parameters, one technique used for parameter estimation is called the method of moments [4]. For a univariate X, if n samples $x_1, x_2, ..., x_n$ are drawn, then the sample moments are computed as

$$\hat{\mu}_k = \frac{1}{n} \sum_{i=1}^{n} x_i^k. \tag{2.9}$$

To determine θ, h raw moments are equated to the corresponding sample moments i.e.

$$\mu_k = \hat{\mu}_k, \quad \forall k = 1, ..., h. \tag{2.10}$$

This equation represents a system of h equations in the h unknowns of θ. θ can be determined by solving this system.

Chapter 3
MOMENT MATCHING FORMULATION

We will now present the idea of moment matching, which is how we will determine vehicle positions from a scalar function. In this chapter, the moment matching formulation is described mathematically and is illustrated through several examples.

3.1 Moments and Moment Matching

3.1.1 Function Moments

Let $f(x,t)$ be a scalar function of time t and spatial variable $x = [x_1 \ \cdots \ x_m] \in \mathbb{R}^m$. We are interested in functions which measure scalar quantities such as concentration and intensity. So, we consider these scalar quantities to be on a scale of zero and up. On this scale, if we are at a points where there is no quantity to measure, we will have $f(x,t) = 0$. Otherwise, we are at a point where there is some measurable quantity and $f(x,t) > 0$. Accordingly, we only consider functions $f(x,t)$ which map to the non-negative real numbers, i.e. $f : \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}_+$.

Define $\Omega \subseteq \mathbb{R}^m$ to be the region of space we are interested in. In the next few steps, we will require that $f(x,t)$ is integrable. However, this restriction can be ignored if we use the numerical approximation techniques described in Section 3.1.5. We introduce $\bar{f}(x,t)$ to be $f(x,t)$ normalized so it integrates over Ω to one, i.e.

$$\bar{f}(x,t) = \frac{f(x,t)}{\int_\Omega f(x,t)dx}. \tag{3.1}$$

Note that if $\int_\Omega f(x,t)dx = 0$ (which implies $f(x,t) = 0$ over all of Ω), then (3.1) will blow up. In this case, we can simply skip (3.1) and define $\bar{f}(x,t) = 0$ over Ω. Next, we use the monomial notation in (2.1) to define moments of $\bar{f}(x,t)$ as

$$\mu_\alpha = \int_\Omega x^\alpha \bar{f}(x,t)dx \tag{3.2}$$

where $\alpha \in \mathbb{N}^m$. The order of μ_α is defined to be the degree of the monomial x^α.

3.1.2 Vehicle Moments

Consider n vehicles in $\mathbb{R}^m$, each with position $p_i = [p_{i,1} \quad \cdots \quad p_{i,m}]$ for $i = 1, ..., n$. Denote the augmented vector of all vehicle positions as $p = [p_1 \quad \cdots \quad p_n]$. Now, moments of the vehicle configurations can be defined. For a fixed $\alpha \in \mathbb{N}^m$, a given moment is calculated by averaging all n monomials p_i^α which gives the equation

$$\nu_\alpha(p) = \frac{1}{n} \sum_{i=1}^{n} p_i^\alpha. \tag{3.3}$$

3.1.3 Moment Matching

The idea of "moment matching" can now be understood as setting $\mu_\alpha = \nu_\alpha(p)$ for a constant exponent vector α. This vector α uniquely defines a moment. To solve for p, we create a system of mn polynomials by matching mn moments. By representing each moment by its exponent vector α, we can define the set of all moments as $\mathcal{A}$, which is a set of mn vectors α. The moment matching formulation can now be described by the equation

$$\mu_\alpha = \nu_\alpha(p), \quad \forall \alpha \in \mathcal{A}. \tag{3.4}$$

This system of mn polynomials in the mn independent variables of p is assumed to generate a zero-dimensional ideal. If the ideal is not zero-dimensional, that would mean there are an infinite number of solutions (Section 4.1 provides more information). To select the exponent vectors in $\mathcal{A}$, we use the first mn vectors in the graded lexicographic order, not including the zero vector. We can express this choice of $\mathcal{A}$ mathematically as

$$\mathcal{A} = \{\alpha^{[1]}, \alpha^{[2]}, ..., \alpha^{[mn]}\} \tag{3.5}$$

where $\alpha^{[i]}$ denotes the i^{th} vector in the graded lexicographic ordering, not including the zero vector.

As an example of a moment matching system, consider the case of three vehicles in two dimensions. The six moment matching equations from (3.4) using $\mathcal{A}$ from (3.5) are:

$$\begin{aligned}
\frac{1}{3}(p_{1,1} + p_{2,1} + p_{3,1}) &= \mu_{1,0} \\
\frac{1}{3}(p_{1,2} + p_{2,2} + p_{3,2}) &= \mu_{0,1} \\
\frac{1}{3}(p_{1,1}^2 + p_{2,1}^2 + p_{3,1}^2) &= \mu_{2,0} \\
\frac{1}{3}(p_{1,1}p_{1,2} + p_{2,1}p_{2,2} + p_{3,1}p_{3,2}) &= \mu_{1,1} \\
\frac{1}{3}(p_{1,2}^2 + p_{2,2}^2 + p_{3,2}^2) &= \mu_{0,2} \\
\frac{1}{3}(p_{1,1}^3 + p_{2,1}^3 + p_{3,1}^3) &= \mu_{3,0}.
\end{aligned} \tag{3.6}$$

A set of vehicle positions can be determined by (3.4) if and only if there exist real solutions to the system. Otherwise, the solutions would have imaginary parts and therefore lose their physical interpretation. This idea is formalized in Theorem 3.1. Section 4.1 describes in further detail the possibility of having no real solutions to a system.

Theorem 3.1. At a given time t^*, n vehicle positions can be determined whose moments match those of a function $\bar{f}(x, t^*)$ for all moments in $\mathcal{A}$, if and only if there exists a real solution p to (3.4).

Proof. If vehicle positions can be determined whose moments match those of a scalar function $\bar{f}(x, t^*)$ for all moments in $\mathcal{A}$, then by definition, those positions will be a real solution to (3.4). If there exists a real solution p to (3.4), then p corresponds to a set of n positions in $\mathbb{R}^m$. Furthermore, since p satisfies (3.4), then the moments of p match those of the function. □

It will be shown in later chapters that with $\mathcal{A}$ defined by (3.5), the system in (3.4) may not have real solutions. In this case, we may want to consider the more general problem of determining if there exists some set of moments (out of the infinite number of possible sets) for which real positions can be found for the moment matching system. So we change our definition of $\mathcal{A}$ from the set in (3.5) to *some* set of moments. Theorem 3.2 formalizes this

idea. This concept of using a set of moments other than the set in (3.5) is further discussed in Section 4.3.

Theorem 3.2. At a given time t^*, n vehicle positions can be determined for which mn unique moments match those of a scalar function $\bar{f}(x, t^*)$ if and only if one can find a set of mn unique moments $\mathcal{A}$ which result in (3.4) having a real solution p.

Proof. By Theorem 3.1, for a certain set of moments $\mathcal{A}$, real vehicle positions can be determined whose moments match those of $\bar{f}(x, t^*)$ if and only if a real solution to (3.4) exists. Since any set of mn unique moments can be expressed as a set $\mathcal{A}$, then vehicle positions can be determined for which mn unique moments match those of $\bar{f}(x, t^*)$ if and only if (3.4) has a real solution for a unique set represented by $\mathcal{A}$. □

3.1.4 Analogy to the Method of Moments

The moment matching formulation has a similar mathematical form as the statistical method of moments. This relationship can be seen by considering a generalization of the univariate case described in Section 2.2.2 to the multivariate case. In the multivariate case, the function $\bar{f}(x, t)$ at a given time t is analogous to a probability density function. And the position of each vehicle is analogous to a sample. The difference between the statistical method of moments and multi-vehicle moment matching is that in the statistical case, the unknowns are the parameters which are incorporated into the function integral side of the equations. However, in the multi-vehicle moment matching case, the unknowns are the vehicle positions and are incorporated into the other side of the equations. Figure 3.1 illustrates this idea.

3.1.5 Integral Approximations

Solving for μ_α relies on an integration to be performed over the region Ω. While some scalar functions can be integrated exactly, all simulations in this thesis will use numerical approximations. The approximations use a variation of the midpoint rule. First, Ω is discretized into rectangular regions of equal size. Let $\Delta = [\Delta_1 \quad \cdots \quad \Delta_m]$ be the vector of side lengths

Figure 3.1: An illustration of the method of moments and multi-vehicle moment matching formulation with a 2-d function. The statistical method of moments (left) determines parameters of a function based on samples. The multi-vehicle moment matching idea in this thesis (right) determines vehicle positions based on a function.

in each dimension. Each of the rectangular regions will have a midpoint where the function will be evaluated. Denote x_m and R_m as a midpoint and the discrete region occupied by that midpoint, respectively. Denote the collection of midpoints as M. Consider a normalized function at an instant in time, $\bar{f}(x)$. For sufficiently small discrete regions, the integral over Ω approximates to

$$\mu_\alpha = \int_\Omega x^\alpha \bar{f}(x)dx \approx \sum_{x_m \in M} \int_\Omega x^\alpha \bar{f}(x_m)dx. \tag{3.7}$$

As an example, in the 2-d case, each rectangular region will have side lengths $\Delta = [\Delta_1 \quad \Delta_2]$, and each midpoint will have the form $x_m = [x_{m,1} \quad x_{m,2}]$. For a moment $\alpha = [\alpha_1 \quad \alpha_2]$, this approximation becomes

$$\begin{aligned} \mu_\alpha &\approx \sum_{x_m \in M} \int_{x_{m,1}-\Delta_1/2}^{x_{m,1}+\Delta_1/2} \int_{x_{m,2}-\Delta_2/2}^{x_{m,2}+\Delta_2/2} x_1^{\alpha_1} x_2^{\alpha_2} \bar{f}(x_m) dx_2 dx_1 \\ &= \frac{1}{(\alpha_1+1)(\alpha_2+1)} \sum_{x_m \in M} \bar{f}(x_m) \left[x_1^{\alpha_1+1}\right]_{x_{m,1}-\Delta_1/2}^{x_{m,1}+\Delta_1/2} \left[x_2^{\alpha_2+1}\right]_{x_{m,2}-\Delta_2/2}^{x_{m,2}+\Delta_2/2}. \end{aligned} \tag{3.8}$$

This method was tested on some of the simulation functions, Ω regions and discretization grids described in later sections. The function moments computed using this approximation were then compared to moments computed using numerical integrations (using the integral2 command in MATLAB). All tested functions resulted in percent errors of less than 6% for each of the first twenty moments.

