Path Optimization Using sub-Riemannian Manifolds with Applications to Astrodynamics

by

James K Whiting

S.B., Aeronautics and Astronautics, Massachusetts Institute of Technology (2002)
S.B., Electrical Engineering and Computer Science, Massachusetts Institute of Technology (2002)
S.M., Aeronautics and Astronautics, Massachusetts Institute of Technology (2004)

Submitted to the Department of Aeronautics and Astronautics in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2011

Author ....
Department of Aeronautics and Astronautics
September 24, 2010

Certified by..
Prof. Olivier deWeck
Associate Professor of Aeronautics and Astronautics and Engineering Systems
Thesis Supervisor

Certified by..
Prof. Manuel Martinez-Sanchez
Professor of Aeronautics and Astronautics
Thesis Supervisor

Certified by..
Prof. Ray Sedwick
Assisstant Professor of Aeronautics and Astronautics, University of Maryland
Thesis Supervisor

Accepted by..
Eytan H. Modiano
Chair, Committee on Graduate Students

Path Optimization Using sub-Riemannian Manifolds with Applications to Astrodynamics

by

James K Whiting

Submitted to the Department of Aeronautics and Astronautics
on January 26, 2011, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Differential geometry provides mechanisms for finding shortest paths in metric spaces. This work describes a procedure for creating a metric space from a path optimization problem description so that the formalism of differential geometry can be applied to find the optimal paths. Most path optimization problems will generate a sub-Riemannian manifold. This work describes an algorithm which approximates a sub-Riemannian manifold as a Riemannian manifold using a penalty metric so that Riemannian geodesic solvers can be used to find the solutions to the path optimization problem. This new method for solving path optimization problems shows promise to be faster than other methods, in part because it can easily run on parallel processing units. It also provides some geometrical insights into path optimization problems which could provide a new way to categorize path optimization problems. Some simple path optimization problems are described to provide an understandable example of how the method works and an application to astrodynamics is also given.

Thesis Supervisor: Prof. Olivier deWeck
Title: Associate Professor of Aeronautics and Astronautics and Engineering Systems

Thesis Supervisor: Prof. Manuel Martinez-Sanchez
Title: Professor of Aeronautics and Astronautics

Thesis Supervisor: Prof. Ray Sedwick
Title: Assisstant Professor of Aeronautics and Astronautics, University of Maryland

Acknowledgments

I would like to thank everyone who made this research possible: my committee members - Olivier de Weck, Manuel Martinez-Sanchez, and Ray Sedwick - for providing advice as I navigated the depths of differential geometry; NASA and the MIT School of Engineering (through a TA position for 1.00) for providing funding as I worked through the mess of equations to create a small amount of order in the chaos of Newtonian gravity; my official readers Thomas Lang and Paulo Lozano for feedback as I completed my research; and my unofficial proofreaders Dale Winter, Rachel and Alan Fetters, Sarah Rockwell, and Allen Bryan for helping me to realize where my explanations needed more clarification.

I would like to thank my many friends for making life more enjoyable while I toiled on a seemingly endless task of translating PhD level theoretical math into moderately usable engineering concepts. I would especially like to thank the Boston change ringers for providing a steady rhythm in my life, Tech Squares for reminding me to peel off from my research and shoot the stars every now and then, Metaphysical Plant for reminding me that all hunts for knowledge are eventually completed, and MIT Hillel and TBS for the services and prayers that provided regular cycles in my life to mark the passage of time. I would like to thank my high school cross-country team for training me to have the endurance to keep going forever, my high school math club for encouraging me to study math beyond what I was taught, and the Cows for being good friends.

I would like to thank my parents for everything they have done: providing a good home for me to grow up in, encouraging me to explore the abstract world of ideas, letting me rush off to MIT where I got stuck for years in an endless maze of equations that I may have finally found a path through (suboptimal as it may have been), and most of all for teaching me the value of hard work, honesty, and persistence. I would like to thank my many siblings for living their lives so fully while I was too busy to do so on my own, providing me with many niblings to enjoy, and making the holidays so full of love and excitement. I would like to thank my in-laws for providing some local family and support as Mira and I toiled through the challenges of graduate school. I would like to thank my extended family for being a good and supportive family, and Mira's extended family for welcoming me so quickly and filling our lives with love and happiness. I would also like to thank my son Jesse for making the final months of this work much more exciting than they would have been without him.

And most of all I would like to thank my wife Mira, without whom I would never have made it all the way to the end. She has provided support and encouragement at every step of this process. I have learned how to move the heavens and the earth for her and I would like to remind her that some things really are rocket science. Melanyecce iluvòrënenya oio, mo bhean chéile luinnar silamiradilenya.

Contents

1 Introduction **15**

2 Previous Path Optimization Methods **17**

2.1 Analytical Methods 18

2.1.1 Hamiltonian 18

2.1.2 Lagrangian 18

2.1.3 Differential Inclusion 19

2.1.4 Kurush-Kuhn-Tucker Conditions 19

2.2 Discretization Methods 20

2.2.1 Runge-Kutta Shooting 20

2.2.2 Finite Differences 20

2.2.3 Pseudospectral Methods 20

3 An Introduction to Differential Geometry **21**

3.1 Manifolds 21

3.1.1 Coordinates 22

3.2 Tensors 22

3.2.1 Vectors 22

3.2.2 Covectors 23

3.2.3 General Tensors 23

3.2.4 Summation convention 23

3.2.5 Tensor Fields 24

3.2.6 Flows 24

3.2.7 Wedge Product 24

3.2.8 Inner Product 24

3.3 Bundles 25

3.3.1 Bundles 25

3.3.2 Frames 25

3.3.3 Tangent Spaces . . . 25
3.4 Parallel Transport and Connection . . . 26
3.4.1 Connection . . . 26
3.4.2 Parallel Transport . . . 26
3.5 Derivatives . . . 26
3.5.1 Partial Derivatives . . . 27
3.5.2 Exterior Derivatives . . . 27
3.5.3 Lie Brackets . . . 27
3.5.4 Covariant Derivatives . . . 28
3.6 Distributions . . . 28
3.6.1 Connection and Curvature . . . 28
3.7 Measurements . . . 29
3.7.1 Metric and Cometric . . . 29
3.7.2 Metric Connection . . . 30
3.8 Geodesics . . . 30
3.8.1 Geodesic Divergence . . . 30
3.8.2 Geodesic Uniqueness . . . 30
3.8.3 Singular Geodesics . . . 31
3.9 Branches of Geometry . . . 31
3.9.1 Riemannian Geometry . . . 31
3.9.2 Pseudo-Riemannian Geometry . . . 32
3.9.3 Sub-Riemannian Geometry . . . 32
3.10 Formulas . . . 34
3.10.1 Christoffel Symbols . . . 34
3.10.2 Riemann Curvature Tensor . . . 34
3.10.3 Ricci Curvature Tensor . . . 35
3.10.4 Sectional Curvature . . . 35

4 Path Optimization and sub-Riemannian manifolds 37
4.1 Problem Formulation . . . 37
4.1.1 Manifold . . . 37
4.1.2 Moving Frame . . . 38
4.1.3 Metric . . . 39
4.1.4 Changing Frames . . . 40
4.2 Optimal Paths . . . 41
4.2.1 Uniqueness of Solutions . . . 41
4.3 Discussion . . . 42

4.3.1 Hamiltonian Equivalence 42
4.3.2 3-D Example 44

5 Searching for Geodesics **47**
5.1 Finite Difference Methods 47
5.1.1 Flat Space 47
5.1.2 Curved Riemannian Spaces 49
5.1.3 Computational Efficiency 49
5.2 Pseudospectral Method 51
5.2.1 Computational Efficiency 53
5.3 Stream Processing 53

6 Heisenberg Manifolds **55**
6.1 Definition 55
6.2 3-Dimensional Heisenberg Group 56
6.2.1 Results 58
6.3 Tank 60
6.3.1 Implementing the Algorithm 64
6.3.2 Results 67
6.3.3 Curvature and Geodesic Uniqueness 67
6.4 Summary 74

7 The Astrodynamics Manifold **75**
7.1 Coordinates and Frames 75
7.2 Metric and Cometric 79
7.3 Christoffel Symbols 81
7.4 Lie Brackets 83
7.5 Connection, Curvature, and Torsion 85
7.6 Results 86

8 Conclusions and Future Work **91**
8.1 Comparison to other methods 91
8.2 Future Work 92

A Simulation Code **95**
A.1 Common Functions 95
A.2 Generic Algorithm 95
A.2.1 Setup 95

A.2.2 Finite Difference Calculations 97

A.2.3 Pseudospectral Method 100

A.3 The Heisenberg Manifold 104

A.3.1 Finite Difference Method 104

A.3.2 Pseudospectral Method 105

A.4 The Tank Manifold 105

A.4.1 Problem Specific Code (Pseudospectral) 105

A.4.2 Full Code (Pseudospectral Method) 107

A.4.3 Jacobi Fields 112

A.4.4 Hamiltonian Integration 116

A.4.5 Geodesic Integration 117

A.5 Astrodynamics 120

A.5.1 Finite Difference Method 120

A.5.2 Pseudospectral Method 124

A.5.3 Hamiltonian Integration 125

List of Figures

4-1 The tank manifold 38

4-2 Diagrams showing how an optimal path could be found on a circle or sphere. The upper left diagram shows the optimal path with no constraints. As the penalty function is increased, the path moves towards the valid surface as shown in the upper right diagram. This method will not work if the path is completely orthogonal to the constraints, because the projection operation will not work. 46

6-1 The Heisenberg manifold path optimization problem. Find the shortest path that encloses a fixed area. The enclosed area goes from the origin to the initial point in a straight line, along the path to the final point, and then in a straight line to the origin. 57

6-2 Movement of a single point in the path as the algorithm converges (α moves from .07 towards 0). The horizontal line shows the analytically determined correct location of the point. The "breaks" in the line are from adding more points to the path. 59

6-3 Convergence rate of basic algorithm and extroplated movement algorithm 60

6-4 Convergence rate of the extrapolated movement algorithm with and without final point extrapolation 61

6-5 The tank manifold path optimization problem 62

6-6 Pseudo-code description of the algorithm 65

6-7 Trajectory Variation as a function of β. Higher values of β lead to more movement in the x-direction and shallower steering angles. The lowest β curve is the one closest to being vertical, the highest β curve is the one that moves farthest horizontally. β values here are equally spaced logarithmically from 0.2 to 200. 68

6-8 $\log_{10}$ magnitude of the 2 Jacobi vector fields for a geodesic of the tank manifold with $\beta = 5$ 71

6-9 Angle between the 2 Jacobi vector fields for a geodesic of the tank manifold 72

6-10 Comparison of the original geodesic and the geodesic found by checking the Jacobi condition 73

6-11 Close-up view of the geodesic found from the Jacobi condition, showing the very tight back and forth movements that allow the tank to move nearly sideways 73

7-1 The astrodynamics manifold path optimization problem. This is for a coplanar orbital transfer using a two-body Newtonian gravitational model with point masses. 76

7-2 Δv as a function of dr for a sample trajectory. The Δv is increasing as the algorithm converges, while dr decreases until the penalty is too large, and then dr begins to increase. 88

7-3 Δv ratio (geodesic divided by Hamiltonian) minus one. The graph shows the distribution of the error of the Δv calculation for the 216 generated paths. The table shows the summary statistics of the distribution. The max and min are the ratio without subtracting one. 89

List of Tables

1 List of Symbols 14
2 Notation Directory 14

p	semi-latus rectum
h	modified angular momentum ($\sqrt{p/\mu}$)
e_x	eccentricity in the projected x direction ($e\cos(\omega+\Omega)$)
e_y	eccentricity in the projected y direction ($e\sin(\omega+\Omega)$)
f	true anomaly
Ω	longitude of the ascending node
ω	argument of perihelion
ϖ	longitude of perihelion ($\Omega+\omega$)
L	true longitude ($f+\varpi$)
ξ	$1+e_x\cos L+e_y\sin L=p/r$

The basis directions are radial, tangential, and normal to the orbital plane. This is effectively a spherical coordinate system.

Table 1: List of Symbols

Wedge Product	$a \wedge b$	3.2.7
Inner Product	$< a, b >$	3.2.8
Vector	a^i	3.2.1
Covector	a_i	3.2.2
Summation Convention	$a^i b_i$	3.2.4
Exterior Derivative	da	3.5.2
Lie Bracket	$[x, y]$	3.5.3
Covariant Derivative	∇a	3.5.4
Christoffel Symbols	Γ^i_{jk}	3.10.1

This work deals only with real numbers. While most of what is done here would also apply to complex manifolds, the variable i is not meant to indicate the imaginary unit.

Table 2: Notation Directory

Chapter 1

Introduction

Path optimization problems have been studied for over 300 years. Methods of solving path optimization problems have two parts: an analytical part that derives the optimality conditions and a numerical part that finds a solution to the optimality conditions. This work focuses on a new analytical approach.

A recent paper[1] described the various analytical approaches taken to solving path optimization problems. Many analytical methods use the calculus of variations to derive some local optimality condition. The most common example is adding a costate to create a Hamiltonian system.

The current work is a fundamentally different approach to solving the path optimization problem. Instead of working directly with the equations of motion as prior methods have done, this method is based on formulating the problem geometrically and then using the results of differential geometry to develop an answer. Control problems have a natural geometric interpretation based on using the state-space description of the problem to provide coordinates on a manifold. The cost functions provide a metric for the manifold, which is then sub-Riemannian. The optimal paths are the geodesics of the manifold.

One advantage of using sub-Riemannian geometry to solve path optimization problems is that the features of the manifold can be studied to discover interesting properties of the underlying problem. For example, the curvature of the manifold can be used to determine if a geodesic is unique. If it is not unique, then there may be a different geodesic connecting the same points with a shorter length. Additionally, there is a class of curves called singular geodesics which do not satisfy the geodesic equation but are still length-minimizing curves. These solutions would never be found through calculus of variations based methods because they do not generally satisfy the local optimality conditions.

Previous methods of solving path optimization problems are summarized in chapter 2. This is followed by a brief introduction to differential geometry in chapter 3, which should be sufficient to allow most people with an undergraduate degree in engineering to understand the math in this work. References to more detailed math texts are included for readers who are interested in a more thorough discussion of the mathematical concepts. These two chapters provide the relevant background information for this research.

The most significant part of this work is chapter 4, which explains the relationship between path optimization problems and geometrical manifolds. A method is presented which will produce a sub-Riemannian manifold that corresponds to any path optimization problem. Another method is presented to produce a reasonable penalty metric for the sub-Riemannian manifold that produces a compatible Riemannian manifold. Chapter 5 includes the other major part of this work, which is two related methods for finding geodesics in Riemannian manifolds with penalty metrics.

The next two chapters provide several detailed examples showing how the algorithms work. Chapter 6 includes an example based on the simplest possible sub-Riemannian manifold, the Heisenberg manifold. It also has an example of working through all the calculations for the tank problem, with a detailed description of every step of the algorithm. Chapter 7 goes through the more complicated problem that originally motivated this research: coplanar orbital transfer trajectory optimization. The final chapter summarizes the findings of this research and provides some further research topics that have been opened by this work.

Chapter 2

Previous Path Optimization Methods

A path optimization problem can in general be stated as:

Minimize the cost function

$$J[x, u] = \int_{t_0}^{t_f} F(x(t), u(t), t)dt \tag{2.1}$$

subject to the dynamic constraints

$$\dot{x}(t) = f(x(t), u(t), t) \tag{2.2}$$

and the end point constraints

$$x(t_0) = x_0 \tag{2.3}$$

$$x(t_f) = x_f \tag{2.4}$$

where $x(t)$ is the state-space description of the system and $u(t)$ is the control vector of the system. $x(t)$ and $u(t)$ are both vector-valued functions of time (t).

There are two steps to creating an algorithm to solve path optimization problems. First, an analytical method has to be used to derive conditions that will determine when a path is optimal. Then a discretization method has to be used to convert the problem into one with a finite number of points so that a computer can calculate a solution[1].

2.1 Analytical Methods

2.1.1 Hamiltonian

The Hamiltonian formulation leads to a Hamiltonian function[2][3][4]

$$H(x, u, \lambda, t) = F(x, u, t) + \lambda^T f(x, u, t) \tag{2.5}$$

where F and f are the cost and constraints (as defined in section 2) and λ is the costate.

The solution to the optimal path problem is then

$$\begin{aligned} \dot{x} &= \frac{\partial H}{\partial \lambda} && (2.6) \\ \dot{\lambda} &= -\frac{\partial H}{\partial x} && (2.7) \\ \frac{\partial H}{\partial u} &= 0 && (2.8) \end{aligned}$$

When these equations can be satisfied, the solutions are optimal paths. For cases where the equations cannot be satisfied (normally the result of an overconstrained problem), Pontryagin's minimum principle provides the solution. Pontryagin's principle states that the optimal solution is the feasible solution which minimizes the Hamiltonian.

For an n dimensional optimal control problem, the Hamiltonian method adds an n dimensional costate, which doubles the dimensionality of the optimization problem from n to $2n$. However, the equations can be solved more easily, so the problem is often easier to solve even with twice as many dimensions.

2.1.2 Lagrangian

The Lagrangian formulation defines a Lagrangian function

$$L(x, \dot{x}, \lambda, t) = F(x, \dot{x}, t) + \lambda^T f(x, \dot{x}, t) \tag{2.9}$$

where λ is the vector of Lagrange multipliers and varies with time.

The solution is found by use of the Euler-Lagrange equation

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}}\right) = \frac{\partial L}{\partial x} \tag{2.10}$$

which will lead to the same equations as the Hamiltonian formulation. This method is slightly more restrictive, because the controls do not appear in the Lagrangian, which means that this method can only be used if it is possible to solve for the controls based on the trajectory. This will be the case as long as the controls are linearly independent, so that the equation for $\dot{x} = f(u)$ can be inverted to produce $u = g(\dot{x})$.

The Lagrangian formulation also provides a way of finding singular solution paths. The singular paths are the solution with the Lagrangian

$$L(x, \dot{x}, \lambda, t) = \lambda^T f \tag{2.11}$$

Not all problems have singular solutions. A singular solution occurs when the constraints on a problem are sufficiently restrictive that some paths have no local variations. These paths will not in general satisfy local optimality conditions, but may still be part of an optimal path solution.

This formulation will also lead to a TPBVP (two-point boundary value problem). However, the dimensionality does not necessarily double. The dimensionality of the problem goes from n to $n+k$, where k is the number of constraints imposed on the dynamics.

2.1.3 Differential Inclusion

Differential inclusion is based on allowing functions to produce a range of values instead of a single value. If the control variables can be computed from the changes in state, then the cost function can be calculated from the changes in state. Rather than calculating optimality conditions, the differential inclusion method derives formulas for the cost functions based only on the trajectory.[5] This formulation produces a non-linear programming (NLP) problem.

2.1.4 Kurush-Kuhn-Tucker Conditions

The KKT conditions add another set of constraints for inequalities that must be satisfied by the solution:

$$g(t) \leq 0 \tag{2.12}$$

where $g(t)$ is also a vector-valued function of time. The formulation is then similar to the Lagrangian with more constraints. The solution satisfies the following relations:

$$\nabla F(x, \dot{x}, t) + \mu g(x, \dot{x}, t) + \lambda f(x, \dot{x}, t) = 0 \tag{2.13}$$

$$\mu_i \geq 0 \tag{2.14}$$

$$\sum_i \mu_i g_i = 0 \tag{2.15}$$

where μ is a vector of constants similar to Lagrange multipliers.

2.2 Discretization Methods

2.2.1 Runge-Kutta Shooting

A common method for solving a TPBVP is to integrate a path from one of the points until it either reachs the second point or it is clear that it will not reach the second point. The convergence of shooting methods is relatively slow, because it is difficult to determine how to modify the initial trajectory in a way that will modify the endpoint in a desired manner.

2.2.2 Finite Differences

A continuous differential equation can be approximated by finite difference equations at a set of discrete points. Finite difference equations approximate the derivative of a function by the differences of the value of the function at nearby points. For example, dy/dx at point n could be approximated by $(y_{n+1} - y_n)/(x_{n+1} - x_n)$.

2.2.3 Pseudospectral Methods

A pseudospectral method uses a set of orthogonal functions to approximate a continuous function at a series of grid points. This can often provide better estimates of the derivatives of the function at the grid points than finite difference equations. Some common orthogonal functions include Fourier series (sines and cosines), Chebyshev polynomials, and Legendre Polynomials. For some of these polynomials, it makes more sense to have non-uniform spacing between the discretization points. This allows the points to be located such that the orthogonal functions are easier to compute (for example, nodes can be placed where many of the functions are equal to 0).

Chapter 3

An Introduction to Differential Geometry

This chapter provides a brief introduction to differential geometry. Differential geometry is the study of geometrical objects and how they change. There are many texts on the subject that provide a more comprehensive description and derivation of these concepts and formulas. Spivak's series on differential geometry[6] provides a thorough explanation of differential geometry. Differential Geometry for Physicists[7] is particularly good for people without a strong background in theoretical math. There are a few books on sub-Riemannian geometry, including one by Montgomery[8] and another by Calin and Chang[9]. The Riemannian geometry book by Manfredo Perdigão do Carmo[10] has a detailed discussion of the mathematics behind geodesics.

3.1 Manifolds

A manifold is a set of connected points. Each point in a manifold is near some set of points (called the neighborhood of that point). The neighborhood of each point is similar to a Euclidean space. The dimension of the manifold at each point is the dimension of the Euclidean space that it resembles. A smooth manifold has the same dimension at every point.

Mathematical functions can be defined which have some specified value at every point in a manifold. A smooth function does not vary much within a small neighborhood of any point in the manifold. A differentiable function has continuous derivatives at all points in the manifold.

Manifolds can be divided into submanifolds. Each submanifold has all the properties of the original manifold except that it does not contain all the points (but all the points it does contain must be connected).

3.1.1 Coordinates

Coordinates are a set of functions that taken together uniquely identify the points in a manifold. The minimum number of functions required to do this for an n-dimensional manifold is n. A coordinate chart is a set of coordinate functions that span a portion of a manifold - meaning that they uniquely identify every point in that part of the manifold. An atlas is a set of charts that taken together span the entire manifold. Some manifolds can have a single chart that spans the entire manifold, while other manifolds (such as a sphere) cannot.

Any set of functions with the uniqueness property define a valid coordinate system. This means that there are no "natural" or "intrinsic" coordinates for a manifold. The coordinates will also not in general have any relationship to the geometrical properties of the manifold (other than the dimension).

Coordinates are used to identify the points in a manifold. Without the coordinates, the manifold still exists in some abstract sense. The geometrical properties of the manifold are the same in every coordinate system. What the coordinates provide is a means of turning geometry problems into algebra and calculus problems. With coordinates on a manifold, there is some way of measuring how close points are.

3.2 Tensors

3.2.1 Vectors

A vector is a geometrical object with a length and a direction.

A vector space is a set of vectors that is closed over addition and scalar multiplication. Being closed over addition means that any two vectors in the vector space can be added and the result will also be in the vector space. Being closed over scalar multiplication means that any vector in the vector space can have its length multiplied by a scalar to produce another vector which is also in the vector space. A vector space also has a linear operation for addition and scalar multiplication.

All vectors in a vector space have the same dimension, which is also the dimension of the vector space. Each point in a manifold has a vector space attached to it, which has the same dimension as the manifold at that point.

When writing out the components of a vector, the indices of the components are raised:

$$\mathbf{v} = \sum v^i \mathbf{e}_i \tag{3.1}$$

where $\mathbf{e}_i$ are the basis vectors.

3.2.2 Covectors

A covector is a linear function from a vector space to the real numbers. One result of this definition is that covectors also form a vector space, which is why they are also called dual vectors. A vector is also a linear function from a covector space to the real numbers. An informal explanation of vectors and covectors is that they function as row and column vectors (however that is not a mathematically precise definition). An alternate name for a covector is a 1-form.

When writing out the components of a covector, the indices of the components are lowered:

$$\mathbf{a} = \sum a_i \mathbf{e}^i \tag{3.2}$$

where $\mathbf{e}^i$ are the basis covectors.

3.2.3 General Tensors

A tensor is a multi-linear geometrical object that is made of some number of vectors and covectors. A (k, l) tensor is a linear function that maps k vectors and l covectors to the real numbers, which also makes it the product of k covectors and l vectors. In particular, a $(1, 0)$ tensor is a covector and a $(0, 1)$ tensor is a vector. Vectors and covectors are combined through the tensor product to make tensors. A $(n, 0)$ tensor is also called an n-form.

Tensors are geometrical objects, so they have an abstract geometrical existence that is independent of any coordinate systems. A tensor space can be defined at each point in a manifold for every type of tensor. In general, the tensor space is only defined at a point, so tensors at different points cannot be directly compared.

Tensor components are written with their indices raised and lowered in accordance with which indices correspond to vectors and which ones correspond to covectors.

3.2.4 Summation convention

When writing out tensor formulas, it is common to want to sum the different components to produce something like a dot product. For example, with the vector v and the covector a, the product $a(v)$ is written in components as

$$a(v) = a_i v^i = \sum_i a_i v^i \tag{3.3}$$

so the summation is implied and unwritten. Only repeated indices are summed, and only when one is raised and the other is lowered.

3.2.5 Tensor Fields

A field is a mapping of some type of tensor to every point of a manifold. A smooth field has objects that vary smoothly in the neighborhood of every point in the manifold.

A smooth vector field is the solution to a differential equation (it specifies a derivative at every point).

3.2.6 Flows

The flow of a vector field is the set of paths that are produced by using the vector field as a tangent vector at each point. If a fluid were flowing in the manifold with the velocity at each point given by the vector field, then the flow of the vector field would be the flow of the fluid.

3.2.7 Wedge Product

The wedge product (denoted $a \wedge b$) is a mathematical operation defined on two tensors. The precise mathematical definition of the wedge product is fairly complicated, but it has properties similar to the cross product. The wedge product is a linear antisymmetric operator, meaning that it has the following properties:

$$a \wedge b = -b \wedge a \tag{3.4}$$

$$(a+b) \wedge c = (a \wedge c) + (b \wedge c) \tag{3.5}$$

$$a \wedge a = 0 \tag{3.6}$$

where a, b, and c are tensors. The wedge product of a k-form and an l-form is a $(k+l)$-form. These are the properties that matter for this work, and a more detailed description is available in any book on differential geometry.

3.2.8 Inner Product

The inner product (denoted $< a, b >$) is a mathematical operation defined on two vectors from a vector space with a metric that produces a real number (metrics are discussed more in section 3.7.1). If the metric is the 2-form $g_{\alpha\beta}$, then the inner product is defined as:

$$< a, b >= g_{\alpha\beta} a^\alpha b^\beta \tag{3.7}$$

The inner product is symmetric and linear in Riemannian and sub-Riemannian geometry.

3.3 Bundles

3.3.1 Bundles

A bundle is a manifold with a tensor space at every point in the manifold. The total space of a bundle is the set of points in the manifold combined with the vector space at every point - in other words the total space is every possible tensor at every point in the manifold. A bundle has a projection operation which associates every point-tensor combination in the total space with the correct point in the manifold.

A bundle is itself a manifold, where every point-tensor combination can be considered a point. A tensor field is then a submanifold of the bundle, where every point has only one tensor associated with it.

3.3.2 Frames

Frames are similar to coordinate systems in that they provide numerical values for geometrical objects. A frame is a set of linearly independent tensors that form a basis of a tensor space. Every tensor in that tensor space can then be described uniquely by the linear combination of the basis tensors that is equal to the desired tensor. This allows tensors to be numerically compared and manipulated.

A smooth frame field is called a moving frame. A moving frame assigns a frame to every point in a smoothly varying manner. Typically only the vector moving frame is specified, because the covector frame can be determined from the vector frame (given a metric), and all other tensor frames can be determined from the vector and covector frames.

Any smooth frame field can be used as a moving frame. Any moving frame can be modified by rotations and stretching to produce another moving frame. Like coordinates, there is no "natural" or "intrinsic" moving frame on a bundle. However, every coordinate system produces a natural moving frame by differentiating the coordinate functions.

A moving frame and coordinate system together produce a coordinate system on a bundle.

3.3.3 Tangent Spaces

The tangent space of a manifold is the vector space of all tangent vectors. A tangent vector is a vector which points in a direction that is tangent to a feasible path in the manifold. For most types of manifolds, the tangent space is of the same dimension as the manifold and feasible paths run through the point to all neighboring points. For sub-Riemannian manifolds, the tangent space is of lower dimension and there are some neighboring points that are not directly connected by a feasible path in the manifold. The tangent space at every point forms a vector bundle on the manifold, called the tangent bundle.

3.4 Parallel Transport and Connection

Every point in a manifold has its own tensor spaces (for vectors, covectors, and other types of tensors). The basis vectors and covectors are defined independently at each point in the manifold. This means that there is no intrinsic way to compare tensors at different points in the manifold. The main topic of differential geometry is how to connect the basis vectors and covectors at different points in a manifold in a geometrically reasonable way. Once the basis vectors and covectors can be compared at nearby points, it is possible to move tensors along a path and calculate how they are changing in a geometrically meaningful way.

3.4.1 Connection

The connection provides a one-to-one mapping of tensors from the tensor space at a point to the tensor space of a nearby point. Two tensors at different points in the manifold are geometrically identical if and only if the connection says they are the same. The connection provides a way of "connecting" the tensor space at each point to the tensor space at nearby points.

3.4.2 Parallel Transport

Parallel transport is a geometrical operation that moves a tensor along a path in a manifold while keeping it geometrically constant. It uses the local transport of the connection to move from one point to the next in the manifold. The connection is like a differential equation that specifies how things change locally, while the parallel transport is like an integration of that equation to go from the beginning of a path to the end.

In general, parallel transport will be path dependent. The only exception is when the curvature of the manifold is 0, in which case all paths connecting two points will provide the same mapping of vectors from the initial point to the final point.

3.5 Derivatives

In differential geometry, different points in a manifold cannot be directly compared. In particular, the tangent space at each point is a unique vector space that is not directly related to any other tangent space. This means that there is not in general any natural way to measure derivatives along a vector or a path. Instead there are several reasonable ways to define derivatives.

3.5.1 Partial Derivatives

A partial derivative can be calculated by ignoring the differences between different tangent spaces and treating the space as Euclidean. This is mathematically useful as a component in calculating other derivatives, which can be defined in terms of how they modify the partial derivative. The notation for a partial derivative is the same as in regular calculus: ∂_x is the partial derivative in the x direction.

3.5.2 Exterior Derivatives

The exterior derivative operator only applies to n-forms. It converts an n-form to an $(n+1)$-form, including converting a function (0-form) to a 1-form. The exterior derivative is written as d. For any form a, $d(da) = 0$.

