SubRiemannian geodesics and cubics for efficient quantum circuits

Michael Swaddle
School of Physics

Supervisors

Lyle Noakes
School of Mathematics and Statistics

Jingbo Wang
School of Physics

September 6, 2017
This thesis is presented for the requirements of the degree of Master of Philosophy of the University of Western Australia.
Acknowledgements

We would like to thank Harry Smallbone, Kooper de Lacy, and Liam Salter for valuable thoughts and discussion. This research was supported by an Australian Government Research Training Program (RTP) Scholarship.
Abstract

Nielsen et al. first argued that efficient quantum circuits can be obtained from special curves called geodesics. Previous work focused mainly on computing geodesics in Riemannian manifolds equipped with a penalty metric where the penalty was taken to infinity. Taking such limits seems problematic, because it is not clear that all extremals of a limiting optimal control problem can be arrived at as limits of solutions. To rectify this, we examine subRiemannian geodesics, using the Pontryagin Maximum Principle on the special unitary group SU($2^n$) to obtain equations for the normal subRiemannian geodesics. The normal subRiemannian geodesics were found to coincide with the infinite penalty limit.

However, the infinite penalty limit does give all the subRiemannian geodesics. There are potentially abnormal geodesics. These abnormal geodesics, which do not satisfy the normal differential equations. Potentially the abnormal geodesics may have lower energy, and hence generate a more efficient quantum circuit. Sometimes abnormals can be ruled out via contradiction. However in other cases it is possible to construct new equations for abnormals.

In SU(8), the space of operations on three qubits, we allow subRiemannian geodesics to move tangent to directions corresponding to one and two qubit operations. In this case, we find new closed form solutions to the normal geodesics equations. Additionally, several examples of abnormal geodesics are constructed. In higher dimensional groups a series solution to the normal geodesic equations is also provided.

Furthermore we present numerical solutions to the normal geodesic boundary value problem in SU($2^n$) for some $n$. Building on numerical methods found in, we find that a modification of bounding the energy leads to lower energy geodesics. Unfortunately, the geodesic boundary value problem is computationally expensive to solve. We consider a new discrete method to compute geodesics, which is more efficient. Instead of solving the differential equations describing geodesics, a discrete curve is found by optimising the energy and error directly.

Geodesics, however, may not be the most accurate continuous description of an efficient quantum circuit. To refine Nielsen et al.’s original ideas we introduce a new type of variational curve called a subRiemannian cubic. We derive their equations from the Pontryagin Maximum Principle and examine the behaviour of subRiemannian cubics in some simple groups.
Solving the geodesic and cubic boundary value problems is still a difficult task. In the search of a better practical tool, we investigate whether neural networks can be trained to help generate quantum circuits. A correctly trained neural network should reasonably guess the types, number and approximate order of gates required to synthesise a $U$. This guess could then be refined with another optimisation algorithm.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledgements</td>
<td>i</td>
</tr>
<tr>
<td>Abstract</td>
<td>iii</td>
</tr>
<tr>
<td><strong>1 Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Quantum Computation</td>
<td></td>
</tr>
<tr>
<td>1.2 Basic background</td>
<td>2</td>
</tr>
<tr>
<td>1.2.1 Pauli basis</td>
<td>2</td>
</tr>
<tr>
<td>1.2.2 Quqits and generalised Pauli matrices</td>
<td>3</td>
</tr>
<tr>
<td>1.2.3 Lie product formula</td>
<td>3</td>
</tr>
<tr>
<td>1.3 Explicit Circuits</td>
<td>4</td>
</tr>
<tr>
<td>1.3.1 Circuits for basis elements</td>
<td>4</td>
</tr>
<tr>
<td><strong>2 SubRiemannian Geodesics</strong></td>
<td>6</td>
</tr>
<tr>
<td>2.1 SubRiemannian manifolds</td>
<td>6</td>
</tr>
<tr>
<td>2.2 SubRiemannian geodesic equations</td>
<td>8</td>
</tr>
<tr>
<td>2.2.1 Normal case</td>
<td>9</td>
</tr>
<tr>
<td>2.2.2 Abnormal case</td>
<td>11</td>
</tr>
<tr>
<td>2.3 Analytical results</td>
<td>11</td>
</tr>
<tr>
<td>2.3.1 SU(2)</td>
<td>12</td>
</tr>
<tr>
<td>2.3.2 SU(8)</td>
<td>12</td>
</tr>
<tr>
<td>2.3.3 SU(2^n)</td>
<td>15</td>
</tr>
<tr>
<td>2.4 Higher Order PMP</td>
<td>17</td>
</tr>
<tr>
<td>2.4.1 General approach</td>
<td>17</td>
</tr>
<tr>
<td>2.4.2 Abnormal geodesics</td>
<td>20</td>
</tr>
<tr>
<td>2.5 Magnus expansions</td>
<td>21</td>
</tr>
<tr>
<td>2.6 Discretising subRiemannian geodesics</td>
<td>23</td>
</tr>
<tr>
<td><strong>3 SubRiemannian cubics</strong></td>
<td>27</td>
</tr>
<tr>
<td>3.1 Background</td>
<td>27</td>
</tr>
<tr>
<td>3.2 Riemannian cubics</td>
<td>28</td>
</tr>
<tr>
<td>3.3 SubRiemannian cubics</td>
<td>30</td>
</tr>
<tr>
<td>3.3.1 Normal case</td>
<td>32</td>
</tr>
<tr>
<td>3.3.2 Abnormal case – SU(2)</td>
<td>33</td>
</tr>
<tr>
<td>3.3.3 SubRiemannian cubics with tension</td>
<td>33</td>
</tr>
<tr>
<td>3.3.4 Higher Order PMP for cubics</td>
<td>34</td>
</tr>
</tbody>
</table>
1. Introduction

1.1 Quantum Computation

In 1982, Richard Feynman showed that a classical Turing machine would not be able to efficiently simulate quantum mechanical systems \(^7\). Feynman went on to propose a model of computation based on quantum mechanics, which would not suffer the same limitations. Feynman’s ideas were later refined by Deutsch who proposed a universal quantum computer \(^8\). In this scheme, computation is performed by a series of quantum gates, which are the quantum analog to classical binary logic gates. A series of gates is called a quantum circuit \(^9\). Quantum gates act on qubits which are the quantum analog of a classical bit.

Seth Lloyd later proved that a quantum computer would be able to simulate a quantum mechanical system efficiently as long as they evolve according to local interactions \(^10\). Equivalently, this can be stated as: given some special unitary operation \(U \in \text{SU}(2^n)\), \(U^{-1}U = I\), there exists some quantum circuit that approximates \(U\), where \(n\) is the number of qubits. One pertinent question that remains is how to find the circuit which implements this \(U\). In certain situations the circuit can be found exactly. However in general it is a difficult problem, and it is acceptable to approximate \(U\). Previously \(U\) has been found via expensive algebraic means \(^11\)--\(^14\). Another novel attempt at finding an approximate \(U\) has been to use the tools of Riemannian geometry \(^1\)--\(^5\).

Michael Nielsen et al originally proposed calculating special curves called geodesics between two points, \(I\) and \(U\) in \(\text{SU}(2^n)\). Geodesics are fixed points of the energy functional \(^15\). Nielsen et al claimed that when an energy minimising geodesic is discretised into a quantum circuit, this would efficiently simulate \(U\) \(^1\)--\(^5\). In practice however, finding the geodesics is a difficult task. Computing geodesics requires one to solve a boundary value problem in a high dimensional space. Furthermore, Nielsen et al originally formulated the problem on a Riemannian manifold equipped with a so called penalty metric, where the penalty was made large. This complicated solving the boundary value problem \(^6\).

The Nielsen et al approach can be refined by considering subRiemannian geodesics. A subRiemannian geodesic is only allowed to evolve in directions from a horizontal subspace of the tangent space \(^16\). However, geodesics may not provide the most accurate description of an efficient quantum circuit.
A circuit to implement $U$ is considered efficient if it uses a polynomial number of gates. Roughly, the application of a quantum gate can be viewed as an acceleration in $U(2^n)$. To try and more accurately measure this, we investigate another class of curves called cubics \[17\]. Cubics are curves which minimise the mean squared norm of the covariant acceleration.

This approach still involves solving a complicated boundary value problem. For a practical tool, a much faster methodology to synthesise a $U$ is required. With recent advances in computing power, *neural networks* are an attractive option. We propose that a neural network be trained to produce approximate subRiemannian geodesics.

### 1.2 Basic background

#### 1.2.1 Pauli basis

Operations on $n$ qubits are represented as unitary matrices from the unitary group $U(2^n)$. Associated with the Lie group is the Lie algebra. The Lie algebra for $U(2^n)$, $\mathfrak{u}(2^n)$, the space of skew-Hermitian matrices, can be constructed out of Kronecker products of the 3 Pauli matrices;

$$\sigma_0 = I_{2 \times 2}, \quad \sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_2 = \begin{pmatrix} 0 & i \\ -i & 0 \end{pmatrix}, \quad \sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$ 

Then the set of all $n$-fold Kronecker products of the form

$$\frac{i}{\sqrt{2^n}} \sigma_{\mu_1} \otimes \cdots \otimes \sigma_{\mu_n},$$

where $\mu_i = 0, 1, 2, 3$, forms a basis for $\mathfrak{u}(2^n)$. Additionally it is often sufficient to work in $SU(2^n)$ instead. In section (1.3), exponentials of these basis elements are constructed in terms of quantum gates. To form a basis for $\mathfrak{su}(2^n)$ simply exclude the Kronecker product of identity matrices from the basis of $\mathfrak{u}(2^n)$. We use the nomenclature that $m$-body refers to a basis element containing $m \leq n$ Pauli matrices in the Kronecker product. Additionally a basis element will be referred to as an *instruction*.

Furthermore, the set of one and two body basis elements, $\subset \mathfrak{su}(2^n)$ are *bracket generating*. This means repeated Lie brackets, of one and two body matrices can span the whole of $\mathfrak{su}(2^n)$. 

1.2 Basic background

1.2.2 Qudits and generalised Pauli matrices

Qudits are a generalisation of a qubit, and transform under $U(d^n)$, $d > 2$. The Lie algebra for this space can be constructed out $n$-fold Kronecker products of the $d \times d$ generalised Pauli matrices $\lambda$, and the $d \times d$ identity matrix.

Alternatively, the set of all generalised Pauli matrices of size $(d^n) \times (d^n)$ dimension form a basis, but the Kronecker product adds additional structure, as it neatly separates the number of qudits the exponential of the instruction acts on.

Let $S_{rs}$ represent a $d \times d$ matrix with a 1 in the $r, s$ entry, and 0 elsewhere. Define the following set of matrices over $\mathbb{C}^{d \times d}$:

- For $1 \leq r \leq d - 1$:
  \[ \lambda_1^r = \frac{2}{r(r+1)} \left( \sum_{s=1}^{m} S_{ss} - rS_{r+1 r+1} \right) \]

- For $1 \leq s < r \leq d$:
  \[ \lambda_2^{rs} = -i(S_{sr} - S_{rs}) \]

- For $1 \leq r < s \leq d$:
  \[ \lambda_3^{rs} = S_{rs} + S_{sr} \]

Along with the $d \times d$ identity matrix, the set

\[ \frac{i}{\sqrt{d}} \{ I, \lambda_1^r, \lambda_2^{rs}, \lambda_3^{rs} \} \]

forms a basis for $u(d)$ \cite{18}. This is extended to a basis for $u(d^n)$ by taking Kronecker products with the $d \times d$ identity matrix in a way analogous to $u(2^n)$. Like the qubit case, it is convenient to consider $SU(d^n)$ instead.

1.2.3 Lie product formula

The Lie product formula gives a rudimentary method to decompose arbitrary $U \in SU(2^n)$ in terms of products of matrix exponentials

\[ \exp \left( \sum_{i=1}^{m} c_i t \tau_i \right) = \lim_{n \to \infty} (\exp(c_1 t/n \tau_1) \ldots \exp(c_m t/n \tau_m))^n. \]

If this expansion is truncated at some finite $n$,

\[ \exp \left( \sum_{i=1}^{m} c_i t \tau_i \right) = (\exp(c_1 t/n \tau_1) \ldots \exp(c_m t/n \tau_m))^n + O(t/n), \]
it can be used to approximate $U$. Higher order variants of the Lie product formula exist
\[
\exp \left( \sum_{i=1}^{m} c_i t \tau_i \right) = \lim_{n \to \infty} \left( \exp(c_m t/2n \tau_m) \right) \ldots \\
\ldots \exp(c_2 t/2n \tau_2) \exp(c_1 t/n \tau_1) \exp(-c_2 t/2n \tau_2) \\
\ldots \exp(-c_m t/2n \tau_m) \right)^n,
\]
where truncation has error $O(t/n^2)$. The Lie product formula and its variants can be used as a rudimentary starting point to construct circuits. One significant disadvantage is that given $\log(U) = c_i \tau_i$ in general has $\dim(\log(U)) \propto 4^n - 1$ components, a circuit generated by the Lie product is hopelessly impractical. If a circuit requires exponentially many operations, then there is little gain over classical simulation. One option is to examine $U$ with some structure which enables the design of efficient quantum circuits. Another option is to use the tools of Riemannian geometry, which is investigated in the following chapters (2-4).