3.2 Polynomial Solver Benchmarking

Many of the methods for polynomial root solving in Section 2.1.2 are implemented computationally. Since we are just interested in finding the roots of the system, the goal of this section is to determine the fastest possible root-solving method. We compared what are perhaps the two most popular methods for root-solving: Gröbner basis techniques and polynomial homotopy continuation. We compared two software packages for each of these methods.

3.2.1 Software Packages

For computing Gröbner bases, we used the software packages SINGULAR [9] and FGb [10]. SINGULAR is free and open-source computer algebra system developed by researchers at the University of Kaiserslautern in Germany. It is a standalone program. FGb is library for computing Gröbner bases developed by Jean-Charles Faugère at INRIA (French Institute for Research in Computer Science and Automation). It is implemented in C/C++ and in Maple. Tests in this thesis used the Maple implementation. FGb's methods are considered state-of-the-art for Gröbner basis computations. In both software packages, polynomials were ordered using graded reverse lexicographic ordering. Gröbner bases were computed in SINGULAR using the "groebner" command which uses a "heuristically chosen method". Gröbner bases were computed in FGb/Maple using the "Basis" command and the "fgb" option.

For polynomial homotopy continuation, we used the software packages PHCpack [22] and Bertini [1]. PHCpack is a free and open-source homotopy continuation solver for polynomial systems. It was developed by Jan Verschelde at the University of Illinois at Chicago. Bertini is another free and open-source homotopy continuation solver for polynomial systems. It was developed by researchers at the University of Notre Dame and Colorado State University. Computations in PHCpack used the "black-box" option and those in Bertini used default options.

Note that while homotopy methods are designed to solve for the roots of a system, finding a Gröbner basis is not the same as solving for the roots. It is more of a first step. After computing a Gröbner basis, the roots would have to be solved for using a technique such as triangular decomposition. This implies that root solving using a Gröbner basis technique would take longer than the computation of the Gröbner basis itself. However, we did not take this into account since we determined homotopy methods were faster than the Gröbner basis computations anyway.

3.2.2 Benchmarking Test

The function used for the benchmarking simulations is the 2-d scalar function

$$f(x,y) = \exp\left[-a\left(r_c(x,y) - r_{in} - b\sin\left[k\theta_c(x,y) + d\right]\right)^2\right] \tag{3.9}$$

where

$$r_c(x,y) = \sqrt{(x-x_c)^2 + (y-y_c)^2} \tag{3.10}$$

and

$$\theta_c(x,y) = \text{atan2}\left(y - y_c, x - x_c\right). \tag{3.11}$$

The atan2 function is defined to be the four-quadrant arctangent function. Note that in order to make Equations 3.9, 3.10 and 3.11 more intuitive, we are using $[x \;\; y] \in \mathbb{R}^2$ as opposed to the $x \in \mathbb{R}^m$ which we have been using to denote spatial positions. $f(x,y)$ creates a "rose" curve centered at (x_c, y_c). The functions $r_c(x,y)$ and $\theta_c(x,y)$ shift the center of the rose to (x_c, y_c). Note that this function is independent of time so the variable t used in Section 3.1.1 can be ignored for this example.

The parameters for the rose function were chosen as $x_c = 7$, $y_c = 11$, $a = .3$, $b = 3$, $d = .5$, $k = 3$ and $r_{in} = 2$. The region Ω was selected to be a $[-20,\ 20] \times [-20,\ 20]$ grid consisting of 800×800 elements. These parameters create the function shown in Figure 3.2 (the plot window is smaller than Ω to better show the function).

Function moments were calculated using the integration approximation in (3.8). Using the graded lexicographic order, the first 10 moments were calculated. The results are shown

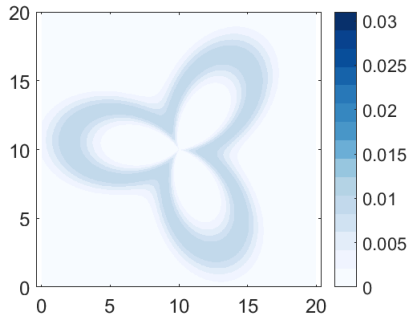

Figure 3.2: The function used for benchmarking polynomial root solvers.

in Table 3.1. Using these moments, the polynomial systems were solved using the moment

$\mu_{1,0}$	$\mu_{0,1}$	$\mu_{2,0}$	$\mu_{1,1}$	$\mu_{0,2}$	$\mu_{3,0}$	$\mu_{2,1}$	$\mu_{1,2}$	$\mu_{0,3}$	$\mu_{4,0}$
7.00	11.00	59.08	77.00	131.08	564.66	668.21	907.52	1645.22	5868.82

Table 3.1: Moments of the benchmarking function (rounded to two decimal places).

matching formulation. All simulations were run on a Linux workstation with a 3.5GHz AMD FX-6300 six-core processor and 32GB of RAM. Results are shown in Table 3.2. The results

vehicles (polynomials)	SINGULAR	FGb/Maple	Bertini	PHCpack
2 (4)	1s	1s	3s	1s
3 (6)	1s	1s	3m 13s	1s
4 (8)	1s	1s	>12h	4s
5 (10)	>12h	>12h	-	1m 49s
6 (12)	-	-	-	2h 13m 34s
7 (14)	-	-	-	>12h

Table 3.2: Time to compute the reduced Gröbner basis or solve for roots of the example system using different computational algebraic geometry software packages.

show that computing a Gröbner basis hits a wall when the system reaches 10 polynomials. Additionally, during their attempts to compute the 10 polynomial system, both SINGULAR and FGb were using very large amounts of memory. Regarding the homotopy solvers, there was large difference in computing time between Bertini and PHCpack. PHCpack was faster and could handle larger systesms. It is clear that among both Gröbner basis techniques and homotopy solvers, PHCpack is the overall fastest method for solving these types of polynomial systems. As a result, all of the simulations in this thesis were done using PHCpack unless otherwise noted. Simulations were done using MATLAB and PHClab [12], which interfaces MATLAB with PHCpack.

As a note, the speed of calculations using PHCpack can be further increased by using a multiple thread option. By doing this, the times in the last column of Table 3.2 were decreased by a factor of about three. However, multiple thread options were not included in Table 3.2 to keep all tests consistent.

3.3 Simulation: Various Functions with Different Numbers of Vehicles

Simulations were performed on various functions to test the moment matching formulation. The first set of simulations were done in 2-d and compared three different functions. These functions are: the rose function from (3.9), a distribution which resembles the letter A and a function which is the sum of 2-d Gaussian distributions. The rose had parameters of $x_c = 10$, $y_c = 10$, $a = .7$, $b = 2$, $d = 5$, $k = 4$ and $r_{in} = 6$. The letter A was manually defined to be uniform everywhere inside the shape and zero everywhere else. The 2-d sum of Gaussians distribution was the function

$$f(x,y) = \sum_{i=1}^{n_G} \exp\left[-\left(\frac{(x-x_{0,i})^2}{2\sigma_{x,i}^2} + \frac{(y-y_{0,i})^2}{2\sigma_{y,i}^2}\right)\right]. \tag{3.12}$$

In this equation n_G is the number of Gaussian functions in the summation. Additionally, $x_{0,i}$, $y_{0,i}$, $\sigma_{x,i}$, and $\sigma_{y,i}$ for all $i = 1, ..., n_G$ are the respective center in the x and y directions and standard deviations in the x and y directions for each of the Gaussians. The example in this section used three Gaussians with respective parameters $x_0 = 5, 2, 15$, $y_0 = 4, 10, 7$,

$\sigma_x = 2, .5, 1$ and $\sigma_y = 1, .5, 3$. For all functions, the region Ω was selected to be a $[-20,\ 20] \times [-20,\ 20]$ grid consisting of 800×800 elements. Results are shown in Figures 3.3, 3.4 and 3.5.

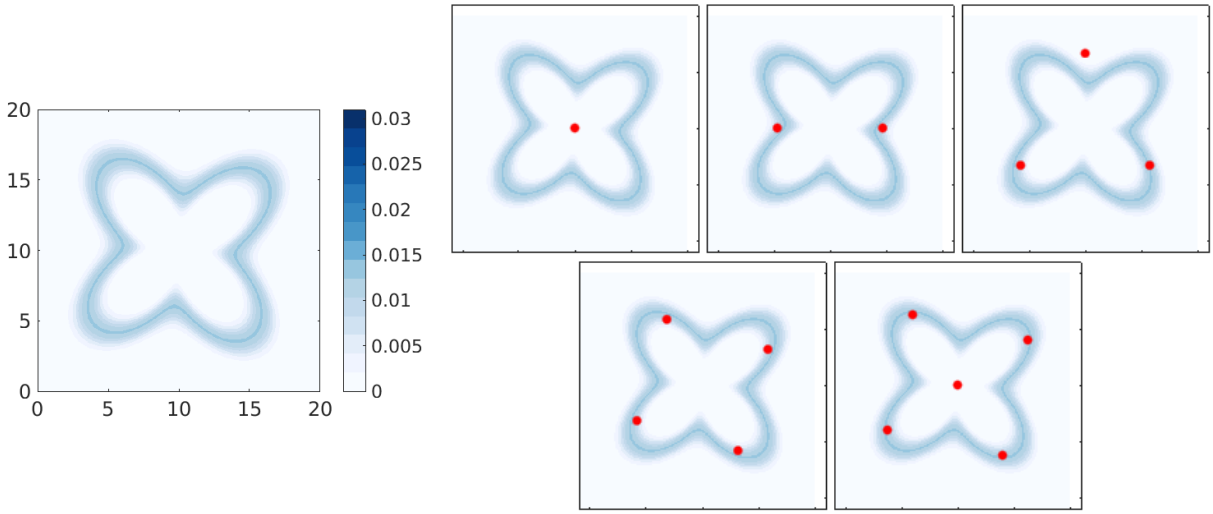

Figure 3.3: A ring function and vehicle positioning for one through five vehicles.