For a function f, df is the standard differential of the function - it is a linear operator on vectors that produces the directional derivative of the function in the direction of the vector, which is the gradient of the function.

$$df(X) = \langle \nabla f, X \rangle \tag{3.8}$$

where in this case, ∇f is the gradient of f, and $\langle X, Y \rangle$ is the dot product.

For an n-form $a = f dx_1 \wedge dx_2 \wedge \ldots \wedge dx_i = f dx_I$, the exterior derivative in coordinates is

$$da = \sum \frac{\partial f}{\partial x_i} dx_i \wedge dx_I \tag{3.9}$$

3.5.3 Lie Brackets

The Lie derivative uses vector fields to equate different points in the manifold. A common use of the Lie derivative is to calculate the Lie bracket of two vector fields. The Lie bracket measures the change in a vector field as it moves along the flow of a second vector field. The notation for the Lie bracket of vector fields a and b is $[a, b]$. The Lie Bracket is anti-symmetric on its two arguments, so $[a, b] = -[b, a]$. If two vector fields commute, then $[a, b] = [b, a] = 0$. The vector fields are only said to commute when the Lie bracket is 0 everywhere, not just at some points.

Physically, if two vector fields commute then movement along the two vector fields can happen in any order and the same point will be reached. This is the case in Euclidean coordinates. An example of non-commuting vector fields would be dx and dr as part of the standard coordinate vector fields for Cartesian and polar coordinates. Moving along the dx flow a distance of x and then along the dr flow a distance r does not in general lead to the same point as moving along the dr flow a distance r and then moving along the dx flow a distance x. The only exception is when the path starts on the x axis, in which $dr = dx$.

Any set of moving frames where all pairs of basis vector fields commute can be used to form a coordinate system by integrating the basis vectors from a chosen origin. A moving frame where some or all of the basis vector fields do not commute cannot be used to define a coordinate system.

In coordinates:

$$[a,b]^i = a^j \partial_j b^i - b^j \partial_j a^i \tag{3.10}$$

3.5.4 Covariant Derivatives

Covariant derivatives use parallel transport to make the tangent space at different points equivalent, which then allows derivatives to be calculated in a natural way. The covariant derivative is indicated with the ∇ operator: ∇_v is the covariant derivative in the v direction. In coordinates,

$$(\nabla_v u)^i = \partial_v u^i + \Gamma^i_{jk} v^j u^k = v^j \frac{\partial u^i}{\partial x^j} + \Gamma^i_{jk} v^j u^k \tag{3.11}$$

where u and v are vectors. If u is instead a covector, then

$$(\nabla_v u)_i = \partial_v u_i - \Gamma^k_{ij} v^i u_k = v^j \frac{\partial u_i}{\partial x^j} - \Gamma^k_{ij} v^j u_k \tag{3.12}$$

Higher order tensors have similar formulas, with the Christoffel symbol term ($\Gamma^i_{jk} \ldots$, defined in section 3.10.1) subtracted for lower indices and added for rasied indices.

3.6 Distributions

A distribution of rank k is a smooth k-dimensional subbundle of the tangent space on a manifold. A rank k distribution in an n-dimensional manifold can be defined by the $n-k$ linearly independent 1-forms θ_i with the property that $\theta_i(X) = 0$ for all vectors X in the distribution. A distribution is called involutive if $[X, Y]$ is in the distribution for all vectors X and Y in the distribution. A sub-Riemannian manifold is a Riemannian manifold with the tangent space restricted to a non-involutive distribution.

3.6.1 Connection and Curvature

A distribution has a curvature and connection that are based on the geometry of the distribution independent of any metric. These are based on the Carnot structure equations.

For a rank k distribution spanned by the one forms $\theta_1, \ldots\theta_k$, the connection is the matrix of one forms ω such that

$$d\theta_i = \sum_j \omega^i_j \wedge \theta_j + \Theta^i_j \tag{3.13}$$

where Θ^i_j is the torsion matrix of 2-forms for the distribution. The distribution curvature is

$$\Omega^i_j = d\omega^i_j + \omega^i_k \wedge \omega^k_j \tag{3.14}$$

The distribution connection, torsion, and curvature determine how the distribution twists along the manifold that it is embedded in.

3.7 Measurements

While it is possible to solve some geometrical problems by using drawings, many problems are easier to solve if the geometrical objects are measured so that the problems can be solved using algebra and calculus. Coordinates and moving frames provide the measurements, but it is important to remember that any chosen coordinate system or moving frame is not intrinsic to the manifold and that the numerical results only have meaning when attached to the chosen coordinate system and moving frame. Additionally, the geometrical answers will be the same regardless of the choice of coordinates or moving frames.

3.7.1 Metric and Cometric

A metric defines the inner product of a vector space. That means that a metric is a (2,0) tensor because it is a function which takes two vectors and produces a real number (the value of their dot product). Since the metric is a geometrical object, the numerical values will depend on the frame that has been chosen, but the computed lengths of vectors will be the same in all frames.

A cometric is the inverse of a metric. It is a (0,2) tensor which takes two covectors and produces a real number. In any given frame, the metric and cometric are the matrix inverses of each other.

A metric on a manifold is a (2,0) tensor field that provides a metric at every point in the manifold.

A Riemannian metric is a smooth metric field that is positive definite at all points in the manifold. A pseudo-Riemannian metric is similar except that instead of being positive definite, the metric is only required to be non-degenerate. A positive definite metric has the property that every vector has a positive length, which will be the case if the metric written out as a matrix is positive definite. A non-degenerate metric is one in which there are no vectors other than the zero vector that have a dot product of zero with all other vectors. This will be the case if the metric has the same rank as the dimension of the manifold (meaning that all the rows of the metric matrix are linearly independent).

Path lengths can be found by integrating the tangent vector along the path. Similar computational methods can be used to find areas, volumes, etc.

3.7.2 Metric Connection

A metric connection is a connection that keeps the dot products of vectors constant as both vectors are parallel transported along any path in the manifold. For any smooth metric on the manifold, it should be possible to define a metric connection.

3.8 Geodesics

A geodesic is a geometrically straight line on a manifold. It is formed by moving along a tangent vector and parallel transporting the tangent vector along the path. Geodesics will not generally have a simple equation like they do in Euclidean spaces. If the curvature of a manifold is zero everywhere, then the manifold is Euclidean and it is possible to use Cartesian coordinates and moving frames. Any manifold with non-zero curvature anywhere is non-Euclidean and will have at least some geodesics that do not have a linear equation.

A geodesic is the locally shortest path between two points in a manifold. It is possible that more than one geodesic connects two points in a manifold, in which case only one of them is the global minimum. There are some conditions that can be checked to prove that all geodesics are global minimums.

3.8.1 Geodesic Divergence

Two nearby geodesics will follow different paths. The divergence between geodesics is described by the Jacobi equation:

$$\frac{D^2}{dt^2}X = -R(X,T)T \tag{3.15}$$

where X is the vector describing the difference between geodesics at length t, T is the tangent vector to the geodesic at length t, and R is the Riemann curvature tensor (defined in section 3.10.2). This equation is only valid when the geodesics are still close enough that the Riemann tensor will not be significantly different on the two geodesics.

3.8.2 Geodesic Uniqueness

In a Riemannian manifold, any two points can be connected by at least one geodesic. Since each geodesic is locally length minimizing, it is useful to determine when geodesics are unique and when they are not. If only one geodesic connects two points on a manifold, then it is the global minimum path length between the points. If multiple geodesics connect two points, then one of them will be the global minimum (although it is also possible to have multiple geodesics with the same length, for example the great circles on a sphere connecting antipodal points).

The simplest case where geodesics are unique is when the sectional curvature for every surface at every point is negative. That will lead to the Jacobi equation taking the form:

$$\frac{D^2}{dt^2}X = KX \tag{3.16}$$

where K is a positive function. When X is positive, it will become more positive. When X is negative, it will become more negative. This means that the divergence between geodesics will grow as they spread out farther. If the sectional curvature is 0 everywhere then the second derivative is 0 and the geodesics will diverge at a linear rate (this corresponds to a Euclidean space). In both of these cases, the geodesics will diverge from each other, so they will never intersect a second time.

The only way for the geodesics to converge is if the sectional curvature is positive at some point along the geodesic. If this happens, then it is possible for some geodesics to converge at multiple points. An example of such a case is a sphere, which has a constant positive curvature. All the lines of longitude on the globe are geodesics that intersect at the poles.

Jacobi Fields

The Jacobi equation can be used to generate Jacobi vector fields[10]. A Jacobi field is a vector field that satisfies the Jacobi equation. The Jacobi fields can be used to determine when geodesics are unique.

Conjugate points on a geodesic are points where a Jacobi field is 0 at both points. Geodesics can only intersect at multiple points if the points are conjugate to each other.

3.8.3 Singular Geodesics

In some sub-Riemannian manifolds, it is possible to have only one path connecting two points in the manifold. This path is then the minimum length path, as no other paths exist. Since it is a minimal length path, it is a geodesic, but in general it will not satisfy the geodesic equation. These geodesics are called singular geodesics[11].

3.9 Branches of Geometry

3.9.1 Riemannian Geometry

Riemannian geometry is the main branch of differential geometry; it is the study of Riemannian manifolds. A Riemannian manifold has a smooth, positive-definite metric and a tangent space that has the same dimension as the manifold. Riemannian manifolds have many useful properties that make them relatively easy to work with. In particular, every Riemannian manifold has a torsion-free metric compatible connection, called the Levi-Civita connection.

3.9.2 Pseudo-Riemannian Geometry

A pseudo-Riemannian manifold is like a Riemannian manifold, but the metric is only required to be non-degenerate instead of positive definite. The spacetime of General Relativity is a pseudo-Riemannian manifold. Because of the application to Relativity, this is the most studied area of applied differential geometry

3.9.3 Sub-Riemannian Geometry

A sub-Riemannian manifold is like a Riemannian manifold, except the tangent space is of lower or equal dimension to the manifold. This means that Riemannian manifolds are actually a subset of sub-Riemannian manifolds. In order for a manifold to be sub-Riemannian when it is not Riemannian, the tangent space has to change from point to point. The Lie brackets of the basis vectors for the tangent space have to include some components that are outside of the tangent space. This is an equivalent condition to the problem having non-Holonomic constraints. This is a complicated statement which can best be described by an example.

The state of a car can be described by its position (x,y), the angle the car is pointing in (θ), and the angle of the steering wheel (ϕ). The car can move forward or backwards along a circle (ds) and can change the radius of the circle, but cannot translate sideways or rotate in place. The total space is four dimensional, but the tangent space is only two dimensional. All points in the space can still be reached through some path. The tangent space restriction makes some problems difficult to solve, such as parallel parking.

- When the car is pointing along dx, $\theta = 0$
- $d\theta = a[d\phi, dx]$ for some value of a
- $dy = b[d\theta, dx] = ab[[d\phi, dx], dx]$ for some values of a,b

Vertical and Horizontal Vector Spaces

In a sub-Riemannian manifold, it is possible to partition the vector spaces into a horizontal part and a vertical part. The horizontal part is the tangent space at each point. The vertical part is everything else. All vectors can be divided into their horizontal part and their vertical part, although one of these will be 0 for some vectors.

Compatible Riemannian Manifolds

In a sub-Riemannian manifold, lengths are only defined for horizontal vectors. The vertical vectors have an undefined length. It is possible to create a Riemannian manifold such that all the horizontal vectors have the same length and all the vertical vectors have a defined length. The manifold can be further restricted by requiring all the vertical vectors to be orthogonal to all the horizontal vectors (which means their dot products are 0). A manifold with these properties is called a compatible Riemannian manifold. Every sub-Riemannian manifold has an infinite number of compatible Riemannian manifolds, because the lengths for the vertical vectors can be any arbitrary function. The only change between the sub-Riemannian manifold and the various compatible Riemannian manifolds is the metric. Each compatible Riemannian manifold defines a compatible Riemannian metric.

Creating a compatible Riemannian manifold allows all the results of Riemannian geometry to be applied to a manifold which is similar to the sub-Riemannian manifold. A particularly useful family of compatible metrics is the penalty metrics, which have vertical vector lengths approaching infinity. In the sub-Riemannian metric, the vertical vectors have infinite length. Therefore the penalty metrics approach the sub-Riemannian metric as the penalty terms approach infinity. This also means that the geodesics in the penalty manifolds approach the sub-Riemannian geodesics.

The Step of a Distribution

Every sub-Riemannian manifold has a tangent distribution of some dimension n. A new distribution can be formed by adding the Lie brackets of the tangent vectors to the original distribution, which will generally increase the dimensionality of the distribution. The step of the distribution is the number of times that the Lie brackets have to be added to increase the dimension of the distribution all the way to the dimension of the manifold (plus one for the original vectors). A Riemannian manifold has a step 1 distribution.

For the car manifold, the distribution has a 2-dimensional tangent space (ds and $d\phi$). The Lie bracket is $d\theta = a[ds, d\phi]$ for some scaling constant a. This brings the tangent space up to three dimensions (ds, $d\phi$, and $d\theta$). The Lie bracket on these three vectors adds the fourth dimension through $[ds, d\theta]$.

The growth vector of a distribution describes the number of dimensions spanned by the distribution plus the Lie brackets of the distribution. For the car problem, the growth vector is $(2, 3, 4)$.

Categories of sub-Riemannian Manifolds

The simplest type of sub-Riemannian manifold is a Heisenberg manifold. A Heisenberg manifold has the following properties:[9]

- The manifold is step 2 everywhere
- The tangent space distribution is fully spanned by m orthonormal vector fields X_i
- There are $p = n - m$ locally defined 1-forms ω_α with $\omega_\alpha(X_i) = 0$, which satisfy the nonvanishing conditions Det $\{\omega_a([X_i, X_j])\}_{ij} \neq 0$

where n is the total dimension of the manifold and m is the dimension of the tangent space ($m < n$). One consequence of these properties is that the rank of the distribution (m) has to be even for a manifold to be a Heisenberg manifold. If the rank is odd, then the skew symmetric matrix defined in property 3 will always have a determinant of 0. Additionally, Heisenberg manifolds have no singular geodesics.[9]

It is worth noting that the definition of a Heisenberg manifold depends only on the tangent space of the manifold and is independent of the metric.

A Grushin manifold is like a Riemannian manifold that has a metric which is singular at some places in the manifold (for example, if it contains a term such as $1/x^2$). A Hormander manifold is a sub-Riemannian manifold which is step 3 or greater. More details about different types of sub-Riemannian manifolds can be found in [9].

3.10 Formulas

3.10.1 Christoffel Symbols

The Christoffel symbols are used to measure how much the covariant derivatives differ from partial derivatives. The formula for calculating the Christoffel symbols is

$$\Gamma^i_{jk} = \frac{1}{2} g^{im} \left(\frac{\partial g_{jm}}{\partial x^k} + \frac{\partial g_{mk}}{\partial x^j} - \frac{\partial g_{jk}}{\partial x^m} \right) \tag{3.17}$$

with g_{ij} as defined in section 3.2.8.

3.10.2 Riemann Curvature Tensor

The Riemann curvature tensor describes how a manifold is different locally from a flat manifold. Geometrically, the Riemann tensor is defined as

$$R(u, v)w = \nabla_u \nabla_v w - \nabla_v \nabla_u w - \nabla_{[u,v]} w \tag{3.18}$$

The Riemann tensor measures the noncommutivity of the covariant derivative. If it is zero everywhere, that means that the covariant derivative is commutative, so parallel transport is independent of path and depends only on the endpoints. This makes the space equivalent to a Euclidean space.

The formula for calculating the components is

$$\begin{aligned} R^i_{jkl} &= \frac{\partial}{\partial x^k} \Gamma^i_{jl} - \frac{\partial}{\partial x^l} \Gamma^i_{jk} + \Gamma^i_{km} \Gamma^m_{jl} - \Gamma^i_{lm} \Gamma^m_{jk} && (3.19) \\ R_{ijkl} &= \left(\frac{\partial^2 g_{il}}{\partial x^j \partial x^k} + \frac{\partial^2 g_{jk}}{\partial x^i \partial x^l} - \frac{\partial^2 g_{ik}}{\partial x^j \partial x^l} - \frac{\partial g_{jl}}{\partial x^i \partial x^k} \right) + g_{mn} \left(\Gamma^m_{jk} \Gamma^n_{il} - \Gamma^m_{jl} \Gamma^n_{ik} \right) && (3.20) \end{aligned}$$

The Riemann tensor has n^4 components, but it has the following symmetries:

$$R_{ijkl} = R_{klij} = -R_{jikl} = -R_{ijlk} \tag{3.21}$$

In addition, the Bianchi identities further reduce the number of independent components:

$$\begin{aligned} R_{ijkl} + R_{iljk} + R_{iklj} &= 0 && (3.22) \\ \nabla_m R^i_{jkl} + \nabla_l R^i_{jmk} + \nabla_k R^i_{jlm} &= 0 && (3.23) \end{aligned}$$

These identities leave only $n^2(n^2-1)/12$ independent components in the Riemann tensor. The following chart describes the number of independent components as a function of the dimension of the manifold.

dimension	components
1	0
2	1
3	6
4	20
5	50
6	105

3.10.3 Ricci Curvature Tensor

The Ricci tensor is a contraction of the Riemann tensor.

$$R_{ij} = R^k_{ikj} = g^{kl}R_{ikjl} = g^{kl}R_{kilj} = \frac{\partial\Gamma^k_{ij}}{\partial x^k} - \frac{\partial\Gamma^k_{ik}}{\partial x^j} + \Gamma^k_{ij}\Gamma^l_{kl} - \Gamma^l_{ik}\Gamma^l_{ik}\Gamma^k_{jl} \tag{3.24}$$

The Ricci tensor is also symmetric ($R_{ij} = R_{ji}$).

3.10.4 Sectional Curvature

The sectional curvature is the Gaussian curvature of a 2-dimensional submanifold within the manifold. If the sectional curvature of all surfaces at all points is non-positive, then every geodesic is unique in the sense that only one curve satisfying the geodesic equation connects any two points on the manifold. This condition means that geodesics are global optima instead of just local optima.

It can be calculated as

$$K(u,v) = \frac{\langle R(u,v)v, u\rangle}{\langle u,u\rangle\langle v,v\rangle - \langle u,v\rangle^2} \tag{3.25}$$

where u and v are vectors in the 2-dimensional submanifold.

Chapter 4

Path Optimization and sub-Riemannian manifolds

4.1 Problem Formulation

Many path optimization problems can be formulated using the language of sub-Riemannian geometry. This formulation provides a convenient and straightforward method of finding the optimal path. A simple problem of a tank on a flat surface will be used to illustrate the method. Any other vehicle that can rotate in place and move freely along a plane with a cost function based on the total distance moved and the total angular rotation would be mathematically identical. Figure 4-1 shows a diagram of the problem.

4.1.1 Manifold

The manifold for a path optimization problem is the same as the state-space formulation for the problem. Each state corresponds to one dimension of the manifold and provides one coordinate function.

For the tank, the state can be described using three variables. The position of the tank is given by its x and y coordinates and the current pointing direction is given by an angle θ. Other coordinates could be chosen, such as using polar coordinates for the position. The only requirement for the coordinates is that they must have a unique value (as a set) for every point in the manifold. It is useful to choose coordinates that are natural and simple for the problem. For example, Cartesian coordinates are often simpler than polar coordinates, but polar coordinates would generally be simpler for rotational problems. The best coordinates to use will depend on the nature of the problem.

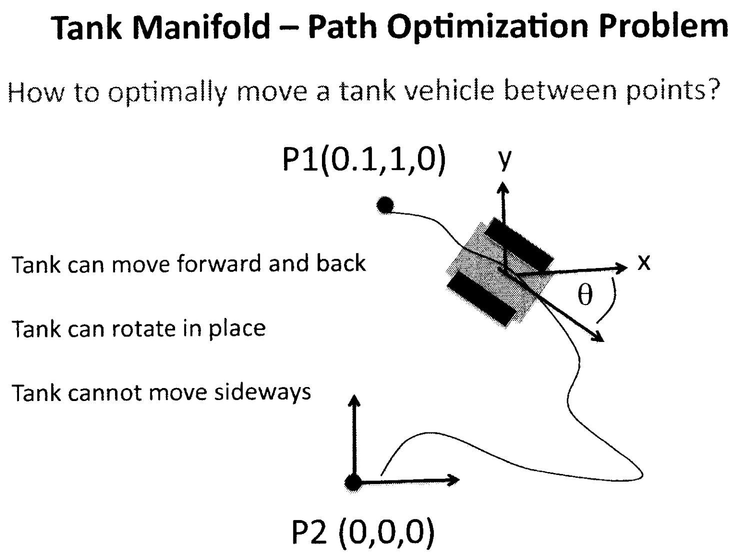

Tank Manifold – Path Optimization Problem
How to optimally move a tank vehicle between points?
P1(0.1,1,0)
y
x
θ
Tank can move forward and back
Tank can rotate in place
Tank cannot move sideways
P2 (0,0,0)

Figure 4-1: The tank manifold

4.1.2 Moving Frame

A moving frame for the problem has to be chosen in order to find the metric. The moving frame should be chosen to produce a simple metric. This can be done by examining the cost function of the problem. For the tank problem, there is a cost for driving and a cost for turning. These are the only feasible motions, so the tangent space is two dimensional.

When solving a problem where the tangent space is of lower dimension than the total space, additional vectors have to be chosen to complete the moving frame. The additional vectors can be any vectors that fully span the vertical portion of the vector space. It is best to choose vectors that are in some natural sense orthogonal to the feasible movements.

For the tank, one more vector has to be chosen. A reasonable choice is moving the tank sideways. This is naturally orthogonal in the sense that there is no way to make these motions in any state the tank will be in. Another possible option would be translation in the x direction, but this is sometimes the same as driving the tank forward, which makes it a bad choice. The above choice of a vector is also relatively simple so it will not complicate the math. Another possible option would be rotating the tank about some fixed point (which changes the pointing direction of the tank and its location), but this would lead to a more complicated metric.

For the tank, the entire moving frame is **a** representing the basis vector for moving the tank forward, **b** representing the basis vector for rotations in place, and **c** representing the basis vector for translating the tank sideways.

4.1.3 Metric

The sub-Riemannian metric is given by the cost function. The lengths and angles of vectors in the vertical sub-bundle are not well defined for a sub-Riemannian manifold, so they can be chosen arbitrarily. The simplest choice is to define the basis vectors of the moving frame to be orthogonal to the horizontal space. This choice will change the metric based on which vertical vectors are chosen, but in the limit as their lengths are made infinite, all of these metrics will converge to the same sub-Riemannian metric. The lengths of the vertical basis vectors should depend on a parameter, so they can be written as $1/\alpha$ or something similar.

Any function that can be determined by just the path through the state-space can be used as a cost function. The metric is the derivative of the cost function, which makes the metric a linear function of the tangent vector at each point, but this does not impose any constraints on the cost function. However, if the cost function depends on the history of the path as well as the current state, the cost function will be complicated and the method might not work well. An example of this would be if the cost function for driving a car depends on the amount of fuel in the car, so that driving around a circle back to the same point would lead to a different local cost function. This can be solved by adding all variables that the cost function depends on to the state-space description of the problem (which would mean adding the amount of fuel in the car as a variable in the description of the car's state).

The cost function for the tank is a cost of one to move the tank a unit of distance and a cost of β to rotate the tank one radian. The parameter β represents the ratio of the cost of steering to the cost of moving. A high value of β will lead to solutions with little steering and lots of back and forth movements, while a low value of β will lead to tight turns with little movement. These two operations are naturally orthogonal, so $\mathbf{a} \cdot \mathbf{b} = 0$. The lengths of the other basis vector will be set to $1/\alpha$ with α small. The metric is diagonal in this moving frame.

Using the column vector $\xi = (a, b, c)$ to represent the physical motion, the cost function is $\xi^T g(\xi)\xi$. $g(\xi)$ is the diagonal matrix with entries $(1, 1/\alpha, \beta)$. So the cost function is

$$\text{cost} = \xi^T g(\xi)\xi = \begin{bmatrix} \mathbf{a} & \mathbf{b} & \mathbf{c} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/\alpha & 0 \\ 0 & 0 & \beta \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \end{bmatrix} \tag{4.1}$$

4.1.4 Changing Frames

In order to change frames to the coordinate-based frame, the relationship between the coordinate vectors and the physical movement vectors must be determined. This is done by determining the differential equations that relate the movements to the state changes. For the tank,

$$\begin{aligned}
\mathbf{x} &= \cos\theta\mathbf{a} - \sin\theta\mathbf{c} && (4.2)\\
\mathbf{y} &= \sin\theta\mathbf{a} + \cos\theta\mathbf{c} && (4.3)\\
\theta &= \mathbf{b} && (4.4)
\end{aligned}$$

In matrix form, this is

$$\mathbf{X} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ \theta \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \end{bmatrix} = T\xi \qquad (4.5)$$

The inverse transformation is

$$\xi = \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ \theta \end{bmatrix} = T^{-1}\mathbf{X} \qquad (4.6)$$

The metric in the new moving frame is derived from the old metric multiplied by the columns of the inverse transformation matrix. For example, $\mathbf{x}$ is the same vector as $\cos\theta\mathbf{a} - \sin\theta\mathbf{c}$, so $g_{xx} = \cos^2\theta + (1/\alpha)\sin^2\theta$. The other entries are all generated in the same way. In equations,

$$\begin{aligned}
cost &= (T^{-1}\mathbf{X})^T g(\xi) T^{-1}\mathbf{X} && (4.7)\\
&= \mathbf{X}^T (T^{-1})^T g(\xi) T^{-1}\mathbf{X} && (4.8)\\
&= \mathbf{X}^T g(X)\mathbf{X} && (4.9)
\end{aligned}$$

$$g(X) = (T^{-1})^T g(\xi) T^{-1} \qquad (4.10)$$

$$g(X) = \begin{bmatrix} \cos^2\theta + (1/\alpha)\sin^2\theta & (1 - 1/\alpha)\cos\theta\sin\theta & 0 \\ (1 - 1/\alpha)\cos\theta\sin\theta & \sin^2\theta + (1/\alpha)\cos^2\theta & 0 \\ 0 & 0 & \beta \end{bmatrix} \qquad (4.11)$$

The cometric is the inverse of the metric and is produced in a similar way from the rows of the basis transformation matrix:

$$g(X)^{-1} = Tg(\xi)^{-1}T^T \tag{4.12}$$

$$g^{-1} = \begin{bmatrix} \cos^2\theta + \alpha\sin^2\theta & (1-\alpha)\cos\theta\sin\theta & 0 \\ (1-\alpha)\cos\theta\sin\theta & \sin^2\theta + \alpha\cos^2\theta & 0 \\ 0 & 0 & 1/\beta \end{bmatrix} \tag{4.13}$$

This is now a Riemannian metric on a Riemannian manifold.

4.2 Optimal Paths

The optimal paths for the control problem correspond to the geodesics of the manifold. Any method which will find geodesics in a Riemannian manifold can be used to find the optimal control trajectories. One method for finding the geodesics is given in chapter 5.

4.2.1 Uniqueness of Solutions

As stated in section 3.8.2, the geodesics of a Riemannian manifold are unique if the sectional curvature of every two-dimensional submanifold is negative at every point. For many control problems, the resulting manifold is likely to have sectional curvatures that are sometimes positive and sometimes negative. This will sometimes produce unique geodesics, but there may be some points that are connected by multiple geodesics.

Unless the sectional curvatures are always negative, Jacobi fields have to be used to determine if the geodesics are unique.[1] This will not in general be possible to do analytically, but it can be done numerically.

Once a geodesic has been found that connects two points, Jacobi fields can be generated along the geodesic. The initial value of all the Jacobi fields can be set to 0. Then the geodesic is unique if the Jacobi fields are all linearly independent along the geodesic. If the Jacobi fields are not linearly independent, then it is possible to create a Jacobi field by adding some of the Jacobi fields together so that the new Jacobi field is 0 at a second point.

It is easiest to generate the Jacobi fields by using a slightly modified version of the Jacobi equation. Starting at the initial point with an orthonormal basis that includes a vector tangent to the geodesic, the basis can be parallel transported along the geodesic. Since parallel transport preserves length and angles, these vectors will remain orthonormal. Then there will be $n-1$ basis

[1]Jacobi fields are described in section 3.8.2

vectors that are perpendicular to the geodesic (where n is the dimension of the manifold). Each of these can be used to generate a Jacobi field. Any other Jacobi field will be generated by some linear combination of these $n-1$ fields. The rewritten Jacobi equation is

$$J(t) = \sum_i f_i(t) e_i(t) \tag{4.14}$$

where e_i is the parallel transported basis vectors.

Defining

$$a_{ij}(t) = \langle R(\gamma'(t), e_i(t))\gamma'(t), e_j(t)\rangle \tag{4.15}$$

leads to the differential equation

$$f_j''(t) + \sum_i a_{ij}(t) f_i(t) = 0 \tag{4.16}$$

with the initial conditions $f_i(0) = 0$ and $f_i'(0) = 1$ for one of the functions and $f_i'(0) = 0$ for the other $n-2$ functions. The differential equation can be numerically integrated along the geodesic for each of the $n-1$ Jacobi vector fields to determine their values. If they are linearly independent at all points along the geodesic, then the geodesic is a global minimum. If they are not linearly independent at some point along the geodesic, then they provide the tangent vector at the initial point which will produce a second geodesic which should converge with the first geodesic at the conjugate point.[10]

4.3 Discussion

It is important to consider when this method is expected to work. Mathematically, this method is identical to formulating the path optimization using a Hamiltonian method. Any control problem that can be solved with a Hamiltonian method can also be solved by geodesic search.