1.3 Explicit Circuits

1.3.1 Circuits for basis elements

For each basis element, $\tau_i$ in the Pauli basis for $\mathfrak{su}(2^n)$, it is possible to construct a circuit for the exponential, $\exp(\vartheta \tau_i)$ in $\text{SU}(2^n)$. The first step is to observe $\exp(i \vartheta \sigma_3 \otimes \cdots \otimes \sigma_3)$ has the following circuit representation

\[
\begin{array}{c}
\cdots \\
\cdots \\
R_3(\vartheta)
\end{array}
\]

where $\cdots$ denotes staggered CNOT gates. When there is an identity matrix in the place of a $\sigma_3$, then the staggered CNOT gates skip the respective qubit. For example $\exp(i \vartheta \sigma_3 \otimes I \otimes \sigma_3)$, has the circuit

\[
\begin{array}{c}
\cdots \\
R_3(\vartheta)
\end{array}
\]
To obtain circuits for exponentials which contain $\sigma_1$ and $\sigma_2$ terms, make use of the Hadamard gate, denoted $H$, and the $Y$ gate. They have the following matrix representations:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad Y = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix},$$

and have the following properties

$$Y \sigma_3 Y^\dagger = \sigma_2, \quad H \sigma_3 H^\dagger = \sigma_1.$$

These gates can be used to swap the indices in the Kronecker product $i\sigma_3 \otimes \cdots \otimes \sigma_3$. For example $\exp(i \vartheta \sigma_1 \otimes I \otimes \sigma_2)$ has the circuit

![Circuit Diagram](image)

This can be seen from the formula for the exponential of a single basis element. Since the circuit for

$$\exp(i \vartheta \sigma_3 \otimes \cdots \otimes \underbrace{\sigma_3}_\text{j-th entry} \otimes \cdots \otimes \sigma_3)$$

is known, to swap the $j$-th sigma, simply apply the $H$ or $Y$ gate to the $j$-th qubit. For example, to obtain a $\sigma_1$ in the $j$th entry, let $\tilde{H} = I_{2\times 2} \otimes \cdots \otimes H \otimes \cdots I_{2\times 2}$

$$\tilde{H} \exp(i \vartheta \sigma_3 \otimes \cdots \otimes \sigma_3 \otimes \cdots \otimes \sigma_3) \tilde{H}^\dagger$$

$$= \tilde{H} (\cos(\vartheta) I_{2^n \times 2^n} + i \sin(\vartheta) (\sigma_3 \otimes \cdots \otimes \sigma_3 \otimes \cdots \otimes \sigma_3)) \tilde{H}^\dagger$$

$$= \cos(\vartheta) I_{2^n \times 2^n} + i \sin(\vartheta) (\sigma_3 \otimes \cdots \otimes \sigma_1 \otimes \cdots \otimes \sigma_3)$$

$$= \exp(i \vartheta \sigma_3 \otimes \cdots \otimes \sigma_1 \otimes \cdots \otimes \sigma_3)$$

Note that such structures will apply to qudits. Instead of using the qubit CNOT gate, the above constructions apply equally with two qudit based operations. These gates can be the building blocks for more complicated circuits. In the Nielsen approach, these circuit elements would be used to discretise geodesics.
2. SubRiemannian Geodesics

2.1 SubRiemannian manifolds

It has been argued \cite{1, 3, 19} that efficient quantum circuits correspond to special curves called \textit{geodesics}. Previous work has focused on computing geodesics in Riemannian manifolds (for a good reference to Riemannian geometry see \cite{20}), equipped with a penalty metric where the penalty needs to be taken to infinity \cite{1, 3, 19, 21}. Taking such limits seems problematic, because it is not clear that all extremals of a limiting optimal control problem can be arrived at as limits of solutions. This can be rectified by using \textit{subRiemannian geometry}

Instead of applying a large penalty to the undesirable instructions, they can be forbidden entirely. To do this, the problem can be recast in terms of subRiemannian geometry. In subRiemannian geometry curves are only allowed to move tangent to a \textit{horizontal} subspace of the tangent space. However if this horizontal subspace is bracket generating, then the curve is able to join any two arbitrary points in the subRiemannian manifold \cite{16}. In the context of quantum computing, allowed directions come from a \textit{universal gate set}. A universal gate set is a set of unitary matrices which generate $\text{SU}(2^n)$ \cite{9}. The matrix logarithms of universal gates would form a basis for a bracket generating subset of $\text{su}(2^n)$. For clarity define the matrix logarithm of a gate as an \textit{instruction}. As seen previously, the Pauli basis forms a bracket generating set. Other work \cite{22, 23} has considered qudit based quantum computers respectively, which can be considered as operations in $\text{SU}(3^n)$ or $\text{SU}(d^n)$. The equations describing subRiemannian geodesics given here also apply to these groups.

Given a differentiable manifold $M$ a \textit{distribution} $\Delta$, is a subbundle of the tangent bundle $TM$, $\Delta \subset TM$. A velocity field, $\dot{x}$ is called \textit{horizontal} if $\dot{x} \in \Delta$. Let $g$ be a smooth positive-definite quadratic form on $\Delta$. A subRiemannian manifold is then defined by the collection $(M, \Delta, g)$. A subRiemannian manifold has a naturally defined metric, taken to be the infimum of the lengths $\mathcal{L}$, of curves $x$, joining two points

$$d(a, b) = \inf \mathcal{L}[x] = \inf \int_0^1 dt \ (g(\dot{x}, \dot{x}))^{\frac{1}{2}},$$

where $x(0) = a$, and $x(1) = b$. When $M$ is a Lie group $G$, $\Delta$ can be identified with a bracket generating subset of the Lie algebra of $G$, $\mathfrak{g}$. This can be done through \textit{left-} or \textit{right-Lie reduction}. To follow the convention set in the Schrödinger equation, we
will use right-Lie reduction.

For simplicity, from now on let $\Delta \subset \mathfrak{g}$ be the right-Lie reduction of the distribution.

To compute subRiemannian geodesics, it is much easier to find stationary points of the energy functional

$$E[x] = \frac{1}{2} \int_0^1 dt \, g(\dot{x}, \dot{x}).$$

This is because the energy functional is invariant under reparameterisations. There are infinitely many curves of the same energy but with lengths dependent on parameterisation. Minimising the energy finds the minimal length curve uniquely parameterised by arc length.

The subRiemannian geodesic equations on a Lie group $G$ may be derived from the Pontryagin Maximum Principle (PMP) \[24\]. Traditionally the PMP applies to dynamical systems $x \in \mathbb{R}^n$ \[25\]. While it can be written for systems on smooth manifolds \[26\], this adds additional complexity. To simplify things, use the traditional PMP by noting that $SU(2^n) \subset \mathbb{R}^{4^n}$. Then given that $x$ is defined by the Schrödinger equation

$$\dot{x} = ux$$

since $u$ is in $\mathfrak{su}(2^n)$, $x$ will always be contained in $SU(2^n)$.

Given some dynamical system

$$\dot{x} = p(x, u), \quad x \in \mathbb{R}^n, \quad u \in \Delta \subset \mathbb{R}^m$$

$$x(a) = x_0, \quad x(b) = x_1,$$

to minimise the functional

$$S[x] = \int_a^b dt \, \psi(x, u)$$

we can use the Pontryagin Maximum Principle.

**Theorem 2.1.1 (Pontryagin Maximum Principle).** The optimal control $u$ and state trajectory $x$ must minimise the Hamiltonian $H$,

$$H(u, x, \lambda, \nu) = \lambda(p(x, u)) + \nu \psi(x, u)$$

so that

$$H(u, x, \lambda, \nu) = \max_{u^* \in \Delta} H(u^*, x, \lambda, \nu),$$

(2.2)
for all permissible controls $u^*$. Additionally the costate equations

$$
\dot{\lambda} = -dH_x, \\
(\lambda, \nu) \neq (0,0), \\
\nu \leq 0,
$$

for some $\lambda \in TM^*$ and, $\nu \in \mathbb{R}$. must be satisfied.

$\Delta$ as a subset of $\mathfrak{g}$ can also be identified with a real vector space, which satisfies the PMP. When the PMP succeeds in finding minima of $S[x]$ the solutions are termed normal. However if $\nu = 0$ typically the PMP does not provide enough information to determine fixed points. The PMP is analogous to Lagrange multipliers, where instead of scalar Lagrange multipliers, they are now functions. An extension of the PMP has been proposed, which attempts to deal with the abnormal case [27].

2.2 SubRiemannian geodesic equations

Define the subRiemannian energy as

$$
\mathcal{E}[x] = \frac{1}{2} \int_0^1 dt \langle \dot{x}, \dot{x} \rangle_\mathcal{J} = \frac{1}{2} \int_0^1 dt \langle u, u \rangle_\mathcal{J},
$$

where we are interested in simple metrics of the form $g := \langle \ldots, \ldots \rangle_\mathcal{J}$, which corresponds to the Killing form

$$
\langle W, V \rangle_\mathcal{J} = \langle \mathcal{J}W, V \rangle = \text{tr}(\mathcal{J}WV^\dagger)
$$

but there is the requirement $V, W \in \Delta$. $\mathcal{J}$ is the inertia transformation. For quantum computing, $\mathcal{J}$ simply weights the cost of moving in a particular basis direction. To calculate $\mathcal{J}W$, let $W$ be the matrix in $\mathfrak{g}$. If

$$
W = \sum_i w_i \tau_i,
$$

then

$$
\mathcal{J}W = \sum_i c_i w_i \tau_i,
$$

where $c_i$ is the cost moving in the $\tau_i$ direction. The $c_i$ can be defined by the physical cost of implementing $\exp(\tau_i)$. For example single body directions correspond to a single gate, so could be weighted with cost 1. Two body directions can require at most five gates so could cost 5.
To minimise $\mathcal{E}[x]$, form the PMP Hamiltonian,
\[ H(\nu, \lambda, x, u) = \lambda(ux) + \frac{\nu}{2} \langle u, u \rangle_J. \] (2.7)
where $\lambda$ is in the dual of the tangent space at $TG_x$, denoted $\lambda \in T^*G_x$ and $\nu \in \mathbb{R}$. By the PMP, find the $(\lambda, \nu) \neq (0, 0), \nu \leq 0$ which minimises (2.7), subject to the constraint $\dot{\lambda} = dH_x$. Clearly
\[ \dot{\lambda}(z) = -dH(z)_x = -\lambda(uz). \] (2.8)

Now define a $\Lambda^*$ in the dual $\mathfrak{g}^*$ of $\mathfrak{g}$, $\Lambda^*(v) =: \lambda(R(x)v)$, where $R$ stands for right multiplication, and $v \in \mathfrak{g}$. Corresponding to $\Lambda^*$, define $\Lambda \in \mathfrak{g}$ by the inner product $\Lambda^*(v) =: \langle \Lambda, v \rangle$, where $\langle \ldots, \ldots \rangle$ corresponds to the usual bi-invariant inner product on $\mathfrak{g}$.

First differentiating $\Lambda^*(v)$,
\[ \dot{\Lambda}^*(v) = \dot{\lambda}(vx) + \lambda(v\dot{x}) = \dot{\lambda}(vx) + \lambda(vux). \]

Using the constraint (2.8), this can be rearranged to give
\[ \dot{\Lambda}^*(v) = -\lambda(vux) + \lambda(vux) = -\Lambda^*(uv) + \Lambda^*(vu). \]

As $\Lambda^*$ is a linear function,
\[ \dot{\Lambda}^*(v) = \Lambda^*([v, u]). \]

Finally the pull back onto the inner product on $\mathfrak{g}$
\[ \langle \dot{\Lambda}, v \rangle = \langle \Lambda, [v, u] \rangle. \]

As $\langle \ldots \rangle$ is bi-invariant, $\langle X, [Y, Z] \rangle = \langle X, [-Z, Y] \rangle = \langle [X, -Z], Y \rangle$,
\[ \dot{\Lambda} = -[\Lambda, u]. \] (2.9)

By the PMP there are two cases to examine.

2.2.1 Normal case

In the normal case, the PMP requires $\nu < 0$. Without loss of generality, choose $\nu = -1$, as the equations can be rescaled by a constant. Then
\[ H = \lambda(ux) - \frac{1}{2} \langle u, u \rangle = \Lambda^*(u) - \frac{1}{2} \langle u, u \rangle_J. \]
Now identifying $\lambda$ with a $\Lambda$

$$H = \langle \Lambda, u \rangle - \frac{1}{2} \langle u, u \rangle_J.$$ 

Minimal controls occur when $dH_u(u^*) = 0$, $\forall u^* \in \Delta$. $dH_u(u^*)$ denotes the pushforward or total derivative of $H$ at $u$ in the direction of $u^*$. We require this to be zero $\forall u^* \in \Delta$ otherwise there will be no unique minima. Computing the derivative,

$$\langle \Lambda, u^* \rangle - \langle u^*, u \rangle_J = \langle \Lambda - J u, u^* \rangle = 0$$

Therefore minima occur when $\text{proj}_{\Delta}(\Lambda) = J u$. Denote the orthogonal complement of $\Delta$ by $\perp$. Denote the projection of $\Lambda$ onto $\perp$, as $\Lambda_{\perp}$. When the metric is the restriction of the bi-invariant metric, the normal equations can be written as

$$\dot{u} + \Lambda_{\perp} = -[\Lambda_{\perp}, u].$$

**Lemma 2.2.1.** Normal subRiemannian geodesics have constant norm.

**Proof.** Taking the inner product of the Euler equations with respect to $u$,

$$\langle \dot{\Lambda}, u \rangle = \langle \dot{\Lambda}^\perp + J \dot{u}, u \rangle = \langle [u, \Lambda], u \rangle.$$ 

By properties of $\langle , \rangle$, clearly $\langle \dot{u}, u \rangle_J = 0$. Integrating, $\langle u, u \rangle_J = C^2_1$, $C_1 \in \mathbb{R}$, hence $\|u\| = C_1$. Additionally the same calculation can be performed for $\Lambda_{\perp}$, $\langle \Lambda^\perp, \Lambda^\perp \rangle = C^2_2$. 

Previous work has considered geodesics on a Riemannian manifold equipped with a penalty metric. It has been suggested that taking the infinite penalty limit coincides with optimal solutions [6]. Let $u$ still denote the desired instructions $u \in \Delta$ and now $v \in \perp$, the Riemannian geodesic equations can be written as [1]

$$\dot{u} + p \dot{v} = (1 - p)[v, u]$$
$$\dot{x} = (u + v)x.$$ 

Only if $v \to \Lambda_{\perp}/p$ as $p \to \infty$, then these equations coincide with the normal equations found by the PMP. However taking the limit of a penalty metric geodesic does not always give optimal results. There are cases where the minimising solution, to the original functional, is not a solution of the equations given by the limit of the penalty metric. Some examples can be found in [28]. The PMP however may provide enough information to rule out abnormals.
2.3. Analytical results

2.2.2 Abnormal case

The abnormal case occurs when the PMP fails to provide sufficient information to determine minima or maxima. For the abnormal case, \( \nu = 0, \lambda \neq 0 \) must minimise \( H(0, \lambda, x, u) = \lambda(ux) \). Equivalently, the optimal \( u \) minimises the quantity:

\[
H = \langle \Lambda, u \rangle,
\]

where \( \Lambda \neq 0 \). As before, differentiating \( H \), we find

\[
\langle \Lambda, u^* \rangle = 0,
\]

\( \forall u^* \in \Delta \). The only way this can occur is if \( \Lambda \) is in the orthogonal complement to \( \Delta \), denoted \( \perp \), so \( \text{proj}_{\Delta} \Lambda = 0 \). Let the component of \( \Lambda \) in the orthogonal complement be denoted by \( \Lambda^\perp \). In this situation (2.9) becomes

\[
\hat{\Lambda} = -[\Lambda, u]. \tag{2.10}
\]

For certain distributions, finding a contradiction to (2.10) can rule out abnormals. If this is not possible then the abnormal case could have a lower energy, which may lead to a more efficient quantum circuit.