Some features of the results are as follows. For the rose function, four and five vehicles seems to capture the shape and symmetry of the function with one point in the center and one on each "petal" of the flower. Interestingly, each point is slightly offset from the center of the petal. For the letter A function, the shape is better represented by an increasing number of vehicles. Finally, it is hard to draw any conclusions from the sum of Gaussians function for any number of vehicles. Additionally, many of the vehicles are at points where the value of $f(x)$ is near zero.

3.4 Simulation: Tracking a Function Over Time

It has been shown that vehicle positions can be determined for different functions. An application of this idea is to "track" a time-varying function by determining positions at discrete timesteps. For example, consider the rose function from (3.9) modified so it varies

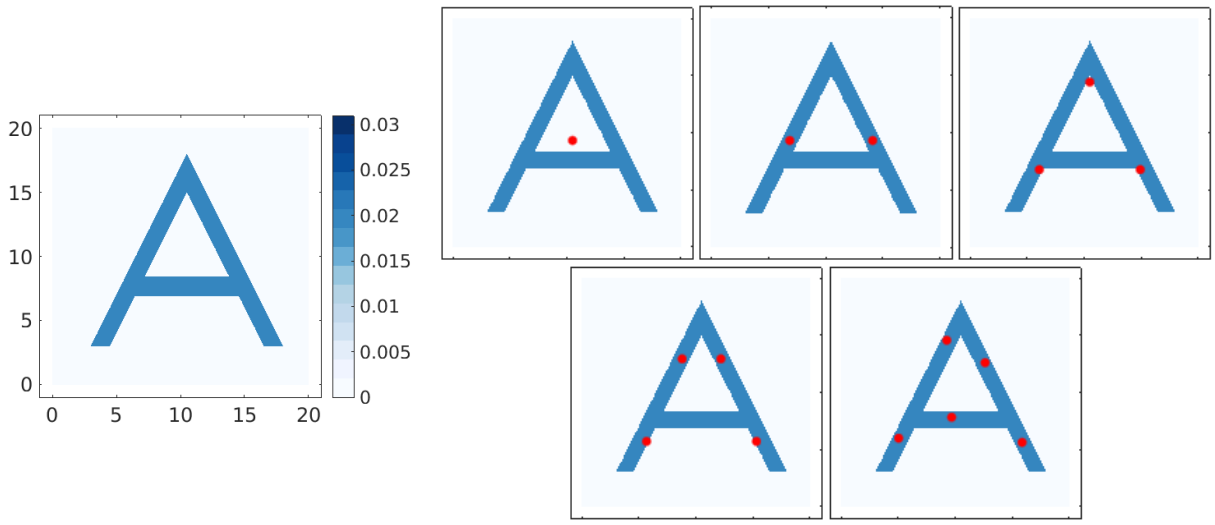

Figure 3.4: The letter A function and vehicle positioning for one through five vehicles.

with time. Define this function as

$$f(x,y,t) = \exp\left[-a\left(r_c(x,y,t) - r_{in} - ct - b\sin\left[k\theta_c(x,y,t) + \omega t\right]\right)^2\right] \tag{3.13}$$

where

$$r_c(x,y,t) = \sqrt{(x - x_0 - v_x t)^2 + (y - v_0 - v_y t)^2} \tag{3.14}$$

and

$$\theta_c(x,y,t) = \operatorname{atan2}\left(y - y_0 - v_y t, x - x_0 - v_x t\right). \tag{3.15}$$

This equation describes a rose that translates with velocities (v_x, v_y) and rotates at a frequency ω. Additionally, the constant c creates a time-dependent effect on the shape of the function.

A simulation was performed with parameters $x_0 = -10$, $y_0 = -10$, $v_x = 1.2$, $v_y = 1.6$, $a = .3$, $b = 3$, $c = .4$, $\omega = .3$, $k = 4$ and $r_{in} = 4$. The region Ω was considered to be a $[-30,\ 30] \times [-30,\ 30]$ grid discretized into 1200×1200 elements. The function was evaluated at times $t = 0, 2, 4, 6, 8, 10$ and five vehicles were used. Figure 3.6 shows the results. For all timesteps the vehicle positions are similar; there is one vehicle at the center and one on each lobe of the function.

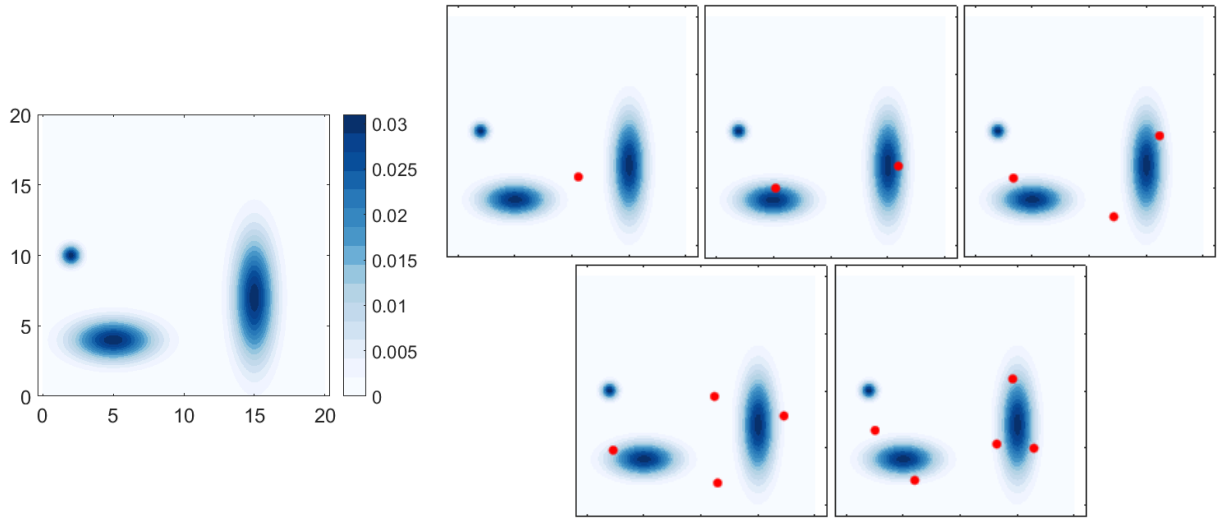

Figure 3.5: A sum of Gaussians function and vehicle positioning for one through five vehicles.

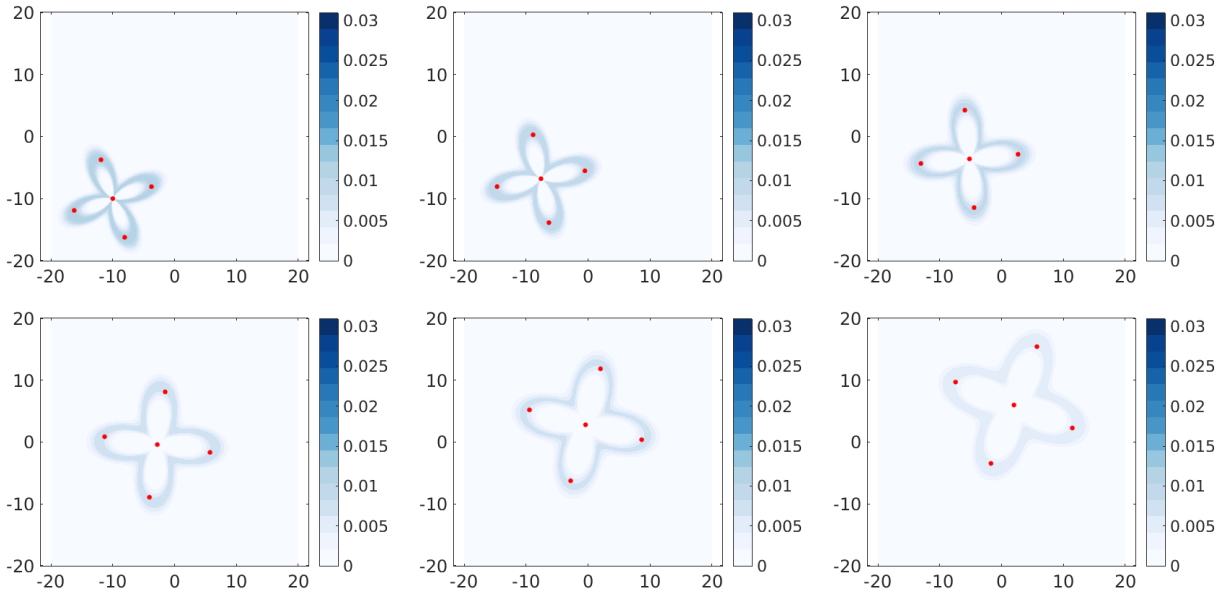

Figure 3.6: Position determination for five vehicles and a dynamic rose function. Top row: (l to r) $t = 0$, $t = 2$, $t = 4$. Bottom row: (l to r) $t = 6$, $t = 8$, $t = 10$.