4.3.1 Hamiltonian Equivalence

Starting with a manifold with a cometric g^{ab}, we can define a function $H = \frac{1}{2} g^{ab} p_a p_b$ and call this a Hamiltonian function. Using the standard notation from Hamiltonian mechanics,

$$\dot{x}^a = \frac{\partial H}{\partial p_a} = g^{ab} p_b \tag{4.17}$$

$$\dot{p}_a = -\frac{\partial H}{\partial x^a} = -\frac{1}{2}\frac{\partial g^{bc}}{\partial x^a} p_b p_c \tag{4.18}$$

The second derivative of x is

$$\begin{aligned}
\ddot{x}^a &= \dot{g}^{ab}p_b + g^{ab}\dot{p}_b \\
&= \frac{\partial g^{ab}}{\partial x^i}\dot{x}^i p_b - \frac{1}{2}g^{ab}\frac{\partial g^{ci}}{\partial x^b}p_c p_i \\
&= \frac{\partial g^{ab}}{\partial x^i}\dot{x}^i p_b - \frac{1}{2}g^{ab}g_{cj}g_{il}\frac{\partial g^{ci}}{\partial x^b}\dot{x}^j\dot{x}^l \\
&= -g^{aj}g^{bk}\frac{\partial g_{jk}}{\partial x^i}\dot{x}^i g_{bl}\dot{x}^l + \frac{1}{2}g^{ab}\frac{\partial g_{jl}}{\partial x^b}\dot{x}^j\dot{x}^l \\
&= -g^{aj}\frac{\partial g_{jk}}{\partial x^i}\dot{x}^i\dot{x}^k + \frac{1}{2}g^{ab}\frac{\partial g_{jl}}{\partial x^b}\dot{x}^j\dot{x}^l \\
&= -g^{ai}\frac{\partial g_{ic}}{\partial x^b}\dot{x}^b\dot{x}^c + \frac{1}{2}g^{ai}\frac{\partial g_{bc}}{\partial x^i}\dot{x}^b\dot{x}^c \\
&= -g^{ai}\left(\frac{\partial g_{ic}}{\partial x^b} + \frac{\partial g_{ib}}{\partial x^c} - \frac{\partial g_{bc}}{\partial x^i}\right)\dot{x}^b\dot{x}^c \\
&= -\Gamma^a_{bc}\dot{x}^b\dot{x}^c
\end{aligned} \tag{4.19}$$

which is the standard Riemannian geodesic equation[12]. This shows the mathematical equivalence of the two methods, which are both based on finding geodesics in a Riemannian or sub-Riemannian manifold. This also provides an alternate method for determining the cometric of a control problem if the Hamiltonian solution has already been found.

The Hamiltonian method of path optimization is based on creating a symplectic manifold which corresponds to the covector bundle of the base manifold. In other words, the Hamiltonian costate is the component representation of the tangent covector. The Hamiltonian function is a calculation of the tangent covector length, which is why the Hamiltonian function is constant along an optimal path. The path on the manifold is found by projecting the covector onto the tangent space and parallel transporting the tangent covector along the path. This produces an optimal trajectory because a geodesic is found by following a tangent vector as it is parallel transported along the path it generates.

In general, the sub-Riemannian cometric is singular, so it cannot be inverted to produce a metric. The cometric is used to convert a covector into a vector, which is why the costate can be used to find the tangent vector for the path. The metric is used to convert a vector into a covector, which is why the derivatives of the state do not uniquely determine the costate. The rank of the cometric is the same as the dimension of the tangent space in the control problem. If the manifold dimension is n and the tangent space dimension is k, then there are $n-k$ more dimensions in the covector space than in the vector space. In principle, it would be possible to use only $n-k$ costates in the Hamiltonian method because the other k costates could be found from the derivatives of the state.

Differences

There are two major differences between the methods. The first is that the Hamiltonian method works with no modifications even if the cometric is singular, which makes the metric degenerate. The second difference is in how the geodesic equation is solved.

The Hamiltonian method solves the geodesic equation by choosing a tangent covector and parallel transporting it along a path. This will produce a geodesic, but there is no way to control where the geodesic will go. The geodesic search method (described in chapter 5) will always produce a path from the starting point to the ending point, but may not fully converge to a geodesic in a reasonable amount of time.

In a Riemannian manifold, a tangent vector at a point fully defines a geodesic path. However, in a sub-Riemannian manifold, the tangent vector does not have enough information to fully specify a unique geodesic. Any vertical vector can be added to the tangent vector without changing its projection into the tangent space. All of these different vectors will appear the same in the sub-Riemannian manifold, but will produce different geodesic paths. It is this fact which complicates the solution process for sub-Riemannian control problems.

The Hamiltonian method solves this problem by using the tangent covector instead of the tangent vector. Since the cometric is singular but still finite, the tangent covector space is complete. The projection of a covector onto the tangent space (performed by the cometric) produces the proper tangent vector at each point. Parallel transporting the covector will then produce a geodesic.

The penalty metric method solves the problem by allowing small deviations from the actual tangent space. These small deviations permit the tangent vector space to be complete. In the limit as the penalty terms become infinite, the tangent space collapses back to the sub-Riemannian tangent space.

4.3.2 3-D Example

A representation of how this method (described in chapter 5) works can be demonstrated by considering the problem of how to find the shortest path between 2 points on a sphere. The method presented here is equivalent to drawing a straight line between the two points as an initial estimate of the path (this corresponds to having a very low penalty term). The path is easy to find and optimal in some sense, but it does not actually satisfy the constraints of the problem. As the penalty term is increased, the path is penalized for failing to meet the constraint. The result of this is that the path is projected onto the surface of the sphere, producing the geodesic of the original problem. It is important to note that this projection operation is ambiguous when the path is connecting two antipodal points on the sphere, because there is no particular direction to move the path in to better meet the constraints. The method presented here does not have a means to correct the projection operation for cases where the non-constrained path is perpendicular to the tangent space.

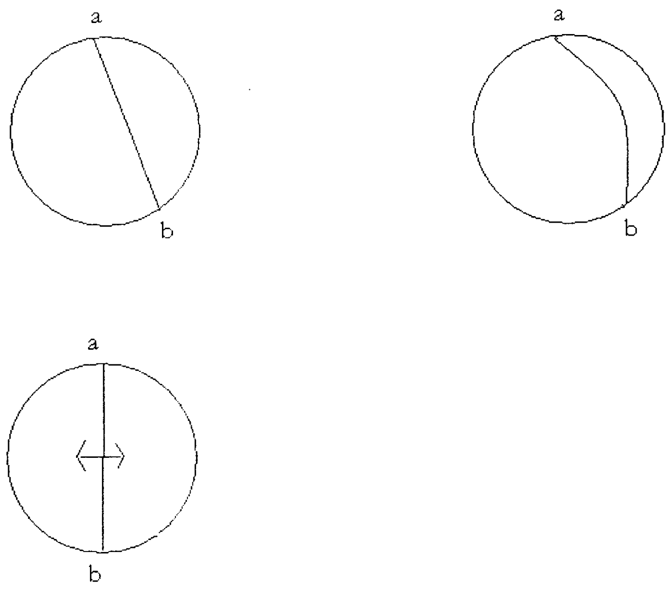


Figure 4-2: Diagrams showing how an optimal path could be found on a circle or sphere. The upper left diagram shows the optimal path with no constraints. As the penalty function is increased, the path moves towards the valid surface as shown in the upper right diagram. This method will not work if the path is completely orthogonal to the constraints, because the projection operation will not work.

Chapter 5

Searching for Geodesics

The geodesic equation is

$$\frac{d^2x^a}{ds^2} + \Gamma^a_{bc}\frac{dx^b}{ds}\frac{dx^c}{ds} = 0 \tag{5.1}$$

with the terms summed over all values of b and c and the equation applied separately for each value of a (a, b, and c take on values corresponding to each dimension of the problem).

In general this cannot be solved analytically. Previous methods of solving the geodesic equation involve discretizing the manifold.

The algorithm developed here solves the geodesic equation directly by moving each point of the solution until it is in the right place. The geodesic equation is modified to include an error term F^a

$$\frac{d^2x^a}{ds^2} + \Gamma^a_{bc}\frac{dx^b}{ds}\frac{dx^c}{ds} = F^a \tag{5.2}$$

For a geodesic, the error term will be 0. For any other curve, the error term will provide information about how to modify the curve to make it closer to a geodesic.

5.1 Finite Difference Methods

5.1.1 Flat Space

For a Euclidean space, the modified geodesic equation is

$$\frac{d^2x^a}{ds^2} = F^a \tag{5.3}$$

Consider the curve $x = s^2$ from $(s, x) = (0, 0)$ to $(s, x) = (1, 1)$. The derivatives are

$$\frac{dx}{ds} = 2s \tag{5.4}$$

$$\frac{d^2x}{ds^2} = 2 \tag{5.5}$$

which means that F^a has the constant value of 2. The geodesic with the same endpoints is $x = s$. For all points on the curve, $s > s^2$, so the values of x should be increased. Adding some multiple of F^a onto each point in the path will move the path closer to a geodesic. This is the general principle of this method.

If the line is discretized into some number of points, the derivatives can be calculated numerically. Using a finite difference method, the derivatives are

$$\frac{dx}{ds} = \frac{x_{n+1} - x_{n-1}}{s_{n+1} - s_{n-1}} \tag{5.6}$$

$$\frac{d^2x}{ds^2} = \frac{(x_{n+1} - x_n)/(s_{n+1} - s_n) - (x_n - x_{n-1})/(s_n - s_{n-1})}{s_{n+1} - s_{n-1}} \tag{5.7}$$

with constant step sizes in s, this becomes

$$\frac{dx}{ds} = \frac{x_{n+1} - x_{n-1}}{\Delta s} \tag{5.8}$$

$$\frac{d^2x}{ds^2} = \frac{x_{n+1} + x_{n-1} - 2x_n}{2(\Delta s)^2} \tag{5.9}$$

Putting this into the geodesic equation for flat space,

$$\frac{(x_{n+1} + x_{n-1})/2 - x_n}{(2\Delta s)^2} = F \tag{5.10}$$

The only way that F can be 0 is if $x_n = (x_{n+1} + x_{n-1})/2$, which is the expected equation for a straight line. When F is not 0, then the point x_n should be moved. Using Δx_n to represent the amount that x_n should be moved for the next iteration of the method,

$$\Delta x_n = \frac{(x_{n+1} + x_{n-1})/2 - x_n}{K(\Delta s)^2} \tag{5.11}$$

where K is a stiffness parameter that determines how much each point moves from one step to the next. If $K\Delta s^2 = 1$ then each point will be moved to the midpoint of its adjacent neighbors at each step. However, those neighbors will also have moved, so it will still take multiple iterations before the algorithm converges. If the stiffness is too high, then the points will not move far enough at each step. If the stiffness is too low, then the points will overshoot at each step and oscillate around the correct solution. If $K\Delta s^2 < 1/2$, then the points will overshoot by a larger amount than the current error terms, which means the error terms will grow with each iteration of the algorithm.

Having the correct stiffness is critical to the performance of the algorithm. If the stiffness is too high, then the algorithm will converge too slowly. When first starting the algorithm, it is best to have a low stiffness, so that the points can move quickly to somewhere near their correct locations. But once the algorithm has come close to converging, the stiffness has to be increased so that the points will not overshoot or oscillate. The stiffness should be adapted as the algorithm converges. One way to do this is with an overrelaxtion method - increasing the stiffness whenever the point overshoots, which can be determined by looking for a sign change in Δx_n between iterations, and decreasing the stiffness whenever the point moves in the same direction for two iterations.

5.1.2 Curved Riemannian Spaces

The algorithm also works for curved Riemannian spaces. For an n-dimensional curve, the derivatives for all coordinates have to be calculated. The calculations are still the same as before, but in curved space the Christoffel symbols are generally not equal to 0.

A simple curved space is the hyperbolic plane, which is a surface with constant negative curvature. The algorithm has been tested on this surface and finds geodesics with no difficulties.

5.1.3 Computational Efficiency

The finite difference algorithm converges fairly quickly for Riemannian spaces. The convergence rate is determined by how many points the path has and by how stiff the path is (based on the K parameter). The stiffness can be varied based on how the path is changing, which will automatically tune it to the proper value for the manifold.

If a high accuracy is desired for the path, then the answer will have to have a large number of points. Adding points to the path increases the effective stiffness, because more points have to be moved and at each iteration the points will only move a small amount (because the points will be close to their neighbors). The most computationally efficient way to find paths with a large number of points is to start with a small number of points and then add points as the path converges. As an added benefit, more points can be added to the parts of the path with more curvature, which is where a higher accuracy is needed most.

Adding points can be done in two ways. One way is to add points to the entire path such that the new step size is constant, but smaller than the previous step size. The other way is to only add points to some part of the path, which will lead to a variable step size.

The benefit of adding points to only some parts of the path is that the grid can be left coarse where it does not affect the accuracy of the curve. The benefit of adding points everywhere is that it is computationally easier, so it might be faster even with more points.

Variable Step Size

Allowing a variable step size can provide a more efficient algorithm in some cases. However, if the step sizes are not constant, then the derivatives are a little more complicated to calculate. A quadratic polynomial can be fit to the three points, which will provide the derivatives. The following substitutions are used in this derivation:

$$X_1 = x_{n-1} - x_n \tag{5.12}$$

$$X_2 = x_{n+1} - x_n \tag{5.13}$$

$$S_1 = s_{n-1} - s_n \tag{5.14}$$

$$S_2 = s_{n+1} - s_n \tag{5.15}$$

The equations to be solved are

$$aS_1^2 + bS_1 = X_1 \tag{5.16}$$

$$aS_2^2 + bS_2 = X_2 \tag{5.17}$$

$$b(S_2 - S_2^2/S_1) = X_2 - (S_2^2/S_1^2)X_1 \tag{5.18}$$

$$b = \frac{S_1^2X_2 - S_2^2X_1}{S_1S_2(S_1 - S_2)} \tag{5.19}$$

$$aS_1^2 + \frac{S_1^2X_2 - S_2^2X_1}{S_2(S_1 - S_2)} = X_1 \tag{5.20}$$

$$aS_1^2 = \frac{-S_1^2X_2 + S_2^2X_1 + S_1S_2X_1 - S_2^2X_1}{S_2(S_1 - S_2)} \tag{5.21}$$

$$aS_1^2 = \frac{-S_1^2X_2 + S_1S_2X_1}{S_2(S_1 - S_2)} \tag{5.22}$$

$$aS_1^2 = \frac{S_1}{S_2}\frac{S_2X_1 - S_1X_2}{S_1 - S_2} \tag{5.23}$$

$$a = \frac{1}{S_1S_2}\frac{S_2X_1 - S_1X_2}{S_1 - S_2} \tag{5.24}$$

$$a = \frac{X_1/S_1 - X_2/S_2}{S_1 - S_2} \tag{5.25}$$

$$a = \frac{X_2/S_2 - X_1/S_1}{S_2 - S_1} \tag{5.26}$$

The derivatives are

$$\frac{dx}{ds} = b \tag{5.27}$$

$$= \frac{S_1^2X_2 - S_2^2X_1}{-S_1S_2(S_2 - S_1)} \tag{5.28}$$

$$= \frac{(s_{n+1} - s_n)^2(x_n - x_{n-1}) + (s_n - s_{n-1})^2(x_{n+1} - x_n)}{(s_n - s_{n-1})(s_{n+1} - s_n)(s_{n+1} - s_{n-1})} \tag{5.29}$$

$$\begin{aligned}\frac{d^2x}{ds^2} &= 2a \qquad (5.30)\\ &= 2\frac{(x_{n+1}-x_n)/(s_{n+1}-s_n)-(x_n-x_{n-1})/(s_n-s_{n-1})}{s_{n+1}-s_{n-1}} \qquad (5.31)\end{aligned}$$

This modified algorithm has been tested with both flat space and the hyperbolic plane. In both cases it converges well to the same solution as with constant step sizes.

Adding Points

When adding points to the path, the most useful places to add more points is to the parts of the path where the pointwise approximation is least accurate. These will be the areas where the second derivatives are changing most rapidly, because the path is effectively interpolating quadratic polynomials between the points. Another useful place to add more points is where the manifold itself has a higher curvature. Higher curvature will lead to faster changes in the Christoffel symbols, which means that the approximation will be less accurate. Further work will be required to determine if there is a good way to determine where to add points so as to minimize the number of points needed.

A much simpler algorithm is to just add a point between the two most separated points. This will add points to the path so that they are almost evenly spaced, but requires using algorithms similar to what will be required for more adaptive point adding, so it is a good intermediate step.

A simulation was run that started with a path in the hyperbolic plane with only 5 points. Whenever the total movement of all the points in the path was small enough, another point was added. This led to convergence with 1000 points in the final path in about 7500 iterations. By comparison, starting with 1000 points requires about 900,000 iterations, (about 120 times as many iterations). Using variable step sizes reduces the computational rate by about a factor of four, but it is still much faster because an approximate solution is found with a small number of points and then the grid is made finer. However, it is even faster to double the number of points each time the solution converges and maintain a constant step size for any given number of points.

5.2 Pseudospectral Method

Pseudospectral methods are based on using some number of orthogonal functions to approximate the curve. The functions are evaluated at the points on the path to determine where the path goes and what its derivatives are. Once the point locations and derivatives are computed, the algorithm is identical for calculating the virtual forces and point movements.

There are several different sets of orthogonal polynomials that have been used for pseudospectral methods. The most common ones are Legendre polynomials, Lagrange polynomials, Chebyshev polynomials, and Jacobi polynomials. For this research, the Chebyshev polynomials were used. The Chebyshev polynomials of the first kind are defined as

$$T_0(x) = 1 \tag{5.32}$$
$$T_1(x) = x \tag{5.33}$$
$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \tag{5.34}$$

The Chebyshev polynomials of the second kind are defined as

$$U_0(x) = 1 \tag{5.35}$$
$$U_1(x) = 2x \tag{5.36}$$
$$U_{n+1}(x) = 2xU_n(x) - U_{n-1}(x) \tag{5.37}$$

The derivatives are:

$$\frac{d}{dx}T_n(x) = nU_{n-1}(x) \tag{5.38}$$
$$\frac{d^2}{dx^2}T_n(x) = n\frac{(n+1)T_n(x) - U_n(x)}{x^2-1} \tag{5.39}$$

The path and its derivatives can be described as a sum of the first m Chebyshev polynomials:

$$f(x) = \sum_{n=0}^{m} a_n T_n(x) \tag{5.40}$$
$$f'(x) = \sum_{n=1}^{m} n a_n U_{n-1}(x) \tag{5.41}$$
$$f''(x) = \sum_{n=2}^{m} n a_n \frac{(n+1)T_n(x) - U_n(x)}{x^2-1} \tag{5.42}$$

This allows the first and second derivatives to be computed algebraically once the path has been fit by the polynomials. If the collocation points are used, then the best fit to the path can be computed by a process similar to a Fourier transform: the sum of the path function multiplied by each polynomial function at all the points provides the coefficient of each polynomial (with some correction to normalize the results). The collocation points are the solution of the equation

$$x = -\cos(a\pi/(m-1)) \tag{5.43}$$

with a taking on the values from 0 to $m-1$.

The computational benefit of the pseudospectral method is that it converts a calculus problem into a linear algebra problem. The current point locations are multiplied by a matrix to generate the Chebyshev coefficients, which are multiplied by another matrix to generate the first and second derivatives. These values are then combined through the geodesic equation to produce the virtual force vectors, which provide the amount that each point should be moved in each iteration of the algorithm (as described in section 5.1).

Using pseudospectral methods to solve differential equations with constant coefficients converts the differential problem into a relatively simple linear algebra problem of adding and inverting matrices. However, the geodesic equation does not have constant coefficients, because the Christoffel symbols change depending on what points the path goes through. It may be possible to perform some kind of convolution procedure to multiply the Christoffel symbol functions in the differential equation to produce a pure linear algebra problem. If this can be done, then the path optimization problem could be transformed into a small number of matrix operations.

5.2.1 Computational Efficiency

As with finite difference methods, there is a tradeoff between accuracy and speed. Using more points (and therefore more orthogonal functions) requires larger matrices and more computation (roughly $O(n^2)$). However, using fewer points reduces the range of functions that can be accurately represented by the polynomials. As with the finite difference methods, the most efficient way to use pseudospectral methods is to start with a small number of points and then add more as the solution converges. For most of the path optimization problems tried in this research, only the first ten coefficients are significantly different from zero, so eliminating the remaining polynomials would lead to only a tiny error. However, increasing the number of points still increases the accuracy of the solution, because the differential equation is solved at more points along the path.

5.3 Stream Processing

One computational advantage of both forms of this algorithm is that they are easily adapted for stream processing. Stream processing is a form of vector processing that applies the same operation to many data items. Many graphics processors can implement stream processing with as many as 128 parallel processing units. This provides performance of up to 1000 billion floating point operations per second (GFLOPs). By contrast, traditional CPUs can only compute at a rate of about 25 GFLOPs per core, making the stream processors up to 40 times faster.

The algorithms described here are well suited to stream processing because the exact same operations are performed on every member of the data set (each point in the path). The finite difference algorithm only requires information from three points for each point's computations, while the pseudospectral method reduces to linear algebra operations that can be run on parallel execution units easily. Both algorithms require computing the Christoffel symbol values, which have no cross-dependencies between different points. Many other path optimization algorithms cannot be run fully in parallel, because they require information from other parts of the path. For example, shooting methods can only integrate from each point to the next point, so they cannot take advantage of parallel execution capabilities.

Chapter 6

Heisenberg Manifolds

Heisenberg manifolds are the simplest and most regular sub-Riemannian manifolds. This chapter provides a description of two Heisenberg manifolds that the geodesic search algorithm can be applied to. The first example has an analytic solution, so the numerical solutions can be checked for accuracy and the convergence properties can be determined. The second example is a little more complicated, but the math is still simple enough that the equations can be written out fully and still be understood, so it provides a good problem for explaining the algorithm in more detail. The geodesic search algorithm works well for these two examples, and it is likely that it will work well for any other Heisenberg manifold.

6.1 Definition

Heisenberg manifolds are an important category of sub-Riemannian manifolds. Heisenberg manifolds locally resemble the Heisenberg group, similar to how Riemannian manifolds are locally Euclidean. A sub-Riemannian manifold with a tangent space of dimension m and a total space of dimension $n = m + p$ is a Heisenberg manifold if it has the following properties:[9]

1. There are m locally defined vector fields $\{X_i\}$ such that the tangent space is fully spanned by $\{X_i\}$

2. The vector fields $(\{X_i\})$ are orthonormal

3. There are $p \geq 1$ locally defined 1-forms ω_α with $\omega_\alpha(X_i) = 0$, which satisfy the nonvanishing conditions $\mathrm{Det}\{\omega_\alpha([X_i, X_j])_{ij}\} \neq 0$

4. If local vector fields $\{X_i\}$ and $\{Y_i\}$ are defined on local charts, then both vector fields fully span the tangent space where the charts overlap

There are some other properties that can be derived from this definition. In particular, a Heisenberg manifold is step 2 everywhere and the dimension of the tangent space has to be even.[9]

One important consequence of the nonvanishing condition is that all Heisenberg manifolds satisfy the strong bracket generating condition.

6.2 3-Dimensional Heisenberg Group

The simplest Heisenberg manifold is the 3-dimensional Heisenberg group. This manifold has tangent vectors

$$\mathbf{a} = \partial_x - (y/2)\partial_t \tag{6.1}$$

$$\mathbf{b} = \partial_y + (x/2)\partial_t \tag{6.2}$$

The Lie Bracket of these two vectors

$$[\mathbf{a}, \mathbf{b}] = \partial_t \tag{6.3}$$

The Heisenberg manifold describes a path in the x, y plane with the t coordinate representing the area enclosed by the path (defined as $\int r^2 d\theta$ in polar coordinates). The path length is equal to the path length in the x, y plane. The geodesics are therefore the paths which enclose the desired total area (specified by the t coordinate) with a minimum perimeter (see Figure 6-1).

Applying the methodology of the previous chapters, we need to add a third vector to span the full space of the manifold. $\mathbf{c} = \partial_t$ is a suitable choice because it is not in the tangent space and is computationally easy to work with. $\mathbf{a}$ and $\mathbf{b}$ both have a length of 1, while $\mathbf{c}$ has a length of ∞ (which will be approximated as $1/\alpha$) because it represents a physically infeasible movement. The basis change matrices are

$$\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ y/2 & -x/2 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{t} \end{bmatrix} \tag{6.4}$$

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{t} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -y/2 & x/2 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \end{bmatrix} \tag{6.5}$$

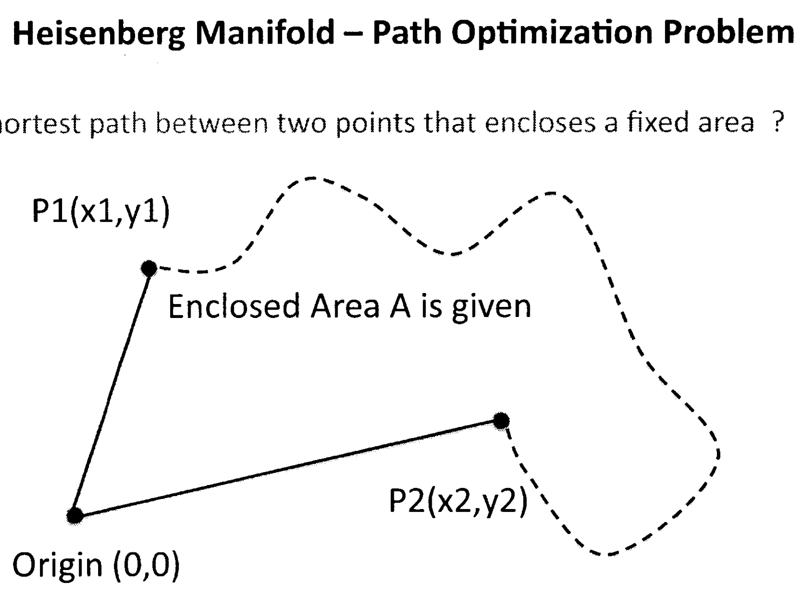

Heisenberg Manifold – Path Optimization Problem
Shortest path between two points that encloses a fixed area ?
P1(x1,y1)
Enclosed Area A is given
P2(x2,y2)
Origin (0,0)

Figure 6-1: The Heisenberg manifold path optimization problem. Find the shortest path that encloses a fixed area. The enclosed area goes from the origin to the initial point in a straight line, along the path to the final point, and then in a straight line to the origin.

So the metric in the (x, y, t) system is

$$g = \begin{bmatrix} 1 + y^2/4\alpha & -xy/4\alpha & y/2\alpha \\ -xy/4\alpha & 1 + x^2/4\alpha & -x/2\alpha \\ y/2\alpha & -x/2\alpha & 1/\alpha \end{bmatrix} \tag{6.6}$$

$$g^{-1} = \begin{bmatrix} 1 & 0 & -y/2 \\ 0 & 1 & x/2 \\ -y/2 & x/2 & (x^2 + y^2)/4 + \alpha \end{bmatrix} \tag{6.7}$$

The corresponding connection coefficients are:

$$\Gamma^x = \begin{bmatrix} 0 & y/4\alpha & 0 \\ y/4\alpha & -x/2\alpha & 1/2\alpha \\ 0 & 1/2\alpha & 0 \end{bmatrix} \tag{6.8}$$

$$\Gamma^y = \begin{bmatrix} -y/2\alpha & x/4\alpha & -1/2\alpha \\ x/4\alpha & 0 & 0 \\ -1/2\alpha & 0 & 0 \end{bmatrix} \tag{6.9}$$

$$\Gamma^t = \begin{bmatrix} -xy/4\alpha & (x^2-y^2)/8\alpha & -x/4\alpha \\ (x^2-y^2)/8\alpha & xy/4\alpha & -y/4\alpha \\ -x/4\alpha & -y/4\alpha & 0 \end{bmatrix} \tag{6.10}$$

The resulting geodesic equations are:

$$\frac{d^2x}{ds^2} + \frac{y}{2\alpha}\frac{dx}{ds}\frac{dy}{ds} - \frac{x}{2\alpha}\left(\frac{dy}{ds}\right)^2 + \frac{1}{\alpha}\frac{dy}{ds}\frac{dt}{ds} = 0 \tag{6.11}$$

$$\frac{d^2y}{ds^2} - \frac{y}{2\alpha}\left(\frac{dx}{ds}\right)^2 + \frac{x}{2\alpha}\frac{dx}{ds}\frac{dy}{ds} - \frac{1}{\alpha}\frac{dx}{ds}\frac{dt}{ds} = 0 \tag{6.12}$$

$$\frac{d^2t}{ds^2} - \frac{xy}{4\alpha}\left(\frac{dx}{ds}\right)^2 + \frac{x^2-y^2}{4\alpha}\frac{dx}{ds}\frac{dy}{ds} - \frac{x}{2\alpha}\frac{dx}{ds}\frac{dt}{ds} + \frac{xy}{4\alpha}\left(\frac{dy}{ds}\right)^2 - \frac{y}{2\alpha}\frac{dy}{ds}\frac{dt}{ds} = 0 \tag{6.13}$$

$$\frac{d^2t}{ds^2} - \frac{x}{2\alpha}\left(\frac{y}{2}\frac{dx}{ds} - \frac{x}{2}\frac{dy}{ds} + \frac{dt}{ds}\right)\frac{dx}{ds} - \frac{y}{2\alpha}\left(\frac{y}{2}\frac{dx}{ds} - \frac{x}{2}\frac{dy}{ds} + \frac{dt}{ds}\right)\frac{dy}{ds} = 0 \tag{6.14}$$

$$\frac{d^2x}{ds^2} + \frac{1}{\alpha}\left(\frac{y}{2}\frac{dx}{ds} - \frac{x}{2}\frac{dy}{ds} + \frac{dt}{ds}\right)\frac{dy}{ds} = 0 \tag{6.15}$$

$$\frac{d^2y}{ds^2} - \frac{1}{\alpha}\left(\frac{y}{2}\frac{dx}{ds} - \frac{x}{2}\frac{dy}{ds} + \frac{dt}{ds}\right)\frac{dx}{ds} = 0 \tag{6.16}$$

$$\frac{d^2t}{ds^2} - \frac{1}{2\alpha}\left(\frac{y}{2}\frac{dx}{ds} - \frac{x}{2}\frac{dy}{ds} + \frac{dt}{ds}\right)\left(x\frac{dx}{ds} + y\frac{dy}{ds}\right) = 0 \tag{6.17}$$

The solution to this system of equations projects to circles in the (x, y) plane. This is the expected result that a circle has the lowest perimeter of any curve that encloses a specified area. Using the algorithm described in the previous chapter produces the geodesic paths rapidly even when the initial conditions are randomized.

6.2.1 Results

The sample problem used for this section is the path from $(x, y) = (1, 0)$ to $(x, y) = (-1, 0)$ with the enclosed area equal to $pi/2$, which is a half circle centered at the origin with radius of 1. A few different algorithm variations were tried to compare how quickly they converge and to demonstrate the convergence properties of the method.