For an example where abnormals can be ruled out, consider SU(2) where \( \Delta = \text{span}\{i\sigma_1, i\sigma_2\} \) and \( \perp = \text{span}\{i\sigma_3\} \). In this case, for \( u \in \Delta \) and \( \Lambda^\perp \in \perp \), then \( [\Lambda^\perp, u] \in \Delta \). As \( \Lambda^\perp \) must be non zero, the only choice for \( u \) to satisfy (2.10) is \( u = 0 \). The same argument can be made for SO(3).

This can be generalised to so called *fat* distributions, where for any \( V \in \perp \), with a slight abuse of notation \( [\Delta, V] \subset \Delta \). Abnormals can be ruled out in these cases via contradiction. Clearly any \( [\Lambda, u] \) will be contained in \( \Delta \). However we require \( \hat{\Lambda} = [\Lambda, u] \), and so the only possibilities to satisfy this equation are \( u = 0 \), the trivial case, or \( \Lambda = 0 \). This contradicts the PMP as \( (\Lambda, \nu) \neq (0, 0) \), so there are no abnormals.

2.3 Analytical results

Analytical expressions for normal geodesics can be found in several groups. In these cases the commutation relationships allowed the equations to be separated into simpler components. In general this does not occur.
2.3. Analytical results

2.3.1 SU(2)

Normal case

SubRiemannian geodesics in SU(2) are well understood, for example [29]. However we use SU(2) as an example to recall a technique due to Noakes [30]. Let $\Delta = \frac{i}{\sqrt{2}}\text{span}\{\sigma_1, \sigma_2\}$ and $\perp = \frac{i}{\sqrt{2}}\text{span}\{\sigma_3\}$. In this case equation 2.3 separates directly into

\[
\begin{align*}
\dot{u} &= -[\Lambda^\perp, u], \\
\dot{\Lambda}^\perp &= 0
\end{align*}
\]

which can be integrated directly to give

\[
\Lambda^\perp = \Lambda^\perp_0, \quad u = \text{Ad}(\exp(-\Lambda^\perp_0 t))u_0.
\]

To solve Schrödinger’s equation for $x$, use the work of Noakes [30].

Given the equation $\dot{Z} = [Z, W]$, if $Z$ and $W$ are known and there exists a $y$ such that $\text{Ad}(y)(Z) = Z_0$ then the solution to the Schrödinger equation, $\dot{x} = Wx$ is given by Noakes to be

\[
x = y^\dagger \exp\left(\int_0^t ds \, v(s)\right)
\]

where

\[
v = \text{Ad}(y^\dagger)(\dot{y}y^\dagger + W)
\]

Note the changes in sign to Noakes, as we are computing right invariant geodesics.

Clearly the choice of $y = \exp(\Lambda^\perp_0 t)$ satisfies this. This gives $v = (\Lambda^\perp_0 + u_0)t$, and so subRiemannian geodesics in (right invariant) SU(2) are given by

\[
x = \exp(-\Lambda^\perp_0 t) \exp((\Lambda^\perp_0 + u_0)t)x_0,
\]

which is the product of two one parameter subgroups. This would also cover SO(3) through a change of basis.

2.3.2 SU(8)

Normal case

In $\mathfrak{su}(8)$, $\Delta$ is the span of one and two body Pauli matrices. $\perp$ is the linear span of the three bodies Pauli matrices. Specifically $\Delta = \frac{1}{2\sqrt{2}}\text{span}\{\sigma_i^r, \sigma_j^l, \sigma_k^e\}$. Denote
$u = S + D$ to be the one and two body components respectively. Furthermore let $\Lambda^\perp = T$. In this case equation (2.9) separates into the three equations

\begin{align*}
\dot{S} &= 0, \quad (2.15) \\
\dot{D} &= -[T, D], \quad (2.16) \\
\dot{T} &= -[T, S] \quad (2.17)
\end{align*}

Clearly $S = S_0$ a constant. To integrate the other equations make the following observation. Given an equation of the form $\dot{Z} = [W, Z]$, where $W$ is known, $W, Z \in \mathfrak{g}$, solutions are given by $Z = \text{Ad}(w)Z_0$, where $w$ solves the equation $\dot{w} = Ww$, with $w(0) = I$, and $Z_0$ is a constant. The solution to the three body equation is immediately

$$T = \text{Ad}(\exp(S_0t))T_0.$$ 

The two bodies now satisfy

$$\dot{D} = -[\text{Ad}(\exp(S_0t))T_0, D].$$

To integrate this look for the $w : [0, 1] \to SU(8)$ such that

$$\dot{w} = -\text{Ad}(\exp(S_0t))(T_0)w.$$ 

This equation has the solution

$$w = \exp(S_0t) \exp(-(S_0 + T_0)t),$$

recalling $w(0) = I$. Therefore the solution to the two body equation is given by

$$D = \text{Ad}(\exp(S_0t) \exp(-(S_0 + T_0)t))(D_0).$$

Now the geodesic equation can be integrated using the work of Noakes \cite{30} to give

$$x = \exp(S_0t) \exp(-(S_0 + T_0)t) \exp((S_0 + D_0 + T_0)t)x_0.$$ \quad (2.18)

Interestingly this geodesic is a product of three one parameter subgroups. Furthermore this extends naturally to $SU(d^3)$, where the allowed set is one and two qudit instructions. These solutions can also cover $SU(3)$ and $SO(4)$ with particular distributions.

**Abnormal case**

In $SU(8)$ the abnormal case is given by $\langle \Lambda, u^* \rangle = 0, \forall u^* \in \Delta$. This occurs when $\Lambda$ is in the orthogonal complement of $\Delta$. In this case the costate equation can be written as

$$\dot{\Lambda}^\perp = -[\Lambda^\perp, u].$$
While there is not an explicit equation for \( u \), examples of abnormals can be constructed in SU(8), which have the same energy as normal solutions.

As before, let the allowed set be \( \Delta = \text{span}\{\frac{1}{\sqrt{2}}\sigma_i^1, \frac{1}{\sqrt{2}}\sigma_j^3\sigma_k^1\} \). Now consider a \( u \) in the allowed set, \( u = \frac{1}{\sqrt{2}} S_i^0 \sigma_i^1 + \frac{1}{\sqrt{2}} D_{ijk} \sigma_i^1 \sigma_j^3 \sigma_k^1 = S + D \), and \( \Lambda^\perp = \frac{1}{\sqrt{2}} T_{rst}^i \sigma_i^3 \sigma_j^3 \sigma_k^1 = T \).

For arbitrary \( T \) and \( D \) the terms \( [T_{rst}^i \sigma_i^3 \sigma_j^3 \sigma_k^1, D_{ijk} \sigma_i^1 \sigma_j^3] \) are always contained in \( \Delta \) (see Appendix A). To avoid a contradiction, as \( \Lambda^\perp = T \) must be perpendicular to \( \Delta \), it must be that \( [T, D] = 0 \).

If \( [T, D] = 0 \), at least one of the \( T_{rst}^{ijk} \neq 0 \) is required by the PMP. In \( SU(8) \) there are subspaces of the two body directions which will commute with a single \( T \).

From the normal case, \( \dot{D} = [T, D] \) and so constant \( D = D_0 \), in the abnormal case, is covered by the normal equations. Now we construct variations of \( D = D_0 \) which preserve the commutation relationship. For example suppose \( D \) is also an isospectral-flow, so \( D = \text{Ad}(d)D_0 \), where \( d : [0, 1] \to SU(8) \). Since \( T \) is also an isospectral-flow, the solution to the abnormal equation can be written as \( T = \text{Ad}(\tau)T_0 \), where \( \tau : [0, 1] \to SU(8) \). If \( T_0 \) and \( D_0 \) are chosen so \( [T_0, D_0] = 0 \), then one possibility is that \( d = \tau \). \( \tau \) however must be chosen however so \( T \) always remains perpendicular to \( \Delta \). Hence \( \tau \) must be the exponential of single body terms, for example \( \tau = \exp(S_0t) \).

However such a solution for \( T \) is also given by the normal case.

Likewise the abnormal equations do not give any information about the single body components. The normal solution for \( S \) is when the \( S_i^1 \) are all constant \( S = \frac{1}{\sqrt{2}} (S_0)^i_j \sigma_i^1 = S_0 \). The abnormal equation for \( T \) can then be satisfied by \( T = \exp(-S_0t)T_0 \exp(S_0t) \). Note this is always contained in \( \perp \) and still satisfies the normal geodesic equations.

A more complicated \( S \) is given by rotations of the form,

\[
S = \text{Ad}(\exp(-S'_0t)S_0),
\]

where \( S'_0 = \frac{1}{\sqrt{2}} (S'_0)^i_j \sigma_i^1 \) is also a single body. \( S \) in this case has constant norm, like the normal case, but now

\[
T = \text{Ad}(\exp(-S'_0t) \exp(-(S_0 - S'_0)t))T_0,
\]

which is not given by the normal case. This \( T \) is always contained in \( \perp \) as

\[
\text{Ad}(\exp(-S'_0t) \exp(-(S_0 - S'_0)t)),
\]

simply rotates \( T_0 \) in \( \perp \). Furthermore

\[
D = \text{Ad}(\exp(-S'_0t) \exp(-(S_0 - S'_0)t))D_0.
\]
Likewise, this will remain in $\Delta$ by a similar argument.

Clearly this choice for $S, D$ and $T$ are not given by the normal geodesic equations, but in $\text{SU}(8)$ it is a rotation of a normal solution by $\exp(-S_0^t)$. More exotic solutions could easily be created by adding extra rotations. For practical implementation as a circuit, the constant single body controls would be simpler, and it is already covered by the normal case.

### 2.3.3 SU($2^n$)

**Normal case**

As before the allowed set is composed of one and two bodies

$$u = S + D.$$ 

Now denote the odd $m > 1$ bodies as $T$ and the even $n > 2$ bodies as $F$. In this situation equation (2.9) separates into

$$\dot{S} = 0$$

$$\dot{D} + \dot{F} = [D, T] + [S, F]$$

$$\dot{T} = [D, F] + [S, T].$$

It is not possible to completely integrate these equations in closed form. Clearly $S = S_0$ as before. Now let $D = \text{Ad}(\exp(S_0 t))\hat{D}$, $F = \text{Ad}(\exp(S_0 t))\hat{F}$ and $T = \text{Ad}(\exp(S_0 t))\tilde{T}$ which removes the Lie brackets of $T$ and $F$, with $S_0$.

Now the equations become

$$\dot{T} = [\hat{D}, \hat{F}],$$

$$\dot{D} + \dot{F} = [\hat{D}, S_0] + [\hat{D}, T].$$

Now set $Z = \hat{D} + \hat{F}$, and let $p$ denote the projection onto the allowed set. This gives the two coupled equations

$$\dot{T} = [p(Z), Z],$$

$$\dot{Z} = [p(Z), S_0] + [p(Z), \tilde{T}].$$

Potentially these equations can be solved with a power series. A power series may be a useful computational tool for generating approximate solutions. Let

$$\tilde{T} = \sum_{i=0}^{\infty} q_i t^i,$$

$$Z = \sum_{i=0}^{\infty} z_i t^i.$$
2.3. Analytical results

Figure 2.1: Truncated series results vs exact result in SU(8). Figure shows the control function in the $\sigma_3 \otimes \sigma_3 \otimes I$ direction. $n$ is the number of terms in the series expansion. Control functions in other directions displayed similar behaviour.

where $z_i$ and $q_i$ are matrices. To find the coefficients, compare the powers of $t$

$$
\sum_{r=1}^{\infty} rq_r t^{r-1} = \sum_{j=0}^{\infty} \sum_{k=0}^{\infty} [p(z_j), z_k] t^{j+k}
$$

$$
\sum_{r=1}^{\infty} rz_r t^{r-1} = \sum_{j=0}^{\infty} [p(z_j), S_0] t^j + \sum_{j=0}^{\infty} \sum_{k=0}^{\infty} [p(z_j), q_k] t^{j+k}
$$

Computing the first few terms

$$q_1 = [p(z_0), z_0]$$

$$q_2 = \frac{1}{2} ([p(z_1), z_0] + [p(z_0), z_1]) ,$$

$$q_3 = \frac{1}{3} ([p(z_2), z_0] + [p(z_1), z_1] + [p(z_0), z_1]) ,$$

and likewise

$$z_1 = [p(z_0), S_0 + q_0],$$

$$z_2 = \frac{1}{2} ([p(z_1), S_0 + q_0] + [p(z_0), q_1]) ,$$

$$z_3 = \frac{1}{3} ([p(z_2), S_0 + q_0] + [p(z_1), q_1] + [p(z_0), q_2]) .$$

In general for $k \geq 1$,

$$q_k = \frac{1}{k} \left( \sum_{r=0}^{k-1} [p(z_{k-1-r}), z_r] \right)$$

$$z_k = \frac{1}{k} \left( [p(z_{k-1}), S_0 + q_0] + \sum_{r=1}^{k-1} [p(z_{k-1-r}), q_r] \right)$$

Figure 2.1 shows the series expansions to various orders, compared with the exact result for randomly chosen initial conditions. For convergence, first note that the
2.4. Higher Order PMP

2.4.1 General approach

The PMP fails in the abnormal case when the derivative $dH_u(u^*) = 0$ does not give enough information to determine $u$. This is because there are not enough constraints in the original control problem. We propose a new method to try and find $u$ in the...
abnormal case, when the original state equation

\[ \dot{x}_1 = f(u, x_1), \]

is affine in \( u \). Specifically, let

\[ \dot{x}_1 = G(x_1, t) + K(x_1, t)u \]

where \( x_1 : [t_1, t_2] \to \mathbb{R}^n \), \( u : [t_1, t_2] \to \mathbb{R}^p \). \( G(x_1) \) is an \( n \times 1 \) matrix valued function of \( x_1 \), and \( K(x_1) \) is an \( n \times p \) matrix valued function of \( x_1 \). Furthermore we restrict ourselves to the case where the original optimal control problem, was to minimise an action of the form

\[ S[x_1] = \frac{1}{2} \int dt \ u(t)^T g(x_1)u(t), \]

where \( g(x_1) \) is a Riemannian metric, namely a position-dependent, positive definite, symmetric matrix. Briefly, the new method is to take any constraints found in the abnormal case and re-apply the PMP with these constraints. The original PMP Hamiltonian \( H \) we now denote \( H^{(1)} \) is simply

\[ H^{(1)} = \lambda_1 \cdot (G(x_1) + K(x_1)u) + \frac{\nu_1}{2} u(t)^T g(x_1)u(t). \]

where \( \lambda_1 \) denotes the co-vector in the dual of \( \mathbb{R}^n \), \( \lambda_1 : [t_1, t_2] \to \mathbb{R}^{n^*} \). By the PMP \( \lambda_1 \) is required to satisfy

\[ \dot{\lambda}_1 = - \left( \frac{\partial H^{(1)}}{\partial x_1} \right)^T \]

which denotes the gradient with respect to components of \( x \). In the normal case there will always be enough information to determine \( u \). While the state equation is linear in \( u \), the integrand of the action is quadratic, leaving a \( u \) after computing the derivative \( dH_u \). This means \( u \) can be solved for in terms of \( \lambda_1 \). However in the abnormal case, the quadratic term will always vanish, which is where the problem occurs. However it does give some information. In the abnormal case we require \( \nu_1 = 0 \), \( \lambda \neq 0 \) and

\[ \lambda \cdot (K(x)\bar{u}) = 0, \]

\( \forall \bar{u} \in \mathbb{R}^p \). Therefore, we require \( \lambda^1 \) to be perpendicular to all the columns of \( K(x) \) for all \( t \in [t_1, t_2] \). This is equivalent to requiring the \( p \) vector,

\[ \chi(t) = \begin{pmatrix} \langle \lambda_1, k_1 \rangle \\ \langle \lambda_1, k_2 \rangle \\ \vdots \\ \langle \lambda_1, k_p \rangle \end{pmatrix}, \]
where \( k_i \) denotes the \( i \)-th column, to be zero, \( \chi_1(t) = 0 \forall t \in [t_1, t_2] \). Clearly this is still affine in the original \( u \). The new method is to add this constraint on \( \lambda_1 \) back into PMP.