Chapter 4
EXTENDING THE MOMENT MATCHING FORMULATION

The moment matching formulation creates a system of polynomials which must be solved. In some cases, the nature of the polynomial system presents obstacles to finding real roots. In this chapter, we present some of these obstacles and then describe how they can be overcome by extending the moment matching formulation.

4.1 Potential Root Finding Challenges

4.1.1 Positive-Dimensional Ideals

If real roots to a system of polynomials exist, there is a possibility that there are an infinite number of real roots. In algebraic geometry terms, this means the ideal generated by the polynomial system is positive-dimensional. In this case, more constraints must be put on the system in order to find a solution. One example of when this would occur is if the system erroneously included two identical polynomials.

4.1.2 Redundant Roots (Non-Unique Configurations)

If the ideal generated by the polynomial system is zero-dimensional, it will have no real roots, one real root, or some finite number of real roots. In fact, for $n > 1$, a moment matching polynomial systems with one real root will most likely have multiple real roots. This can be shown by considering the form of (3.3). If any position vectors p_i and p_j for $i, j \in \{1, ..., n\}$, $i \neq j$ are permuted, the value of $\nu_\alpha(p)$ will remain the same. This is true regardless of α. Using this logic, it follows that if a polynomial system defined by the moment matching formulation has a real root, it will have up to $n!$ roots which correspond to permutations of that root. The number of permuted roots will depend on how many positions in the

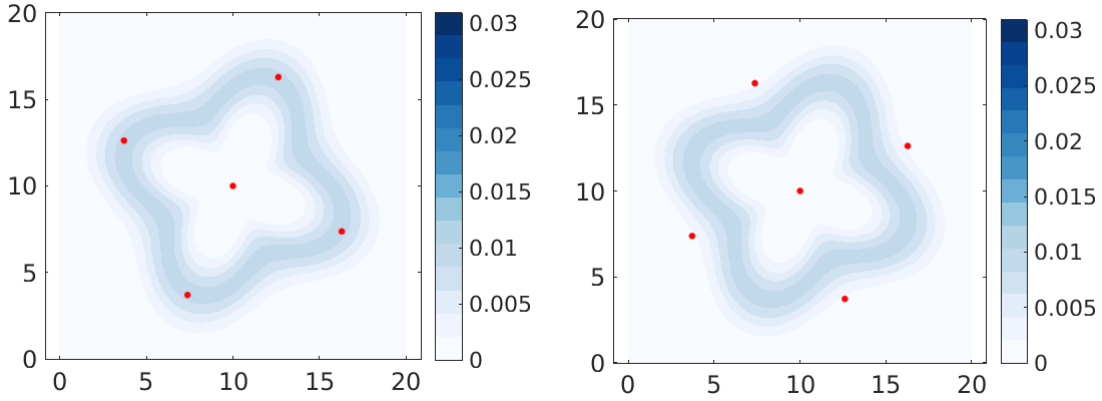

Figure 4.1: The two unique configurations of five vehicles in the domain of a ring function.

configuration are identical. Fortunately, in implementation these permutation solutions don't pose a big obstacle. After computing the roots using a polynomial solver, permutations can be removed using a simple sorting function.

4.1.3 Unique Configurations

Another situation is when, after reducing the set of solutions to a permutation-invariant set, there are still multiple solutions. For example, consider the rose equation from (3.9) with parameters $x_c = 10$, $y_c = 10$, $a = .3$, $b = 3$, $d = 5$, $k = 4$ and $r_{in} = 4$. There are two real, permutation-invariant solutions which are plotted in Figure 4.1. The configuration on the left includes one vehicle at the center and one vehicle at each of the four lobes of the function. The configuration on the right shows one vehicle at the center and the others at points where the function is close to zero. Regarding the application we are considering, placing sensors to characterize a dynamic function, the configuration on the right may be unfavorable since all sensors are measuring close to zero. However, the moment matching formulation does not distinguish between the two configurations.

4.1.4 No Real Roots

As noted in Section 2.1, the real numbers are not an algebraically closed field. For some functions, the moment matching formulation will result in polynomial systems with no real roots. This is a significant obstacle to implementing the moment matching idea. As an example, consider the sum of Gaussians function in (3.12) with only one Gaussian. Define the parameters of this function to be $x_0 = -5$, $y_0 = -7$, $\sigma_x = 1$, $\sigma_y = 3$ and Ω to be a $[-20,\ 20] \times [-20,\ 20]$ grid consisting of 800×800 elements. This distribution is shown in Figure 4.2. In this distribution, moment matching finds real positions for configurations with

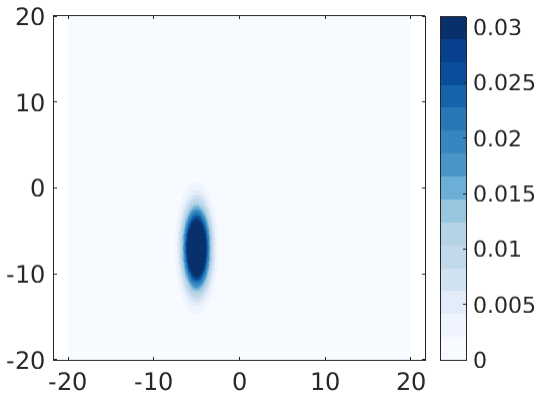

Figure 4.2: A distribution which leads to real positions for one, two or three vehicles but no real positions for four or five vehicles.

one, two or three vehicles. However for four or five vehicles, there are no real positions. A proposed method to overcome this obstacle is described in Section 4.3.

4.2 Configuration Sorting

As noted in Section 4.1.3, there may be multiple unique configurations which satisfy the polynomial system. We propose the following simple method to choose a configuration. Since the application for vehicle-positioning is sensing, we will pick the configuration which has largest summation of $\bar{f}(x,t)$ over all of the positions. To express this mathematically,

first denote the set of all configurations p which satisfy the moment matching polynomial system in (3.4) as $\mathcal{P}$, that is

$$\mathcal{P} = \{p \in \mathbb{R}^{mn} \mid \mu_\alpha = \nu_\alpha(p), \quad \forall \alpha \in \mathcal{A}\}. \tag{4.1}$$

Then, denoting an optimal configuration(s) as p^* we have

$$p^* = \arg\max_{p \in \mathcal{P}} \sum_{i=1}^{n} \bar{f}(p_i, t). \tag{4.2}$$

Note that in the equation above, as described in Section 4.1.2, there will be multiple values of p^* which correspond to the $n!$ vehicle permutations of any vector p. However this doesn't pose any real mathematical or practical problem.

As an example of this method, we revisit the configuration in Figure 4.1, which is described in Section 4.1.3. This method selects the configuration on the left, which is a more reasonable configuration for sensor placement.

4.3 Different Orderings of Moments

The formulation in this thesis has created a system of mn polynomials in mn variables by selecting the first mn moments ordered by the graded lexicographic ordering. Unfortunately, as shown in Section 4.1.4, there may not be real roots of this system. We now note that there are an infinite number of moments of a function. So while the graded lexicographic order is an intuitive method to select mn moments, any number of mn moments will provide mn polynomial constraints. So, in order to get around the issue of a distribution with no real roots, we will select mn moments in another way.

A simple alternative method of selecting moments is to continue using the graded lexicographic ordering but neglect the moment of index mn and replace it with a moment further in the ordering. More precisely, the set of moments $\mathcal{A}$ from (3.5) becomes

$$\mathcal{A} = \{\alpha^{[1]}, \alpha^{[2]}, ..., \alpha^{[mn-1]}, \alpha^{[mn+i]}\}, \quad i \in \{1, 2, ...\} \tag{4.3}$$

where $\alpha^{[i]}$ denotes the i^{th} vector in the graded lexicographic ordering, not including the zero vector. This can be an effective tool for overcoming the issue of no real roots.

As an example, we reconsider the example function from Section 4.1.4 which was shown in Figure 4.2. Real positions for the $n = 4$ and $n = 5$ case can now be solved by considering different moments. Real solutions were found when the index of the last moment was switched from index mn to $mn+3$ in the four vehicle case and to $mn+1$ in the five vehicle case. The results are shown in Figure 4.3.

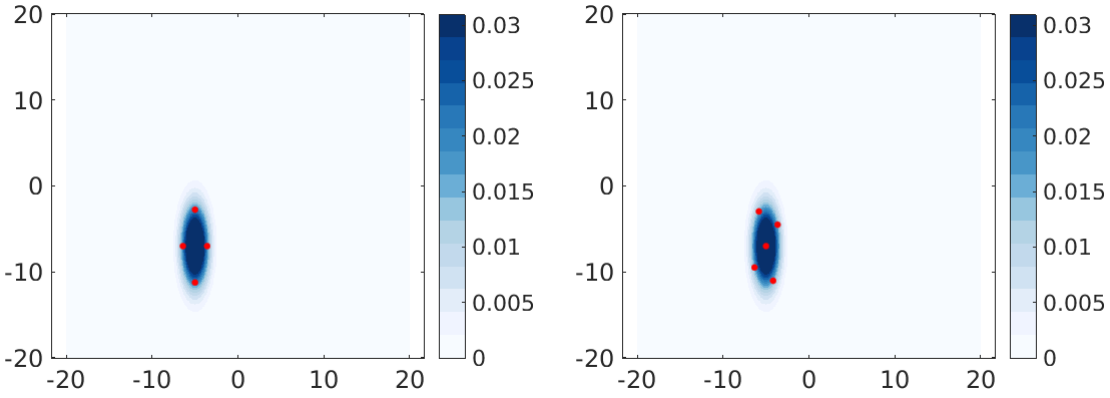

Figure 4.3: The function from Section 4.1.4 which had no real roots for $n = 4$ and $n = 5$ vehicles. Real roots can be determined when a different set of moments are used for $n = 4$ (left) and $n = 5$ (right).