Figure 6-2 shows the way that a single point in the path moves as the algorithm converges. The value of α is decreased as the algorithm runs, which increases the penalty function and forces the path to approach the sub-Riemannian geodesic. At high values of α, the points are all moderately far away from the desired location, but they move closer as α is decreased. The graph shows that the points approach the expected location approximately as a linear function of α. This suggests a

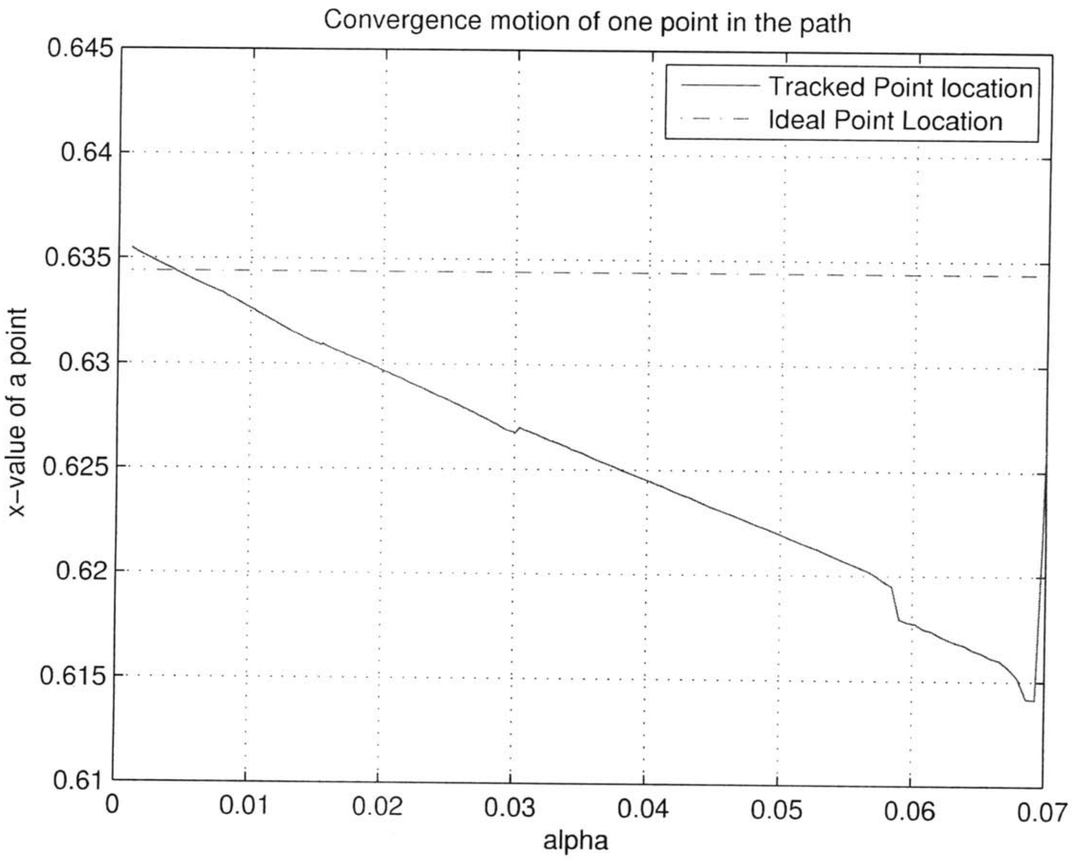


Figure 6-2: Movement of a single point in the path as the algorithm converges (α moves from .07 towards 0). The horizontal line shows the analytically determined correct location of the point. The "breaks" in the line are from adding more points to the path.

useful variation on the algorithm. At each step of the algorithm, as α is decreased, the algorithm can estimate how far each point will be moved based on how far each point has moved over the past several iterations.

Figure 6-3 compares the convergence rate of the basic algorithm to the convergence rate of the algorithm that extrapolates how far each point will move at each iteration step. Adding the extrapolation allows the algorithm to converge to a more accurate solution. Both algorithms are limited in how far they converge by the number of points used and the numerical limits of the computers calculating the paths.

Another variation based on extrapolating the position of each point allows stopping the algorithm early. Instead of actually trying to have the algorithm decrease α all the way to 0, the trajectory of each point can be tracked and a linear extrapolation can be used to predict where the point would end up if the algorithm were allowed to run all the way to $\alpha = 0$. Figure 6-4 shows how the convergence changes. In this case, the most recent 30 iterations are used to extrapolate the location

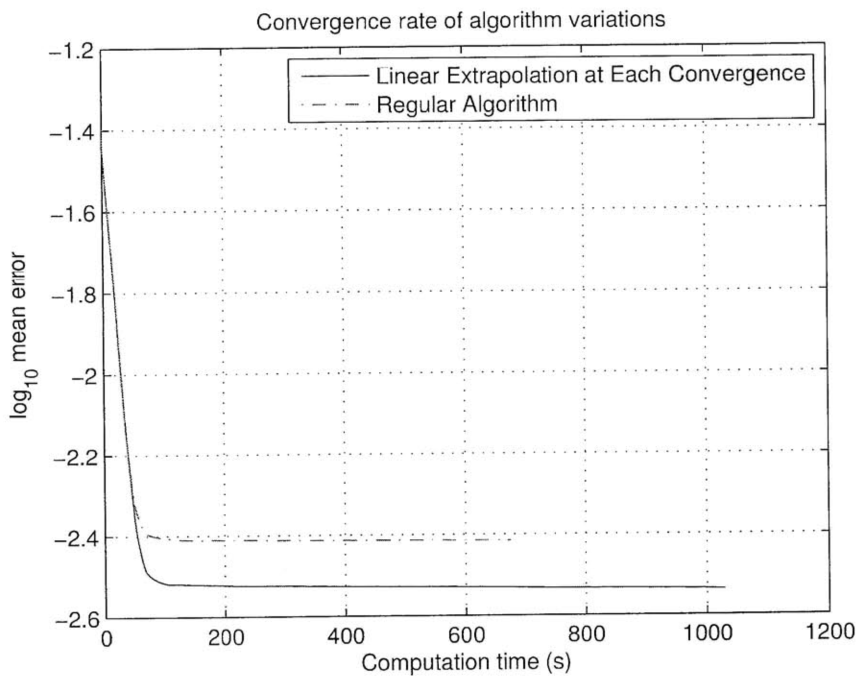


Figure 6-3: Convergence rate of basic algorithm and extroplated movement algorithm

of each point with $\alpha = 0$. The algorithm produces better results, but not in an entirely steady way. The extrapolation eventually converges to the same total error as not extrapolating, after having achieved a lower error if the algorithm had been stopped. In general, an analytical solution will not be available to compare the results to. While extrapolating the final point positions is likely to provide solutions much faster, more work needs to be done to determine how to tell which estimate of the final path is the best one to use.

6.3 Tank

The tank problem described earlier provides a good example of a path optimization problem that can be solved with this algorithm. The math for the tank problem is more interesting than the math for the Heisenberg manifold, but still simple enough to be understandable.

The coordinate system used to describe the problem is (x, y, θ), describing the position in the (x, y) plane and the current angle the tank is pointing in. The basis vectors are

$$\mathbf{a} = \cos\theta\partial_x + \sin\theta\partial_y \tag{6.18}$$

$$\mathbf{b} = \partial_\theta \tag{6.19}$$

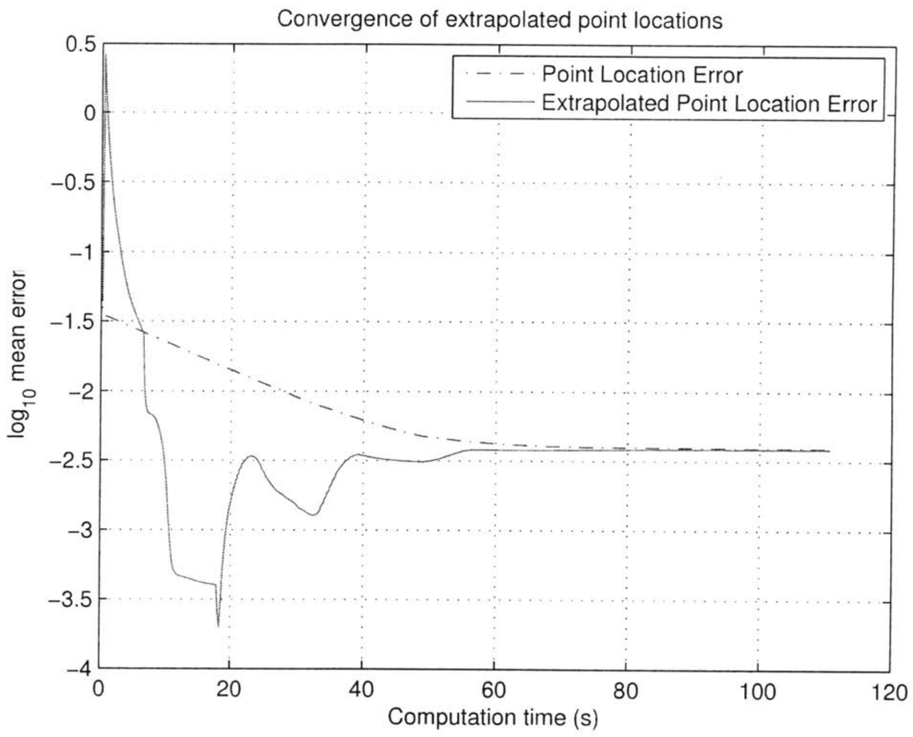


Figure 6-4: Convergence rate of the extrapolated movement algorithm with and without final point extrapolation

The bracket of these two vectors is

$$[\mathbf{a}, \mathbf{b}] = \sin\theta\partial_x - \cos\theta\partial_y \tag{6.20}$$

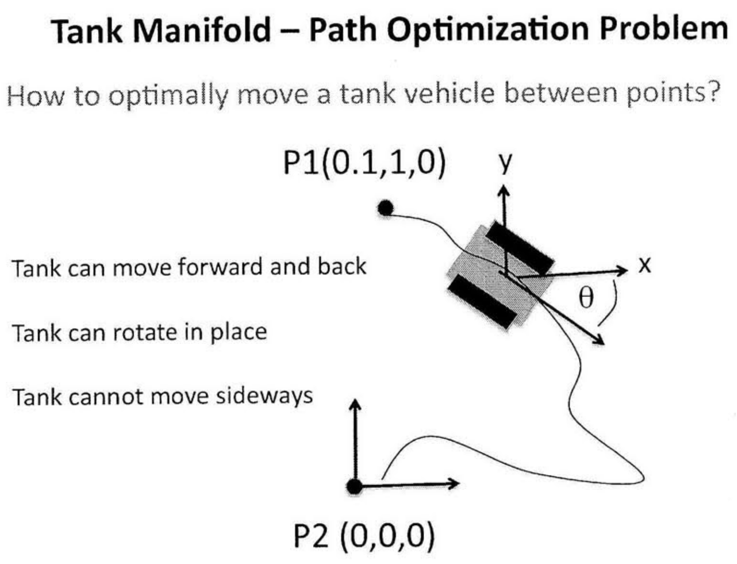
A third basis vector can be added to span the total space, such as $\mathbf{c} = -\sin\theta\partial_x + \cos\theta\partial_y$. $\mathbf{a}$ has a length of 1, $\mathbf{b}$ has a length of β, and $\mathbf{c}$ has a length of $1/\alpha$.

The metric can then be translated from $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ to $(\mathbf{x}, \mathbf{y}, \theta)$ with a basis change transformation. The matrix that changes the basis is

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ \theta \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \end{bmatrix} \tag{6.21}$$

The inverse transformation is

$$\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ \theta \end{bmatrix} \tag{6.22}$$



Tank Manifold – Path Optimization Problem
How to optimally move a tank vehicle between points?
P1(0.1,1,0)
y
x
θ
Tank can move forward and back
Tank can rotate in place
Tank cannot move sideways
P2 (0,0,0)

Figure 6-5: The tank manifold path optimization problem

This matrix also has the one form $\omega = -\sin\theta dx + \cos\theta dy$ that defines the distribution through the relation $\omega(X_i) = 0$. $\omega([\mathbf{a}, \mathbf{b}]) = -1$, so

$$\mathrm{Det}\omega([X_i, X_j]) = \mathrm{Det}\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} = -1 \tag{6.23}$$

which proves that this manifold satisfies the nonvanishing condition and is a Heisenberg manifold.

The metric in coordinates is

$$g = \begin{bmatrix} \cos^2\theta + (1/\alpha)\sin^2\theta & (1-1/\alpha)\cos\theta\sin\theta & 0 \\ (1-1/\alpha)\cos\theta\sin\theta & \sin^2\theta + (1/\alpha)\cos^2\theta & 0 \\ 0 & 0 & \beta \end{bmatrix} \tag{6.24}$$

$$g^{-1} = \begin{bmatrix} \cos^2\theta + \alpha\sin^2\theta & (1-\alpha)\cos\theta\sin\theta & 0 \\ (1-\alpha)\cos\theta\sin\theta & \sin^2\theta + \alpha\cos^2\theta & 0 \\ 0 & 0 & 1/\beta \end{bmatrix} \tag{6.25}$$

The corresponding connection coefficients are:

$$\begin{aligned}
\Gamma^x_{xx} &= 0 && (6.26)\\
\Gamma^x_{xy} &= 0 && (6.27)\\
\Gamma^x_{x\theta} &= (\cos^2\theta + \alpha\sin^2\theta)(1/\alpha - 1)\sin\theta\cos\theta - \frac{1-\alpha}{2}\cos\theta\sin\theta(1/\alpha - 1)(\cos^2\theta - \sin^2\theta)\\
&= \frac{1/\alpha - 1}{2}(\cos^2\theta + \sin^2\theta + \alpha\cos^2\theta + \alpha\sin^2\theta)\cos\theta\sin\theta\\
&= \frac{1/\alpha - \alpha}{2}\cos\theta\sin\theta && (6.28)\\
\Gamma^x_{yy} &= 0 && (6.29)\\
\Gamma^x_{y\theta} &= \frac{1}{2}(\cos^2\theta + \alpha\sin^2\theta)(1 - 1/\alpha)(\cos^2\theta - \sin^2\theta) + (1-\alpha)(1 - 1/\alpha)\cos^2\theta\sin^2\theta\\
&= \frac{1}{2}(\cos^2\theta - \alpha\sin^2\theta)(1 - 1/\alpha) && (6.30)\\
\Gamma^x_{\theta\theta} &= 0 && (6.31)\\
\Gamma^y_{xx} &= 0 && (6.32)\\
\Gamma^y_{xy} &= 0 && (6.33)\\
\Gamma^y_{x\theta} &= (1-\alpha)(1/\alpha - 1)\sin^2\theta\cos^2\theta - \frac{1}{2}(\sin^2\theta + \alpha\cos^2\theta)(1/\alpha - 1)(\cos^2\theta - \sin^2\theta)\\
&= \frac{1}{2}(1/\alpha - 1)(\sin^2\theta\cos^2\theta + \sin^4\theta - \alpha\sin^2\theta\cos^2\theta - \alpha\cos^4\theta)\\
&= \frac{1}{2}(1/\alpha - 1)(\sin^2\theta - \alpha\cos^2\theta) && (6.34)\\
\Gamma^y_{yy} &= 0 && (6.35)\\
\Gamma^y_{y\theta} &= \frac{1-\alpha}{2}(1 - 1/\alpha)(\cos^2\theta - \sin^2\theta)\cos\theta\sin\theta + (\sin^2\theta + \alpha\cos^2\theta)(1 - 1/\alpha)\cos\theta\sin\theta\\
&= \frac{(1+\alpha)\cos\theta\sin\theta}{2}(1 - 1/\alpha) && (6.36)\\
\Gamma^y_{\theta\theta} &= 0 && (6.37)\\
\Gamma^\theta_{xx} &= \frac{1 - 1/\alpha}{\beta}\cos\theta\sin\theta && (6.38)\\
\Gamma^\theta_{xy} &= \frac{1 - 1/\alpha}{2\beta}(\sin^2\theta - \cos^2\theta) && (6.39)\\
\Gamma^\theta_{x\theta} &= 0 && (6.40)\\
\Gamma^\theta_{yy} &= \frac{1/\alpha - 1}{\beta}\cos\theta\sin\theta && (6.41)\\
\Gamma^\theta_{y\theta} &= 0 && (6.42)\\
\Gamma^\theta_{\theta\theta} &= 0 && (6.43)
\end{aligned}$$

$$\Gamma^x = \begin{bmatrix} 0 & 0 & \frac{1/\alpha-\alpha}{2}\cos\theta\sin\theta \\ 0 & 0 & \frac{1-1/\alpha}{2}(\cos^2\theta-\alpha\sin^2\theta) \\ \frac{1/\alpha-\alpha}{2}\cos\theta\sin\theta & \frac{1-1/\alpha}{2}(\cos^2\theta-\alpha\sin^2\theta) & 0 \end{bmatrix} \tag{6.44}$$

$$\Gamma^y = \begin{bmatrix} 0 & 0 & \frac{1/\alpha-1}{2}(\sin^2\theta-\alpha\cos^2\theta) \\ 0 & 0 & \frac{\alpha-1/\alpha}{2}\cos\theta\sin\theta \\ \frac{1/\alpha-1}{2}(\sin^2\theta-\alpha\cos^2\theta) & \frac{\alpha-1/\alpha}{2}\cos\theta\sin\theta & 0 \end{bmatrix} \tag{6.45}$$

$$\Gamma^\theta = \begin{bmatrix} \frac{1-1/\alpha}{\beta}\cos\theta\sin\theta & \frac{1-1/\alpha}{2\beta}(\sin^2\theta-\cos^2\theta) & 0 \\ \frac{1-1/\alpha}{2\beta}(\sin^2\theta-\cos^2\theta) & \frac{1/\alpha-1}{\beta}\cos\theta\sin\theta & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{6.46}$$

6.3.1 Implementing the Algorithm

With the geometrical objects calculated, it is now possible to actually implement the algorithm. A pseudo-code description of the algorithm is given in Figure 6-6. The first step in the algorithm is setting up the conditions of the problem. Some initial guess of a reasonable path has to be provided for the algorithm to work with. A linear interpolation from the initial state to the final state works well. The initial value of α has to be set to a reasonable value, which can be determined by trying different values to see how well the algorithm converges. Starting with a high value of α will force the algorithm to go through extra iterations to reduce α more, while starting with too low of a value of α will prevent the algorithm from converging. For the tank, an initial value of $\alpha = 0.05$ was used. An initial value for the stiffness also has to be set, which should also be determined by trial and error. For the tank, an initial value of 10^4 was used.

With the initial problem setup completed, the main algorithm loop can be started. The first step in the loop is calculating all the derivatives. This can be done using either finite difference methods or pseudospectral methods (or any other method that can compute derivatives from a sequence of points). The first and second derivatives are put into the modified geodesic equation to calculate the virtual forces on each point.

$$F^i = \frac{d^2x^i}{ds^2} + \Gamma^i_{jk}\frac{dx^j}{ds}\frac{dx^k}{ds} \tag{6.47}$$

For example, the x direction force for the tank problem is

$$\begin{aligned} F^x &= \frac{d^2x}{ds^2} + 2\left[\frac{1/\alpha-\alpha}{2}\cos\theta\sin\theta\right]\frac{dx}{ds}\frac{d\theta}{ds} + 2\left[\frac{1}{2}(\cos^2\theta-\alpha\sin^2\theta)(1-1/\alpha)\right]\frac{dy}{ds}\frac{d\theta}{ds} & (6.48) \\ &= \frac{d^2x}{ds^2} + (1/\alpha-\alpha)\frac{\sin 2\theta}{2}\frac{dx}{ds}\frac{d\theta}{ds} + (\cos^2\theta-\alpha\sin^2\theta)(1-1/\alpha)\frac{dy}{ds}\frac{d\theta}{ds} & (6.49) \end{aligned}$$

The factor of two shows up because $(dx/ds)(d\theta/ds) = (d\theta/ds)(dx/ds)$ and the Christoffel symbols

```
%Setup initial values
X=initial path guess vector
alpha=0.05
stiff=10^4 %vector valued
%outer loop
while (penalty should be increased) and (path is still converging)
  dX=calculate derivatives vector
  dX2=calculate second derivatives vector
  G=calculate connection coefficients vector
  F=dX2+G*dX*dX %vector valued terms
  move=F/stiff %vector valued terms
  move=max(limit,move) %prevent any one point from moving too much
  X=X+move
  foreach point where sign(move)=sign(prev move)
    stiff=stiff*.95
  forall other points
    stiff=stiff*1.5
  end
  if sum(move)<1e-3
    save solution
    alpha=alpha*.95
  end
end
if path stopped converging
  use last saved converged path
else
  use last path
end
```

Figure 6-6: Pseudo-code description of the algorithm

are symmetric, so the two terms can be added together.

Once the virtual forces have been computed, the next step in the algorithm is to move all the points (except for the first and last points, which represent the unchanging boundary conditions of the problem). The previous movement is saved for comparison, and then the forces are divided by the stiffness, which is different for every point, to calculate the new movement. It is generally a good idea to limit how far each point can move at one step, to prevent minor numerical instabilities from causing the path to explode out and become divergent. For the tank, individual steps were limited to a movement of 0.01 in each direction (the distance between two consecutive points was not limited, just the actual movement from one iteration of the algorithm to the next).

After moving the points, the stiffness vectors are updated. Any point that moves in the same direction two iterations in a row will have its stiffness decreased, and any point that changes direction will have its stiffness increased. The stiffness is stored separately for each direction at each point, and each stiffness parameter is updated separately, so a point could have its x direction stiffness increased while its y direction stiffness is decreased. The stiffness parameters are multiplied by a factor just under one to decrease them and a factor over one to increase them. If these numbers are too high (such as 0.99 and 5) then the path will not move enough at each iteration and the convergence rate will be very slow. If the numbers are too low (such as 0.5 and 1.01) then the path will fail to converge. For the tank, the factors are 0.95 and 1.5. The stiffness parameters are also limited by a maximum and minimum value. As the path converges, the stiffness parameters will approach the maximum value. Once the value of α is decreased (a few steps later in the algorithm), these high stiffnesses have to be brought back down. If the maximum value is too high, then a large number of iterations will be required each time that α is reduced so that the path can be modified again. The maximum value for the tank is 10^{15}. The minimum value for the tank is 1.

The above steps are the main loop of the algorithm: calculate derivatives, calculate forces, move points, and update stiffness. This is the entire algorithm for a Riemannian manifold. However, with a sub-Riemannian manifold with a penalty metric, the penalty has to be increased. Each setting of the penalty provides a different Riemannian manifold that approximates the sub-Riemannian manifold. Once the path has converged enough, the penalty should be increased. The penalty term is $1/\alpha$, so it is increased by reducing α by multiplying it by a factor lower than 1. For the tank, this factor is 0.95. Once the penalty term is increased too far, the algorithm will no longer converge, so the solution should be saved before the penalty is increased.

In addition to increasing the penalty, it can be useful to increase the number of points used to represent the path. Using more points will lead to a slower convergence rate, but it also makes the algorithm more stable, especially with a higher penalty term. The most efficient way to balance these considerations is to start with a small number of points and then add points as the penalty is increased. That will provide rapid convergence with a small number of points initially and a more fully specified path with more stable convergence as the path converges to the solution. Additionally, using more points allows the path to converge to a more accurate solution.

6.3.2 Results

Putting these relations into the geodesic equation and using the algorithm previously described produces optimal trajectories for the control problem. Varying the value of β changes the cost of steering, with higher values making steering more expensive.

The effect of changing β is shown in Figure 6-7. For all trajectories, the tank begins pointing in the x direction at $x = 0.1$ and $y = 1$ and ends pointing the same way with x and y both equal to 0. The movement is nearly a sideways translation, but with some x movement to reduce the symmetry of the problem (if x started and ended at 0, then some solutions would start by increasing x and others would start by decreasing x, because neither direction would be naturally preferred). The lower values of β make translation more expensive relative to steering, so the trajectory uses sharp turns and moves the tank along a nearly vertical path. The higher values of β make steering more expensive, so the tank moves further in the x direction to avoid having to turn as far to obtain the necessary translation in the y direction.

In general, this sort of result is expected in control problems that have a tunable cost. There is a whole family of control problems represented by the same metric, with some parameter that determines which control problem is actually being solved. In the case of the tank problem, different vehicles could have differing costs for steering relative to moving. The math is fundamentally the same and the algorithm does not change for any of these problems. The solution changes as expected, by having more movement in the lower cost directions and less movement in the higher cost directions.

6.3.3 Curvature and Geodesic Uniqueness

The non-zero components of the Riemann tensor for the tank problem are:

$$
\begin{aligned}
R^x_{xxy} &= \frac{(\alpha-1)^3\sin(2\theta)}{8\alpha^2\beta} && (6.50)\\
R^x_{xyx} &= -R^x_{xxy} && (6.51)\\
R^x_{yxy} &= \frac{(1-1/\alpha)^2}{4\beta}(\alpha\sin^2\theta+\cos^2\theta) && (6.52)\\
R^x_{yyx} &= -R^x_{yxy} && (6.53)\\
R^x_{\theta x\theta} &= \frac{(\alpha^2-1)\cos(2\theta)}{2\alpha} - \frac{(\alpha-1)^2}{4\alpha} && (6.54)\\
R^x_{\theta\theta x} &= -R^x_{\theta x\theta} && (6.55)\\
R^x_{\theta y\theta} &= (\alpha-1/\alpha)\cos(t)\sin(t)) && (6.56)
\end{aligned}
$$

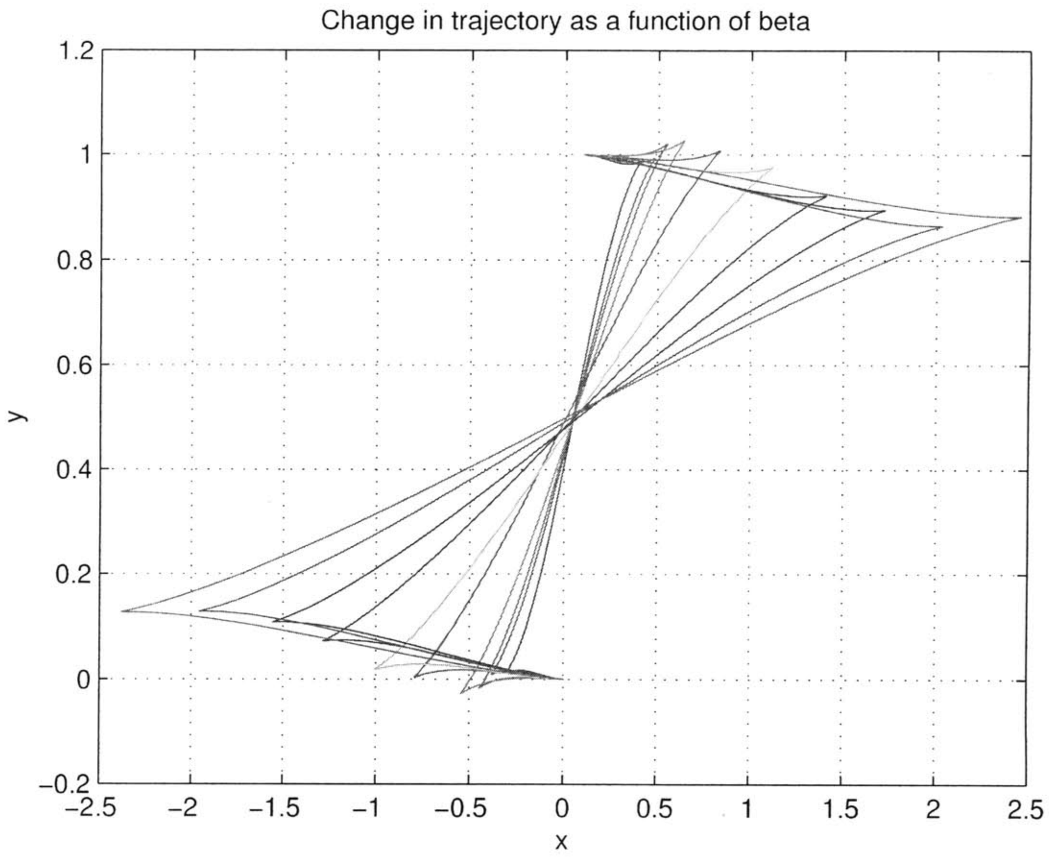


Figure 6-7: Trajectory Variation as a function of β. Higher values of β lead to more movement in the x-direction and shallower steering angles. The lowest β curve is the one closest to being vertical, the highest β curve is the one that moves farthest horizontally. β values here are equally spaced logarithmically from 0.2 to 200.

$$R^x_{\theta\theta y} = -R^x_{\theta y\theta} \tag{6.57}$$

$$R^y_{xyx} = ((-1+\alpha)^2(1+\alpha+(-1+\alpha)\cos(2t)))/(8\alpha^2\beta) \tag{6.58}$$

$$R^y_{xxy} = -R^y_{xyx} \tag{6.59}$$

$$R^y_{yyx} = \frac{(\alpha-1)^3\sin(2\theta)}{8\alpha^2\beta} = R^x_{xxy} \tag{6.60}$$

$$R^y_{yxy} = -R^y_{yxy} \tag{6.61}$$

$$R^y_{\theta x\theta} = (\alpha - 1/\alpha)\cos(t)\sin(t)) = R^x_{\theta y\theta} \tag{6.62}$$

$$R^y_{\theta y\theta} = -((-1+\alpha)(-1+\alpha+2(1+\alpha)\cos(2t)))/(4\alpha) \tag{6.63}$$

$$R^y_{\theta\theta x} = -R^y_{\theta x\theta} \tag{6.64}$$

$$R^y_{\theta\theta y} = ((-1+\alpha)(-1+\alpha+2(1+\alpha)\cos(2t)))/(4\alpha) \tag{6.65}$$

$$R^\theta_{xx\theta} = -((-1+\alpha)(-1+\alpha^2+(1+6\alpha+\alpha^2)\cos(2t)))/(8\alpha^2\beta) \tag{6.66}$$

$$
\begin{aligned}
R^{\theta}_{xy\theta} &= -((-1+\alpha)(1+6\alpha+\alpha^2)\sin(2t))/(8\alpha^2\beta) && (6.67)\\
R^{\theta}_{x\theta x} &= ((-1+\alpha)(-1+\alpha^2+(1+6\alpha+\alpha^2)\cos(2t)))/(8\alpha^2\beta) && (6.68)\\
R^{\theta}_{x\theta y} &= ((-1+\alpha)(1+6\alpha+\alpha^2)\sin(2t))/(8\alpha^2\beta) && (6.69)\\
R^{\theta}_{yx\theta} &= -((-1+\alpha)(1+6\alpha+\alpha^2)\sin(2t))/(8\alpha^2\beta) && (6.70)\\
R^{\theta}_{yy\theta} &= ((-1+\alpha)(1-\alpha^2+(1+6\alpha+\alpha^2)\cos(2t)))/(8\alpha^2\beta) && (6.71)\\
R^{\theta}_{y\theta x} &= ((-1+\alpha)(1+6\alpha+\alpha^2)\sin(2t))/(8\alpha^2\beta) && (6.72)\\
R^{\theta}_{y\theta y} &= -((-1+\alpha)(1-\alpha^2+(1+6\alpha+\alpha^2)\cos(2t)))/(8\alpha^2\beta) && (6.73)
\end{aligned}
$$

This shows some of the many symmetries of the Riemann tensor, such as the anti-symmetry on the last two components ($R^a_{bcd} = -R^a_{bdc}$). In this particular problem, all the components of the Riemann tensor are of the form $a + b\sin 2\theta$ or $a + b\cos 2\theta$, but this would not be expected to happen for general control problems.