Analogous to finite dimensional Lagrange multipliers, for each constraint we add a Lagrange multiplier. As long as we only check the cases where \( \lambda_1 \) satisfies \( \chi_1(t) = 0 \) reapplying the PMP with these extra constraints hopefully gives extra equations for \( u \).

Constraints in the PMP are given by state equations. So to add the \( \chi_1 \) constraint, we upgrade \( \chi_1 \) and \( \lambda_1 \) to state vectors. A state equation for \( \chi_1 \) is found by the derivative, \( \dot{\chi}_1 \), and requiring the initial condition \( \chi_1(0) = 0 \). Clearly \( \dot{\chi}_1 \) will still be affine in the control, and there will be no \( \dot{u} \) terms. An equation for \( \dot{\lambda}_1 \) is already given by the PMP. Hence we then take our new state to be

\[
x_2(t) = (x_1(t), \lambda_1(t), \chi_1(t)),
\]

where \( x_2 : [t_1, t_2] \to \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^p \). Then re-apply the PMP looking to minimise \( S[x_1] \) with respect to \( u \) subject to,

\[
\dot{x}_2 = (\dot{x}_1, \dot{\lambda}_1, \dot{\chi}_1).
\]

Note this system will still be affine in \( u \). Now the new PMP Hamiltonian can be written as

\[
H^{(2)} = \lambda_2 \cdot (\dot{x}_1, \dot{\lambda}_1, \dot{\chi}_1) + \frac{\nu_2}{2} u^T g(x_1) u,
\]

where \( \lambda_2 : [t_1, t_2] \to (\mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^p)^* \) is required to satisfy

\[
\dot{\lambda}_2 = \left( \left( \frac{\partial H^{(2)}}{\partial x_1} \right)^T, \left( \frac{\partial H^{(2)}}{\partial \lambda_1} \right)^T, \left( \frac{\partial H^{(2)}}{\partial \chi} \right)^T \right).
\]

Clearly there are abnormal and normal cases of this to examine. In the abnormal case, \( dH^{(2)}_u \) will always be affine in \( u \), so \( u \) can be solved for in terms of \( \lambda_2 \) in \( dH^{(2)}_u = 0 \). However in the abnormal case of \( H^{(2)} \) we will find new orthogonality constraints for \( \lambda_2 \), which we will set as \( \lambda_2 \). If this does not lead to a contradiction, for example the SU(2) case, we will upgrade these and \( \lambda_2 \) to state variables and repeat the procedure. This will give a chain of \( H^{(k)} \) and \( \lambda_k \), where at each level we will call the control \( u \) found \( k \)-th abnormal. Suppose the original state vector was of dimension \( m \). Then the new state will have dimension \( 2m + p \), as a \( \chi_k \) will always be of dimension \( p \), and \( \lambda_k \) of dimension \( m \). More than doubling the dimension at each step means this procedure is expensive for control problems in large dimensions.
2.4.2 Abnormal geodesics

Previously it was found that $\dot{\Lambda}_1 = -[\Lambda_1, u_1]$. In the abnormal case, $\Lambda$ was required to satisfy $\langle \Lambda_1^1, u^* \rangle = 0$, $\forall u^* \in \Delta$, so by the PMP we require $\text{proj}_\Delta (\Lambda_1^1) = 0$. As an example of the previous method, we will find equations for $2$-abnormal geodesics.

Along with the equation for $\Lambda_1^1$, this constraint is upgraded to a state equation $\chi_1 = \frac{d}{dt} \text{proj}_\Delta (\Lambda_1^1) = -\text{proj}_\Delta ([\Lambda_1^1, u])$. Then the new iterated PMP Hamiltonian is given by

$$H^{(2)} = \lambda_1^2 (u_1 x) - \lambda_2^2 ([\Lambda_1^1, u_1]) - \lambda_3^2 (\text{proj}_\Delta ([\Lambda_1^1, u_1])) + \frac{\nu_2}{2} ||u_1||^2.$$ 

where $\lambda_i^2$ are new costates, so $\lambda_1^2 \in T^* G_x$, $\lambda_2^2 \in g^*$ and $\lambda_3^2 \in \Delta^*$. Computing the derivatives

$$\dot{\lambda}_1^2 (v_1) = -\lambda_1^2 (v_1 x),$$

$$\dot{\lambda}_2^2 (v_2) = \lambda_2^2 ([v_2, u_1]) + \lambda_3^2 (\text{proj}_\Delta ([v_2, u_1])),$$

$$\dot{\lambda}_3^2 (v_3) = 0,$$

where $v_1 \in TG_x$, $v_2$, and $v_3 \in g$. These equations can be rewritten as equations on $g$. $\lambda_1^2$ can be identified with a $\Lambda_1^2 \in g$ as before. Additionally $\lambda_2^2$ can be identified with a $\Lambda_2^2 \in g$ via $\lambda_2^2 (v) = \langle \Lambda_2^2, v \rangle$ where $v \in g$. Similarly for $\lambda_3^2$ can be identified with a $\Lambda_3^2$.

$$\dot{\Lambda}_1^2 = -[\Lambda_1^2, u_1],$$

$$\dot{\Lambda}_2^2 = -[\Lambda_2^2, u_1] - [\text{proj}_\Delta (\Lambda_2^2), u_1],$$

$$\dot{\Lambda}_3^2 = 0.$$ 

Therefore take $\Lambda_3^2$ as a constant in $\Delta$. As usual there are two cases to examine, the normal case $\nu \leq 0$ and the abnormal case $\nu = 0$. Additionally $H^{(2)}$ can be written in terms of inner products.

$$H^{(2)} = \langle \Lambda_1^2 - [\Lambda_2^2, \Lambda_1^1] - \Lambda_3^2, u_1 \rangle + \frac{\nu}{2} \langle u_1, u_1 \rangle.$$ 

In the normal case, without loss of generality, $\nu_2 = -1$. Maxima of the Hamiltonian occur when $dH^{(2)}_{\Lambda_1^1} (u^*) = 0$, $\forall u^* \in \Delta$. Therefore

$$\langle \Lambda_1^2 - [\Lambda_2^2, \Lambda_1^1] - \Lambda_3^2 - u_1, u^* \rangle = 0.$$ 

Hence this requires

$$\text{proj}_\Delta (\Lambda_1^2 - [\Lambda_2^2, \Lambda_1^1]) - \Lambda_3^2 = u_1.$$
This gives enough equations to determine $u_1$. In the abnormal case, $\nu_2 = 0$, a similar situation as the original abnormal case occurs,

$$\text{proj}_\Delta (\Lambda^2_1 - \{\Lambda^2_2, \Lambda^1_1\} - \Lambda^2_3) = 0.$$ 

Then the process can be repeated. The $\Lambda^2_i$ can be upgraded to state variables, along with a new state for the projection. One significant problem with this approach is that the dimension more than doubles at each iteration. For SU(8) this becomes a problem in a 252 dimension space. There is also no guarantee that these abnormals have lower energy. For the quantum computing application it might be better to accept the possibly higher energy normal geodesics. However this approach might be useful for other control problems.

### 2.5 Magnus expansions

The Magnus Expansion uses Picard fixed point iteration to solve the matrix Schrödinger equation as a power series which respects the geometric structure [31]–[34]. As an extension, the same technique can be used to constructs series solutions for Lax equations. Furthermore, this will also yield a solution for the Schrödinger equation, giving a general series solution for subRiemannian geodesics.

Recall the derivative of the matrix exponential is given by

$$\frac{d}{dt} \exp(X) = \left( \frac{I - \exp(-\text{ad}_X)}{\text{ad}_X} \right) \dot{X} \exp(X) = \text{dexp}_X(\dot{X}) \exp(X),$$

where the operator, dexp and its inverse can be computed by a power series

$$\text{dexp}_X := \left( \frac{I - \exp(-\text{ad}_X)}{\text{ad}_X} \right) = \sum_{k=0}^{\infty} \frac{(-1)^k}{(k+1)!} (\text{ad}_X)^k \quad (2.19)$$

$$\text{dexp}_X^{-1} := \sum \frac{B_k}{k!} \text{ad}_X^k, \quad (2.20)$$

where $B_k$ is the $k$–th Bernoulli number. Consider the Lax equation

$$\dot{Z} = [p(Z), Z].$$

As $Z$ is an isospectral flow, there exists a $W \in G$ such that $Z = WZ_0W^\dagger$. Additionally let $p$ be the orthogonal projection onto the allowed set $\Delta$,

$$p(X) = \sum_{\tau_i \in \Delta} \langle X, \tau_i \rangle \tau_i.$$ 

Now $W$ must satisfy

$$\dot{W} = p(WZ_0W^\dagger)W = p(\text{Ad}(W)(Z_0))W.$$
Suppose \( W = \exp(\Omega) \exp(Z_0 t) \). The \( \exp(Z_0 t) \) term is included to increase the accuracy of \( \Omega \) constructed from the series expansion, and to match known solutions. Then

\[
dexp_\Omega(\Omega) \exp(Z_0 t) + \exp(\Omega) \exp(Z_0 t) Z_0 = p(\Ad(\exp(\Omega)(Z_0))) \exp(\Omega) \exp(Z_0 t),
\]

\[
dexp_\Omega(\Omega) = p(\Ad(\exp(\Omega))(Z_0)) - \Ad((\exp(\Omega))(Z_0)).
\]

Let \( p^\perp \) be the negative of the orthogonal projection onto \( \perp \). Now \( \Omega \) satisfies the equation

\[
\dot{\Omega} = dexp_\Omega^{-1}(p^\perp(\Ad(\exp(\Omega))(Z_0))) = \sum_{k=0}^{\infty} \frac{B_k}{k!} \text{ad}_{\Omega}^k \left( p^\perp(\Ad(\exp(\Omega))(Z_0)) \right).
\]

This can be written as the integral equation

\[
\Omega(t) = \sum_{k=0}^{\infty} \frac{B_k}{k!} \int_0^t ds \text{ad}_{\Omega(s)}^k \left( p^\perp(\Ad(\exp(\Omega))(Z_0)) \right),
\]

assuming \( \Omega(0) = 0 \). To find an approximate solution perform the Picard iterative process

\[
\Omega[0] = 0,
\]

\[
\Omega[1] = \int_0^t ds \ p^\perp(Z_0) = p^\perp(Z_0)t,
\]

\[
\Omega[n+1] = \int_0^t ds \sum_{k=0}^{\infty} \frac{B_k}{k!} \text{ad}_{\Omega[n]}^k \left( p^\perp(\Ad(\exp(\Omega[n]))(Z_0)) \right).
\]

Assuming small \( t \), \( t \leq 1 \), the calculations can be simplified by truncating the power series for \( \text{dexp}^{-1} \) and \( \Ad(\exp) \) at the order of accuracy as the iterated solution. This is because we would like \( \Omega[n] \) to be accurate to order \( O(t^n) \). A \( k \)-fold nesting of Lie brackets of \( \Omega[n] \) will at least have a \( O(t^k) \) term. So we should ignore \( n + 1 \) nested Lie brackets as they will contribute higher order terms. Then, the procedure for iterating can be taken to be

\[
\Omega[n+1] = \int_0^t ds \sum_{k=0}^{n} \frac{B_k}{k!} \text{ad}_{\Omega[n]}^k \ p^\perp \left( \sum_{j=0}^{n} \frac{1}{j!} \text{ad}_{\Omega[n]}^j(Z_0) \right).
\]

By linearity of \( p^\perp \) this at least simplifies to

\[
\Omega[n+1] = \int_0^t ds \sum_{k=0}^{n} \sum_{j=0}^{n} \frac{B_k}{j!k!} \text{ad}_{\Omega[n]}^k \ p^\perp \left( \text{ad}_{\Omega[n]}^j(Z_0) \right).
\]

Given that \( \Omega[1] \) is a first order polynomial, additional iterated solutions will be higher order polynomials. Furthermore if the \( \Omega[n] \) term is accurate to \( O(t^n) \), we should only
take terms up to $O(t^n)$ in the integrand, so after integration we are left with $O(t^{n+1})$ order terms. The second order solution is given by

$$\Omega[2] = p^+(Z_0)t + \frac{1}{2}p^+([p^+(Z_0), Z_0])t^2.$$  

To find the third order solution, we ignore $O(t^3)$ terms in the integrand.

$$\Omega[3] = p^+(Z_0)t + \frac{1}{2}p^+([p^+(Z_0), Z_0])t^2 + \frac{1}{3}\left(\frac{1}{2}p^+([Z_0, p^+(Z_0)], p^+(Z_0)) + \frac{1}{4}[p^+(p^+(Z_0), Z_0), p^+(Z_0)]\right)t^3.$$  

As a check, this matches exactly with the truncated solution in SU(8). Recall in that case $\exp(\Omega) = \exp(S_0 t) \exp(-(S_0 + T_0) t)$ exactly. Expanding this with the Baker-Campbell-Hausdorff formula to $O(t^3)$ we find $\Omega[3]$.  