4.4 Domain Subdivision

As shown in Section 3.2, the moment matching formulation is limited computationally and can only calculate the positions of up to five vehicles within a reasonable amount of time. Ideally, we would like this formulation to incorporate much larger numbers of vehicles. Also, as illustrated by the sum of Gaussians in Fig. 3.5, it is apparent that five vehicles may not provide enough detail to characterize a complicated function.

4.4.1 One Dimensional Domain Subdivision

The following method is proposed to implement larger numbers of vehicles. The idea is to divide up Ω into regions and place vehicles in each of the regions. If the same number

of vehicles are to be placed in each region, then it makes sense to ensure that each region contains the same "amount" of the function $\bar{f}(x,t)$. So, we divide Ω up so that $\bar{f}$ integrates to the same value in each region. Mathematically speaking, if Ω is to be divided into κ regions $R_1, R_2, ..., R_\kappa$, then

$$\Omega = \bigcup_{i=1}^{\kappa} R_i \tag{4.4}$$

and

$$\int_{R_i} \bar{f}(x)dx = \frac{1}{\kappa}\int_{\Omega} \bar{f}(x)dx = \frac{1}{\kappa}, \quad \forall i = 1, 2, ..., \kappa. \tag{4.5}$$

There are many different possible ways to divide up Ω. A simple method is to divide into adjacent rectangular regions. One dimension must first be chosen along which divide. This will determine $n_\kappa - 1$ points along that dimension which are the locations where divisions occur. Figure 4.4 shows an example of a Gaussian divided into five regions along the horizontal and vertical axes.

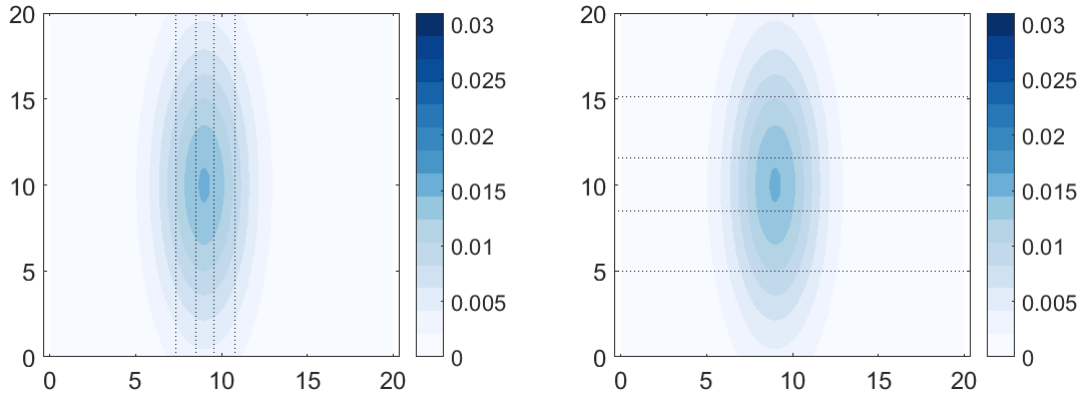

Figure 4.4: Subdivision of a Gaussian into five regions, divided along the horizontal axis (left) and vertical axis (right).

Using the method above, it will be possible to divide Ω into an arbitrary number of regions. However, for increasing values of κ, the regions will become very "skinny" i.e. one dimension is much smaller than the other(s). This may be undesirable if we want to keep all of the vehicles in a region close or if we want easier visualization of the regions.

4.4.2 Multi-Dimensional Domain Subdivision

In order to make the regions more square or cubic, we can apply the subdivision technique consecutively to each of the m dimensions. To do this, we first divide Ω along one dimension into κ_1 regions $R_1, R_2, ..., R_{\kappa_1}$. Then, we subdivide each of these regions along the next dimension into κ_2 regions. If $m = 3$, we have the option of performing a final subdivision along the last dimension. This method insures that $\bar{f}(x,t)$ will integrate the same over each region.

In the 2-d example, each region R_i for $i = 1, ..., \kappa_1$ will be divided into κ_2 sub-regions, which we will denote as $R_{i,1}, R_{i,2}, ..., R_{i,\kappa_2}$ for $i = 1, ..., \kappa_1$. A total of $\kappa_1\kappa_2$ sub-regions will be created. So, for multi-dimensional subdivision in 2-d, we have the following equations:

$$\Omega = \bigcup_{i=1}^{\kappa_1} \left(\bigcup_{j=1}^{\kappa_2} R_{i,j} \right) \tag{4.6}$$

and

$$\int_{R_{i,j}} \bar{f}(x)dx = \frac{1}{\kappa_1\kappa_2} \int_{\Omega} \bar{f}(x)dx = \frac{1}{\kappa_1\kappa_2}, \quad \forall i = 1, ..., \kappa_1, \quad \forall j = 1, ..., \kappa_2. \tag{4.7}$$

Figure 4.5 shows an example of a function being subdivided into twenty regions by dividing only along one axis as well as doing a multi-dimensional subdivision. While it is nearly impossible to distinguish each region in the left plot, the regions in the right plot are distinguishable and do not have as dramatic aspect ratios.

Once the domain is subdivided, vehicles can be placed in each of the regions. To do this, each region is considered a separate entity. The same methodology in Section 3.1 is used to place vehicles. This is done by replacing Ω in Equations 3.1 and 3.2 with the corresponding region as the domain of integration. Let n_R denote the number of vehicles placed in each region. For the example of a multidimensional subdivision in a two dimensional space, there will be a total of $\kappa_1\kappa_2 n_R$ vehicles placed in the entire domain Ω.

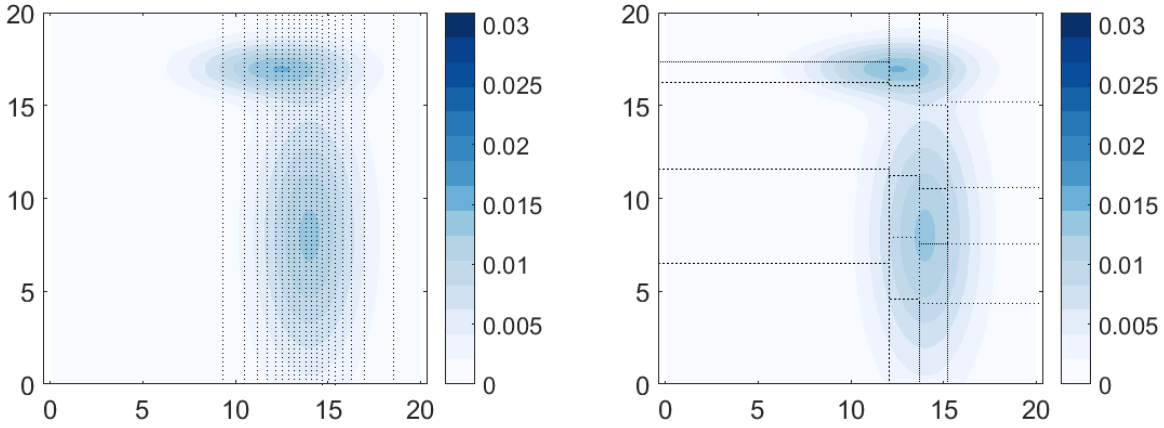

Figure 4.5: Subdivision of a sum of Gaussians function into twenty regions by dividing along the horizontal axis (left) and by dividing along the horizontal axis then vertical axis with $\kappa_1 = 4$ and $\kappa_2 = 5$ (right).

4.4.3 Simulations

We now return to the ring, letter A and sum of Gaussians functions which were shown with one through five vehicles in Section 3.3. The multi-dimensional subdivision technique can implement larger numbers of vehicles to characterize these functions. Figures 4.6, 4.7, and 4.8 show different numbers of vehicles and different subdivision schemes for each function.

To find real roots for some of these examples, it sometimes required using the method in

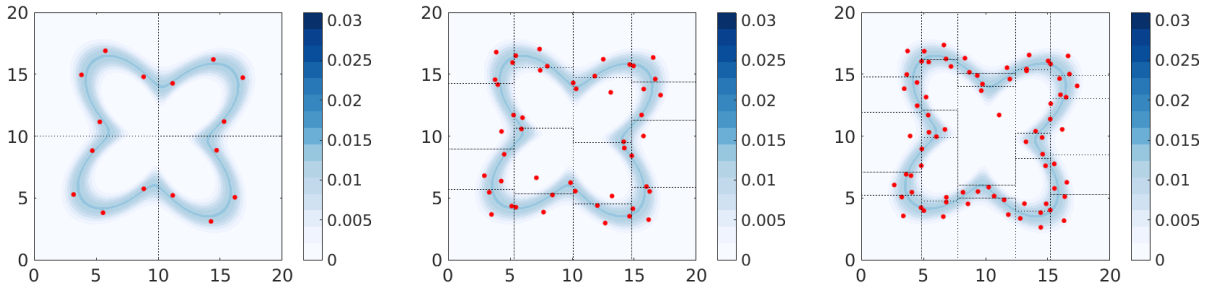

Figure 4.6: The ring function from Section 3.3 with vehicles placed using the subdivision method. Left: $n_R = 4$, $\kappa_1 = 2$, $\kappa_2 = 2$. Center: $n_R = 3$, $\kappa_1 = 4$, $\kappa_2 = 4$. Right: $n_R = 3$, $\kappa_1 = 5$, $\kappa_2 = 5$.