With the Riemann tensor calculated, it is possible to integrate the Jacobi equation:

$$\frac{d^2J}{ds^2} = -R(T,J)T \qquad (6.74)$$

where J is the Jacobi vector being integrated and T is the tangent vector to the geodesic that defines the integration path. In components, this equation is

$$\frac{d^2J^i}{ds^2} = -R^i_{jkl}T^jJ^kT^l \qquad (6.75)$$

with implied summation over j, k, and l. For example, the x component of this equation (leaving out the 0's in the Riemann tensor and taking advantage of symmetry) is

$$
\begin{aligned}
\frac{d^2J^x}{ds^2} &= -R^x_{xxy}T^x(J^xT^y - J^yT^x) - R^x_{yxy}T^y(J^xT^y - J^yT_x)\\
&\quad -R^x_{\theta x\theta}T^\theta(J^xT^\theta - J^\theta T^x) - R^x_{\theta y\theta}T^\theta(J^yT^\theta - J^\theta T^y)\\
&= -(R^x_{xxy}T^x + R^x_{yxy}T^y)(J^xT^y - J^yT^x)\\
&\quad -R^x_{\theta x\theta}T^\theta(J^xT^\theta - J^\theta T^x) - R^x_{\theta y\theta}T^\theta(J^yT^\theta - J^\theta T^y) \qquad (6.76)
\end{aligned}
$$

Integrating this equation provides a Jacobi vector field along the geodesic. The Jacobi field provides information about how two nearby geodesics will evolve relative to each other. If two geodesics start at the same point, but move in different directions, then a Jacobi field can be used to determine whether they intersect at a second point or not. The Jacobi vector (J) is set to 0 at the initial point, with the initial value of its first derivative equal to the difference between the tangent vectors of the two geodesics. If the Jacobi field is ever 0 again, it means that the geodesics will intersect again at the point where the Jacobi vector is 0. Since both paths are geodesics, either one could be found by the algorithm. They are both local minima in the path optimization problem, but in general one of them will be shorter, so they need to be checked to determine which one is the global minimum.

Points along a geodesic where a Jacobi field is 0 are called conjugate points. A geodesic with no conjugate points cannot intersect another geodesic in two points, which means that it is a global minimum as well as a local minimum, at least within the region where the Jacobi equation accurately describes the deviation between nearby geodesics - which is the region where the curvature tensor does not vary too much.

A geodesic is therefore globally unique (or at least unique in a region) if there is no Jacobi field which is equal to 0 at two points along the geodesic. Because the Jacobi equation is a linear function of the Jacobi vector, different solutions can be combined linearly to find new solutions. With an initial condition of $J(0) = 0$, there are $n - 1$ linearly independent values for $J'(0)$ because using $J'(0) = T$ will produce a Jacobi field that represents the null divergence between a geodesic and itself. Integrating these $n - 1$ Jacobi fields will produce all the information necessary to find every Jacobi field along the geodesic. If these $n - 1$ Jacobi fields remain linearly independent along the entire geodesic, then the geodesic is unique. If at some point, the Jacobi fields are not linearly independent, then the geodesic is not unique and the combination of Jacobi fields which produces a conjugate point provides information about how to find an alternate geodesic connecting the same end points.

Figure 6-8 shows a graph of the magnitude of the Jacobi vectors along a geodesic in the tank manifold (corresponding to the same initial and final positions as in section 6.3.2 with $\beta = 5$). The vectors grow exponentially, showing that once two geodesics start to diverge, the distance between them will always grow larger.

Once the Jacobi vectors have been computed, there are a few ways to check for linear independence. In this case, the check used was to calculate the dot product of the vectors and use that to compute the angle between them. This works well when there are only two vector fields, but may not work as well with more vectors (each pair has to be checked separately). The Gramian matrix can be used with more vectors. The Gramian matrix is defined as

$$G_{ij} = \langle J_i, J_j \rangle \tag{6.77}$$

which means that the dot product of the i-th and j-th vectors is at the (i, j) location in the matrix. This will be a symmetric matrix. If the vectors are not linearly independent, then the determinant of the matrix will be zero. It will generally be worth checking anywhere along the path where the determinant is very small to see if the Jacobi vectors will actually produce a second geodesic.

Whatever method is used to check for linear independence has to be done at every point along the path. If the path as found is defined by only a small number of points, then an interpolating

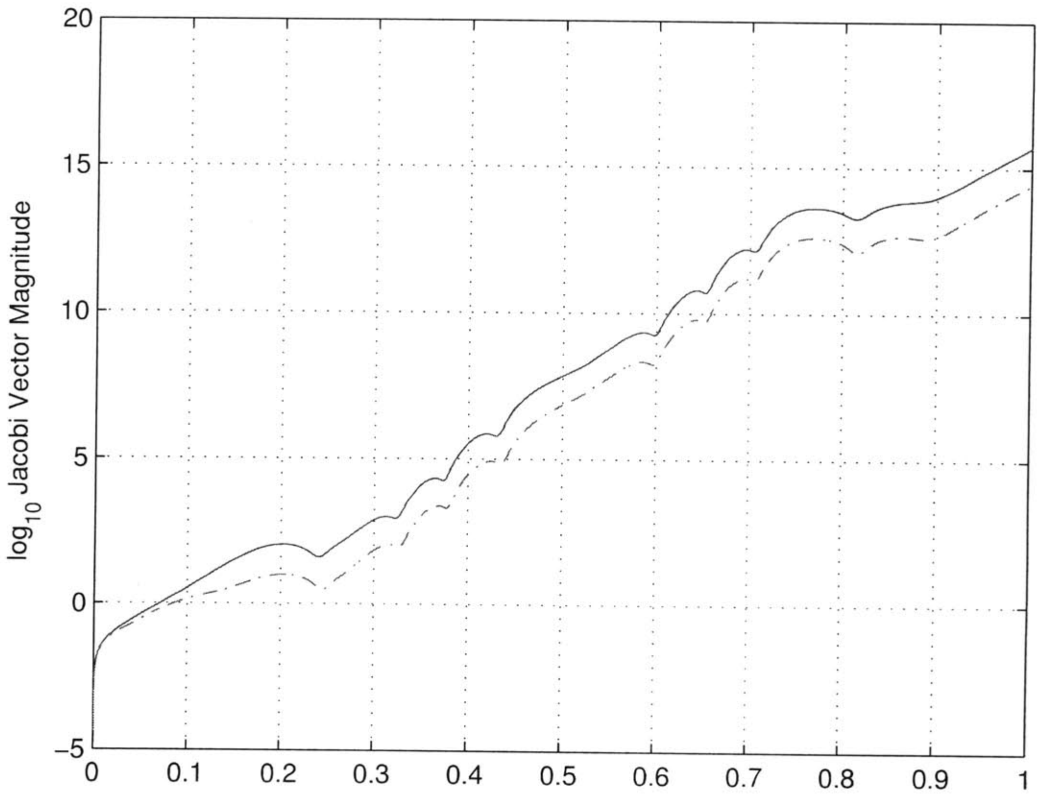


Figure 6-8: $\log_{10}$ magnitude of the 2 Jacobi vector fields for a geodesic of the tank manifold with $\beta = 5$

function should be used to determine that path at more points to increase the numerical accuracy of the integration. Otherwise, the integration may not be accurate enough to verify that the vectors remain linearly independent and the vectors will not even be properly checked except at a few points along the path. For the example problem, an interpolation function was used to provide 100,000 points along the path from the 30 points calculated in the pseudospectral method.

The angle between the Jacobi vectors for the example problem (shown in figure 6-9) decreases to as little as half a degree. Given that the Jacobi field is only completely accurate if the curvature is constant, it is worth checking to see if there is actually a geodesic which intersects the original geodesic a second time. This will also provide a detailed example of how to check for uniqueness and what to do if the Jacobi field condition is not satisfied.

If a point is found where the vectors are not linearly independent, then the linear combination of vectors which will produce a zero vector has to be calculated. In the example problem, this was calculated to be $J_1 - 18.23J_2$. The initial derivatives of this sum of vectors $(dJ_1(0) - 18.23dJ_2(0))$ was then used as the tangent vector for a path. That path was then integrated forward and compared

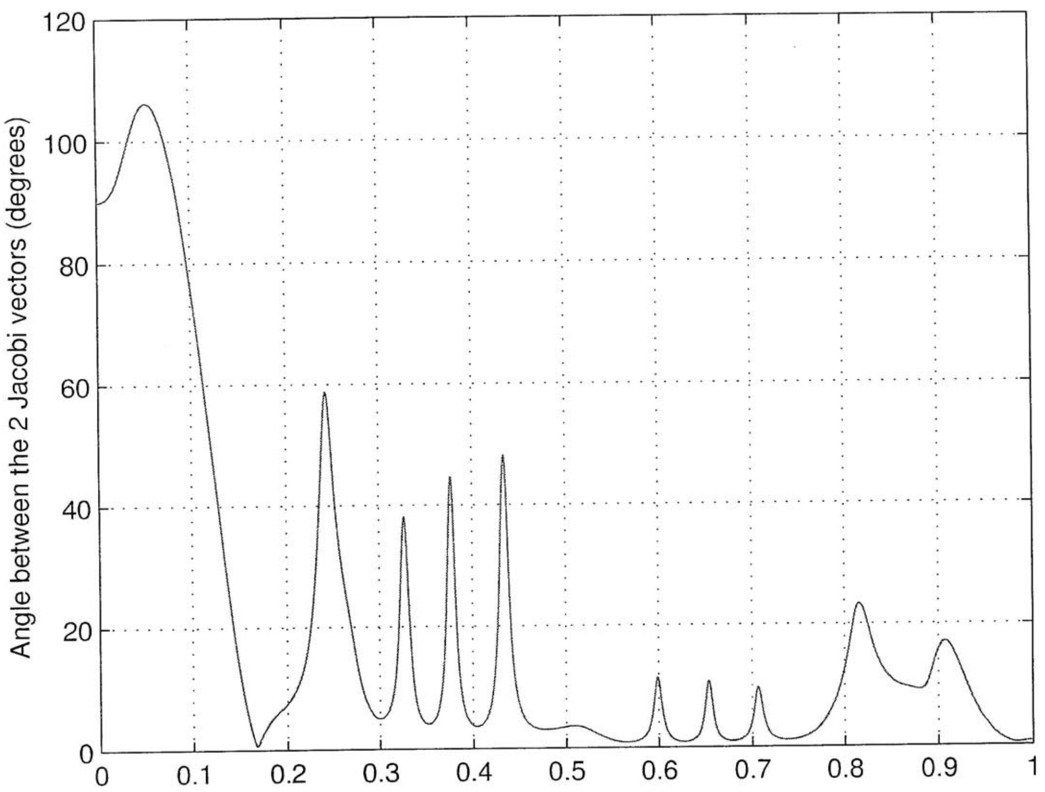


Figure 6-9: Angle between the 2 Jacobi vector fields for a geodesic of the tank manifold

to the original geodesic.

The second geodesic does not quite intersect the original geodesic, however it does pass close to the same endpoint. Both geodesics are plotted in figure 6-10. The original geodesic is a z-shaped path, while the second geodesic has a large amount of moving back and forth to produce a path that is almost a sideways translation (shown in figure 6-11). The large amount of back and forth translation leads to a higher overall cost by a significant amount - the total cost of the original geodesic is 27.308 while the total cost of the second geodesic is 37,283, over one thousand times as high. While it is probably possible to modify this geodesic so that it meets the original one at the endpoint, it is clearly going to still have a higher cost. This means that the original geodesic can be taken as a global minimum.

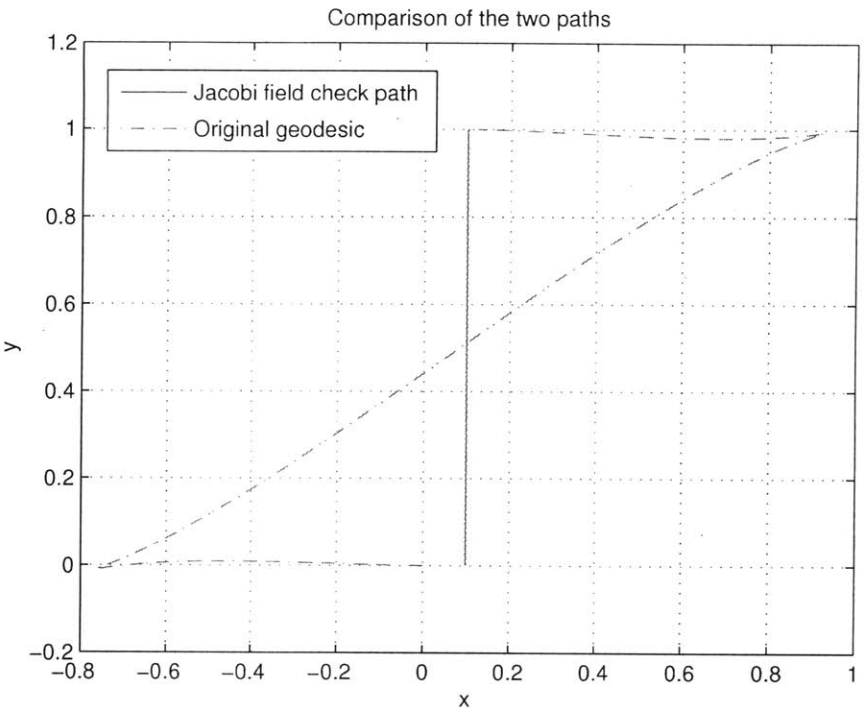


Figure 6-10: Comparison of the original geodesic and the geodesic found by checking the Jacobi condition

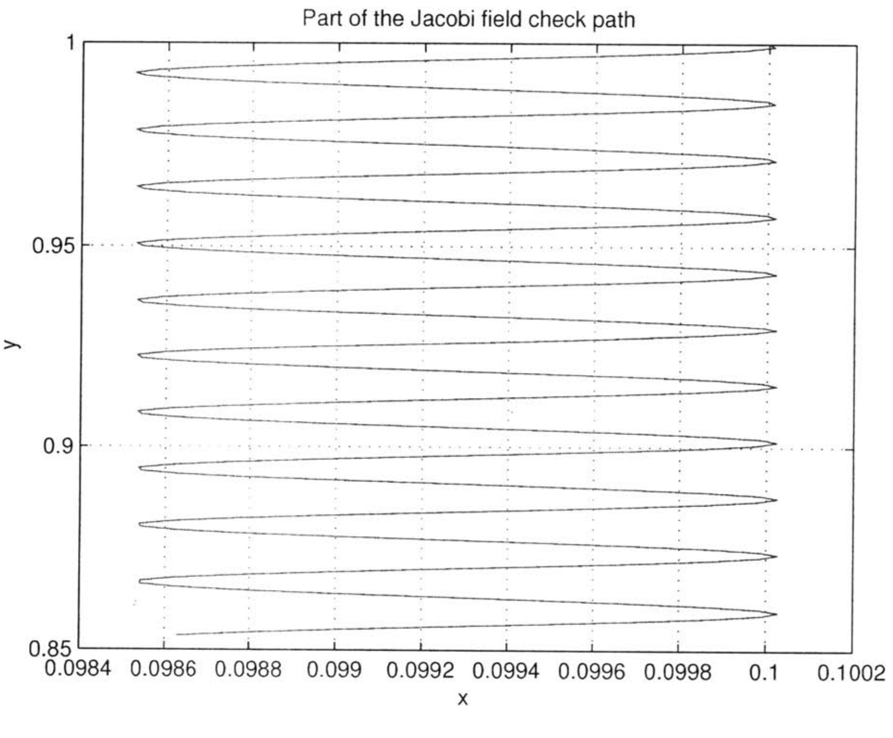


Figure 6-11: Close-up view of the geodesic found from the Jacobi condition, showing the very tight back and forth movements that allow the tank to move nearly sideways

6.4 Summary

Heisenberg manifolds are the simplest type of sub-Riemannian manifolds. They represent some path optimization problems. The algorithm described in chapters 4 and 5 works with no difficulties on Heisenberg manifolds to quickly find optimal trajectories. Two example Heisenberg manifolds were analyzed to demonstrate the accuracy and convergence properties of the method. The examples were worked through in detail to provide a description of the algorithm.

Chapter 7

The Astrodynamics Manifold

The astrodynamics manifold is the mathematical representation of all possible orbits, with each orbit represented by a line (each position within an orbit is represented by a point). There are actually many different astrodynamics manifolds, with each one corresponding to a representation of the orbits around some real central body with a non-ideal gravitational field. Assuming two-body Newtonian gravity with point masses produces the simplest astrodynamics manifold which is a good approximation to the real astrodynamics manifold for most planets and stars. Restricting the orbits to a plane provides a further simplification of the manifold.

7.1 Coordinates and Frames

In order to calculate any numerical properties, coordinates and a moving frame have to be selected for the astrodynamics manifold. The coordinates are derived from the orbital elements described below. The moving frame is based on the physical movements in space - thrusting in 2 perpendicular directions, coasting along the orbit, and moving radially. The basis vector that corresponds to coasting includes both the position and velocity changes such that none of the orbital elements change except for the angular position.

For the purposes of describing motions and orbital elements in real space, the directions are:

x	the standard x direction
y	the standard y direction
r	the radial direction, positive outward
t	the tangential direction, positive in the direction of orbital motion

The orbital elements to be used are

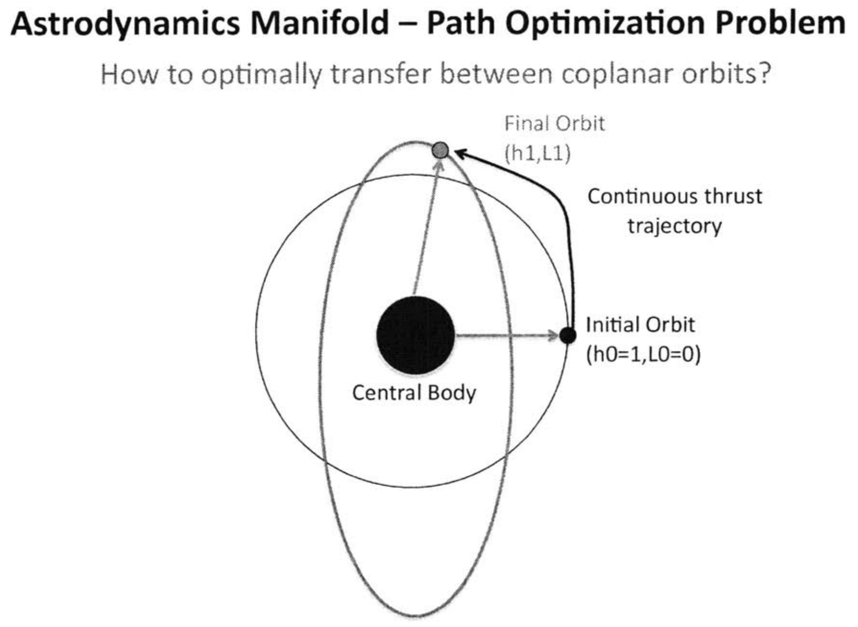

Astrodynamics Manifold – Path Optimization Problem
How to optimally transfer between coplanar orbits?
Final Orbit
(h1,L1)
Continuous thrust trajectory
Initial Orbit
(h0=1,L0=0)
Central Body

Figure 7-1: The astrodynamics manifold path optimization problem. This is for a coplanar orbital transfer using a two-body Newtonian gravitational model with point masses.

h	angular momentum
e_r	radial component of the eccentricity vector
e_t	tangential component of the eccentricity vector
e_x	x-component of the eccentricity vector
e_y	y-component of the eccentricity vector
L	true longitude

The basis vectors are

da_r	radial thrusting
da_t	tangential thrusting
da_x	x-direction thrusting
da_y	y-direction thrusting
dt	natural movement along the orbit
dr	radially outward translation with no velocity change
ξ	$= 1 + e_r = 1 + e_x \cos L + e_y \sin L$

In terms of position and velocity, the orbital elements are

$$h = rv_t \tag{7.1}$$

$$\begin{aligned}
e_r &= r v_t^2 - 1 && (7.2)\\
e_t &= -r v_r v_t && (7.3)\\
e_x &= r_x v_y^2 - r_y v_x v_y - \frac{r_x}{r} && (7.4)\\
e_y &= r_y v_x^2 - r_x v_x v_y - \frac{r_y}{r} && (7.5)\\
L &= \text{atan}\frac{r_y}{r_x} && (7.6)
\end{aligned}$$

where the atan function is one which gives the correct four-quadrant angle. The simpler form of the eccentricity components in radial/tangential form is the reason those components are being used initially.

The derivatives of these elements are:

$$\begin{aligned}
\frac{dh}{da_r} &= 0 && (7.7)\\
\frac{dh}{da_t} &= r && (7.8)\\
\frac{dh}{dt} &= 0 && (7.9)\\
\frac{dh}{dr} &= v_t && (7.10)
\end{aligned}$$

For the next two derivations, I am going to abuse the notation and pretend that $de_r/dt = 0$ without thrusting, even though the rotating frame causes e_r and e_t to vary cyclically. I am actually interested in calculating e_x and e_y. which do not have this cyclical variation, but the differential equations for e_r and e_t are easier to derive.

$$\begin{aligned}
\frac{de_r}{da_r} &= 0 && (7.11)\\
\frac{de_r}{da_t} &= 2 r v_t && (7.12)\\
\frac{de_r}{dt} &= 0 && (7.13)\\
\frac{de_r}{dr} &= v_t^2 && (7.14)
\end{aligned}$$

$$\begin{aligned}
\frac{de_t}{da_r} &= -r v_t && (7.15)\\
\frac{de_t}{da_t} &= -r v_r && (7.16)\\
\frac{de_t}{dt} &= 0 && (7.17)\\
\frac{de_t}{dr} &= -v_r v_t && (7.18)
\end{aligned}$$

$$\frac{dL}{da_r} = 0 \tag{7.19}$$
$$\frac{dL}{da_t} = 0 \tag{7.20}$$
$$\frac{dL}{dt} = \frac{\xi^2}{h^3} \tag{7.21}$$
$$\frac{dL}{dr} = 0 \tag{7.22}$$

Translating these into orbital elements:

$$r = \frac{h^2}{\xi} \tag{7.23}$$
$$v_r = \frac{-e_t}{h} \tag{7.24}$$
$$v_t = \frac{\xi}{h} \tag{7.25}$$

$$\frac{dh}{da_r} = 0 \tag{7.26}$$
$$\frac{dh}{da_t} = \frac{h^2}{\xi} \tag{7.27}$$
$$\frac{dh}{dt} = 0 \tag{7.28}$$
$$\frac{dh}{dr} = \frac{\xi}{h} \tag{7.29}$$

$$\frac{de_r}{da_r} = 0 \tag{7.30}$$
$$\frac{de_r}{da_t} = 2h \tag{7.31}$$
$$\frac{de_r}{dt} = 0 \tag{7.32}$$
$$\frac{de_r}{dr} = \frac{\xi^2}{h^2} \tag{7.33}$$

$$\frac{de_t}{da_r} = -h \tag{7.34}$$
$$\frac{de_t}{da_t} = \frac{he_t}{\xi} \tag{7.35}$$
$$\frac{de_t}{dt} = 0 \tag{7.36}$$

$$\frac{de_t}{dr} = \frac{e_t\xi}{h^2} \tag{7.37}$$

$$\frac{dL}{da_r} = 0 \tag{7.38}$$

$$\frac{dL}{da_r} = 0 \tag{7.39}$$

$$\frac{dL}{dt} = \frac{\xi^2}{h^3} \tag{7.40}$$

$$\frac{dL}{dr} = 0 \tag{7.41}$$

The metric from the velocity-position basis vectors is

$$g_{rv} = diag(1, 1, 1/\beta, \alpha) \tag{7.42}$$

where α and β are parameters that describes how close to the true astrodynamics manifold the metric is. The correct values are both 0, but higher values allow the computations to be performed. If the limit of some quantity exists as α and β approach 0, then that limit is the correct value of that quantity for the astrodynamics manifold.

7.2 Metric and Cometric

The metric can then be translated from (v_r, v_t, r, t) to (h, e_r, e_t, L) with a basis change transformation. The matrix that changes the basis is

$$\begin{bmatrix} dh \\ de_r \\ de_t \\ dL \end{bmatrix} = \begin{bmatrix} 0 & h^2/\xi & \xi/h & 0 \\ 0 & 2h & \xi^2/h^2 & 0 \\ -h & he_t/\xi & e_t\xi/h^2 & 0 \\ 0 & 0 & 0 & \xi^2/h^3 \end{bmatrix} \begin{bmatrix} da_r \\ da_t \\ dr \\ dt \end{bmatrix} \tag{7.43}$$

The inverse transformation is

$$\begin{bmatrix} da_r \\ da_t \\ dr \\ dt \end{bmatrix} = \begin{bmatrix} e_t/h^2 & 0 & -1/h & 0 \\ -\xi/h^2 & 1/h & 0 & 0 \\ 2h/\xi & -h^2/\xi^2 & 0 & 0 \\ 0 & 0 & 0 & h^3/\xi^2 \end{bmatrix} \begin{bmatrix} dh \\ de_r \\ de_t \\ dL \end{bmatrix} \tag{7.44}$$

It is more useful to use the x and y components of the eccentricity vector because the r and t components change as a function of L.

$$\begin{bmatrix} dh \\ de_x \\ de_y \\ dL \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos L & -\sin L & 0 \\ 0 & \sin L & \cos L & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} dh \\ de_r \\ de_t \\ dL \end{bmatrix} \tag{7.45}$$

$$\begin{bmatrix} dh \\ de_r \\ de_t \\ dL \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos L & \sin L & 0 \\ 0 & -\sin L & \cos L & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} dh \\ de_x \\ de_y \\ dL \end{bmatrix} \tag{7.46}$$

It is useful to define the terms $E_x = e_x + \cos L$ and $E_y = e_y + \sin L$

$$\begin{bmatrix} dh \\ de_x \\ de_y \\ dL \end{bmatrix} = \begin{bmatrix} 0 & h^2/\xi & \xi/h & 0 \\ h\sin L & h\cos L + hE_x/\xi & \xi E_x/h^2 & 0 \\ -h\cos L & h\sin L + hE_y/\xi & \xi E_y/h^2 & 0 \\ 0 & 0 & 0 & \xi^2/h^3 \end{bmatrix} \begin{bmatrix} da_r \\ da_t \\ dr \\ dt \end{bmatrix} \tag{7.47}$$

$$\begin{bmatrix} da_r \\ da_t \\ dr \\ dt \end{bmatrix} = \begin{bmatrix} e_t/h^2 & (\sin L)/h & -(\cos L)/h & 0 \\ -\xi/h^2 & (\cos L)/h & (\sin L)/h & 0 \\ 2h/\xi & -(h^2\cos L)/\xi^2 & -(h^2\sin L)/\xi^2 & 0 \\ 0 & 0 & 0 & h^3/\xi^2 \end{bmatrix} \begin{bmatrix} dh \\ de_x \\ de_y \\ dL \end{bmatrix} \tag{7.48}$$

These matrices allow converting numerical representations of vectors from one basis to the other. Since the metric for one set of basis vectors is known, the metric for the other set can be calculated. Converting each of the basis vectors in orbital elements to their numerical representation in physical movements allows their dot products to be determined.