In general $\Omega$ can be represented by a power series.

$$\Omega = \sum_{k=0}^{\infty} \Omega_k t^k,$$

where the $\Omega_k$ term involves permutations of $(k-1)$-deep Lie brackets and $k$ projections.

$W$ also provides a solution to the Schrödinger equation,

$$\dot{x} = p(WZ_0W^+)x.$$  

Suppose $y$ can be found such that $x = Wy$. Differentiating

$$p(WZ_0W^+)y + W\dot{y} = p(WZ_0W^+)Wy.$$  

Clearly the only solution is $y = y_0$ a constant, so $x = W$. This series for $W$ will be useful for numerical calculations, as it can be used to estimate initial conditions. It can also be used to construct numerical integrators which respect the unitary property of $x$. This can be achieved by replacing the integral with a quadrature.

### 2.6 Discretising subRiemannian geodesics

Nielsen et al proved three lemmas guaranteeing that Riemannian geodesics can be approximated to arbitrary precision with a quantum circuit. The first lemma is automatically satisfied by subRiemannian geodesics. Our control function $u$ is always
in terms of one and two body gates, it does not need to be approximated. The remaining two lemmas can be modified for subRiemannian geodesics. Namely that the error of the approximation can be controlled by choosing the size of discretisation. Additionally, as a departure from previous work, we choose a more natural definition of error.

For a small time interval \([0, \Delta t]\), let \(U = x(\Delta t)\), be the unitary matrix generated by the control \(u\). Now define the average control \(\tilde{u}\),

\[
\tilde{u} = \frac{1}{\Delta t} \int_0^{\Delta t} dt \ u(t),
\]

and let \(\tilde{U} = \exp(\tilde{u}\Delta t)\) be an approximation to \(U\). We define the error to be given by the length of the minimal bi-invariant Riemannian geodesic joining \(U\) and \(\tilde{U}\). This can be computed by

\[
\text{err}(\tilde{U}, U) = \| \log(U^\dagger \tilde{U}) \|,
\]

where \(\log\) is taken on the principal branch, and \(\|V\| = \sqrt{\text{tr}(VV^\dagger)}\) is the Riemannian inner product. While err does not respect any subRiemannian structure, a subRiemannian metric might be crudely approximated by replacing \(\|\ldots\|\) with a penalty metric \(\|\ldots\|_p\).

First, the error of approximating \(U\) by \(\tilde{U}\) is bounded

**Lemma 2.6.1.** \(\text{err}(U, \tilde{U})\) is \(O(\Delta t^2)\).

**Proof.** Using the BCH formula, let \(U^\dagger = \exp(V)\), correspond to the exact solution to the Schrödinger equation, \(\dot{x} = ux\). Now

\[
\text{err}(U, \tilde{U}) = \| \log(U^\dagger \exp(\tilde{u}\Delta t)) \|
\]

\[
= \left\| V + \tilde{u}\Delta t + \frac{1}{2}[V, \tilde{u}\Delta t] + \frac{1}{12}([V, [V, \Delta t\tilde{u}]] + \Delta t^2[\tilde{u}, [\tilde{u}, V]]) + \ldots \right\|.
\]

If \(\int_0^{\Delta t} dt \|u\| = \Delta t C_1 \leq \pi\), then the Magnus expansion, as a solution for \(x\), in \(\dot{x} = ux\), is guaranteed to converge. So choosing a sufficiently small \(\Delta t\), \(V\) is given by the Magnus expansion as

\[
V = -\int_0^{\Delta t} dt \ u - \frac{1}{2} \int_0^{\Delta t} dt_1 \int_0^{\Delta t} dt_2 [u(t_1), u(t_2)] + O(\Delta t^3),
\]

Denoting the higher order integrals with the notation \(O(\Delta t) \sim \int_0^{\Delta t} dt\). Additionally
2.6. Discretising subRiemannian geodesics

by lemma (2.2.1) \(\|u\| = C_1\). Now

\[
\begin{align*}
\text{err}(U, \bar{U}) &= \left\| -\frac{1}{2} \int_0^\Delta t dt_1 \int_0^{t_1} dt_2 [u(t_1), u(t_2)] + O(\Delta t^3) \right. \\
&\quad - \frac{1}{4} \left[ \int_0^\Delta t dt_1 \int_0^{t_1} dt_2 [u(t_1), u(t_2)] + O(\Delta t^3), \bar{u}\Delta t] + O(\Delta t^3) \right\| \\
\text{err}(U, \bar{U}) &\leq C_1^2 \Delta t^2 + \|O(\Delta t^3)\|.
\end{align*}
\]

Then

\[
\lim_{\Delta t^2 \to 0} \frac{\text{err}(U, \bar{U})}{\Delta t^2} \leq C_1^2.
\]

To ensure the overall error is bounded by \(\varepsilon^2\) we must choose \(\Delta t \leq \varepsilon/C_1\).

Therefore the error of approximating \(U\) by constant controls can be controlled by making \(\Delta t\) sufficiently small. It is currently unknown how \(C_1\) changes with respect to the dimension, the target and the distribution in general.

To convert a constant segment \(\bar{U}\) into a quantum circuit, use the Lie Product formula. First divide \([0, \Delta t]\) into \(M\) segments of length \(\Delta t^2\). Then

\[
\bar{U} = (\bar{U}_{\Delta t^2})^M + O(\Delta t^2),
\]

where

\[
\bar{U}_{\Delta t^2} = \exp(u_1 \tau_1 \Delta t^2) \cdots \exp(u_m \tau_m \Delta t^2).
\]

From Chapter (1), a two body gate requires 2 CNOT gates and at most 5 single qubit gates. Also note that the single body gates can be written as phase gates. In \(\mathfrak{su}(2^n)\) there are \(m = (9/2) n(n - 1) + 3n\) one and two body basis elements. So in total there are at most \(q = (63/2) n(n - 1) + 3n\) fundamental gates (assuming for now phase gate can be implemented directly) required to implement a single step of the Lie Trotter formula.

**Lemma 2.6.2.** \(\text{err}(\bar{U}, \bar{U}_{\Delta t^2}^M)\) is \(O(\Delta t^3)\).

**Proof.** First by the Lie product formula

\[
\exp(\bar{u}_1 \tau_1 \Delta t^2) \cdots \exp(\bar{u}_m \tau_m \Delta t^2) = \exp(\bar{u}\Delta t^2 + O(\Delta t^4 m^2)).
\]

Next

\[
\left( \exp(\bar{u}\Delta t^2 + O(\Delta t^4 m^2)) \right)^M = \exp(\bar{u}\Delta t + O(\Delta t^2 m)).
\]
Then
\[
\text{err}(\tilde{U}, \tilde{U}_{\Delta t}^M) = \| \log(\exp(-\tilde{u}\Delta t) \exp(\tilde{u}\Delta t + O(\Delta t^2 m))) \| \\
= \| - [\Delta t \tilde{u}, O(\Delta t^2 m)] + O(\Delta t^4) \| \\
\leq 2C_1 m \Delta t \| O(\Delta t^2) \| + \| O(\Delta t^4) \|.
\]

Therefore as \( \Delta t^3 \to 0 \),
\[
\frac{\text{err}(\tilde{U}, \tilde{U}_{\Delta t}^M)}{\Delta t^3} \leq 2C_2 m,
\]
where \( C_2 \) is another constant at most order \( O(C_1^2) \).

Overall to approximate a general \( x(1) = U \) with error \( \varepsilon \), in terms of one and two body gates, divide \([0, 1]\) into small segments of length \( O(\varepsilon / mc_1) \). Clearly each local section of \( U \) can be approximated by \( O(q^2 c_1^2) \) gates, and so overall \( O(q^3 c_1^3) \) gates. Note \( c_1 \) may become quite large. A lower bound is determined by the fact that the subRiemannian distance is bounded below by the Riemannian distance. So at least \( c_1 \geq \| \log(U) \| \). However for an arbitrary \( U \), the subRiemannian distance is expected to scale exponentially, and hence the number of segments and gates required.
3. SubRiemannian cubics

3.1 Background

While the energy of a geodesic is a reasonable measure of complexity, we believe there is a better description. Consider the simple geodesic \( x = \exp(\vartheta \tau_j t) \), where \( \vartheta \in \mathbb{R} \) is some constant variable. The energy of \( x \) is \( \frac{1}{2} \vartheta^2 \). Physically, \( x \) can be implemented by CNOT gates and a phase gate which depends on \( \vartheta \). If we are only counting CNOTs and phase gates, \( \vartheta \) does not add extra complexity, but it does change the energy. So the energy of \( x \) includes contributions from phenomena that do not contribute to computational complexity. One measure of complexity is to count the number of operations in an algorithm. The continuous description of complexity should try to mimic this. Hence a more appropriate measure would be to count the number of changes in direction. On a Riemannian manifold one way this can be achieved is with the following functional

\[
S[x] = \int_0^1 dt \langle \nabla_t \dot{x}, \nabla_t \dot{x} \rangle_J,
\]

where \( \nabla_t \) denotes the covariant derivative. Also \( \langle A, A \rangle_J = \langle JA, A \rangle \) and \( J \) is the inertia transform. When the Lie reduction of \( \dot{x} \) is restricted to a \( \Delta \), the PMP can be used to determine stationary points. However \( \langle \cdot, \cdot \rangle_J \) does not need to be a subRiemannian metric. The Lie reduction of the covariant acceleration does not have to be in \( \Delta \). Measuring the norm squared of the covariant acceleration in perpendicular directions is important, because changing directions in certain ways may be more expensive.

In the Riemannian case it is true that geodesics are cubics with zero initial velocity and acceleration. Riemannian geodesics are fixed points of the Riemannian cubic functional. However, this might not necessarily be true on a subRiemannian manifold. We study subRiemannian cubics out of mathematical interest and hopefully as a better measure of complexity of a quantum circuit.

Let \( x \in G \), and \( \dot{x} \in TG_x \). Denote the right-Lie reduction of \( \dot{x} \) as \( \dot{x} \),

\[
\dot{x} = \dot{x} x^{-1}.
\]

Now define \( X \) st \( \dot{x} = X^i \tilde{e}_i x \) where \( e_j \) is a vector field in \( TG_x \). For a vector field \( Y(t) = Y^i e_i \) along \( x \), the right-Lie reduction of the covariant derivative is defined by

\[
\nabla_t Y = \dot{Y}^i \tilde{e}_i + X^i Y^j \nabla_{e_i} e_j.
\]
The right-Lie reduction of the covariant derivative for two vector fields, for right invariant $G$, is given by

$$\tilde{\nabla}_e e_j = \frac{1}{2}([\tilde{e}_j, \tilde{e}_i] + h(\tilde{e}_i, \tilde{e}_j)),$$

where the standard notation $[W, V] = WV - VW$ applies. Note the order of the Lie bracket in the definition. The right-Lie reduction of a Lie bracket $[e_i, e_j] = -[\tilde{e}_i, \tilde{e}_j] = [\tilde{e}_j, \tilde{e}_i]$. We take into account the factors of $-1$ in our definition.

To save space, the notation

$$h(V, W) = B(V, W) + B(W, V)$$

(3.1)

where $B$ is the bi-linear operator defined by

$$\langle B(V, W), Z \rangle_{\mathcal{J}} := \langle [Z, W], V \rangle_{\mathcal{J}}$$

$$= \langle [W, Z], \mathcal{J}V \rangle$$

$$= \langle W, [Z, \mathcal{J}^{-1}(V)] \rangle$$

$$= \langle [\mathcal{J}V, W], Z \rangle$$

$$= \langle \mathcal{J}^{-1}([\mathcal{J}V, W]), \mathcal{J}Z \rangle$$

$$= \langle \mathcal{J}^{-1}([\mathcal{J}(V), W]), Z \rangle_{\mathcal{J}},$$

has been used. Therefore

$$B(V, W) = \mathcal{J}^{-1}([\mathcal{J}V, W]),$$

(3.2)

which gives

$$\tilde{\nabla}_Y \tilde{Y} = \tilde{Y} + \frac{1}{2}([\tilde{Y}, X] + h(X, Y)).$$

(3.3)

When $Y = \dot{x}$ this equation reduces to

$$\tilde{\nabla}_t \dot{x} = \dot{X} + \mathcal{J}^{-1}[\mathcal{J}X, X]$$

(3.4)

This is the negative of the left-Lie reduction of $\nabla_t \dot{x}$ as seen in [17], because we used right-Lie reductions.