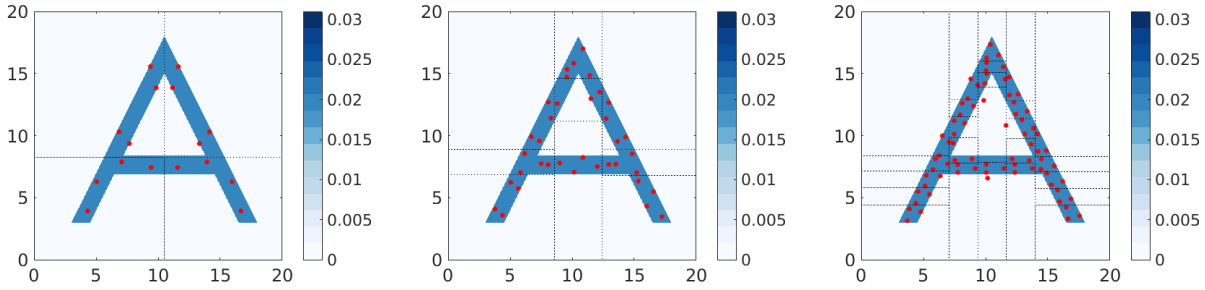

Figure 4.7: The letter A function from Section 3.3 with vehicles placed using the subdivision method. Left: $n_R = 4$, $\kappa_1 = 2$, $\kappa_2 = 2$. Center: $n_R = 4$, $\kappa_1 = 3$, $\kappa_2 = 3$. Right: $n_R = 3$, $\kappa_1 = 5$, $\kappa_2 = 5$.

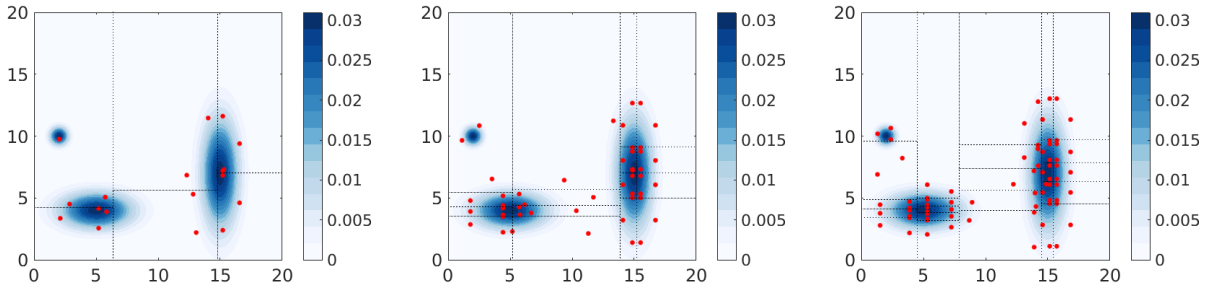

Figure 4.8: The sum of Gaussians function from Section 3.3 with vehicles placed using the subdivision method. Left: $n_R = 3$, $\kappa_1 = 3$, $\kappa_2 = 2$. Center: $n_R = 3$, $\kappa_1 = 4$, $\kappa_2 = 4$. Right: $n_R = 3$, $\kappa_1 = 5$, $\kappa_2 = 5$.

Section 4.3 to find different sets of moments. Also, other tests showed that attempting to use four or five vehicles per subdivision was less successful in determining real roots than using one, two or three. In general, the results show that the vehicle positions characterize the distribution well. One noticeable feature of the sum of Gaussians results is that the symmetry of the function causes a sort of alignment to happen of some positions in the vertical and horizontal directions. This is especially noticeable in the center and right plots in Figure 4.8.

Chapter 5
GEOMETRIC POLYNOMIAL CONSTRAINTS

The approach we have developed so far involves solving a system of polynomials in which the variables represent vehicle positions. To solve this system, we use methods from computational algebraic geometry. The motivation behind this chapter is to incorporate different kinds of polynomials into the system which represent geometric constraints on vehicle positions. Since these resulting systems are still polynomial, they can be solved using the same computational approach we have been using.

5.1 Bounding Vehicles by Second-Order Squares Moments

So far, moment equations have been used to relate vehicle positions to a function of interest. In this section, another interpretation of moment equations will be demonstrated. It will be shown that second-order squares moments have a useful geometric interpretation. Restricting these moments allows a group of vehicles to be bound a certain distance from the average position of the group. This bounding could be useful for collision avoidance, communication, or other purposes.

Consider the second-order squares moment equations. In m dimensions there will be m of these equations, one corresponding to each dimension. These equations are of the form $\frac{1}{n}(p_{1,i}^2 + p_{2,i}^2 + \cdots + p_{n,i}^2)$ where i is the index of the dimension being considered.

Lemma 5.1. Consider $n \geq 2$ vehicles in m dimensions. Denote the coordinates of the vehicles in the i^{th} dimension as $q_1, q_2, ..., q_n$ (i.e. $q_j = p_{j,i} \ \ \forall j = 1, ..., n$). Denote the average position in the i^{th} dimension as $\bar{q}$. Consider all possible configurations of vehicles with a given $\bar{q}$ and at least one element u units from $\bar{q}$ (i.e. $|q_j - \bar{q}| = u$ for some $j \in \{1, ..., n\}$). The minimum second-order moment in the i^{th} dimension for all of these configurations is

$\bar{q}^2 + \frac{u^2}{n-1}$.

Proof. The vehicles' average position in the i^{th} dimension is

$$\bar{q} = \frac{1}{n}\left(q_1 + q_2 + ... + q_n\right) \tag{5.1}$$

and their second-order moment in the i^{th} dimension is

$$\frac{1}{n}\left(q_1^2 + q_2^2 + ... + q_n^2\right). \tag{5.2}$$

Without loss of generality designate the element u units from $\bar{q}$ as q_n. There are two possibilities, either $q_n = \bar{q} + u$ or $q_n = \bar{q} - u$. First consider the case where

$$q_n = \bar{q} + u. \tag{5.3}$$

Solving for q_{n-1} in the average position equation yields

$$q_{n-1} = -q_1 - ... - q_{n-2} + (n-1)\bar{q} - u. \tag{5.4}$$

We can use this equation to eliminate q_{n-1} from the second-order squares moment equation. We define ν_2 as the second-order squares moment of a configuration with average position $\bar{q}$ and one position u units from $\bar{q}$. Defining $q = [q_1 \quad q_2 \quad \cdots \quad q_{n-2}]^T$ we have

$$\nu_2(q; \bar{q}, u) = \frac{1}{n}\left[q_1^2 + ... + q_{n-2}^2 + \left(-q_1 - ... - q_{n-2} + (n-1)\,\bar{q} - u\right)^2 + \left(\bar{q} + u\right)^2\right] \tag{5.5}$$

which is a function of variable q and parameters $\bar{q}$ and u. Now note that

$$\frac{\partial \nu_2}{\partial q_i} = \frac{1}{n}\left[2q_i + 2\left(q_1 + ... + q_{n-2} - (n-1)\bar{q} + u\right)\right], \quad \forall i = 1, ..., n-2 \tag{5.6}$$

so the gradient of (5.5) is

$$\nabla_q \nu_2 = \frac{2}{n}\left[\left(I + \mathbf{1}\mathbf{1}^T\right)q + \left(-(n-1)\bar{q} + u\right)\mathbf{1}\right] \tag{5.7}$$

where $\mathbf{1}$ is the $n-2$ dimensional column vector of 1s and I is the $(n-2)\times(n-2)$ identity matrix. The Hessian of (5.5) is

$$\mathcal{H}_q\left(\nu_2\right) = \frac{2}{n}\left(I + \mathbf{1}\mathbf{1}^T\right). \tag{5.8}$$

Since $I \succeq 0$ and $\mathbf{1}\mathbf{1}^T \succ 0$ then $\mathcal{H}_q(\nu_2) \succeq 0$ and ν_2 is a strictly convex function. ν_2 will have a minimum at its critical point which can be found by setting (5.7) equal to zero and solving for q. The Sherman-Morrison formula can be used to invert $\left(I + \mathbf{1}\mathbf{1}^T\right)$. The result is

$$q = [(n-1)\bar{q} - u]\left(I - \frac{1}{n-1}\mathbf{1}\mathbf{1}^T\right)\mathbf{1} \tag{5.9}$$

$$= \left(\bar{q} - \frac{u}{n-1}\right)\mathbf{1}. \tag{5.10}$$

Now plugging this result into (5.4) yields $q_{n-1} = \bar{q} - \frac{u}{n-1}$. So the minimum of (5.5) occurs when

$$q_i = \bar{q} - \frac{u}{n-1},\ q_n = \bar{q} + u, \quad \forall i = 1, ..., n-1. \tag{5.11}$$

Plugging this result into (5.5) yields

$$\min_q \nu_2(q; \bar{q}, u) = \frac{1}{n}\left[(n-1)\left(\bar{q} - \frac{u}{n-1}\right)^2 + (\bar{q} + u)^2\right] \tag{5.12}$$

$$= \bar{q}^2 + \frac{u^2}{n-1}. \tag{5.13}$$

Now consider the second case of $q_n = \bar{q} - u$. If (5.3) is replaced with $q_n = \bar{q} - u$, then the same steps (i.e. (5.4) through (5.13)) can be followed. The results will show that the configuration positions in this case (i.e. those shown in (5.11) for the previous case) are $q_i = \bar{q} + \frac{u}{n-1},\ q_n = \bar{q} - u, \quad \forall i = 1, ..., n-1$. With this configuration, the same minimum moment is obtained as in (5.13). □

The configurations resulting in $\min_q \nu_2(q; \bar{q}, u)$ from Lemma 5.1 are shown in Figure 5.1 for the six vehicle case.

Theorem 5.2. Consider $n \geq 2$ vehicles in m dimensions. Denote the average position in the i^{th} dimension as $\bar{q}$, as in Lemma 5.1. If the second-order squares moment in the i^{th} dimension is less than or equal to $\bar{q}^2 + \frac{u^2}{n-1}$, then each vehicle is a distance u or less from $\bar{q}$ in the i^{th} dimension.