Using the transformation of basis matrix, the metric in orbital elements coordinates is

$$g = \begin{bmatrix} \frac{e_t^2+\xi^2}{h^4} + \frac{4h^2}{\xi^2\beta} & \frac{-E_x}{h^3} - \frac{2h^3\cos L}{\xi^3\beta} & \frac{-E_y}{h^3} - \frac{2h^3\sin L}{\xi^3\beta} & 0 \\ \frac{-E_x}{h^3} - \frac{2h^3\cos L}{\xi^3\beta} & \frac{1}{h^2} + \frac{h^4\cos^2 L}{\xi^4\beta} & \frac{h^4\cos L\sin L}{\xi^4\beta} & 0 \\ \frac{-E_y}{h^3} - \frac{2h^3\sin L}{\xi^3\beta} & \frac{h^4\cos L\sin L}{\xi^4\beta} & \frac{1}{h^2} + \frac{h^4\sin^2 L}{\xi^4\beta} & 0 \\ 0 & 0 & 0 & \frac{h^6\alpha}{\xi^4} \end{bmatrix} \tag{7.49}$$

$$g = \begin{bmatrix} (e_t^2+\xi^2)/h^4 & -E_x/h^3 & -E_y/h^3 & 0 \\ -E_x/h^3 & 1/h^2 & 0 & 0 \\ -E_y/h^3 & 0 & 1/h^2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$+ \begin{bmatrix} 4h^2/\xi^2 & -2h^3\cos L/\xi^3 & -2h^3\sin L/\xi^3 & 0 \\ -2h^3\cos L/\xi^3 & h^4\cos^2 L/\xi^4 & h^4\cos L\sin L/\xi^4 & 0 \\ -2h^3\sin L/\xi^3 & h^4\cos L\sin L/\xi^4 & h^4\sin^2 L/\xi^4 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \frac{1}{\beta}$$

$$+ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & h^6/\xi^4 \end{bmatrix} \alpha \tag{7.50}$$

$$g^{-1} = \begin{bmatrix} \frac{h^4}{\xi^2} + \frac{\beta\xi^2}{h^2} & \frac{h^3}{\xi^2}(E_x + \xi\cos L) + \frac{\beta\xi^2 E_x}{h^3} & \frac{h^3}{\xi^2}(E_y + \xi\sin L) + \frac{\beta\xi^2 E_y}{h^3} & 0 \\ \frac{h^3}{\xi^2}(E_x + \xi\cos L) + \frac{\beta\xi^2 E_x}{h^3} & h^2 + \frac{2h^2 E_x\cos L}{\xi} + \frac{h^2 E_x^2}{\xi^2} + \frac{\beta\xi^2 E_x^2}{h^4} & \frac{h^2[\xi(E_y\cos L + E_x\sin L) + E_x E_y]}{\xi^2} + \frac{\beta\xi^2 E_x E_y}{h^4} & 0 \\ \frac{h^3}{\xi^2}(E_y + \xi\sin L) + \frac{\beta\xi^2 E_y}{h^3} & \frac{h^2[\xi(E_y\cos L + E_x\sin L) + E_x E_y]}{\xi^2} + \frac{\beta\xi^2 E_x E_y}{h^4} & h^2 + \frac{2h^2 E_y\sin L}{\xi} + \frac{h^2 E_y^2}{\xi^2} + \frac{\beta\xi^2 E_y^2}{h^4} & 0 \\ 0 & 0 & 0 & \frac{\xi^4}{h^6\alpha} \end{bmatrix} \tag{7.51}$$

$$g^{-1} = \frac{h^2}{\xi^2} \begin{bmatrix} h^2 & h(E_x + \xi\cos L) & h(E_y + \xi\sin L) & 0 \\ h(E_x + \xi\cos L) & \xi^2 + 2E_x\xi\cos L + E_x^2 & \xi(E_y\cos L + E_x\sin L) + E_x E_y & 0 \\ h(E_y + \xi\sin L) & \xi(E_y\cos L + E_x\sin L) + E_x E_y & \xi^2 + 2E_y\xi\sin L + E_y^2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$+ \frac{\beta\xi^2}{h^4} \begin{bmatrix} h^2 & hE_x & hE_y & 0 \\ hE_x & E_x^2 & E_x E_y & 0 \\ hE_y & E_x E_y & E_y^2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\xi^4}{h^6\alpha} \end{bmatrix} \tag{7.52}$$

7.3 Christoffel Symbols

The corresponding connection coefficients are:

$$\Gamma^h_{hh} = \frac{4}{h} \tag{7.53}$$

$$\Gamma^h_{hx} = \frac{-3\cos L}{\xi} \tag{7.54}$$

$$\Gamma^h_{hy} = \frac{-3\sin L}{\xi} \tag{7.55}$$

$$\Gamma^h_{hL} = \frac{h^6 e_t}{\beta\xi^5} - \frac{2e_t}{\xi} + \frac{e_t}{2\xi^2} + \frac{\beta\xi^2 e_t}{2h^6} \tag{7.56}$$

$$\Gamma^h_{xx} = \frac{2h\cos^2 L}{\xi^2} \tag{7.57}$$

$$\Gamma^h_{xy} = \frac{h\sin 2L}{\xi^2} \tag{7.58}$$

$$\Gamma^h_{xL} = \frac{h}{2\xi^2}(e_y + 2\sin L + 2e_t\cos L) + \frac{\beta\xi^2\sin L}{2h^5} - \frac{h^7 e_t\cos L}{2\xi^6\beta} \tag{7.59}$$

$$\Gamma^h_{yy} = \frac{2h\sin^2 L}{\xi^2} \tag{7.60}$$

$$\Gamma^h_{yL} = \frac{-h^7 e_t\sin L}{2\beta\xi^6} + \frac{h(e_x\cos 2L + e_y\sin 2L - 2E_x)}{2\xi^2} - \frac{\beta\xi^2\cos L}{2h^5} \tag{7.61}$$

$$\Gamma^h_{LL} = \frac{h^9\alpha}{\xi^6} - \frac{h^3\alpha\beta}{\xi^2} \tag{7.62}$$

$$\Gamma^x_{hh} = \frac{6(e_x + \cos L)}{h^2} \tag{7.63}$$

$$\Gamma^x_{hx} = \frac{-3 - 6e_x\cos L - 3\cos 2L - 2\xi}{2h\xi} \tag{7.64}$$

$$\Gamma^x_{hy} = \frac{-3(e_x + \cos L)\sin L}{h\xi} \tag{7.65}$$

$$\begin{aligned}\Gamma^x_{hL} &= \frac{h^5(E_y + \xi e_y + e_t e_x)}{\xi^5\beta} + \frac{E_x e_t\xi^2\beta}{2h^7} \\ &+ (-4e_x e_y\cos L - 2(1+2e_x^2)e_y\cos 2L - 2e_x e_y\cos 3L + \sin L + 4e_x^2\sin L \\ &\quad +2e_x\sin 2L + 2e_x^3\sin 2L - 2e_x e_y^2\sin 2L + e_x^2\sin 3L - e_y^2\sin 3L)/(2h\xi^2)\end{aligned} \tag{7.66}$$

$$\Gamma^x_{xx} = \frac{2(e_x + \cos L)\cos^2 L}{\xi^2} \tag{7.67}$$

$$\Gamma^x_{xy} = \frac{(e_x + \cos L)\sin 2L}{\xi^2} \tag{7.68}$$

$$\begin{aligned}\Gamma^x_{xL} &= \frac{-h^6(E_y + \xi e_y + e_t e_x)\cos L}{2\xi^6\beta} + \frac{E_x\xi^2\beta\sin L}{2h^6} + (8e_x e_y + 11e_y\cos L \\ &\quad +4e_x e_y\cos 2L + e_y\cos 3L + 7e_x\sin L + 6\sin 2L - 4e_x^2\sin 2L - e_x\sin 3L)/(8\xi^2)\end{aligned} \tag{7.69}$$

$$\Gamma^x_{yy} = \frac{2(e_x + \cos L)\sin^2 L}{\xi^2} \tag{7.70}$$

$$\begin{aligned}\Gamma^x_{yL} &= \frac{-h^6(E_y + \xi e_y + e_t e_x)\sin L}{2\xi^6\beta} - \frac{E_x\xi^2\beta\cos L}{2h^6} + (-6 - 8e_x^2 - 17e_x\cos L \\ &\quad +(4e_x^2 - 6)\cos 2L + e_x\cos 3L + e_y\sin L + 4e_x e_y\sin 2L + e_y\sin 3L)/(8\xi^2)\end{aligned} \tag{7.71}$$

$$\Gamma^x_{LL} = \frac{h^8\alpha(E_x + \xi\cos L)}{\xi^6} - \frac{h^2 E_x\alpha\beta}{\xi^2} \tag{7.72}$$

$$\Gamma^y_{hh} = \frac{6(e_y + \sin L)}{h^2} \tag{7.73}$$

$$\Gamma^y_{hx} = \frac{-3(e_y + \sin L)\cos L}{h\xi} \tag{7.74}$$

$$\Gamma^y_{hy} = \frac{-3 - 6e_y\sin L + 3\cos 2L - 2\xi}{2h\xi} \tag{7.75}$$

$$\begin{aligned}\Gamma^y_{hL} &= \frac{-h^5(E_x + \xi e_x - e_t e_y)}{\xi^5 \beta} - \frac{E_y e_t \xi^2 \beta}{2h^7} - [\cos L + 2e_x(1+2e_y^2)\cos 2L \\ &\quad +(e_x^2 - e_y^2)\cos 3L + 4e_y e_t + 2e_y(1 - e_x^2 + e_y^2)\sin 2L + 2e_x e_y \sin 3L]/(2h\xi^2)\end{aligned} \tag{7.76}$$

$$\Gamma^y_{xx} = \frac{2(e_y + \sin L)\cos^2 L}{\xi^2} \tag{7.77}$$

$$\Gamma^y_{xy} = \frac{(e_y + \sin L)\sin 2L}{\xi^2} \tag{7.78}$$

$$\begin{aligned}\Gamma^y_{xL} &= \frac{h^6(E_x + \xi e_x - e_t e_y)\cos L}{2\xi^6 \beta} + \frac{E_y \xi^2 \beta \sin L}{2h^6} + (6 + 8e_y^2 - e_x \cos L \\ &\quad +(4e_y^2 - 6)\cos 2L + e_x \cos 3L + 17 e_y \sin L - 4e_x e_y \sin 2L + e_y \sin 3L)/(8\xi^2)\end{aligned} \tag{7.79}$$

$$\Gamma^y_{yy} = \frac{2(e_y + \sin L)\sin^2 L}{\xi^2} \tag{7.80}$$

$$\begin{aligned}\Gamma^y_{yL} &= -\frac{h^5(E_x + \xi e_x - e_t e_y)}{\xi^5 \beta} + \frac{E_y e_t \xi^2 \beta}{2h^7} - [(1+4e_y^2)\cos L + 2e_x(1+2e_y^2)\cos 2L \\ &\quad +e_x^2 \cos 3L - e_y^2 \cos 3L - 4e_x e_y \sin L + 2e_y \sin 2L \\ &\quad -2e_x^2 e_y \sin 2L + 2e_y^3 \sin 2L + 2e_x e_y \sin 3L]/(2h\xi^2)\end{aligned} \tag{7.81}$$

$$\Gamma^y_{LL} = \frac{h^8 \alpha (E_y + \xi \sin L)}{\xi^6} - \frac{h^2 \alpha \beta E_y}{\xi^2} \tag{7.82}$$

$$\Gamma^L_{hh} = \frac{(e_x \sin L - e_y \cos L)\xi^4}{h^{10}\alpha} + \frac{4(e_y \cos L - e_x \sin L)\xi}{h^4 \alpha \beta} \tag{7.83}$$

$$\Gamma^L_{hx} = \frac{-\xi^4 \sin L}{2h^9 \alpha} - \frac{3e_t \cos L + \xi \sin L}{h^3 \alpha \beta} \tag{7.84}$$

$$\Gamma^L_{hy} = \frac{\xi^4 \cos L}{2h^9 \alpha} - \frac{3e_t \sin L - \xi \cos L}{h^3 \alpha \beta} \tag{7.85}$$

$$\Gamma^L_{hL} = \frac{3}{h} \tag{7.86}$$

$$\Gamma^L_{xx} = \frac{E_y \cos L + e_t \cos^2 L}{h^2 \xi \alpha \beta} \tag{7.87}$$

$$\Gamma^L_{xy} = \frac{-3e_x \cos L - 2\cos 2L + e_x \cos 3L + 3e_y \sin L + e_y \sin 3L}{4h^2 \xi \alpha \beta} \tag{7.88}$$

$$\Gamma^L_{xL} = \frac{-2\cos L}{\xi} \tag{7.89}$$

$$\Gamma^L_{yy} = \frac{-E_x \sin L + e_t \sin^2 L}{h^2 \xi \alpha \beta} \tag{7.90}$$

$$\Gamma^L_{yL} = \frac{-2\sin L}{\xi} \tag{7.91}$$

$$\Gamma^L_{LL} = \frac{2(x \sin L - y \cos L)}{\xi} \tag{7.92}$$

7.4 Lie Brackets

It is also useful to compute the Lie brackets for the horizontal vectors (everything except dr). The formula for the Lie bracket in coordinates is

$$[X,Y]^i = X^j \frac{\partial Y^i}{\partial x^j} - Y^j \frac{\partial X^i}{\partial x^j} \tag{7.93}$$

$$\begin{aligned}
[da_r, da_t]^h &= h\sin L\frac{-h^2\cos L}{\xi^2} - h\cos L\frac{-h^2\sin L}{\xi^2} \\
&= 0 && (7.94) \\
[da_r, da_t]^{e_x} &= h\sin L\frac{h\xi\sin^2 L + he_t\cos L\sin L}{\xi^2} - h\cos L\frac{-h\xi\sin L\cos L + he_t\sin^2 L}{\xi^2} - \frac{h^2\sin L}{\xi} \\
&= 0 && (7.95) \\
[da_r, da_t]^{e_y} &= h\sin L\frac{-h\xi\cos L\sin L - he_t\cos^2 L}{\xi^2} - h\cos L\frac{h\xi\cos^2 L - he_t\cos L\sin L}{\xi^2} + \frac{h^2\cos L}{\xi} \\
&= 0 && (7.96) \\
[da_r, da_t]^L &= 0 && (7.97) \\
[da_r, da_t] &= (0,0,0,0) && (7.98)
\end{aligned}$$

$$\begin{aligned}
[da_r, dt]^h &= 0 && (7.99) \\
[da_r, dt]^{e_x} &= -\frac{\xi^2}{h^3}h\cos L \\
&= -\frac{\xi^2\cos L}{h^2} && (7.100) \\
[da_r, dt]^{e_y} &= -\frac{\xi^2}{h^3}h\sin L \\
&= -\frac{\xi^2\sin L}{h^2} && (7.101) \\
[da_r, dt]^L &= -h\sin L\frac{2\xi\cos L}{h^3} + h\cos L\frac{2\xi\sin L}{h^3} \\
&= 0 && (7.102) \\
[da_r, dt] &= \frac{-\xi^2}{h^2}(0, \cos L, \sin L, 0) && (7.103)
\end{aligned}$$

$$\begin{aligned}
[da_t, dt]^h &= \frac{-\xi^2}{h^3}\frac{-h^2(e_y\cos L - e_x\sin L)}{\xi^2} \\
&= \frac{e_y\cos L - e_x\sin L}{h} \\
&= \frac{e_t}{h} && (7.104) \\
[da_t, dt]^{e_x} &= \frac{-\xi^2}{h^3}\left(-2h\sin L + \frac{-he_t\cos L + he_x\cos L\sin L + he_y\sin^2 L}{\xi} + \frac{he_t\sin L(e_y\cos L - e_x\sin L)}{\xi^2}\right) \\
&= \frac{1}{h^2}\left(2\xi^2\sin L - 2e_x\xi\cos L\sin L + e_y\xi(\cos^2 L - \sin^2 L) - (e_y\cos L - e_x\sin L)^2\sin L\right) \\
&= \frac{1}{h^2}\left(2\xi^2\sin L + \xi(e_t\cos L - e_r\sin L) - e_t^2\sin L\right) && (7.105) \\
[da_t, dt]^{e_y} &= \frac{-\xi^2}{h^3}\left(2h\cos L + \frac{-he_t\sin L - he_x\cos^2 L - he_y\cos L\sin L}{\xi} - \frac{he_t\cos L(e_y\cos L - e_x\sin L)}{\xi^2}\right)
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{h^2}\left(-2\xi^2\cos L + 2e_y\xi\cos L\sin L + e_x\xi(\cos^2 L - \sin^2 L) + (e_y\cos L - e_x\sin L)^2\cos L\right) \\
&= \frac{1}{h^2}\left(-2\xi^2\cos L + \xi(e_r\cos L + e_t\sin L) + e_t^2\cos L\right) && (7.106) \\
[da_t, dt]^L &= \frac{h^2}{\xi}\frac{-3\xi^2}{h^4} + \frac{2h\xi\cos L - he_t\sin L}{\xi}\frac{2\xi\cos L}{h^3} + \frac{2h\xi\sin L + he_t\cos L}{\xi}\frac{2\xi\sin L}{h^3} \\
&= \frac{\xi}{h^2} && (7.107)
\end{aligned}
$$

7.5 Connection, Curvature, and Torsion

In addition, the connection 1-form (ω), curvature 2-form (Ω) and torsion 2-form (Θ) can be calculated.

$$d\theta = -\omega \wedge \theta + \Theta \tag{7.108}$$

$$\Omega = d\omega + \omega \wedge \omega \tag{7.109}$$

where θ is the basis 1-forms for a moving frame.

$$
\begin{aligned}
da_r &= \frac{e_t}{h^2}dh + \frac{\sin L}{h}de_x - \frac{\cos L}{h}de_y && (7.110) \\
da_t &= -\frac{\xi}{h^2}dh + \frac{\cos L}{h}de_x + \frac{\sin L}{h}de_y && (7.111) \\
dr &= \frac{2h}{\xi}dh - \frac{h^2\cos L}{\xi^2}de_x - \frac{h^2\sin L}{\xi^2}de_y && (7.112) \\
dt &= \frac{h^3}{\xi^2}dL && (7.113) \\
d(da_r) &= -\frac{\sin L}{h^2}dh\wedge de_x + \frac{\cos L}{h^2}dh\wedge de_y - \frac{\sin L}{h^2}de_x\wedge dh + \frac{\cos L}{h^2}de_y\wedge dh \\
&\quad -\frac{e_r}{h^2}dL\wedge dh + \frac{\cos L}{h}dL\wedge de_x + \frac{\sin L}{h}dL\wedge de_y \\
&= dL\wedge da_t + \frac{1}{h^2}dL\wedge dh && (7.114) \\
d(da_t) &= -\frac{\cos L}{h^2}dh\wedge de_x - \frac{\sin L}{h^2}dh\wedge de_y - \frac{\cos L}{h^2}de_x\wedge dh - \frac{\sin L}{h^2}de_y\wedge dh \\
&\quad -\frac{e_t}{h^2}dL\wedge dh - \frac{\sin L}{h}dL\wedge de_x + \frac{\cos L}{h}dL\wedge de_y \\
&= dL\wedge da_r && (7.115) \\
d(dr) &= -\frac{2h\cos L}{\xi^2}dh\wedge de_x - \frac{2h\cos L}{\xi^2}de_x\wedge dh - \frac{2h\sin L}{\xi^2}dh\wedge de_y - \frac{2h\sin L}{\xi^2}de_y\wedge dh \\
&\quad -\frac{2he_t}{\xi^2}dL\wedge dh - h^2\frac{-\xi\sin L - 2e_t\cos L}{\xi^3}dL\wedge de_x - h^2\frac{\xi\cos L - 2e_t\sin L}{\xi^3}dL\wedge de_y \\
&= -\frac{e_t}{\xi}dL\wedge dr + \frac{h^2\sin L + e_t\cos L/\xi}{\xi^2}dL\wedge de_x - \frac{h^2\cos L - e_t\sin L/\xi}{\xi^2}dL\wedge de_y \\
&= -\frac{2e_t}{\xi}dL\wedge dr + \frac{he_t}{\xi^2}dL\wedge dh + \frac{h^3}{\xi^2}dL\wedge da_r && (7.116)
\end{aligned}
$$

$$\begin{aligned} d(dt) &= \frac{3h^2}{\xi^2}dh \wedge dL - \frac{2h^3 \cos L}{\xi^3}de_x \wedge dL - \frac{2h^3 \sin L}{\xi^3}de_y \wedge dL \\ &= \frac{2h}{\xi}dr \wedge dL - \frac{h^2}{\xi^2}dh \wedge dL \end{aligned} \tag{7.117}$$

$$\omega = \begin{bmatrix} 0 & -dL & -\frac{h^3}{\xi^2}dL & 0 \\ dL & 0 & 0 & 0 \\ 0 & 0 & \frac{2e_t}{\xi}dL & 0 \\ 0 & 0 & \frac{2h}{\xi}dL & hdh \end{bmatrix} \tag{7.118}$$

$$\Theta = \begin{bmatrix} \frac{1}{h^2}dL \wedge dh \\ 0 \\ \frac{he_t}{\xi^2}dL \wedge dh \\ 0 \end{bmatrix} \tag{7.119}$$

$$d\omega = \begin{bmatrix} 0 & 0 & \frac{3h^2}{\xi^2}dh \wedge dL - \frac{2h^3 \cos L}{\xi^3}de_x \wedge dL - \frac{2h^3 \sin L}{\xi^3}de_y \wedge dL & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{-2\xi \sin L + 2e_t \cos L}{\xi^2}de_x \wedge dL + \frac{2\xi \cos L + 2e_t \sin L}{\xi^2}de_y \wedge dL & 0 \\ 0 & 0 & \frac{-2h}{\xi^2}(\cos L de_x + \sin L de_y) \wedge dL + \frac{2}{\xi^2}dh \wedge dL & 0 \end{bmatrix} \tag{7.120}$$

$$\omega \wedge \omega = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{2h^2}{\xi}dh \wedge dL & 0 \end{bmatrix} \tag{7.121}$$

$$\Omega = \begin{bmatrix} 0 & 0 & \frac{3h^2}{\xi^2}dh \wedge dL - \frac{2h^3 \cos L}{\xi^3}de_x \wedge dL - \frac{2h^3 \sin L}{\xi^3}de_y \wedge dL & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{-2\xi \sin L + 2e_t \cos L}{\xi^2}de_x \wedge dL + \frac{2\xi \cos L + 2e_t \sin L}{\xi^2}de_y \wedge dL & 0 \\ 0 & 0 & \frac{-2h}{\xi^2}(\cos L de_x + \sin L de_y) \wedge dL + \frac{2+2h^2\xi}{\xi^2}dh \wedge dL & 0 \end{bmatrix} \tag{7.122}$$

7.6 Results

With all of the properties of the astrodynamics manifold computed, it is possible to use the algorithm to compute geodesics. In order to determine whether the generated trajectories are actually optimal, they need to be compared to some known optimal trajectories. This was done by using a Hamiltonian shooting method to generate a family of trajectories with different starting conditions. The resulting trajectories provided a large number of trajectory optimization problems that could be solved using the geodesic method.

The Hamiltonian integrator was initialized with a variety of initial values for the costate (changing p_h and p_{e_x}) while keeping the initial state constant. The path was integrated forward until $L > 1$ or

100,000 points had been integrated. All paths where the final value of L was greater than 0.1 were included in this analysis. This provided 216 paths for comparison.

The endpoints of each path were then used to define a path optimization problem for the geodesic search algorithm. The geodesic algorithm was run using a pseudospectral method with Chebyshev polynomials and 30 collocation points. The value of β was decreased until it was determined that the solution was not getting better (this is explained in the next paragraph). For each value of β, the algorithm found the best solution to the geodesic equation (starting with the solution for the previous value of β). This geodesic was then analyzed to calculate the horizontal and vertical cost of the trajectory.

The path with the smallest vertical cost was considered to be the best path, because it violates the constraints by the least. Figure 7-2 shows the typical evolution of the horizontal cost (Δv, the actual cost) and vertical cost ($\int dr$, the constraint violation cost) of a trajectory. The horizontal cost increases as the vertical cost is decreased because enforcing the constraints forces the path to take a more expensive, but also more feasible route. Eventually, the constraints begin to dominate the path selection and further increases to the constraints do not improve the path accuracy. This can be seen in the graph at the point where the vertical costs start to increase. At that point, the constraints are being enforced too strictly and the algorithm no longer has enough freedom to search for a better path, so the path already found is used.

After computing all of the paths, the Δv's were compared to determine how accurate the geodesic method is. Figure 7-3 shows the CDF (cumulative distribution function) of the Δv ratio. This ratio is determined by dividing the Δv of the path determined by the geodesic method by the Δv of the path determined by the Hamiltonian method and subtracting one to show the error. These would not in general be expected to be identical, in part because the Hamiltonian method uses several tens of thousands of points while the geodesic method uses only thirty points, so the geodesic path is by its nature more approximate. Most of the Δv calculations appear to be within one percent of the Hamiltonian Δv, which is fairly close for such a coarse path approximation.

The algorithm works well with small transfers, but with larger transfers it needs too many points, which increases the computational cost, making the algorithm impractically slow. This is most likely due to the high curvature of the astrodynamics manifold, particularly in the L direction (which has 4th order trigonometric functions). The high curvature requires more points because the connection coefficients vary rapidly from one point to another, so a large number of points are required in order for the equations to accurately represent the problem.

The result of this is that the method as presented works well for small transfers (less than 2 revolutions), but is impractical for long spiral transfers. Shooting methods tend to work well for long spiral transfers, but are unstable with a small number of revolutions. This means that prior methods of optimizing orbital transfers work well where this one is currently computationally inefficient (but

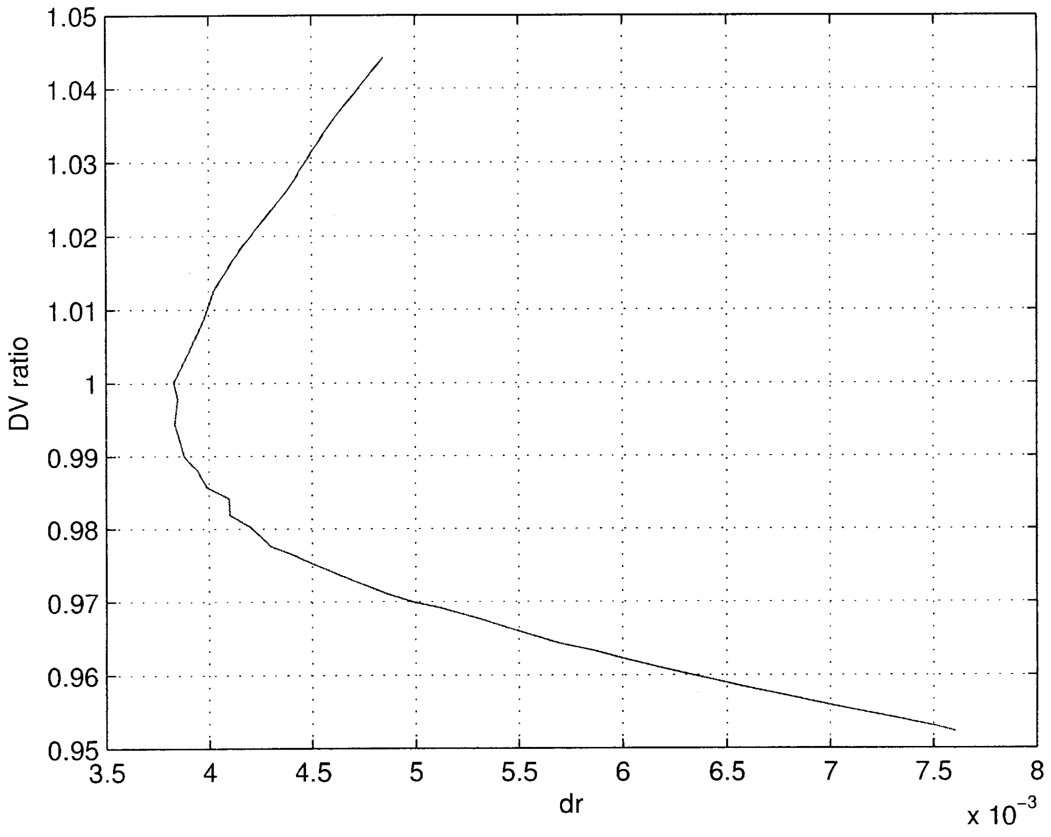


Figure 7-2: Δv as a function of dr for a sample trajectory. The Δv is increasing as the algorithm converges, while dr decreases until the penalty is too large, and then dr begins to increase.

may still be improved), while other methods do not tend to work very well for transfers that the geodesic search method does work well for. More research is required to find the crossover between methods in terms of computational efficiency, accuracy, and stability.

The method as presented also works for a wide range of possible thrust levels. It will work best if the thrust level is allowed to vary continuously with no limits, but it will also work fine if the thrust is constrained to be in some range (varying continuously from a minimum to a maximum). Turning the engine off completely for part of the trajectory might be numerically difficult for the algorithm, but it should be possible to produce approximate solutions for such problems by turning the engine off whenever the thrust is below some threshold.

Statistic	Value
mean	.0076
standard deviation	0.00465
median	0.0070
within 1%	155/216
within 2%	213/216
max	1.02266
min	0.9764

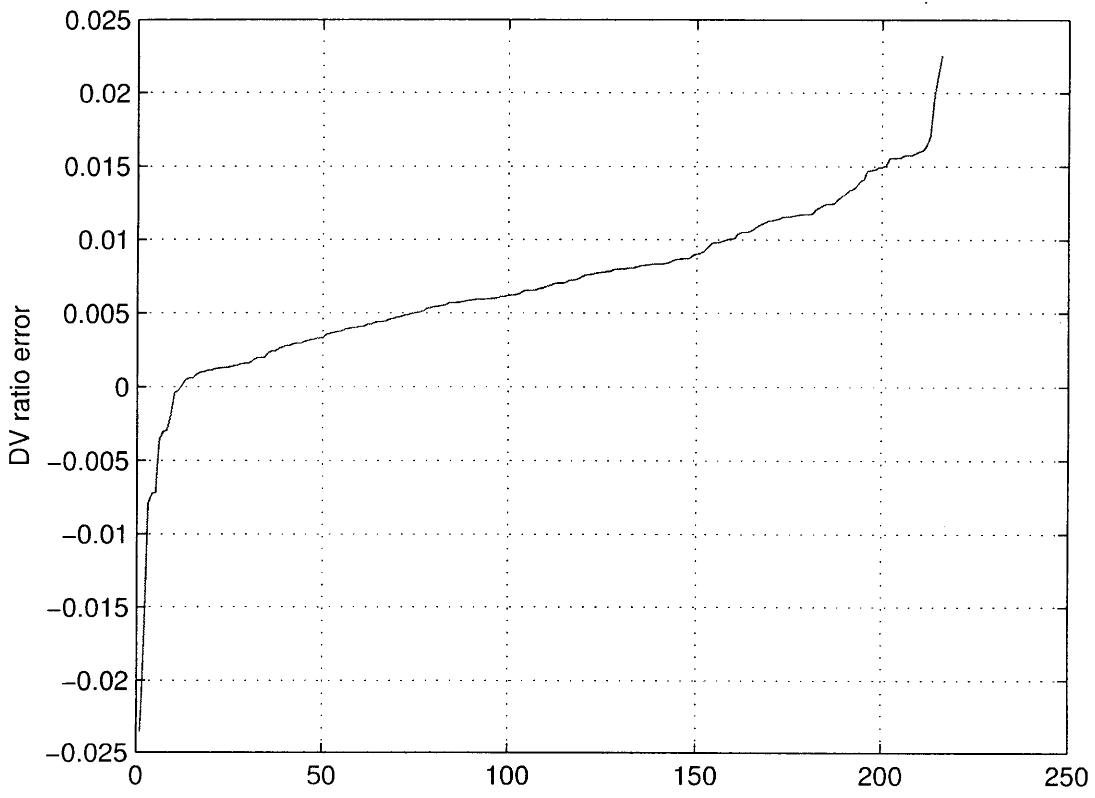


Figure 7-3: Δv ratio (geodesic divided by Hamiltonian) minus one. The graph shows the distribution of the error of the Δv calculation for the 216 generated paths. The table shows the summary statistics of the distribution. The max and min are the ratio without subtracting one.

Chapter 8

Conclusions and Future Work

Most path optimization problems can be represented geometrically as sub-Riemannian manifolds. The concepts of sub-Riemannian geometry can then be applied to better understand the path optimization problems. Additionally, the geodesics of the manifold are the solutions to the path optimization problems.

A method has been presented for characterizing the sub-Riemannian manifold that corresponds to a given path optimization problem. This method should be flexible enough to work for any path optimization problem. The cometric could be used to generate the Hamiltonian function so that the path optimization problem can be solved using a Hamiltonian method. Alternatively, the penalty metric can be used to find the geodesic equation for the path optimization problem.

An algorithm has been presented which can find geodesics in sub-Riemannian manifolds using a penalty metric. The algorithm converges well for manifolds with small amounts of curvature. Manifolds with large curvature require more points in the path, which slows down the convergence rate.

The geometrical properties of the Astrodynamics manifold have been calculated. Based on these properties, it seems that astrodynamics is a Hörmander manifold. It is step 2 everywhere, but does not have the strong bracket generating property. The curvature is very high in some places, which makes the convergence rate somewhat slow for the algorithm as described.