### 3.2 Riemannian cubics

Before examining the subRiemannian case, we investigate the Riemannian case in the metric, $\langle \cdot, \cdot \rangle_{\mathcal{J}}$.

$$S[x] = \int_a^b dt \langle \nabla_t \dot{x}, \nabla_t \dot{x} \rangle_{\mathcal{J}}.$$  

(3.5)
It can be shown that the critical points of equation (3.5) are given by
\[\nabla^3 x + R(\nabla x, \dot{x})\dot{x} = 0. \tag{3.6}\]

Given that \(x\) must satisfy the Schrödinger equation \(\dot{x} = Vx\), we can find an equation for \(V\) via right-Lie reduction. From the previous section we found that
\[\nabla_t x = \dot{V} + \frac{1}{2} h(V, V),\]
and then
\[\nabla_t^2 x = \frac{d}{dt}(\nabla_t x) + \frac{1}{2} \left( [\nabla_t x, V] + h(V, \nabla_t x) \right)\]
\[= \frac{d}{dt}(\dot{X} + \frac{1}{2} h(V, V)) + \frac{1}{2} \left( [\dot{V} + \frac{1}{2} h(V, V), V] + h(V, \dot{V} + \frac{1}{2} h(V, V)) \right)\]
\[= \dot{\dot{V}} + \frac{3}{2} h(V, \dot{V}) + \frac{1}{2} [\dot{V}, V] + \frac{1}{4} [h(V, V), V] + \frac{1}{4} h(V, h(V, V))\]
where the derivative of \(h\) is given by
\[\frac{d}{dt}h(X, Y) = \frac{d}{dt}(B(X, Y) + B(Y, X)) = h(X, Y) + h(X, Y)\]

Finally
\[\nabla_t^3 x = \frac{d}{dt}(\nabla_t^2 x) + \frac{1}{2} \left( [\nabla_t^2 x, V] + h(V, \nabla_t^2 x) \right)\]
\[= \frac{d}{dt}(\dot{\dot{V}} + \frac{3}{2} h(V, \dot{V}) + \frac{1}{2} [\dot{V}, V] + \frac{1}{4} [h(V, V), V] + \frac{1}{4} h(V, h(V, V)))\]
\[= \ddot{V} + \dot{V} + \frac{3}{2} h(V, \dot{V}) + \frac{1}{2} [\dot{V}, V] + \frac{1}{4} [h(V, V), V] + \frac{1}{4} h(V, h(V, V)))\]
\[+ \frac{1}{4} [h(V, V), V] + \frac{1}{4} h(V, h(V, V)) + 5 \frac{1}{4} h(V, h(V, V))\]
\[= \ddot{V} + \dot{V} + \frac{3}{2} h(V, \dot{V}) + 2 h(V, \dot{V})\]
\[+ \frac{1}{4} [h(V, V), V] + \frac{1}{4} h(V, h(V, V)) + \frac{5}{4} h(V, h(V, V))\]
\[+ \frac{5}{4} [h(V, \dot{V}), V] + \frac{1}{4} [\dot{V}, V] + \frac{1}{8} [h(V, V), V] \]
\[+ \frac{1}{8} [h(V, h(V, V)), V] + \frac{1}{4} h(V, [\dot{V}, V])\]
\[+ \frac{1}{8} h(V, [h(V, V), V]) + \frac{1}{8} h(V, h(h(V, V)))\]

Now consider the curvature term \(R(\nabla x, \dot{x})\dot{x}\). The curvature operator is defined by
\[R(V, W)Z = \nabla_V \nabla_W Z - \nabla_W \nabla_V Z - \nabla_{[V, W]} Z.\]
3.3. SubRiemannian cubics

where \( V, W, Z \in TG \). Denote the right Lie reduction as \( \tilde{R} \). Writing this in local coordinates,

\[
R(e_i, e_j)_{ek} = \nabla_{e_i} \nabla_{e_j} e_k - \nabla_{e_j} \nabla_{e_i} e_k - \nabla_{[e_i, e_j]} e_k
\]

\[
\implies \tilde{R}(e_i, e_j)_{ek} = \frac{1}{2}(\tilde{\nabla}_{e_i} e_k, \tilde{e}_i) + h(\tilde{\nabla}_{e_j} e_k, \tilde{e}_j)
\]

\[
- \frac{1}{2}(\tilde{\nabla}_{e_i} e_k, \tilde{e}_j) + h(\tilde{\nabla}_{e_j} e_k, \tilde{e}_j)
\]

\[
- \frac{1}{2}(\tilde{e}_k, [\tilde{e}_j, \tilde{e}_i]) + h(\tilde{e}_k, [\tilde{e}_j, \tilde{e}_i])
\]

\[
= -\frac{1}{4}[[\tilde{e}_i, \tilde{e}_j], \tilde{e}_k] + \frac{1}{4}h(\tilde{e}_j, \tilde{e}_k), \tilde{e}_i] + \frac{1}{4}h(\tilde{e}_i, [\tilde{e}_k, \tilde{e}_j]) + \frac{1}{4}h(\tilde{e}_i, h(\tilde{e}_j, \tilde{e}_k))
\]

\[
- \frac{1}{4}[h(\tilde{e}_i, \tilde{e}_k), \tilde{e}_j] - \frac{1}{4}h(\tilde{e}_j, [\tilde{e}_k, \tilde{e}_i]) - \frac{1}{4}h(\tilde{e}_j, h(\tilde{e}_i, \tilde{e}_k))
\]

\[
+ \frac{1}{2}h([\tilde{e}_i, \tilde{e}_j], \tilde{e}_k)
\]

Then

\[
\tilde{R}(\nabla_{i\dot{x}}, \dot{x}) \dot{x} = -\frac{1}{4}[[\tilde{\nabla}_{i\dot{x}}, V], V] + \frac{1}{4}[h(V, V), \tilde{\nabla}_{i\dot{x}}] + \frac{1}{4}h(\nabla_{i\dot{x}} V, h(V, V))
\]

\[
- \frac{1}{4}[h(\tilde{\nabla}_{i\dot{x}}, V), V] - \frac{1}{4}h(V, [V, \tilde{\nabla}_{i\dot{x}}]) - \frac{1}{4}h(V, h(\tilde{\nabla}_{i\dot{x}}, V)) + \frac{1}{2}h([\tilde{\nabla}_{i\dot{x}}, V], V)
\]

\[
= -\frac{1}{4}[[\tilde{V}, V], V] - \frac{1}{8}[h(\tilde{V}, V), [V, V]] + \frac{1}{4}[h(V, V), \tilde{V}]
\]

\[
+ \frac{1}{4}h(\tilde{V}, h(V, V)) + \frac{1}{8}h(h(V, V), h(V, V))
\]

\[
- \frac{1}{4}[h(\tilde{V}, V), V] - \frac{1}{8}[h(h(V, V), V), V] - \frac{3}{4}h(V, [V, \tilde{V}]) + \frac{3}{8}h(V, [h(V, V), V])
\]

\[
- \frac{1}{4}h(V, h(\tilde{V}, V)) - \frac{1}{8}h(V, h(V, h(V, V))).
\]

When \( J = I \), the bi-invariant metric, substituting all the terms back into the original cubic expression, we find

\[
\tilde{\nabla}^3_i \dot{x} + \tilde{R}(\nabla_{i\dot{x}}, \dot{x}) \dot{x} = \tilde{V} + [\tilde{V}, V] = 0,
\]

which simply gives

\[
\tilde{V} = -[\tilde{V}, V].
\]

This is the negative of the Left-invariant cubics. Specifically when \( \dot{x} = xW \), we find \( W = -V \). This expression then matches the equations for the cubics found by Noakes et al in \[17\].

3.3 SubRiemannian cubics

The equations for subRiemannian cubics in a matrix Lie group \( G \) can also be derived from the PMP. The PMP only applies however when there are no derivatives of a
control function, \( w : [0, 1] \to \Delta \subset g \) in the object functional. When \( x : [0, 1] \to G \) is required to satisfy
\[
\dot{x} = wx,
\]
the Lie reduction of covariant acceleration gives a first order derivative of the control function \( w \),
\[
S[x] = \frac{1}{2} \int_0^1 dt \langle \nabla_t \dot{x}, \nabla_t \dot{x} \rangle_{\mathcal{J}} = \frac{1}{2} \int_0^1 dt \langle \dot{w} + \mathcal{J}^{-1}[\mathcal{J} w, w], w + \mathcal{J}^{-1}[\mathcal{J} w, w] \rangle_{\mathcal{J}}
\]
where \( \langle , \rangle_{\mathcal{J}} \) is a Riemannian metric and \( \mathcal{J} \) corresponds to an inertia transformation. Clearly \( \nabla_t \dot{x} \) is not always in \( \Delta \), so it is important to measure the accelerations in the perpendicular directions. Only \( w \) is constrained to be in \( \Delta \).

One option so we can apply the PMP is to define a new control function \( u : [0, 1] \to \Delta \), and an additional costate \( \vartheta \in g^* \). Then \( w \) is treated as a state variable,
\[
\dot{x} = wx,
\]
\[
\dot{w} = u.
\]
This is so the action, \( S[x] \), contains no derivatives of the control \( u \). Now the PMP Hamiltonian can be written as
\[
H = \lambda(wx) + \vartheta(u) + \frac{\nu}{2} \langle u + \mathcal{J}^{-1}[\mathcal{J} w, w], u + \mathcal{J}^{-1}[\mathcal{J} w, w] \rangle_{\mathcal{J}}.
\]
Where \( \lambda \in T^*G, \vartheta \in g^* \). The PMP requires that \( \dot{\lambda}(z) = -dH(z)x, z \in TG \), and \( \dot{\vartheta}(v) = -dH(v)_w, v \in g \). Computing the derivatives
\[
\dot{\lambda}(z) = -\lambda(wz),
\]
and
\[
\dot{\vartheta}(v) = -\lambda(vx) + \nu(\mathcal{J}^{-1}[\mathcal{J} v, w] + \mathcal{J}^{-1}[\mathcal{J} w, v], u + \mathcal{J}^{-1}[\mathcal{J} w, v])_{\mathcal{J}}.
\]
\( \lambda \) can be identified with a \( \Lambda^* \in g^* \) by \( \lambda(vx) =: \Lambda^*(v) \). \( \Lambda^* \) can then be identified with a \( \Lambda \in g \) via the bi-invariant inner product \( \Lambda^*(v) =: \langle \Lambda, v \rangle, \Lambda \in g \). Note \( \langle , \rangle_{\mathcal{J}} \) does not need to be used as it will only scale both sides of the resulting equations by \( \mathcal{J} \). Likewise \( \vartheta \in g^* \) can be assigned to a \( \Theta \in g \), also by the bi-invariant inner product \( \vartheta(v) =: \langle \Theta, v \rangle \).

Differentiating \( \Lambda^* \),
\[
\dot{\Lambda}^*(v) = \dot{\lambda}(vx) + \lambda(vwx)
\]
\[
= -\lambda(wvx) + \lambda(vwx)
\]
\[
= \Lambda^*([v, w]).
\]
which gives

\[ \langle \Lambda, v \rangle = \langle \Lambda, \{v, w\} \rangle \implies \dot{\Lambda} = -[\Lambda, w]. \]

Furthermore

\[ \langle \dot{\Theta}, v \rangle = -\langle \Lambda, v \rangle - \nu \left( \langle \mathcal{J}[w, u], v \rangle + \langle \mathcal{J}[w, \mathcal{J}^{-1}[\mathcal{J}w, w]], v \rangle \right) \]

\[ - \left( \langle [\mathcal{J}w, u], v \rangle - \langle [\mathcal{J}w, \mathcal{J}^{-1}[\mathcal{J}w, w]], v \rangle \right) \]

\[ \implies \dot{\Theta} = -\Lambda - \nu \left( \mathcal{J}[w, u] + \mathcal{J}[w, \mathcal{J}^{-1}[\mathcal{J}w, w]] \right) \]

\[ - [\mathcal{J}w, u] - [\mathcal{J}w, \mathcal{J}^{-1}[\mathcal{J}w, w]]. \]

### 3.3.1 Normal case

In the normal case \( \nu < 0 \), the optimal control \( u^* \) must maximise \( H \). When the metric is not bi-invariant the costates depend on \( \nu \) which adds additional complexity. For now we will only consider the bi-invariant metric, so \( \mathcal{J} = I \). Then without loss of generality we can set \( \nu = -1 \). Writing \( H \) in terms of inner products,

\[ H = \langle \Lambda, w \rangle + \langle \Theta, u \rangle - \frac{1}{2} \langle u, u \rangle. \]

Maxima occur when \( dH(u^*)_u = 0 \), so

\[ \langle \Theta, u^* \rangle - \langle u, u^* \rangle = 0, \]

\( \forall u^* \in \Delta \). So clearly this gives \( \text{proj}_\Delta(\Theta) = u \). The equations for the costates in \( g \) can then be reduced into a single equation,

\[ \hat{\Theta} = -[\hat{\Theta}, w], \quad (3.7) \]

where \( \Theta = \Theta_\perp + \hat{w} \), and \( \Theta_\perp \in g/\Delta \). As a check of this calculation, let \( \Delta = g \), then necessarily \( \Theta_\perp = 0 \). The resulting normal equations match the bi-invariant Riemannian case, \( \hat{w} = -[\hat{w}, w] \).

There are several conserved quantities in \((3.7)\). First, take the inner product of \((3.7)\) and \( w \), to find \( \langle \hat{w}, w \rangle = \langle \hat{w} + \Theta_\perp, [w, w] \rangle = 0 \). Second, take the inner product of equation \((3.7)\) and \( \hat{w} + \dot{\Theta}_\perp \), to find \( \langle \hat{w}, \hat{w} \rangle + \langle \dot{\Theta}_\perp, \hat{\Theta}_\perp \rangle = 0 \). Integrating these equations,

\[ \langle \hat{w}, w \rangle = C_2 - \frac{1}{2} \langle \hat{w}, \hat{w} \rangle, \quad (3.8) \]

\[ \langle \hat{w}, \hat{w} \rangle + \langle \dot{\Theta}_\perp, \hat{\Theta}_\perp \rangle = C_3. \quad (3.9) \]
3.3. SubRiemannian cubics

Clearly $C_3 > 0$. In general, solutions to (3.7) are difficult to find. One subset of solutions are the linear Lie quadratics

$$w = (q_0 + q_1 t + q_2 t^2)w_0, \quad \Theta_\perp = \Theta_0,$$

where $q_i \in \mathbb{R}$ and $w_0 \in \Delta$ and $\Theta_0 \in \mathfrak{g}/\Delta$ are constant matrices.

3.3.2 Abnormal case - SU(2)

The abnormal case is given by $\nu = 0$. The PMP Hamiltonian can be written as

$$H = \langle \Lambda, w \rangle + \langle \Theta, u \rangle.$$

Computing $dH_u(u^*)$, maxima occur when $(\Theta, u^*) = 0$. This occurs when $\text{proj}_\Delta \Theta = 0$, so $\Theta = \Theta_\perp \in \perp$ where $\perp$ is the orthogonal complement of $\Delta$. Unlike geodesics, even in simple groups such as SO(3) and SU(2), abnormals cannot be ruled out by a contradiction. Equation (3.7) reduces to

$$\hat{\Theta}_\perp = -[\hat{\Theta}_\perp, w]. \quad (3.10)$$

Using SU(2) as an example, let $\mathcal{B}_\Delta = \{\tau_1, \tau_2\}$, and $\mathcal{B}_\perp = \{\tau_3\}$, clearly the only only way to satisfy (3.10) is if $\hat{\Theta}_\perp = 0$. Then $\Theta_\perp = \Theta_0$, a constant. So the PMP does not give any information on how to choose $w$ in this case.