Proof. From Lemma 5.1, the minimum second-order squares moment for all configurations with average position $\bar{q}$ and a vehicle u units from $\bar{q}$ is $\bar{q}^2 + \frac{u^2}{n-1}$. This function is strictly

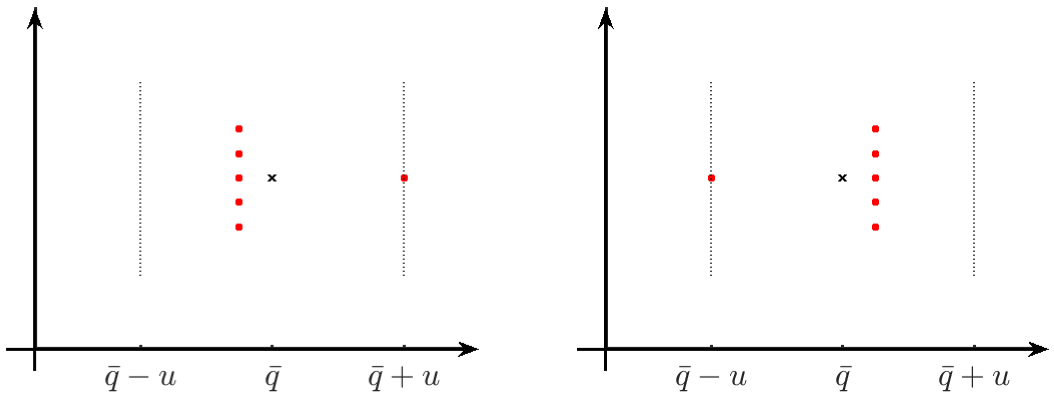

Figure 5.1: For six vehicles, the configurations which lead $\min_q \nu_2(q;\bar{q},u)$ in the horizontal direction, as described in Lemma 5.1. Left: If $q_n = \bar{q}+u$. Right: If $q_n = \bar{q}-u$. "×" represents the average position. Note the positions in the vertical dimension are irrelevant.

increasing in u for $u \geq 0$. So it is true that $\min_q \nu_2(q;\bar{q},\tilde{u}) > \min_q \nu_2(q;\bar{q},u)$ for all $\tilde{u} > u$. Therefore, for a certain configuration q^* with average position $\bar{q}$ and a bound u,

$$\frac{1}{n}\left[(q_1^*)^2 + ... + (q_n^*)^2\right] \leq \min_q \nu_2(q;\bar{q},u) \implies |q_i^* - \bar{q}| \leq u \quad \forall i = 1,...,n. \tag{5.14}$$

□

Note that the converse of (5.14) is not true, i.e. there may be configurations q^* which are bounded by u but for which the second order squares moment is greater than $\min_q \nu_2(q;\bar{q},u)$. One advantage of Theorem 5.2 is that it can be applied to each dimension of an m-dimensional space independently.

As an example of applying Theorem 5.2, consider bounding n vehicles in two dimensions. If we wish to bound these vehicles in the first dimension, note that the second-order squares equation is $\frac{1}{n}(p_{1,1}^2 + \cdots + p_{n,1}^2) = \mu_{2,0}$. Normally, $\mu_{2,0}$ would be defined by the function $\bar{f}(x,t)$. If instead we wanted to ensure all vehicles are a distance u or less from the average position in this dimension, we replace $\mu_{2,0}$ with some value less than or equal to $\bar{q}^2 + \frac{u^2}{n-1}$.

5.2 Other Polynomial Constraints

Taking a look back, our method of determining vehicle positions comes down to creating a system of polynomials with variables representing vehicle positions, then solving the system. In this section it will be shown that polynomial equations other than the moment polynomials can also be used as constraints. There are many types of polynomial constraints that can be used to create a wide variety of geometric constraints. Note that in order to encourage the ideal generated by the polynomial system to remain zero-dimensional, a moment polynomial should be deleted from the system whenever a new polynomial is added. Consider the following types of polynomial constraints which have geometric interpretations.

5.2.1 Two vehicle offset constraint

A polynomial equation can specify that a vehicle remain exactly d units above another vehicle in the dimension indexed by k. Denoting the upper and lower vehicles with indices i and j respectively, the equation is

$$p_{i,k} - p_{j,k} = d. \tag{5.15}$$

5.2.2 Two vehicle distance constraint

The following polynomial equation will place two vehicles, denoted by indices i and j, a distance d apart:

$$(p_{i,1} - p_{j,1})^2 + \cdots + (p_{i,m} - p_{j,m})^2 = d^2. \tag{5.16}$$

5.2.3 Vehicle on a line/plane constraint

A polynomial equation can specify that a vehicle remain on a line or a plane. As an example in 3-d, consider a plane defined by a normal vector $[a \;\; b \;\; c]$ and point $(p_{0,1}, p_{0,2}, p_{0,3})$. The equation for a vehicle with index i restricted to this plane is

$$a(p_{i,1} - p_{0,1}) + b(p_{i,2} - p_{0,2}) + c(p_{i,3} - p_{0,3}) = 0. \tag{5.17}$$

5.2.4 Other geometric shape constraints

There are many other curves and surfaces in two and three-dimensions which are defined by one or more polynomials. Examples include spheres, conic sections and non-piecewise spline functions.

5.2.5 Predetermined shape by moments

It is also possible to encourage a group of vehicles to configure themselves in the form of a certain shape if the moments of that shape are already known.

5.3 Simulation: Obstacle Avoidance by Bounding

Theorem 5.2 and some of the geometric constraints in Section 5.2 are now applied in an example domain which includes an obstacle. In this example, vehicles will be forced to simultaneously track a function and avoid an obstacle. The obstacle is defined to be a rectangular wall with a slot. The wall is four units wide and is centered along the vertical axis. The slot is six units wide and the coordinate of the horizontal centerline of the slot is $s_c = 4$.

The vehicles will follow the moment matching formulation to track a function in the same way as in the dynamic rose tracking in Section 3.4. While tracking, if vehicle positions are predicted to collide with the wall, then the polynomial system is changed to one that ensures that the wall will be avoided. In these cases, the standard system of mn moment matching

polynomials is replaced with the following system:

$$\begin{aligned}
\frac{1}{5}(p_{1,1}+p_{2,1}+p_{3,1}+p_{4,1}+p_{5,1}) &= \mu_{1,0}\\
\frac{1}{5}(p_{1,2}+p_{2,2}+p_{3,2}+p_{4,2}+p_{5,2}) &= s_c\\
\frac{1}{5}(p_{1,2}^2+p_{2,2}^2+p_{3,2}^2+p_{4,2}^2+p_{5,2}^2) &= s_c^2+\frac{u^2}{n-1}\\
p_{1,1}-p_{2,1} &= 2\\
p_{1,1}-p_{3,1} &= 3\\
p_{1,1}-p_{4,1} &= 2\\
p_{1,1}-p_{5,1} &= 3\\
(p_{3,1}-p_{4,1})^2+(p_{3,2}-p_{4,2})^2 &= 16\\
p_{1,2}-p_{2,2} &= 2\\
p_{1,2}-p_{3,2} &= 2.
\end{aligned} \tag{5.18}$$

Define the first dimension to be the horizontal dimension and the second to be the vertical. The first constraint in (5.18) keeps the average position of the group at the centroid of the function in the horizontal dimension. The second constraint keeps the average position of the group at the centerline of the slot. The third constraint uses Theorem 5.2 to ensure the group doesn't go above or below the slot. The final seven constraints specify a combination of offsets and distances between pairs of vehicles and ensure that their are mn polynomial equations specified.

The dynamic rose equation in (3.13) was used as the function with parameters $x_0 = -13$, $y_0 = -2$, $v_x = 2.3$, $v_y = -.3$, $a = 1$, $b = .5$, $c = .2$, $\omega = .3$, $k = 4$ and $r_{in} = 3$. The bound u was set to be 2.9. Five vehicles were used, Ω was set as a $[-30,\ 30] \times [-30,\ 30]$ grid of 1200×1200 elements, and the evalutation times were $t = 0, 2, 4, 6, 8, 10$. The simulation results are shown in Figure 5.2. The results show successful avoidance of the obstacle.

One advantage of this obstacle avoidance method is that while other constraints are imposed, the collective motion of the group can still be set in relation to the function. So, as demonstrated by this example, if we know (or have an estimate of) the horizontal velocity

of the function, we can still track the function in the horizontal direction while obstacle avoidance constraints are imposed.

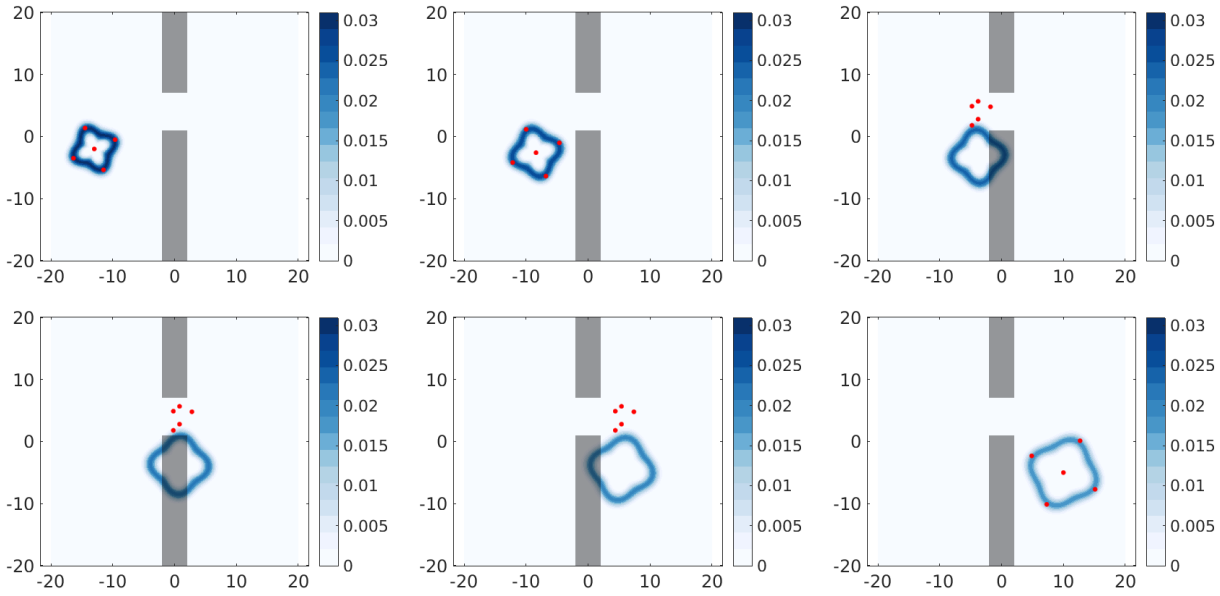

Figure 5.2: Five vehicles tracking a rose function with an obstacle. Top row: (l to r) $t = 0$, $t = 2$, $t = 4$. Bottom row: (l to r) $t = 6$, $t = 8$, $t = 10$.