8.1 Comparison to other methods

The biggest difference between this method and other methods is that this method translates path optimization problems into geometrical manifolds. This reformulates the problem into a different branch of math than previous path optimization methods. There is a large body of work in differential geometry, which has mostly not been applied to path optimization problems. It is likely that studying the different types of manifolds that correspond to different path optimization problems will lead to new insights that may be useful for future work in path optimization. Determining the type of manifold that a problem corresponds to may provide guidance as to what solution method is likely to work best.

Beyond the conceptual differences, this path optimization method is essentially a direct optimization method, but unlike most direct optimization methods it has explicit optimality conditions that provide information about how the path should be improved and when the path has converged to the optimal solution.

Another benefit of this method is that the Jacobi equation can be used to check the solution and determine if it is unique or not (or at least determine a region within the manifold that a given geodesic is unique). Other path optimization methods can only determine if a geodesic is a local optimum and provide no information about how large of a change would be necessary to potentially find a different local minimum.

A major disadvantage of this method is that it requires a lot of algebra. However, the algebra is all straightforward enough that computer programs such as Mathematica can be used for all the symbolic work, so this disadvantage is not actually as severe as it could be.

8.2 Future Work

This work presents a new optimization method. It is not yet clear under what circumstances this method will be better than other methods. It is likely that the method could be improved in many ways. Some possible improvements that will require further work include:

- Determining the effect of picking different vectors for spanning the non-tangent space portion of the manifold
- Determining how to modify the penalty in the metric to achieve fast convergence
- Determining how close the points in the path should be (possibly as a function of the local curvature)
- Determining when to increase the number of points in the path
- Using pseudospectral methods to compute the connection coefficients explicitly as part of the optimization process
- Implementing the geodesic search algorithm with parallel or stream processing

The first item in this list may change the convergence properties of the algorithm. Choosing a different set of vectors to represent the infeasible motions in the problem will change the Riemannian penalty manifolds. In the limit as the penalty is increased to infinity, all of the penalty manifolds will converge to the same sub-Riemannian manifold. However, the properties of the manifold could vary depending on which basis vectors are used. There is no natural guidance for choosing these vectors, so they can be chosen in any way that produces linearly independent vectors. It is likely that choosing different vectors will normally not make much difference, but in some cases it could alter the properties of the problem.

The next three items deal with tradeoffs between the convergence stability and the convergence rate. Increasing the penalty too slowly will waste computational time, but increasing it too quickly will prevent the algorithm from converging. Similarly, having more points than necessary will waste computational resources, but not having enough destabilizes the algorithm. It is likely that for most problems, starting with a small number of points and adding more as the path converges will lead to faster convergence times.

The fifth improvement idea is to calculate the values of the connection coefficients using pseudospectral methods and convolutions or some similar process. This would allow the connection coefficients to be calculated as functions along the path, rather than calculating their numerical value at each point. The biggest obstacle to convergence is that the connection coefficients change depending on the path, so as the path changes, the numerical values in the differential equations defining the path also change. If the connection coefficients were computed as functions of the path, then a method similar to fourier transforms could potentially be used to find the solutions to the differential equations much more quickly. Such a method would probably be substantially faster than any current path optimization methods.

The final improvement is the most straightforward one. The algorithm requires no substantial modifications to be able to take advantage of parallel processing, it only requires the code to be rewritten. It may be even possible to use the same code with linear algebra libraries that take care of the parallel processing.

Other areas of future work include characterizing path optimization problems based on the properties of the sub-Riemannian manifolds they correspond to. There are a few classes of sub-Riemannian manifolds (Heisenberg, Grushin, Hörmander, etc), and it is likely that control problems which lie in different categories have different properties. These differences are likely to matter even when using other optimization algorithms. Furthermore, the curvature and other properties of the manifolds are intrinsic properties of the optimization problems related to the manifolds. Sub-Riemannian geometry is still a relatively new branch of math, and as it is studied further, it is likely that more useful results will be discovered which could provide insights into better methods for solving path optimization problems.

Appendix A

Simulation Code

A.1 Common Functions

This function inserts a new point in the middle of each pair of points in the path.

```
function new = half(old)


[a b]=size(old);
c=(b-1)*2+1;
new=zeros(a,c);
new(:,1:2:c)=old(:,1:b);
new(:,2:2:c)=(old(:,1:b-1)+old(:,2:b))/2;
```

A.2 Generic Algorithm

This section contains the generic algorithm code. For a specific problem, the initial values would have to be tuned and the connection coefficients would have to be entered. The generic code only has space for two dimensions (x and y), but more can be added by adding similar lines.

The setup code is similar for both cases, with the pseudospectral method needing a little bit more code to calculate the polynomials. After the setup code, either the finite difference or pseudospectral code will actually optimize the path.

A.2.1 Setup

```
clear l F2 F3 F4 trackx tracky trackz
%
% Set Initial Values
```

```
%
addn=1;
lastadd=100;
plotting=0;
n=5;
maxn=400;
track=1;
alpha=.07;
stiffup=1.5;
stiffdown=.9;
dl=1/(2*pi);
basestiff=100;
stiffx=ones(1,n)*basestiff;
stiffy=ones(1,n)*basestiff;
%
% Set initial guess for path vectors here
%
x=
y=
x0=
y0=
xf=
yf=
x(1)=x0;
x(n)=xf;
y(1)=y0;
y(n)=yf;
trackx=[];
tracky=[];
count=1;
Fx=x;
Fy=y;
deltax=x*0;
deltay=y*0;
totaldelta=0;
errmin=1;
```

```
%
%The rest is for setting up the Chebyshev polynomials
%It is not needed for the finite difference method
%
cheby=sort(cos((2*(1:n)-1)/2/n*pi));
s=cheby/2+.5;
T=cheby*0+1;
U=T;
T(2,:)=cheby;
U(2,:)=2*cheby;
for count=3:m
  T(count,:)=2*cheby.*T(count-1,:)-T(count-2,:);
  U(count,:)=2*cheby.*U(count-1,:)-U(count-2,:);
end
```

A.2.2 Finite Difference Calculations

```
while max(sqrt(Fx.^2+Fy.^2+Fz.^2))>.01*totaldelta
%
%Finite difference equations
%
  diffx=diff(x);
  dx=[0 (diffx(1:n-2)+diffx(2:n-1))/2 0]./dl;
  d2x=[0 diff(diffx) 0]./(2*dl^2);
  diffy=diff(y);
  dy=[0 (diffy(1:n-2)+diffy(2:n-1))/2 0]./dl;
  d2y=[0 diff(diffy) 0]./(2*dl^2);


%
% Enter connection coefficients here
%
  Fx=d2x+
  Fy=d2y+
%
%Calculate movements from forces
%
  deltax2=deltax;
  deltax=Fx./stiffx;
  samex=(sign(deltax)==sign(deltax2));
  stiffx=stiffx.*samex*stiffdown+stiffx.*(1-samex)*stiffup;
  stiffx=min(1e10,max(stiffx,.01));
  deltay2=deltay;
  deltay=Fy./stiffy;
  samey=(sign(deltay)==sign(deltay2));
  stiffy=stiffy.*samey*stiffdown+stiffy.*(1-samey)*stiffup;
  stiffy=min(1e10,max(stiffy,.01));
%
%This code limits how far the points can move at each step.
%It often stabilizes things, but may just slow down convergence in
%some cases
%
  mdx=mean(min(.1,abs(deltax)));
  mdy=mean(min(.1,abs(deltay)));
  deltax=max(-mdx,min(deltax,mdx));
  deltay=max(-mdy,min(deltay,mdy));
%
%Keep track of where the points were and then move them
%
  px=x;
  py=y;
  x=x+deltax;
  y=y+deltay;
  totaldelta=sum(sqrt((px-x).^2+(py-y).^2));
  count=count+1;
  l=sum(sqrt(diffx.^2+diffy.^2));
%
%These conditions determine when to reduce alpha and when to increase
%the number of points. They should be adjusted to provide a good
%convergence rate while maintaining stability
%
  decalpha=0;
  if (totaldelta*min(10,sqrt(n))<max(alpha,sqrt(alpha)/10))&&(count>lastadd+20)
    alpha=alpha*.99;
    lastadd=count;
    decalpha=1;
%
%Track what the path was each time that the path converges for a new
%value of alpha
%
    [a b]=size(trackx);
    trackx(1:n,b+1)=x;
    tracky(1:n,b+1)=y;
    trackcount(b+1)=count;
%
%Linear extrapolation at each convergence step
%
    if b>20
      x=x+mean(diff(trackx(:,b-8:b+1)'))/2;
      y=y+mean(diff(tracky(:,b-8:b+1)'))/2;
    end
  end
%
%Increase the points if more points are still desired.
%It may be useful to have a condition relating the number of points
%to the value of alpha, such as
%if (n<alpha^(-1))&&(n<maxn)
%
  if (decalpha)&&(n<maxn)
    stiffx=stiffx*10;
    stiffy=stiffy*10;
    addn=[addn count];
    A=half([x;y;deltax;deltay;stiffx;stiffy]);
    x=A(1,:);
    y=A(2,:);
    deltax=A(3,:);
    deltay=A(4,:);
    stiffx=A(5,:);
    stiffy=A(6,:);
    dl=dl/2;
    n=n*2-1;
    lastadd=count;
    trackx=half(trackx')';
    tracky=half(tracky')';
  end
end
```

A.2.3 Pseudospectral Method

```
while (alpha>1e-3)
%
%Calculate the polynomial coefficients. The projection works if the
%collocation points are used, except for the first value
%
  xa=2*(T*x'/n)';
  ya=2*(T*y'/n)';
  xa(1)=mean(x);
  ya(1)=mean(y);
%
%Calculate derivative coefficients
%
  dxb=(1:m-1).*xa(2:m);
  dyb=(1:m-1).*ya(2:m);
  for count=2:m
    d2xa(count)=count*(count-1)*xa(count);
    d2xb(count)=-(count-1)*xa(count);
    d2ya(count)=count*(count-1)*ya(count);
    d2yb(count)=-(count-1)*ya(count);
  end
%
%Calculate derivatives
%
  dx=dxb*U(1:m-1,:)*2;
  dy=dyb*U(1:m-1,:)*2;
  d2x=(d2xa*T+d2xb*U)./(cheby.^2-1)*4;
  d2y=(d2ya*T+d2yb*U)./(cheby.^2-1)*4;
%
```

```
%Calculate forces. Insert connection coefficients here.
%This is the same as for the finite difference method
%
  Fx=d2x+
  Fy=d2y+
  deltax2=deltax;
  deltax=Fx./stiffx;
  samex=(sign(deltax)==sign(deltax2));
  stiffx=stiffx.*samex*stiffdown+stiffx.*(1-samex)*stiffup;
  stiffx=min(1e10,max(stiffx,.01));
  deltay2=deltay;
  deltay=Fy./stiffy;
  samey=(sign(deltay)==sign(deltay2));
  stiffy=stiffy.*samey*stiffdown+stiffy.*(1-samey)*stiffup;
  stiffy=min(1e10,max(stiffy,.01));
%
%This code limits how far the points can move at each step.
%It often stabilizes things, but may just slow down convergence in
%some cases
%
  mdx=mean(min(.1,abs(deltax)));
  mdy=mean(min(.1,abs(deltay)));
  deltax=max(-mdx,min(deltax,mdx));
  deltay=max(-mdy,min(deltay,mdy));
%
%Keep track of where the points were and then move them
%
  px=x;
  py=y;
  x=x+deltax;
  y=y+deltay;
%
%It is difficult to not accidentally move the initial and final
%points with this method, so these lines return them to the
%path endpoints
%
```

```
  x(1)=x0;
  x(n)=xf;
  y(1)=y0;
  y(n)=yf;
  count=count+1;
  totaldelta=sum(deltax.^2+deltay.^2);
%
%These conditions determine when to reduce alpha. They should
%be adjusted to provide a good convergence rate while
%maintaining stability
%
  if (totaldelta<min(alpha/10,alpha^2))&&(count>lastadd+100)
    [a b]=size(trackx);
    xa=(2*T*x'/n)';
    ya=(2*T*y'/n)';
    xa(1)=mean(x);
    ya(1)=mean(y);
    trackx(1:m,b+1)=xa;
    tracky(1:m,b+1)=ya;
    trackcount(b+1)=count;
    tracka(b+1)=alpha;
    alpha=alpha*.99;
    if b>20
      xa=xa+mean(diff(trackx(:,b-8:b+1)'))/2;
      ya=ya+mean(diff(tracky(:,b-8:b+1)'))/2;
      x=xa*T;
      y=ya*T;
    end
    lastadd=count;
%
%Check to see if adding more points would be desirable
%
    if (n<alpha^(-1))&&(n<1e3)
      stiffx=stiffx*10;
      stiffy=stiffy*10;
      addn=[addn count];
      dxa=(2*T*deltax'/n)';
      dya=(2*T*deltay'/n)';
      sxa=(2*T*stiffx'/n)';
      sya=(2*T*stiffy'/n)';
      oldm=m;
%
%Add points slowly at first, then add them faster
%
      if n<100
        n=n+10;
      else
        n=n+100;
      end
      if (n>1e3)
        n=1e3;
      end
      m=n;
%
%If we actually added points, calculate new polynomials
%and point locations
%
      if m>oldm
        cheby=sort(cos((2*(1:n)-1)/2/n*pi));
        s=cheby/2+.5;
        T=cheby*0+1;
        U=T;
        T(2,:)=cheby;
        U(2,:)=2*cheby;
        for count=3:m
          T(count,:)=2*cheby.*T(count-1,:)-T(count-2,:);
          U(count,:)=2*cheby.*U(count-1,:)-U(count-2,:);
        end
%
%Add the new points with all the new polynomials having a
%coefficient of 0
%
```

```
        x=mean(x)+xa(2:oldm)*T(2:oldm,:);
        y=mean(y)+ya(2:oldm)*T(2:oldm,:);
        deltax=mean(deltax)+dxa(2:oldm)*T(2:oldm,:);
        deltay=mean(deltay)+dya(2:oldm)*T(2:oldm,:);
        stiffx=mean(stiffx)+sxa(2:oldm)*T(2:oldm,:);
        stiffy=mean(stiffy)+sya(2:oldm)*T(2:oldm,:);
        lastadd=count;
      end
    end
  end
end
```

A.3 The Heisenberg Manifold

The generic code has been removed to save space. This is the parts that need to be inserted to specify the Heisenberg manifold problem.

A.3.1 Finite Difference Method

```
%
%Set up path
%
x=rand(1,n)*2-1;
y=rand(1,n)*2-1;
x0=1;
xf=-1;
y0=0;
yf=0;
z0=0;
zf=pi/2;
z=(0:n-1)/(n-1)*zf;
%
%Calculate forces
%
Fx=d2x+(y/2.*dx.*dy-x/2.*dy.^2+dy.*dz)/alpha;
Fy=d2y+(-y/2.*dx.^2+x/2.*dx.*dy-dx.*dz)/alpha;
Fz=d2z+(-x.*y/4.*dx.^2+(x.^2-y.^2)/4.*dx.*dy-x/2.*dx.*dz \
        +(x.*y)/4.*dy.^2-y/2.*dy.*dz)/alpha;
%
%Condition for decreasing alpha
%
if (totaldelta*min(10,sqrt(n))<max(alpha,sqrt(alpha)/10))&&(count>lastadd+20)
  alpha=alpha*.99;
%
%Additional conditions for adding points (conditions for decreasing
%alpha must also be met
%
  if (n<maxn)
```

A.3.2 Pseudospectral Method

The setup and force calculations are the same. The only difference from the finite difference method (other than the changes to the generic code) are the conditions for decreasing alpha and adding points.

```
%
%Condition for decreasing alpha
%
  if (totaldelta<min(alpha/10,alpha^2))&&(count>lastadd+100)
    alpha=alpha*.99;
    lastadd=count;
%
%Additional conditions for increasing the number of points
%
    if (n<alpha^(-1))&&(n<1e3)
```

A.4 The Tank Manifold

For this problem, the code will be shown first with the generic code removed to save space, and then the full program will be shown. This will remove any ambiguity about how the generic code is combined with the problem specific code.

A.4.1 Problem Specific Code (Pseudospectral)

```
%
```

```
%initial values
%
n=11;
alpha=0.05;
beta=10^(loop/3-1);
stiffup=1.5;
stiffdown=.95;
%
%path definition
%
x0=.1;
xf=0;
y0=1;
yf=0;
t0=0;
tf=0;
%
%connection coefficients
%
Fxx=(1/alpha-alpha).*sin(t).*cos(t).*dt;
Fxy=(1-1/alpha).*(cos(t).^2-alpha*sin(t).^2).*dt;
Fxt=0;
Fx=d2x+Fxx.*dx+Fxy.*dy+Fxt.*dt;
Fyx=(1/alpha-1).*(sin(t).^2-alpha.*cos(t).^2).*dt;
Fyy=(alpha-1/alpha).*cos(t).*sin(t).*dt;
Fyt=0;
Fy=d2y+Fyx.*dx+Fyy.*dy+Fyt.*dt;
Ftx=(1-1/alpha)/beta.*(cos(t).*sin(t).*dx+(sin(t).^2-cos(t).^2).*dy);
Fty=(1/alpha-1)/beta.*cos(t).*sin(t).*dy;
Ftt=0;
Ft=d2t+Ftx.*dx+Fty.*dy+Ftt.*dt;
%
%Condition for decreasing alpha
%
  if (totaldelta/n<alpha/10)&&(count2>lastadd+1e2)
    lastadd=count2;
```

```
%
%Additional condition for adding points
%
    if (n<alpha^(-1))&&(n<100)
      oldm=m;
      if n<100
        n=n+50;
      end
      if n>100
        n=100;
      end
      m=n;
```

A.4.2 Full Code (Pseudospectral Method)

```
for loop=1:10
clear d2xa d2xb d2ya d2yb d2ta d2tb
addn=1;
lastadd=1;
plotting=0;
movavg=1;
n=11;
m=n;
alpha=0.05;
beta=10^(loop/3-1);
stiffup=1.5;
stiffdown=.95;
x0=.1;
xf=0;
y0=1;
yf=0;
t0=0;
tf=0;
cheby=sort(cos((2*(1:n)-1)/2/n*pi));
s=cheby/2+.5;
T=cheby*0+1;
U=T;
```

```
T(2,:)=cheby;
U(2,:)=2*cheby;
for count=3:m
  T(count,:)=2*cheby.*T(count-1,:)-T(count-2,:);
  U(count,:)=2*cheby.*U(count-1,:)-U(count-2,:);
end
basestiff=1e4;
stiffx=ones(1,n)*basestiff;
stiffy=ones(1,n)*basestiff;
stifft=ones(1,n)*basestiff;
x=x0+(xf-x0)*s;
y=y0+(yf-y0)*s;
t=t0+(tf-t0)*s;
r=x*0;
trackx=[];
tracky=[];
trackt=[];
trackl=[];
trackl2=[];
count2=1;
totaldelta=0;
Fx=x;
Fy=y;
Ft=t;
deltax=0*x;
deltay=0*y;
deltat=0*t;
while ((max(sqrt(Fx.^2+Fy.^2+Ft.^2))>.01*totaldelta)&&(count2<1e5))
  xa=2*(T*x'/n)';
  ya=2*(T*y'/n)';
  ta=2*(T*t'/n)';
  xa(1)=mean(x);
  ya(1)=mean(y);
  ta(1)=mean(t);
  dxb=(1:m-1).*xa(2:m);
  dyb=(1:m-1).*ya(2:m);
```

```
dtb=(1:m-1).*ta(2:m);
for count=2:m
  d2xa(count)=count*(count-1)*xa(count);
  d2xb(count)=-(count-1)*xa(count);
  d2ya(count)=count*(count-1)*ya(count);
  d2yb(count)=-(count-1)*ya(count);
  d2ta(count)=count*(count-1)*ta(count);
  d2tb(count)=-(count-1)*ta(count);
end
dx=dxb*U(1:m-1,:)*2;
dy=dyb*U(1:m-1,:)*2;
dt=dtb*U(1:m-1,:)*2;
d2x=(d2xa*T+d2xb*U)./(cheby.^2-1)*4;
d2y=(d2ya*T+d2yb*U)./(cheby.^2-1)*4;
d2t=(d2ta*T+d2tb*U)./(cheby.^2-1)*4;
Fxx=(1/alpha-alpha).*sin(t).*cos(t).*dt;
Fxy=(1-1/alpha).*(cos(t).^2-alpha*sin(t).^2).*dt;
Fxt=0;
Fx=d2x+Fxx.*dx+Fxy.*dy+Fxt.*dt;
Fyx=(1/alpha-1).*(sin(t).^2-alpha.*cos(t).^2).*dt;
Fyy=(alpha-1/alpha).*cos(t).*sin(t).*dt;
Fyt=0;
Fy=d2y+Fyx.*dx+Fyy.*dy+Fyt.*dt;
Ftx=(1-1/alpha)/beta.*(cos(t).*sin(t).*dx+(sin(t).^2-cos(t).^2).*dy);
Fty=(1/alpha-1)/beta.*cos(t).*sin(t).*dy;
Ftt=0;
Ft=d2t+Ftx.*dx+Fty.*dy+Ftt.*dt;
deltax2=deltax;
deltay2=deltay;
deltat2=deltat;
deltax=Fx./stiffx;
deltay=Fy./stiffy;
deltat=Ft./stifft;
samex=(sign(deltax)==sign(deltax2));
samey=(sign(deltay)==sign(deltay2));
samet=(sign(deltat)==sign(deltat2));
```

```
stiffx=stiffx.*samex*stiffdown+stiffx.*(1-samex)*stiffup;
stiffy=stiffy.*samey*stiffdown+stiffy.*(1-samey)*stiffup;
stifft=stifft.*samet*stiffdown+stifft.*(1-samet)*stiffup;
stiffx=min(1e15,max(stiffx,1));
stiffy=min(1e15,max(stiffy,1));
stifft=min(1e15,max(stifft,1));
maxmove=1e-2;
deltax=max(-maxmove,min(deltax,maxmove));
deltay=max(-maxmove,min(deltay,maxmove));
deltat=max(-maxmove,min(deltat,maxmove));
px=x;
py=y;
pt=t;
cond3=0;
x=x+deltax;
y=y+deltay;
t=t+deltat;
r=x*0;
x(1)=x0;
y(1)=y0;
t(1)=t0;
x(n)=xf;
y(n)=yf;
t(n)=tf;
deltax=x-px;
deltay=y-py;
deltat=t-pt;
count2=count2+1;
totaldelta=sum(sqrt((px-x).^2+(py-y).^2+(pt-t).^2));
if (totaldelta/n<alpha/10)&&(count2>lastadd+1e2)
  [a b]=size(trackx);
  xa=(2*T*x'/n)';
  ya=(2*T*y'/n)';
  ta=(2*T*t'/n)';
  xa(1)=mean(x);
  ya(1)=mean(y);
```

```
ta(1)=mean(t);
trackx(1:m,b+1)=xa;
tracky(1:m,b+1)=ya;
trackt(1:m,b+1)=ta;
trackcount(b+1)=count2;
tracka(b+1)=alpha;
dd=cos(t).*dx+sin(t).*dy;
da=-sin(t).*dx+cos(t).*dy;
db=-r.*cos(t).*dx-r.*sin(t).*dy+dt;
trackl2(b+1)=sum(sqrt(da.^2).*[0 diff(s)]);
trackl(b+1)=sum(sqrt(dd.^2+db.^2*beta).*[0 diff(s)]);
if b>20
  xa=xa+mean(diff(trackx(:,b-8:b+1)'))/2;
  ya=ya+mean(diff(tracky(:,b-8:b+1)'))/2;
  ta=ta+mean(diff(trackt(:,b-8:b+1)'))/2;
end
x=xa*T;
y=ya*T;
t=ta*T;
r=x*0;
alpha=alpha*.95;
lastadd=count2;
if (n<alpha^(-1))&&(n<100)
  dxa=(2*T*deltax'/n)';
  dya=(2*T*deltay'/n)';
  dta=(2*T*deltat'/n)';
  sxa=(2*T*stiffx'/n)';
  sya=(2*T*stiffy'/n)';
  sta=(2*T*stifft'/n)';
  oldm=m;
  if n<100
    n=n+50;
  end
  if n>100
    n=100;
  end
```

```
      m=n;
      cheby=sort(cos((2*(1:n)-1)/2/n*pi));
      s=cheby/2+.5;
      T=cheby*0+1;
      U=T;
      T(2,:)=cheby;
      U(2,:)=2*cheby;
      for count=3:m
        T(count,:)=2*cheby.*T(count-1,:)-T(count-2,:);
        U(count,:)=2*cheby.*U(count-1,:)-U(count-2,:);
      end
      x=mean(x)+xa(2:oldm)*T(2:oldm,:);
      y=mean(y)+ya(2:oldm)*T(2:oldm,:);
      t=mean(t)+ta(2:oldm)*T(2:oldm,:);
      r=x*0;
      deltax=mean(deltax)+dxa(2:oldm)*T(2:oldm,:);
      deltay=mean(deltay)+dya(2:oldm)*T(2:oldm,:);
      deltat=mean(deltat)+dta(2:oldm)*T(2:oldm,:);
      stiffx=mean(stiffx)+sxa(2:oldm)*T(2:oldm,:);
      stiffy=mean(stiffy)+sya(2:oldm)*T(2:oldm,:);
      stifft=mean(stifft)+sta(2:oldm)*T(2:oldm,:);
      addn=[addn count2];
      lastadd=count2;
    end
  end
end
fname=['tank.' '0'+loop-1];
save("-binary",fname);
end
```

A.4.3 Jacobi Fields

This is the code for calculating the Jacobi fields.

```
[a b]=size(trackx);
x=interp1(s,trackx(:,b)'*T,(0:1e5)/1e5,'spline','extrap');
y=interp1(s,tracky(:,b)'*T,(0:1e5)/1e5,'spline','extrap');
t=interp1(s,trackt(:,b)'*T,(0:1e5)/1e5,'spline','extrap');
```

```
dx=diff(x);
dy=diff(y);
dt=diff(t);
alpha=tracka(b);
Ru1112=((-1+alpha).^3.*sin(2.*t))./(8.*alpha.^2.*beta);
Ru1121=-((-1+alpha).^3.*sin(2.*t))./(8.*alpha.^2.*beta);
Ru1212=-((-1+alpha).^2.*(-1-alpha+(-1+alpha).*cos(2.*t)))./(8.*alpha.^2.*beta);
Ru1221=((-1+alpha).^2.*(-1-alpha+(-1+alpha).*cos(2.*t)))./(8.*alpha.^2.*beta);
Ru1313=((-1+alpha).*(1-alpha+2.*(1+alpha).*cos(2.*t)))./(4.*alpha);
Ru1323=((-1+alpha).*(1+alpha).*cos(t).*sin(t))./alpha;
Ru1331=-((-1+alpha).*(1-alpha+2.*(1+alpha).*cos(2.*t)))./(4.*alpha);
Ru1332=-(((-1+alpha).*(1+alpha).*cos(t).*sin(t))./alpha);
Ru2112=-((-1+alpha).^2.*(1+alpha+(-1+alpha).*cos(2.*t)))./(8.*alpha.^2.*beta);
Ru2121=((-1+alpha).^2.*(1+alpha+(-1+alpha).*cos(2.*t)))./(8.*alpha.^2.*beta);
Ru2212=-((-1+alpha).^3.*sin(2.*t))./(8.*alpha.^2.*beta);
Ru2221=((-1+alpha).^3.*sin(2.*t))./(8.*alpha.^2.*beta);
Ru2313=((-1+alpha).*(1+alpha).*cos(t).*sin(t))./alpha;
Ru2323=-((-1+alpha).*(-1+alpha+2.*(1+alpha).*cos(2.*t)))./(4.*alpha);
Ru2331=-(((-1+alpha).*(1+alpha).*cos(t).*sin(t))./alpha);
Ru2332=((-1+alpha).*(-1+alpha+2.*(1+alpha).*cos(2.*t)))./(4.*alpha);
Ru3113=-((-1+alpha).*(-1+alpha.^2+(1+6.*alpha+alpha.^2).*cos(2.*t))) \
    ./(8.*alpha.^2.*beta);
Ru3123=-((-1+alpha).*(1+6.*alpha+alpha.^2).*sin(2.*t))./(8.*alpha.^2.*beta);
Ru3131=((-1+alpha).*(-1+alpha.^2+(1+6.*alpha+alpha.^2).*cos(2.*t))) \
    ./(8.*alpha.^2.*beta);
Ru3132=((-1+alpha).*(1+6.*alpha+alpha.^2).*sin(2.*t))./(8.*alpha.^2.*beta);
Ru3213=-((-1+alpha).*(1+6.*alpha+alpha.^2).*sin(2.*t))./(8.*alpha.^2.*beta);
Ru3223=((-1+alpha).*(1-alpha.^2+(1+6.*alpha+alpha.^2).*cos(2.*t))) \
    ./(8.*alpha.^2.*beta);
Ru3231=((-1+alpha).*(1+6.*alpha+alpha.^2).*sin(2.*t))./(8.*alpha.^2.*beta);
Ru3232=-((-1+alpha).*(1-alpha.^2+(1+6.*alpha+alpha.^2).*cos(2.*t))) \
    ./(8.*alpha.^2.*beta);
J1=[0*x' 0*x' 0*x'];
J2=J1;
if (dy(1)==0)&&(dt(1)==0)
  dJ1(:,1)=[0 1 0];
  dJ2(:,1)=[0 0 1];
elseif (dx(1)==0)&&(dt(1)==0)
  dJ1(:,1)=[1 0 0];
  dJ2(:,1)=[0 0 1];
elseif (dx(1)==0)&&(dy(1)==0)
  dJ1(:,1)=[1 0 0];
  dJ2(:,1)=[0 1 0];
else
  dT=[dx(1) dy(1) dt(1)];
  dT=dT/sqrt(dT*dT');
  dJ1=[1 0 0];
  dJ1=dJ1-(dJ1*dT')*dT;
  dJ1=dJ1/sqrt(dJ1*dJ1');
  dJ2=[0 1 0];
  dJ2=dJ2-(dJ2*dT')*dT-(dJ2*dJ1')*dJ1;
  dJ2=dJ2/sqrt(dJ2*dJ2');
end
dT=[dx(1) dy(1) dt(1)];
dJ1=dJ1*sqrt(dT*dT');
dJ2=dJ2*sqrt(dT*dT');
dJ1(1e5,3)=0;
dJ2(1e5,3)=0;
for count=1:1e5
  J1(count+1,:)=J1(count,:)+dJ1(count,:);
  J2(count+1,:)=J2(count,:)+dJ2(count,:);
  dJ1(count+1,1)=dJ1(count,1)-Ru1112(count)*dx(count)*J1(count,1)*dy(count) \
      -Ru1121(count)*dx(count)*J1(count,2)*dx(count) \
      -Ru1212(count)*dy(count)*J1(count,1)*dy(count) \
      -Ru1221(count)*dy(count)*J1(count,2)*dx(count) \
      +dt(count)*(Ru1313(count)*J1(count,1)*dt(count) \
                  -Ru1323(count)*J1(count,2)*dt(count) \
                  -Ru1331(count)*J1(count,3)*dx(count) \
                  -Ru1332(count)*J1(count,3)*dy(count));
  dJ1(count+1,2)=dJ1(count,2)-Ru2112(count)*dx(count)*J1(count,1)*dy(count) \
      -Ru2121(count)*dx(count)*J1(count,2)*dx(count) \
      -Ru2212(count)*dy(count)*J1(count,1)*dy(count) \
    -Ru2221(count)*dy(count)*J1(count,2)*dx(count) \
    +dt(count)*(Ru2313(count)*J1(count,1)*dt(count) \
                -Ru2323(count)*J1(count,2)*dt(count) \
                -Ru2331(count)*J1(count,3)*dx(count) \
                -Ru2332(count)*J1(count,3)*dy(count));
dJ1(count+1,3)=dJ1(count,3)-Ru3113(count)*dx(count)*J1(count,1)*dt(count) \
    -Ru3123(count)*dx(count)*J1(count,2)*dt(count) \
    -Ru3131(count)*dx(count)*J1(count,3)*dx(count) \
    -Ru3132(count)*dx(count)*J1(count,3)*dy(count) \
    +dy(count)*(Ru3213(count)*J1(count,1)*dt(count) \
                -Ru3223(count)*J1(count,2)*dt(count) \
                -Ru3231(count)*J1(count,3)*dx(count) \
                -Ru3232(count)*J1(count,3)*dy(count));
dJ2(count+1,1)=dJ2(count,1)-Ru1112(count)*dx(count)*J2(count,1)*dy(count) \
    -Ru1121(count)*dx(count)*J2(count,2)*dx(count) \
    -Ru1212(count)*dy(count)*J2(count,1)*dy(count) \
    -Ru1221(count)*dy(count)*J2(count,2)*dx(count) \
    +dt(count)*(Ru1313(count)*J2(count,1)*dt(count)
                -Ru1323(count)*J2(count,2)*dt(count) \
                -Ru1331(count)*J2(count,3)*dx(count) \
                -Ru1332(count)*J2(count,3)*dy(count));
dJ2(count+1,2)=dJ2(count,2)-Ru2112(count)*dx(count)*J2(count,1)*dy(count) \
    -Ru2121(count)*dx(count)*J2(count,2)*dx(count) \
    -Ru2212(count)*dy(count)*J2(count,1)*dy(count) \
    -Ru2221(count)*dy(count)*J2(count,2)*dx(count) \
    +dt(count)*(Ru2313(count)*J2(count,1)*dt(count) \
                -Ru2323(count)*J2(count,2)*dt(count) \
                -Ru2331(count)*J2(count,3)*dx(count) \
                -Ru2332(count)*J2(count,3)*dy(count));
dJ2(count+1,3)=dJ2(count,3)-Ru3113(count)*dx(count)*J2(count,1)*dt(count) \
    -Ru3123(count)*dx(count)*J2(count,2)*dt(count) \
    -Ru3131(count)*dx(count)*J2(count,3)*dx(count) \
    -Ru3132(count)*dx(count)*J2(count,3)*dy(count) \
    +dy(count)*(Ru3213(count)*J2(count,1)*dt(count) \
                -Ru3223(count)*J2(count,2)*dt(count) \
                -Ru3231(count)*J2(count,3)*dx(count) \
                  -Ru3232(count)*J2(count,3)*dy(count));
end
for count=1:length(J1)
  if dot(J1(count,:),J1(count,:))~=0
    uJ1(count,:)=J1(count,:)/sqrt(dot(J1(count,:),J1(count,:)));
  else
    uJ1(count,:)=J1(count,:);
  end
  if dot(J2(count,:),J2(count,:))~=0
    uJ2(count,:)=J2(count,:)/sqrt(dot(J2(count,:),J2(count,:)));
  else
    uJ2(count,:)=J2(count,:);
  end
end
clear a b
M=18:1e-5:18.5;
for count=1:length(M)
  mul=M(count);
  a(count,:)=J1(98359,:)-mul*J2(98359,:);
  b(count)=a(count,:)*a(count,:)';
end
plot(M,log(b));grid on
```