3.3.3 SubRiemannian cubics with tension

The abnormal case might be removed by adding an additional term in the action. With the bi-invariant metric, consider cubics with tension,

$$S[x] = \frac{1}{2} \int_0^1 dt \langle \dot{w}, \dot{w} \rangle + T \langle w, w \rangle$$

where $T \in \mathbb{R}_{\geq 0}$ is the tension. Here $\langle \cdot, \cdot \rangle$ will just be the restriction of the bi-invariant norm to the distribution. The more general case can be handled as before. Another motivation for the tension, is if the norm of a cubic is smaller, it will make it cheaper to approximate in terms of quantum circuit elements. Specifically because a larger step size can be used in approximating $x$. As before, introduce the additional control function $u$ and costate $\vartheta$, so $\dot{x} = wx$, $\dot{w} = u$. Writing the PMP Hamiltonian

$$H = \lambda(wx) + \vartheta(u) + \frac{\nu}{2} \left( \langle u, u \rangle + T \langle w, w \rangle \right).$$

The equations for the co-states can be computed in the same way as the cubics without tension. For the first costate

$$\lambda'(z) = -dH_x(z) = -\lambda(zx)$$
and for the second costate
\[ \dot{\eta}(v) = -dH_w(v) = -\lambda(vx) - \nu \langle v, w \rangle. \]
As before, these can be written as equations in the Lie algebra,
\[ \dot{\Lambda} = -[\Lambda, w] \]
\[ \Lambda = -\Theta - \nu \mathcal{T} w. \]
Writing as a single equation
\[ \dot{\Theta} + \nu \mathcal{T} \dot{w} = -[\hat{\Theta}, w]. \]
In the abnormal case, minima of $H$ occur when $\Lambda$ and $\Theta$ are perpendicular to $w$ as before. All time derivatives of $w$ vanish, giving no extra information. Hence adding the tension constraint does not remove the abnormal case.

3.3.4 Higher Order PMP for cubics

Using the method described in Chapter (2.4), we can attempt to find equations for 2-abnormal cubics in $\text{SU}(2^n)$. As required, introduce the new state equations \[ \dot{x} = \text{proj}_\Delta(\Gamma), \quad \chi(0) = 0, \quad \text{and} \quad \dot{\Gamma} = -[\Gamma, w], \] where $\Gamma = \dot{\Theta}$. Also $\dot{x} = wx$ and $\dot{w} = u$ as before. For now we will only consider cubics in the bi-invariant metric. This control system requires the PMP Hamiltonian
\[ H = \lambda_1(wx) + \lambda_2(u) - \lambda_3([\Gamma, w]) + \lambda_4(\text{proj}_\Delta(\Gamma)) + \frac{\nu}{2} \langle u, u \rangle, \]
where $\lambda_1 \in T^*G$, $\lambda_2, \lambda_3 \in g^*$ and $\lambda_4 \in \Delta^*$. As before equations can be found in $g$ for these quantities. Let $\Lambda_i$ denote the corresponding quantity in $g$.
\[ \dot{\Lambda}_1 = -[\Lambda_1, w] \]
\[ \dot{\Lambda}_2 = -\Lambda_1 - [\Lambda_3, \Gamma] \]
\[ \dot{\Lambda}_3 = -[\Lambda_3, w] + \text{proj}_\Delta(\Lambda_4) \]
\[ \dot{\Lambda}_4 = 0. \]
Clearly it is sufficient to take $\Lambda_4$ as a constant in $\Delta$. In the normal case, to solve for $u$, re-write $H$ in terms of inner products,
\[ H = \langle \Lambda_1, w \rangle + \langle \Lambda_2, u \rangle - \langle \Lambda_3, [\Gamma, w] \rangle + \langle \Lambda_4, \Gamma \rangle + \frac{\nu}{2} \langle u, u \rangle. \]
The optimal $u$ is required to minimise $H$. In the normal case, without loss of generality, take $\nu = -1$ and find $dH_u(u^*) = 0$, $\forall u^* \in \Delta \implies \text{proj}_\Delta(\Lambda_2) = u$. These are complicated coupled differential equations and would need to be solved numerically.
3.4 Normal cubics in three dimensional groups

3.4.1 SU(2)

In SU(2), $\Delta$ is the span of the $x$ and $y$ Pauli matrices, $\Delta = \text{span}\{\tau_1 = i\sigma_1, \tau_2 = i\sigma_2\}$, while $\bot$ is spanned by the Pauli $z$ matrix $\bot = \text{span}\{\tau_3 = i\sigma_3\}$. Let $\hat{\Theta}_\bot = \phi$. The equations for the control functions in $\mathfrak{su}(2)$ expand into two components,

$$\dot{\phi} = -[\dot{w}, w],$$

$$\ddot{w} = -[\dot{\phi}, w].$$

(3.11) can be integrated to give

$$\phi = -[\dot{w}, w] + C_1 \tau_3.$$

SubRiemannian cubics in SU(2) are termed null when $C_1 = 0$. Suppose $w$ can be written as

$$w = \text{Ad}(W)(rw_0),$$

where $r : [0, 1] \to \mathbb{R}$ and $w_0 \in \Delta$. In SU(2), suppose

$$W = \begin{pmatrix} e^{i\varphi} & 0 \\ 0 & e^{-i\varphi} \end{pmatrix},$$

and $\varphi \in \mathbb{R}$. If $\dot{r}(0) = \dot{r}(0) = 0$, then $\varphi = \varphi_0 t$, $\varphi_0 \in \mathbb{R}$, and the solutions reduce to geodesics.

**Lemma 3.4.1.** SubRiemannian geodesics are normal subRiemannian cubics with zero initial velocity in SU(2).

For non zero initial velocity and acceleration, in the limit $t \to \infty$, certain solutions to (3.11), are observed to behave like quadratics. To see this, simplify the problem by taking $w \to \omega \in \mathbb{C}$. Now equation (3.11) is written as a complex equation

$$\ddot{w} = \frac{1}{2} \omega(\dot{\omega} \omega - \dot{\omega} \omega) - iC_1 \omega.$$

In the null case, $C_1 = 0$, in polar coordinates $\omega = re^{i\vartheta}$, the equations separate into coupled radial and angular components

$$\ddot{r} - 3\dot{r}\dot{\vartheta}^2 - 3r\ddot{\vartheta}\dot{\vartheta} = 0,$$

$$3\dot{\vartheta}\dot{r} + 3r\ddot{\vartheta} - r\dot{\vartheta}^3 + r\ddot{\vartheta} = -\dot{\vartheta}r^3.$$

First make a change of variable and let $\dot{\vartheta} = \psi$. Multiply the first equation by $r$ and then integrate to obtain

$$\frac{1}{2} \dot{r}^2 + \ddot{r} - \frac{3}{2} r^2 \psi^2 + C_1 = 0,$$
3.4. Normal cubics in three dimensional groups

Figure 3.1: Radial and angular components of a null cubic control function in $\mathfrak{su}(2)$, with unit initial conditions.

where $C_1$ is another constant. Now in certain conditions the behaviour of the radial component, $r$ is observed to asymptotically be quadratic, while $\theta$ and $\dot{\theta} \to 0$. In this case, in the large time limit, the radial equation behaves like

$$-\frac{1}{2} \ddot{r}^2 + \dot{r}r + C_1 = 0.$$ 

This has solutions

$$r = \frac{(C_2 t + 2C_3)^2 - 2C_1 t^2}{4C_3},$$

where $C_2$ and $C_3$ are other constants. Now the angular equation becomes

$$r \ddot{\varphi} + 3r \dot{\varphi} + (3\dot{r} + r^3)\varphi = 0.$$ 

Recall, the transformed Bessel equation

$$x^2 \frac{d^2 y}{dx^2} + (2p + 1)x \frac{dy}{dx} + (a^2 x^{2r} + b^2)y = 0,$$

has solutions

$$y = x^{-p} \left( D_1 J_{q/r} \left( \frac{a}{r} x^r \right) + D_2 Y_{q/r} \left( \frac{a}{r} x^r \right) \right),$$

where $Y$ and $J$ are Bessel functions, $q = \sqrt{p^2 - b^2}$ and $D_1, D_2 \in \mathbb{R}$. If $r = q_2(t + q_1)^2$, first make the change of variable $z = t + k$, and then this case can be integrated in terms of Bessel functions.
4. Numerical calculations of geodesics and cubics

4.1 Geodesics

Integrating the subRiemannian geodesic equations analytically is a difficult task. Instead, as a more practical approach, geodesics can be computed numerically. While the geodesic equations can be forward integrated easily, solving the boundary value problem adds additional complexity. For quantum computing the geodesics are also required to have minimal energy. Solving this boundary value problem can be treated as a constrained optimisation problem. Specifically finding the initial conditions $\Lambda_0$, with the smallest norm, such that for a trial solution $\tilde{x}$, the error, $\text{err}(\tilde{x}(1), U)$ is small.

Other work has attempted to find subRiemannian geodesics by slowly increasing the penalty $p$ in the Riemannian geodesic equations, using a previous iteration as a guess for the current value of $p$. There is also no guarantee that this method gives an optimal solution. The method proposed here does not rely on increasing $p$ as it solves the subRiemannian equations directly.

From numerical experimentation, it was found that using the trace error, given by

$$\text{err}(x, U) = 1 - \frac{1}{2^n} \text{tr}(U^\dagger x)$$

gave faster convergence. Additionally this definition of error is much easier to calculate compared to the Riemannian distance which involves a matrix logarithm. The only downside being that this error is less accurate, but the results it generates can be refined by switching back to the Riemannian distance definition.

Recall that geodesics can be found by solving the equation

$$\dot{x} = \text{proj}_\Delta(x\Lambda_0 x^\dagger)x$$

where $\Lambda_0 = u_0 + \Lambda_0^\dagger$ and proj is projection onto $\Delta$. While standard numerical integration methods can be used, they will not preserve the unitary property of $x$, (with error order of the integration method). Given that the matrix exponential can be computed accurately given $x$ at step $j$, $x_j$, $x_{j+1}$ can be found by the modified Euler discretisation

$$x_{j+1} = \exp(h \text{proj}_\Delta(x_j\Lambda_0 x_j^\dagger))x_j.$$ 

In a similar fashion higher order geometric integrators, analogous to Runge-Kutta, which respect the group structure are easily constructed.
4.1.1 Direct Search

The direct search method is simply forward solving the geodesic equations and adjusting the initial conditions to minimise the error. This can be done with a quasi-Newton method. The main problem with this approach is that the error function has many local minima, slowing convergence. Further complicating the matter is that there are infinitely many geodesics joining two points with differing energy. Optimal geodesics are solutions with the smallest energy such that the boundary conditions are satisfied. To eliminate nonoptimal solutions, constrained optimisation can be used to systematically rule out higher energy geodesics.

As the energy is completely determined by the norm of the initial conditions, it is simply a matter of constraining this norm at each stage to forbid nonoptimal geodesics. Evidently, the subRiemannian distance is bounded below by the (bi-invariant) Riemannian distance, $d_{\text{sub}}(I,U) \geq d_{\text{Riem}}(I,U)$. Hence for the normal subRiemannian geodesics there is a lower bound on the norm of the initial conditions $\|\text{proj}_\Delta(\Lambda_0)\| = \|u_0\| \geq \|\log(U)\|$. The Riemannian distance is simply the norm of the principal branch of the logarithm, $\|\log(U)\|$.

Now to find subRiemannian geodesics, an initial $\Lambda_0$ is chosen such that $\|u_0\| = c$, where $c$ is larger than the Riemannian distance. A simple initial guess is a scalar multiple of $\log(U)$. This $\Lambda_0$ is adjusted via constrained optimisation (with constraint $c$) until the error is small, $\text{err}(x(1),U) \leq \epsilon$, where $\epsilon$ is some small parameter. Next the constraint is lowered by some small parameter $\delta$, $c \rightarrow c - \delta$. Then the previous $\Lambda_0$ are also adjusted $\Lambda_0 \rightarrow \tilde{\Lambda}_0$, so $\|\tilde{u}_0\| \leq c$. Finally, the $\tilde{\Lambda}_0$ is adjusted again via constrained optimisation, with constraint $c$, so the error is small. This procedure is iterated until the error fails to converge below $\epsilon$, or the norm is equal to the Riemannian distance (then the target is the exponential of elements in the allowed set).

A more sophisticated way to generate an initial guess for $\Lambda_0$, is to use the series solution for $x$, given in the previous chapter, truncated to some order. This also requires optimisation, but can improve overall convergence. Truncating the series to first order, often a choice of $\Lambda_0$, so $\text{err}(y,U) \leq \epsilon$ where $y = \exp(-\text{proj}_\perp(\Lambda_0)) \exp(\Lambda_0)$ would be adequate. The accuracy of this initial guess can be further improved by using Leapfrog, which divides $x$ into a number of shorter segments which can iteratively be improved [36].
4.1. Geodesics

4.1.2 Discrete Geodesics

A significant problem with the direct search method is that the size of the basis for $\mathfrak{su}(2^n)$ scales exponentially. Worse, integrating the geodesic equations requires all the basis matrices to be stored in memory. Even with sparse data structures this presents significant challenges. A much more efficient way to find geodesics is to discretise the problem entirely. First consider the discrete curve $x: \mathbb{R}^{N|\Delta|} \to \text{SU}(2^n)$, composed of $N$ segments,

$$x = \exp(u_1^{1}/N\tau_i) \ldots \exp(u_N^{n}/N\tau_i)$$

where each $u_i^j \in \mathbb{R}$, and $\tau_i \in \Delta$ (repeated indices summed). $x$ will be a discrete subRiemannian geodesic when it minimises the discretised energy functional.

$$\mathcal{E}[x] = \frac{\hbar}{2} \sum_{i=0}^{N-2} \left( \langle u(t_{i+1}), u(t_{i+1}) \rangle + \langle u(t_i), u(t_i) \rangle \right), \quad (4.1)$$

where $u(t_i) = u_i^j \tau_j$. As before, to reach an arbitrary $U \in \text{SU}(2^n)$ exponentially many segments will still be required in general. The main advantage here is that only the allowed set will need to be generated, along with a vector of real coefficients $u_i^j$. This is still significantly smaller than the entire basis for $\mathfrak{su}(2^n)$.

To find the discrete geodesic joining $I$ and $U$, minimise the object function, $F$ which is the sum of the energy and the error,

$$F = \kappa \text{err}(x, U) + \varepsilon \mathcal{E}[x]$$

where $\kappa$ and $\varepsilon$ are experimentally chosen weightings.

One common method to minimise such a function is to use Newton’s or a quasi-Newton method. For this approach $\kappa$ and $\varepsilon$ were found to require scaling with the error function. If they did not change the optimisation procedure would become stuck at a local minimum, with suboptimal error. From experimentation, scalings of $\varepsilon = 10^{-p}\text{err}(x, U) + 10^{-q}$ and $\kappa = 10^3$ gave best results. This is to stop the energy term dominating the error term. Figure (4.1) shows the improved convergence with the adaptive scaling.