Chapter 6

CONCLUSION

6.1 Summary

This thesis has developed a method to position sensing vehicles in the domain of a scalar function of space and time. This method calculates moments of the scalar function as well as moments of a discrete set of points, then sets the corresponding values equal to each other. The resulting system is a set of polynomials which must be solved. This method has analogies to the statistical method of moments.

By comparing several polynomial root solving software packages which either utilize Gröbner bases or polynomial homotopy continuation methods, it was determined that the overall fastest software for solving the polynomial systems was PHCpack, a polynomial homotopy solver. Using PHCpack, it was shown that polynomial systems of up to 10 variables/10 equations could be solved within two minutes. Next, simulations demonstrated the results of the moment matching method. The first simulations showed the resulting positions for one through five vehicles for three different 2-d functions. Another simulation showed the results of a dynamic function of time being "tracked" by five vehicles.

The moment matching method was shown to have several challenges associated with it, many of which have to do with the number of real roots of the polynomial system. One of the biggest challenges is that, using this methodology, some of the polynomial systems do not have real roots. In this case we proposed to redefine the polynomial systems with different moments. In many cases, these redefined systems will have real roots. Another big challenge is the computational limit of polynomial root solvers, which limits the number of vehicles which can be implemented. To get around this, we proposed partitioning the domain of interest into regions and placing a number of vehicles in each region. This was

demonstrated to be an effective method for a variety of different functions.

Finally, we stepped back to note that our methodology for determining vehicle positions reduces to solving a system of polynomials. The last section of the thesis takes advantage of this fact by using other polynomial equations to impose geometric constraints. It was proved that all vehicles could be bound together by restricting the second-order squares moments. Additionally, examples of other polynomials constraints were given which can set the relative distance of a pair of vehicles or place individual vehicles on a geometric surface. In the final section, an example was given which uses moment matching, second-order squares moment bounding and offset constraints to simultaneously track a function of time and avoid an obstacle.

6.2 Future Work

6.2.1 Improvement of the Moment Matching Formulation

As discussed in Section 4.1.4, one of the biggest challenges of this formulation is that some function/moment combinations do not have real roots. An interesting direction of future work would be to attempt to analytically determine conditions which guarantee that a set of equations determined by the moment matching formulation either have or do not have real roots. The Real Nullstellensatz [21], for example, can provide a guarantee that a system of polynomials does not have a real solution.

It was shown that a promising method to rectify systems with no real roots is to use a different combination of moments. The way we proposed doing this was to change the last moment in the graded lexicographic ordering. However, based on simulations, this method would sometimes fail even after trying the next fifteen moments. Accordingly, an interesting direction of future work would be to investigate different schemes for moment selection.

It is also possible that the problem of no real roots could be rectified by removing polynomials from the system, that is, solving systems of polynomials with less equations than variables. The challenge of this approach would be that these systems would most likely gen-

erate positive-dimensional ideals. In this case, it would be worthwhile to investigate methods which can determine *any* real roots of a positive-dimensional system.

Another interesting direction would be to attempt to relax or manipulate the polynomial systems into either a convex or sum of squares optimization problem. If this could be done, it is possible that systems with larger numbers of equations and variables could be solved. Lastly, since this thesis has only showed examples in 2-d, it would also be interesting to perform some simulations with vehicles in 3-d.

6.2.2 Estimation and Control

The formulation in this thesis has assumed that all functions $f(x,t)$ are known exactly. This is unrealistic since the application of this thesis is to characterize or monitor an unknown environmental event. Accordingly, a necessary direction of future work would be to incorporate a method of estimation for the function. This method would depend on factors such as how much information about the function is known and if there is a model of the function.

Additionally, this thesis has only considered determining positions for a group of vehicles, and not controlling the vehicles to move between these positions. The first step in devising a control scheme would be to determine which point each vehicle will go to at the next time point. For example, if vehicle number one is at a position at the first time point, then which of the n positions should it go to at the second time point? Once it is determined which point each vehicle will go to next, a controller would be needed to perform the point-to-point control. There are very simple methods to perform point-to-point control and would be worthwhile to investigate different point-to-point controllers.

BIBLIOGRAPHY

[1] Daniel J. Bates, Jonathan D. Hauenstein, Andrew J. Sommese, and Charles W. Wampler. Bertini: Software for numerical algebraic geometry. Available at bertini.nd.edu with permanent doi: dx.doi.org/10.7274/R0H41PB5.

[2] Calin Belta and Vijay Kumar. Abstraction and control for groups of robots. *IEEE Transactions on Robotics*, 2004.

[3] Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. Cryptanalysis of hfe, multi-hfe and variants for odd and even characteristic. *Designs, Codes and Cryptography*, 2013.

[4] K. O. Bowman and L. R. Shenton. *Estimation: Method of Moments*. John Wiley & Sons, Inc., 2004.

[5] F. Bullo, J. Cortés, and S. Martínez. *Distributed Control of Robotic Networks*. Applied Mathematics Series. Princeton University Press, 2009. Electronically available at http://coordinationbook.info.

[6] Janice L. Coen and Wilfrid Schroeder. The High Park fire: Coupled weather-wildland fire model simulation of a windstorm-driven wildfire in Colorado's Front Range. *Journal of Geophysical Research: Atmospheres*, 120(1):131–146, 2015.

[7] David Cox, John Little, and Donal O'Shea. *Using Algebraic Geometry*. Springer, 2005.

[8] David Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms*. Springer, 2015.

[9] Wolfram Decker, Gert-Martin Greuel, Gerhard Pfister, and Hans Schönemann. SINGULAR 4-0-2 — A computer algebra system for polynomial computations. `http://www.singular.uni-kl.de`, 2015.

[10] Jean-Charles Faugère. FGb: A Library for Computing Gröbner Bases. In Komei Fukuda, Joris Hoeven, Michael Joswig, and Nobuki Takayama, editors, *Mathematical Software - ICMS 2010*, volume 6327 of *Lecture Notes in Computer Science*, pages 84–87, Berlin, Heidelberg, September 2010. Springer Berlin / Heidelberg.

[11] Elizabeth Gross, Heather A. Harrington, Zvi Rosen, and Bernd Sturmfels. Algebraic systems biology: A case study for the Wnt pathway. *Bulletin of Mathematical Biology*, 2015.

[12] Yun Guan and Jan Verschelde. *Software for Algebraic Geometry*, chapter PHClab: A MATLAB/Octave Interface to PHCpack, pages 15–32. Springer New York, New York, NY, 2008.

[13] S. Kalantar and U. Zimmer. Motion planning for small formations of autonomous vehicles navigating on gradient fields. In *Underwater Technology and Workshop on Scientific Use of Submarine Cables and Related Technologies, 2007. Symposium on*, pages 512–519, April 2007.

[14] M.A.K. Khalil, editor. *Atmospheric Methane: Sources, Sinks, and Role in Global Change*. Springer, 1993.

[15] Tõnu Kollo and D. von Rose. *Advanced Multivariate Statistics with Matrices*, volume 579 of *Mathematics and Its Applications*. Springer, 2005.

[16] NOAA Great Lakes Environmental Research Laboratory. Predicting and tracking contaminant spills in the Huron-Erie Corridor. online. Available: http://www.glerl.noaa.gov/pubs/brochures/hecwfs.pdf.

[17] Daniel Marthaler and Andrea L. Bertozzi. Tracking environmental level sets with autonomous vehicles. In Sergiy Butenko, Robert Murphey, and Panos M. Pardalos, editors, *Recent Developments in Cooperative Control and Optimization*, volume 3 of *Cooperative Systems*, pages 317–332. Springer US, 2004.

[18] Dhagash Mehta, Yang-Hui He, and Jonathan D. Hauenstein. Numerical algebraic geometry: A new perspective on string and gauge theories. *Journal of High Energy Physics*, 2012.

[19] P. Ögren, E. Fiorelli, and N.E. Leonard. Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment. *Automatic Control, IEEE Transactions on*, 49(8):1292–1302, Aug 2004.

[20] Luc Rolland. Synthesis of the forward kinematics problem algebraic modeling for the general parallel manipulator: displacement-based equations. 2007.

[21] Bernd Sturmfels. *Solving Systems of Polynomial Equations*, volume 97 of *CBMS Regional Conference Series in Mathematics*. American Mathematical Society, 2002.

[22] Jan Verschelde. Algorithm 795: PHCpack: A general purpose solver for polynomial systems by homotopy continuation. 1999.

[23] Peng Yang, Randy A. Freeman, and Kevin M. Lynch. Multi-agent coordination by decentralized estimation and control. *IEEE Transactions on Automatic Control*, 2008.

[24] J. Yuh. Design and control of autonomous underwater robots: A survey. *Autonomous Robots*, 8(1):7–24.