A.4.4 Hamiltonian Integration

```
loop=0;
alpha=0.05;
beta=10^(loop/3-1);
x=0;
y=0;
t=0;
px=1e-2;
py=1e-1;
pt=0;
ds=1e-3;
count=1;
while count<1e5
```

```
  %dp_a=-1/2*(dg^bc/dx^a)p_bp_c
  c=cos(t(count));
  s=sin(t(count));
  dptx=c*s*px-(c^2-s^2)*py;
  dpty=-c*s*py;
  dpt=dptx*px+dpty*py;
  pt(count+1)=pt(count)+dpt*ds;
  co_g=[c^2 c*s 0;c*s s^2 0;0 0 1/beta];
  dX=co_g*[px py pt(count)]';
  x(count+1)=x(count)+dX(1)*ds;
  y(count+1)=y(count)+dX(2)*ds;
  t(count+1)=t(count)+dX(3)*ds;
  count=count+1;
end
```

A.4.5 Geodesic Integration

```
clear
load tankjacobi
n=length(x);
%dJ3=dJ3*length(x)/n;
addn=1;
lastadd=1;
plotting=1;
movavg=1;
stiffup=1.5;
stiffdown=.95;
x=rand(1,n)*2-1;
y=rand(1,n)*2-1;
t=rand(1,n)*.1-.05;
x(1)=x0;
y(1)=y0;
t(1)=t0;
x(2)=dJ3(1)+x0;
y(2)=dJ3(2)+y0;
t(2)=dJ3(3)+t0;
basestiff=1e8;
```

```
stiffx=basestiff;
stiffy=basestiff;
stifft=basestiff;
totaldelta=0;
Fx=x;
Fy=y;
Ft=t;
deltax=0;
deltay=0;
deltat=0;
last=0;
count=3;
counter=1;
while count<=n
  dl=1/n;
  dx=(x(count)-x(count-2))/2/dl;
  d2x=(x(count)+x(count-2)-2*x(count-1))/(2*dl^2);
  dy=(y(count)-y(count-2))/2/dl;
  d2y=(y(count)+y(count-2)-2*y(count-1))/(2*dl^2);
  dt=(t(count)-t(count-2))/2/dl;
  d2t=(t(count)+t(count-2)-2*t(count-1))/(2*dl^2);
  s=sin(t(count-1));
  c=cos(t(count-1));
  Fxx=(1/alpha-alpha)*s*c*dt;
  Fxy=(1-1/alpha)*(c^2-alpha*s^2)*dt;
  Fxt=0;
  Fx=d2x+Fxx*dx+Fxy*dy+Fxt*dt;
  Fyx=(1/alpha-1)*(s^2-alpha*c^2)*dt;
  Fyy=(alpha-1/alpha)*c*s*dt;
  Fyt=0;
  Fy=d2y+Fyx*dx+Fyy*dy+Fyt*dt;
  Ftx=(1-1/alpha)/beta*(c*s*dx+(s^2-c^2)*dy);
  Fty=(1/alpha-1)/beta*c*s*dy;
  Ftt=0;
  Ft=d2t+Ftx*dx+Fty*dy+Ftt*dt;
  deltax2=deltax;
```

```
deltay2=deltay;
deltat2=deltat;
deltax=-Fx/stiffx;
deltay=-Fy/stiffy;
deltat=-Ft/stifft;
samex=(sign(deltax)==sign(deltax2));
samey=(sign(deltay)==sign(deltay2));
samet=(sign(deltat)==sign(deltat2));
stiffx=stiffx*samex*stiffdown+stiffx*(1-samex)*stiffup;
stiffy=stiffy*samey*stiffdown+stiffy*(1-samey)*stiffup;
stifft=stifft*samet*stiffdown+stifft*(1-samet)*stiffup;
stiffx=min(1e13,max(stiffx,1));
stiffy=min(1e13,max(stiffy,1));
stifft=min(1e13,max(stifft,1));
maxmove=1e-2;
deltax=max(-maxmove,min(deltax,maxmove));
deltay=max(-maxmove,min(deltay,maxmove));
deltat=max(-maxmove,min(deltat,maxmove));
px=x;
py=y;
pt=t;
x(count)=x(count)+deltax;
y(count)=y(count)+deltay;
t(count)=t(count)+deltat;
x(1)=x0;
y(1)=y0;
t(1)=t0;
x(2)=dJ3(1)+x0;
y(2)=dJ3(2)+y0;
t(2)=dJ3(3)+t0;
deltax=x(count)-px(count);
deltay=y(count)-py(count);
deltat=t(count)-pt(count);
totaldelta=sum(sqrt((px-x).^2+(py-y).^2+(pt-t).^2));
if (totaldelta<1e-10)
  count=count+1;
```

```
    stiffx=basestiff;
    stiffy=basestiff;
    stifft=basestiff;
    if count<=n
      x(count)=x(count-1);
      y(count)=y(count-1);
      t(count)=t(count-1);
    end
  end
  counter=counter+1;
  if mod(counter,50)==1
    [counter totaldelta count mean([stiffx stiffy stifft])]
  end
end
```

A.5 Astrodynamics

For this section, only the problem specific code will be given.

A.5.1 Finite Difference Method

```
%
%Initial setup
%
n=5;
beta=1e-1;
stiffup=1.2;
stiffdown=.95;
basestiff=100;
%
%path endpoints are defined before starting
%
x=x0+(xf-x0)*(0:n-1)/(n-1);
y=y0+(yf-y0)*(0:n-1)/(n-1);
h=h0+(hf-h0)*(0:n-1)/(n-1);
L=L0+(Lf-L0)*(0:n-1)/(n-1);
%
```

```
%stopping condition
%
deltar=1;
while (deltar(length(deltar))<1.5*min(deltar))||(count<5e5)
%
%Calculate some parameters to make the lines somewhat readable
%Then calculate forces
%
  X=x+cos(L);
  Y=y+sin(L);
  t=y.*cos(L)-x.*sin(L);
  r=x.*cos(L)+y.*sin(L);
  xi=1+r;
  Fxh=6*X./h.^2.*dh-(3+6*x.*cos(L)+3*cos(2*L)+2*xi)/2./h./xi.*dx*2 \
      -3*X.*sin(L)./h./xi.*dy*2;
  Fxx=2*X.*cos(L).^2./xi.^2.*dx+X.*sin(2*L)./xi.^2.*dy*2;
  Fxy=2*X.*sin(L).^2./xi.^2.*dy;
  FxLbp=beta*(X.*t.*xi.^2/2./h.^7.*dh*2+X.*xi.^2.*sin(L)/2./h.^6.*dx*2 \
             -X.*xi.^2.*cos(L)/2./h.^6.*dy*2-h.^2.*X*alpha./xi.^2.*dL);
  FxLh=(-4*x.*y.*cos(L)-2*(1+2*x.^2).*y.*cos(2*L)-2*x.*y.*cos(3*L) \
        +sin(L)+4*x.^2.*sin(L)+2*x.*sin(2*L)+2*x.^3.*sin(2*L) \
        -2*x.*y.^2.*sin(2*L)+x.^2.*sin(3*L)-y.^2.*sin(3*L))./(2*h.*xi.^2).*dh*2;
  FxLx=(8*x.*y+11*y.*cos(L)+4*x.*y.*cos(2*L)+y.*cos(3*L)+7*x.*sin(L) \
        +6*sin(2*L)-4*x.^2.*sin(2*L)-x.*sin(3*L))./(8.*xi.^2).*dx*2;
  FxLy=(-6-8*x.^2-17*x.*cos(L)+(4*x.^2-6).*cos(2*L)+x.*cos(3*L) \
        +y.*sin(L)+4*x.*y.*sin(2*L)+y.*sin(3*L))./(8.*xi.^2).*dy*2;
  FxLL=h.^8*alpha.*(X+xi.*cos(L))./xi.^6.*dL;
  FxL=FxLh+FxLx+FxLy+FxLL;
  FxLbn=(h.^5.*(Y+xi.*y+t.*x)./xi.^5.*dh*2 \
         -h.^6.*(Y+xi.*y+t.*x).*cos(L)/2./xi.^6.*dx*2 \
         -h.^6.*(Y+xi.*y+t.*x).*sin(L)/2./xi.^6.*dy*2)/beta;
  Fx=d2x+Fxh.*dh+Fxx.*dx+Fxy.*dy+FxL.*dL+(FxLbp+FxLbn).*dL;
  Fyh=6*Y./h.^2.*dh-3*Y.*cos(L)./h./xi.*dx*2 \
     +(-3-6*y.*sin(L)+3.*cos(2*L)-2*xi)/2./h./xi.*dy*2;
  Fyx=2*Y.*cos(L).^2./xi.^2.*dx+Y.*sin(2*L)./xi.^2.*dy*2;
  Fyy=2*Y.*sin(L).^2./xi.^2.*dy;
```

```
FyLbp=beta*(Y.*t.*xi.^2/2./h.^7.*dh*2+Y.*xi.^2.*sin(L)/2./h.^6.*dx*2 \
            -Y.*xi.^2.*cos(L)/2./h.^6.*dy*2-alpha*h.^2.*Y./xi.^2.*dL);
FyLh=-((1+4*y.^2).*cos(L)+2*x.*(1+2*y.^2).*cos(2*L)+x.^2.*cos(3*L) \
       -y.^2.*cos(3*L)-4*x.*y.*sin(L)+2*y.*sin(2*L)-2*x.^2.*y.*sin(2*L) \
       +2*y.^3.*sin(2*L)+2*x.*y.*sin(3*L))./(2*h.*xi.^2).*dh*2;
FyLx=(6+8*y.^2-x.*cos(L)+(4.*y.^2-6).*cos(2*L)+x.*cos(3*L)+17*y.*sin(L) \
      -4*x.*y.*sin(2*L)+y.*sin(3*L))./(8*xi.^2).*dx*2;
FyLy=-(8*x.*y+7*y.*cos(L)-4*x.*y.*cos(2*L)+y.*cos(3*L)+11*x.*sin(L) \
       +6*sin(2*L)-4*y.^2.*sin(2*L)-x.*sin(3*L))/8./xi.^2.*dy*2;
FyLL=h.^8*alpha.*(Y+xi.*sin(L))./xi.^6.*dL;
FyL=FyLh+FyLx+FyLy+FyLL;
FyLbn=(-h.^5.*((1+x.^2-y.^2).*cos(L)+2*x.*(1+y.*sin(L)))./xi.^5.*dh \
       +h.^6.*(X+xi.*x-t.*y).*cos(L)./2./xi.^6.*dx \
       +h.^6.*sin(L).*((1+x.^2-y.^2).*cos(L) \
                       +2*x.*(1+y.*sin(L)))/2./xi.^6.*dy)/beta*2;
Fy=d2y+Fyh.*dh+Fyx.*dx+Fyy.*dy+FyL.*dL+(FyLbp+FyLbn).*dL;
Fhh=4./h.*dh-3*cos(L)./xi.*dx*2-3*sin(L)./xi.*dy*2;
Fhx=2*h.*cos(L).^2./xi.^2.*dx+h.*sin(2*L)./xi.^2.*dy*2;
Fhy=2*h.*sin(L).^2./xi.^2.*dy*2;
FhLbp=beta*(xi.^2.*t/2./h.^6.*dh*2+xi.^2.*sin(L)/2./h.^5.*dx*2 \
            -xi.^2.*cos(L)/2./h.^5.*dy*2-h.^3*alpha./xi.^2.*dL);
FhL=(-2*t./xi+t/2./xi.^2).*dh*2+h/2./xi.^2.*(y+2*sin(L)+2*t.*cos(L)).*dx*2 \
    +h.*(x.*cos(2*L)+y.*sin(2*L)-2*x-2*cos(L))/2./xi.^2.*dy*2 \
    +h.^9*alpha./xi.^6.*dL;
FhLbn=(h.^6.*t./xi.^5.*dh-h.^7.*t.*cos(L)/2./xi.^6.*dx \
       -h.^7.*t.*sin(L)/2./xi.^6.*dy)/beta*2;
Fh=d2h+Fhh.*dh+Fhx.*dx+Fhy.*dy+FhL.*dL+(FhLbp+FhLbn).*dL;
FLh=-t.*xi.^4./h.^10/alpha.*dh-xi.^4.*sin(L)/2./h.^9/alpha.*dx*2 \
    +xi.^4.*cos(L)/2./h.^9/alpha.*dy*2;
FLhb=4*t.*xi./h.^4/alpha.*dh-(3.*t.*cos(L)+xi.*sin(L))./h.^3/alpha.*dx*2 \
    -(3*t.*sin(L)-xi.*cos(L))./h.^3/alpha.*dy*2;
FLxb=(Y.*cos(L)+t.*cos(L).^2)./h.^2./xi/alpha.*dx \
    +(-3*x.*cos(L)-2*cos(2*L)+x.*cos(3*L)+3*y.*sin(L) \
      +y.*sin(3*L))/4./h.^2./xi/alpha.*dy*2;
FLyb=(-X.*sin(L)+t.*sin(L).^2)./h.^2./xi/alpha.*dy;
FLL=3./h.*dh*2-2*cos(L)./xi.*dx*2-2*sin(L)./xi.*dy*2-2*t./xi.*dL;
```

```
  FLb=(FLhb.*dh+FLxb.*dx+FLyb.*dy)/beta;
  FL=d2L+FLh.*dh+FLL.*dL+FLb*0;
%
%Limits on point movement
%
  maxmove=1e-4;
  mdx=mean(min(maxmove,abs(deltax)));
  mdy=mean(min(maxmove,abs(deltay)));
  mdh=mean(min(maxmove,abs(deltah)));
  maxmove=(Lf-L0)/100;
  mdL=mean(min(maxmove,abs(deltaL)));
  deltax=max(-mdx,min(deltax,mdx));
  deltay=max(-mdy,min(deltay,mdy));
  deltah=max(-mdh,min(deltah,mdh));
  deltaL=max(-mdL,min(deltaL,mdL));
%
%Limits on path range
%
  h=max(0.5,min(3,h));
  x=max(-.6,min(.6,x));
  y=max(-.6,min(.6,y));
  L=max(L0,min(Lf,L));
  L(n)=Lf;
%
%Conditions for decreasing beta
%
  savex(mod(count-1,1e3)+1,:)=x;
  savey(mod(count-1,1e3)+1,:)=y;
  saveh(mod(count-1,1e3)+1,:)=h;
  saveL(mod(count-1,1e3)+1,:)=L;
  totaldelta=sum(sqrt((max(savex)-min(savex)).^2+(max(savey)-min(savey)).^2 \
                    +(max(saveh)-min(saveh)).^2+(max(saveL)-min(saveL)).^2));
  if (totaldelta/n<1e-3)&&(count>lastadd+100)
    if (beta>2.5e-4)
      beta=beta*.95;
    else
```

```
      beta=beta*.99;
    end
%
%Additional ondition for adding points
%
    if (n<1000)%&&(n<beta^(-1))
```

A.5.2 Pseudospectral Method

```
%
%Setup that differs from the finite difference method
%
n=30;
m=n;
beta=0.1;
stiffup=1.7;
stiffdown=.95;
%
%Stopping condition
%
while (length(trackdr)<10) \
      ||((trackdr(length(trackdr))<1.2*min(trackdr(10:length(trackdr)))) \
         &&(deltar<10*min(trackdr(10:length(trackdr)))))
%
%Forces are the same as for finite differences
%
%
%Limits on path range
%
  h=max(0.5,min(2,h));
  x=max(-.6,min(.6,x));
  y=max(-.6,min(.6,y));
  L=max(L0,min(Lf,L));
%
%Conditions for decreasing beta
%
  totaldelta=sum(sqrt((max(savex)-min(savex)).^2+(max(savey)-min(savey)).^2 \
                    +(max(saveh)-min(saveh)).^2+(max(saveL)-min(saveL)).^2));
  if (totaldelta/n<min(beta/10,10*beta^2))&&(count2>lastadd+1e2)
    dar=t./ih.^2.*dh+sin(iL)./ih.*dx-cos(iL)./ih.*dy;
    dat=-xi./ih.^2.*dh+cos(iL)./ih.*dx+sin(iL)./ih.*dy;
    dr=2*ih./xi.*dh-ih.^2.*cos(iL)./xi.^2.*dx-ih.^2.*sin(iL)./xi.^2.*dy;
    dt=ih.^3./xi.^2.*dL;
    index=ceil(count2/100);
    deltav(index)=sum(sqrt(dar.^2+dat.^2))+sum(abs(dt))*alpha;
    deltar(index)=sum(abs(dr));
    deltatot(index)=deltav(index)+deltar(index)/beta;
    trackdv(b+1)=deltav(index);
    trackdr(b+1)=deltar(index);
    trackdtot(b+1)=deltatot(index);
    beta=beta*.95;
    lastadd=count2;
%
%Additional conditions for adding points
%
    if (n<beta^(-1))&&(n<30)
      lastadd=count2;
```

A.5.3 Hamiltonian Integration

```
clear Vx Vy Vh VL PL
addn=1;
lastadd=1;
plotting=1;
movavg=1;
n=1e5;
m=n;
alpha=1e-3;
x=x0;
y=y0;
h=h0;
L=L0;
%px=-.1;
%py=0;
```

```
%ph=5;
%pL=0;
dl=1e-6;
count=1;
Vx=x;
Vy=y;
Vh=h;
VL=L;
Px=px;
Py=py;
Ph=ph;
PL=pL;
while (count<n)&&(L<1)
  X=x+cos(L);
  Y=y+sin(L);
  t=y.*cos(L)-x.*sin(L);
  r=x.*cos(L)+y.*sin(L);
  xi=1+r;
  c=cos(L);
  s=sin(L);
  co_g=h^2/xi^2*[h^2 h*(X+xi*c) h*(Y+xi*s);h*(X+xi*c) \
               xi^2+2*X*xi*c+X^2 xi*(Y*c+X*s)+X*Y;h*(Y+xi*s) \
               xi*(Y*c+X*s)+X*Y xi^2+2*Y*xi*s+Y^2];
  D=co_g*[ph;px;py];
  dh=D(1);
  dx=D(2);
  dy=D(3);
  dL=xi^4/h^6/alpha*pL;
  Gxhh=6*X./h.^2;
  Gxhx=-(3+6*x.*cos(L)+3*cos(2*L)+2*xi)/2./h./xi;
  Gxhy=-3*X.*sin(L)./h./xi;
  GxhL=(-4*x.*y.*cos(L)-2*(1+2*x.^2).*y.*cos(2*L)-2*x.*y.*cos(3*L)+sin(L) \
       +4*x.^2.*sin(L)+2*x.*sin(2*L)+2*x.^3.*sin(2*L)-2*x.*y.^2.*sin(2*L) \
       +x.^2.*sin(3*L)-y.^2.*sin(3*L))./(2*h.*xi.^2);
  Gxxx=2*X.*cos(L).^2./xi.^2;
  Gxxy=X.*sin(2*L)./xi.^2;
```

```
GxxL=(8*x.*y+11*y.*cos(L)+4*x.*y.*cos(2*L)+y.*cos(3*L)+7*x.*sin(L) \
      +6*sin(2*L)-4*x.^2.*sin(2*L)-x.*sin(3*L))./(8.*xi.^2);
Gxyy=2*X.*sin(L).^2./xi.^2;
GxyL=(-6-8*x.^2-17*x.*cos(L)+(4*x.^2-6).*cos(2*L)+x.*cos(3*L)+y.*sin(L) \
      +4*x.*y.*sin(2*L)+y.*sin(3*L))./(8.*xi.^2);
GxLL=h.^8*alpha.*(X+xi.*cos(L))./xi.^6;
Gyhh=6*Y./h.^2;
Gyhx=-3*Y.*cos(L)./h./xi;
Gyhy=(-3-6*y.*sin(L)+3.*cos(2*L)-2*xi)/2./h./xi;
GyhL=-((1+4*y.^2).*cos(L)+2*x.*(1+2*y.^2).*cos(2*L)+x.^2.*cos(3*L) \
       -y.^2.*cos(3*L)-4*x.*y.*sin(L)+2*y.*sin(2*L)-2*x.^2.*y.*sin(2*L) \
       +2*y.^3.*sin(2*L)+2*x.*y.*sin(3*L))./(2*h.*xi.^2);
Gyxx=2*Y.*cos(L).^2./xi.^2;
Gyxy=Y.*sin(2*L)./xi.^2;
GyxL=(6+8*y.^2-x.*cos(L)+(4.*y.^2-6).*cos(2*L)+x.*cos(3*L)+17*y.*sin(L) \
      -4*x.*y.*sin(2*L)+y.*sin(3*L))./(8*xi.^2);
Gyyy=2*Y.*sin(L).^2./xi.^2;
GyyL=-(8*x.*y+7*y.*cos(L)-4*x.*y.*cos(2*L)+y.*cos(3*L)+11*x.*sin(L) \
       +6*sin(2*L)-4*y.^2.*sin(2*L)-x.*sin(3*L))/8./xi.^2;
GyLL=h.^8*alpha.*(Y+xi.*sin(L))./xi.^6;
Ghhh=4./h;
Ghhx=-3*cos(L)./xi;
Ghhy=-3*sin(L)./xi;
GhhL=(-2*t./xi+t/2./xi.^2);
Ghxx=2*h.*cos(L).^2./xi.^2;
Ghxy=h.*sin(2*L)./xi.^2;
GhxL=h/2./xi.^2.*(y+2*sin(L)+2*t.*cos(L));
Ghyy=2*h.*sin(L).^2./xi.^2;
GhyL=h.*(x.*cos(2*L)+y.*sin(2*L)-2*x-2*cos(L))/2./xi.^2;
GhLL=h.^9*alpha./xi.^6;
GLhh=-t.*xi.^4./h.^10/alpha;
GLhx=-xi.^4.*sin(L)/2./h.^9/alpha;
GLhy=xi.^4.*cos(L)/2./h.^9/alpha;
GLhL=3./h;
GLxx=0;
GLxy=0;
```

```
GLxL=-2*cos(L)./xi;
GLyy=0;
GLyL=-2*sin(L)./xi;
GLLL=-2*t./xi;
Fhh=Ghhh*dh+Ghhx*dx+Ghhy*dy+GhhL*dL;
Fhx=Ghhx*dh+Ghxx*dx+Ghxy*dy+GhxL*dL;
Fhy=Ghhy*dh+Ghxy*dx+Ghyy*dy+GhyL*dL;
FhL=GhhL*dh+GhxL*dx+GhyL*dy+GhLL*dL;
Fxh=Gxhh*dh+Gxhx*dx+Gxhy*dy+GxhL*dL;
Fxx=Gxhx*dh+Gxxx*dx+Gxxy*dy+GxxL*dL;
Fxy=Gxhy*dh+Gxxy*dx+Gxyy*dy+GxyL*dL;
FxL=GxhL*dh+GxxL*dx+GxyL*dy+GxLL*dL;
Fyh=Gyhh*dh+Gyhx*dx+Gyhy*dy+GyhL*dL;
Fyx=Gyhx*dh+Gyxx*dx+Gyxy*dy+GyxL*dL;
Fyy=Gyhy*dh+Gyxy*dx+Gyyy*dy+GyyL*dL;
FyL=GyhL*dh+GyxL*dx+GyyL*dy+GyLL*dL;
FLh=GLhh*dh+GLhx*dx+GLhy*dy+GLhL*dL;
FLx=GLhx*dh+GLxx*dx+GLxy*dy+GLxL*dL;
FLy=GLhy*dh+GLxy*dx+GLyy*dy+GLyL*dL;
FLL=GLhL*dh+GLxL*dx+GLyL*dy+GLLL*dL;
Fh=Fhh*ph+Fxh*px+Fyh*py+FLh*pL;
Fx=Fhx*ph+Fxx*px+Fyx*py+FLx*pL;
Fy=Fhy*ph+Fxy*px+Fyy*py+FLy*pL;
FL=FhL*ph+FxL*px+FyL*py+FLL*pL;
ph=ph+Fh*dl;
px=px+Fx*dl;
py=py+Fh*dl;
pL=pL+FL*dl;
x=x+dx*dl;
y=y+dy*dl;
h=h+dh*dl;
L=L+dL*dl;
count=count+1;
Vx(count)=x;
Vy(count)=y;
Vh(count)=h;
```

```
  VL(count)=L;
  Px(count)=px;
  Py(count)=py;
  Ph(count)=ph;
  PL(count)=pL;
end
xf=x;
yf=y;
Lf=L;
hf=h;
```

Bibliography

[1] I.M. Ross and F. Fahroo. A perspective on methods for trajectory optimization. In *Proceedings of the AIAA/AAS Astrodynamics Specialist Conference, Monterey, CA*, pages 5–8, 2002.

[2] J.K. Whiting. Orbital Transfer Trajectory Optimization. Master's thesis, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, 2004.

[3] James K Whiting. Three-dimensional low-thrust trajectory optimization, with applications. Huntsville, Alabama, 2003. 39thAIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, AIAA.

[4] Vyacheslav Ptukhov. Optimal multirevolution transfers between non-coplanar elliptical orbits.

[5] V. Coverstone-Carroll and S.N. Williams. Optimal low thrust trajectories using differential inclusion concepts. *Journal of the Astronautical Sciences*, 42:379–393, 1994.

[6] Micahel Spivak. *A Comprehensive Introduction to Differential Geometry*, volume I-V. Publish or Perish, Inc., Houston, Texas, third edition, 1999.

[7] Chris J Isham. *Modern Differential Geometry for Physicists.* World Scientific Publishing Company, Singapore, second edition, 1999.

[8] Richard Montgomery. *A Tour of Subriemannian Geometries, Their Geodesics and Applications.* American Mathematical Society, Providence, Rhode Island, 2002.

[9] Ovidiu Calin and Der-Chen Chang. *Sub-Riemannian Geometry General Theory and Examples.* Cambridge University Press, New York, New York, 2009.

[10] Manfredo Perdigão do Carmo. *Riemannian Geometry.* Birkhäuser, Boston, 1992.

[11] R. Montgomery. A survey of singular curves in sub-Riemannian geometry. *Journal of Dynamical and Control Systems*, 1(1):49–90, 1995.

[12] Jürgen Jost. *Riemannian Geometry and Geometric Analysis.* Springer, Berlin, Germany, fifth edition, 2008.