For $x$ to accurately approximate $U$, the trace error needs to be less than $O(10^{-8})$. The weighting ensures that early in the optimisation procedure, a greater emphasis is placed on the error. Once a non optimal curve joining $I$ and $U$ is approximately found, it can be refined into having lower energy.
The Hessian of $F$ is an expensive function to calculate. This is because the derivatives of the error term,

$$\frac{\partial^2 \text{err}}{\partial u_k^1 \partial u_p^q} = \frac{1}{N^2} \text{tr}(U^\dagger x_1 \ldots \tau_j x_k \ldots \tau_p x_q \ldots x_n),$$

(4.2)

involve many products of matrix exponentials in each entry. The calculation of the gradient of $F$ can be found in Appendix B. To avoid evaluating this expensive function, the Hessian can be approximated at each step. In general the problem will scale poorly to higher dimensions if the Hessian has to be evaluated. Future work should investigate alternative optimisation methods which do not calculate the gradient.

4.1.3 Results

Direct

Figure 4.2 shows two of the 36 control functions generated by the direct search for a randomly chosen target in $\text{SU}(8)$. The first iteration, shown in red, gave an energy of 36.806. The second iteration, shown in blue, gave an energy of 24.74. Beyond this the algorithm failed to converge. The trace error was $\mathcal{O}(10^{-8})$ for both cases. The algorithm was implemented in Mathematica 11.0 and took, over several trials, an average of 9.5 hours on an i7-5960x.

Interestingly, the reduction in norm of the control functions was also often accompanied by a reflection. This type of behaviour is more easily visualised on the sphere $S_2$. Starting from the north pole, a geodesic is a line of longitude. The minimal solution will be a geodesic which is just a single arc. Higher energy geodesics may wrap around the sphere multiple times. The same situation can occur in $\text{SU}(8)$, but this is a 63 dimensional space so the differences may be more subtle.
Figure 4.2: Comparison of two of the 36 control functions after two iterations of the direct search. Red is first iteration, blue is second. Other control functions displayed similar behaviour.

Discrete

Figure 4.3 shows a sample of the discrete control functions generated for randomly chosen targets in SU(16) and SU(32).

Figure 4.3: Same of the control functions for discrete geodesics in SU(16) and SU(32), generated by optimisation.

Direct vs Discrete

For three random unitary matrices in SU(8) we benchmarked the two different methods. Both methods were run until the trace error was $O(10^{-8})$. The discrete curves had $N = 10$ segments. For the direct search a step size of $h = 0.001$ was used. Both algorithms were implemented in Mathematica 11.0 and run on a 3.7 GHz (fixed) i7 5960x. More segments could have been used, but since each trial converged to the desired accuracy this highlights the increased efficiency of the discrete method. It is likely as $n$ increases more segments will have to be used.
Figure 4.4: Comparison of some discrete and direct control functions in SU(8). Red is result of direct search, blue is discrete. Often the discrete result would show a reflection.

<table>
<thead>
<tr>
<th></th>
<th>Timing - Direct</th>
<th>Timing - Discrete</th>
<th>Energy - Direct</th>
<th>Energy - Discrete</th>
</tr>
</thead>
<tbody>
<tr>
<td>$U_1$</td>
<td>9.94 hrs</td>
<td>1.52 hrs</td>
<td>20.175</td>
<td>25.6217</td>
</tr>
<tr>
<td>$U_2$</td>
<td>8.78 hrs</td>
<td>1.45 hrs</td>
<td>25.785</td>
<td>26.541</td>
</tr>
<tr>
<td>$U_3$</td>
<td>1.94 hrs</td>
<td>48.45 mins</td>
<td>23.054</td>
<td>25.399</td>
</tr>
</tbody>
</table>

Comparing some of the control functions generated by the discrete and direct search. Often the discrete result would be a reflection of the exact solution. This is analogous to the situation on the sphere $S_2$. Consider the geodesics from the north to south pole. In this case the geodesics can be thought of as the lines of longitude. To reach the south pole there are infinitely many choices, but they all have equal energy. The energy of the discrete and the exact result was found to be comparable for all targets tested. This then suggests the most efficient way to find subRiemannian geodesics is to use the initial data of a discrete curve as an initial guess for the direct search.

4.2 Cubics
4.2. Cubics

4.2.1 Direct search for cubics

Given an initial velocity and acceleration, \( \dot{\mathbf{v}}(0) \), and \( \ddot{\mathbf{v}}(0) \) to satisfy \( x(1) = U \). The simplest way to calculate cubics is to forward solve the cubic equations and then adjust initial conditions based on the error at the \( x(1) \) boundary. Recall that for subRiemannian geodesics this process was relatively inefficient compared to a discrete method.

4.2.2 Discrete Cubics

Like discrete geodesics, discrete cubics are minima of the discretised cubic functional

\[
S[x] = \frac{\hbar}{2} \sum_{j=1}^{N-1} \langle \dot{\mathbf{v}}(t_j), \dot{\mathbf{v}}(t_j) \rangle + \langle \ddot{\mathbf{v}}(t_{j-1}), \ddot{\mathbf{v}}(t_{j-1}) \rangle
\]

\[
= \sum_{j=1}^{N-1} ||w(t_j) - w(t_{j-1})||^2
\]

(4.3) (4.4)

where \( \langle , \rangle \) is the restriction of the bi-invariant metric. Using other metrics would not change the procedure significantly. Now to find the discrete cubics, numerically minimise the weighted sum

\[
F = \omega \text{err}(x_A(u), U) + \kappa \sum_{j=1}^{N-1} ||w(t_j) - w(t_{j-1})||^2
\]

where \( \kappa \geq 0 \) and \( \omega > 0 \) are chosen experimentally, and err is the error. To solve the continuous cubic equations, the value of \( \dot{\mathbf{v}}(0) \) also needs to be specified. In the discrete case, this can be achieved by adding a constraint on the components so \( w(t_2) - w(t_1) = \hbar \dot{\mathbf{v}}(0) \). For the quantum computing application the need of such a constraint is not clear. The initial acceleration does not particularly correspond to any physical quantity.

If no constraints are given, the optimisation procedure would often converge to a control function, for example Figure [4.5], which has zero acceleration for a duration, and then a sudden spike. This type of behaviour is not apparent from the normal equations given by the PMP. Controls with zero initial acceleration and acceleration, in the normal case, are required to be geodesics. Such behaviour is not apparent in subRiemannian geodesics. This potentially is a numerical example of an abnormal cubic. Compared to geodesics, this could be considered a simpler circuit, as gates can be reused. Approximating the previous geodesics as a circuit, the phase gates change constantly. The curve shown in [4.5] the phase gates only change once. However there is still a significant problem, as the cubic is allowed to move in a linear combination of directions at each time step.
4.2 Cubics

4.2.3 Indefinite Cubics

Unfortunately cubics are still not the most accurate (continuous) description of an efficient quantum circuit. A cubic is still allowed to move in a linear combination of directions in $\Delta$. So at a small time step $x(t+h) \approx \exp(hw^i\tau_i)x(t)$, and each $\exp(hw^i\tau_i)$ requires additional approximation. The naive way to approximate this exponential is to use the Lie product formula. However, there might be an exact product of exponentials which implements the exp of the linear combination in fewer circuit elements than using the Lie product formula.

Clearly the cost of a circuit must depend on the number of changes in direction, but it should be more expensive to move in a linear combination of directions. While the ideal solution is nonsmooth, a small regularisation is an acceptable compromise. As long as it is small it can approximately be removed in a quantum circuit. To try and force curves to move in only one direction, modify the cubic functional by adding the indefinite inner product to the cubic equation,

$$S_I[x] = \frac{1}{2} \int_0^1 dt \langle \nabla_t x, \nabla_t \dot{x} \rangle + \langle w, w \rangle_I$$

where $\langle w, w \rangle_I = w_i I_{ij} w_j$ is what we call the indefinite inner product, $w_i$ is a real coefficient, and $I = J - I$. This is to try and make movements in linear combinations of directions more expensive than moving in a single direction.

As before, the most efficient way to try and compute minima of these functionals is to discretise the problem entirely. We first attempted to find indefinite cubics by adding the indefinite inner product to the discrete object function

$$F = \omega \text{err}(x,U) + \kappa \sum_{j=1}^{N-1} \|w(t_j) - w(t_{j-1})\|^2 + \eta \sum_{j=0}^{N-1} \langle w(t_j), w(t_j) \rangle_I$$
where $\eta$ is another experimentally chosen weighting. It took significantly longer to find the minima of this function. Even for a simple target, for example $U_4 = \exp(i(\sigma_1 \otimes \sigma_2 \otimes \sigma_3 + \sigma_3 \otimes \sigma_2 \otimes \sigma_1))$, the process took 4.1 hours to find a curve joining $I$ to $U_4$, figure (4.6). Note that at each step $i$ the control $w$ approximately takes on a single value, which is the desired behaviour. More complicated targets the time extended significantly. This is because many more segments are required. When each $x_j$ is the exponential of a linear combination, products of consecutive $x_j$ can bracket generate more perpendicular directions. When the $x_j$ are only the exponential of a single allowed instruction, more terms are needed to bracket generate perpendicular directions. This is evident from the Baker-Campbell-Hausdoff formula
\[
\exp(X) \exp(Y) = \exp(X + Y + \frac{1}{2}[X,Y] + \ldots).
\]
Currently it might be more practical to use geodesics to create circuits, and accept some error from approximation. Alternatively one might consider just minimising the error between $U$ and products of exponentials of single instructions. The problem with this approach is that there is no unique way to do this. Cubics and geodesics are unique which means there is a better chance of the discrete optimisation succeeding. Figure (4.6) does show it is indeed possible to generate indefinite cubics. Possibly more advanced optimisation procedures can improve this process in future.

![Figure 4.6](image.png)

Figure 4.6: Sample of indefinite cubic control functions with no constraints on initial conditions in $SU(8)$ for $U_4$. 

5. Neural Networks

5.1 Background

Solving the cubic and geodesic boundary value problems require expensive optimisation procedures. This significantly hinders practical usage. A much faster method needs to be devised. One option is to explore better optimisation methods. Alternatively, with advances in computational power, neural networks might be an attractive option for generating approximate circuits. Due to the high dimension and the non commutativity of the matrix exponential it is very difficult to analytically obtain initial guesses for the previous approaches in Chapter (4). Being able to generate a good initial guess which can later be refined by the previous methods would be a significant step forward.

The most basic type of neural network is a multilayer perceptron. This type of neural network consists of several fully connected layers of neurons. The layers are divided into input, hidden and output layers. Each neuron in a layer is fully connected to all the neurons in the previous layer. A neuron accepts a vectors of real numbers as an input, and outputs a single real number. This output is computed by first taking the weighted sum of the inputs, and then passing this sum to an activation function, which then produces an output. Common activation functions include the logistic function, the hyperbolic tangent (tanh), and the rectified linear unit (ReLu).

\[ S = \sum_i w_i \sigma_i \]

\[ \sigma_4 = f(S) \]

Figure 5.1: A basic neuron, \( \sigma_{1-3} \) are the inputs, \( w_i \) are the input weights, \( f \) is an activation function, \( \sigma_4 \) is the output of the neuron.

Training a neural network uses a technique called backpropagation. Training data consists of inputs and desired outputs. First, the input data is fed through the network. The error between the output, and the desired output is computed. Gradient descent is used to minimise the error by adjusting the weights. However instead of adjusting all the weights simultaneously, the weights are adjusted sequentially from the final output layer to the input layer via the chain rule. More complicated
5.1 Background

Figure 5.2: A very simple neural network, $\sigma_i$ are the inputs / outputs, $w_i$ are the input weights, $f$ is an activation function. Numbered circles denote neurons.

networks exist featuring different connectivity and types of neurons. Two common types are Convolutional Neural Networks (CNNs), and Long Short-Term Memory neural networks (LSTMs).

Recall, the problem is to find $U$ approximately as a product of exponentials

$$U \approx E(c) = \exp(c_1^1 \tau_1) \cdots \exp(c_{m}^1 \tau_{m})$$

$$\cdots \exp(c_1^N \tau_1) \cdots \exp(c_{m}^N \tau_{m}), \quad (5.1)$$

where $E$ we call the embedding function, $c = (c_1^1, \ldots, c_{m}^N)$ and the $\tau_i$ are a basis for a bracket generating subset of the Lie algebra $\Delta \subset \mathfrak{su}(2^n)$ of dimension $m$. Bracket generating means that repeated Lie brackets of terms in $\Delta$ can generate any term in $\mathfrak{su}(2^n)$. Because products of matrix exponentials generate Lie bracket terms

$$\exp(A) \exp(B) = \exp(A + B + \frac{1}{2}[A, B] + \ldots),$$

any $U \in \text{SU}(2^n)$ can be written as Equation (5.1) with sufficiently many products. We restrict ourselves to $U$ which can be written as a product of a polynomial in $n$ terms. An example of such a $\Delta$ could be the matrix logarithms of universal gates.

For convenience it is easier to work with all permutations of Kronecker products of one and two Pauli matrices, so

$$\Delta = \text{span}\{\frac{i}{\sqrt{2^n}}\sigma_i^j, \frac{i}{\sqrt{2^n}}\sigma_i^k\sigma_j^l\},$$

where $\sigma_i^j$ represents the $N$ fold Kronecker product, $I \otimes \cdots \otimes \sigma_i \otimes \cdots \otimes I$, with a $\sigma_i$ inserted in the $j$-th slot and $I$ representing the $2 \times 2$ identity matrix. Exponentials of these basis elements have very simple circuits, for more detail see Chapter (1).

We propose that a neural network be trained to learn $E^{-1}$. The neural network will try to find all the coefficients $c_i^k$ so that the product approximates $U$. In this approach, the neural network takes a unitary matrix $U$ as an input and returns a
list $c$ of $c^k_i$. A segment is a product of $m$ exponentials of each basis element. In total there are $N$ segments. We only examine $U$ which are implementable in a reasonable number of segments. We found that we required two neural networks to achieve this. The first is a Gated Recurrent Unit, GRU, network \cite{37,38} which factors a $U$ into a product of $U_j$, 

$$U \approx U_1 U_2 \ldots U_j \ldots U_N,$$

where each $U_j$ is implementable in polynomially many gates, which we call \textit{global decomposition}. The second is simply several dense fully connected layers, which decomposes the $U_j$ into products of exponentials 

$$U_j \approx \exp(c^1_j \tau_1) \ldots \exp(c^m_j \tau_m),$$

which we term \textit{local decomposition}. These procedures can be done with traditional optimisation methods. But the lack of a good initial guess meant that it took an order of an hour in $\text{SU}(8)$. While the output from the neural network may not implement $U$ to a required tolerance, it does provide a good initial guess as the error will be small.

### 5.2 Training data

To generate the training data, the $c$ should not be chosen randomly. If there is no structure to how $c$ is chosen, it will introduce extra redundancy. More seriously, $E^{-1}$ will not be well defined. There are infinitely many ways to factor a $U$ into some unordered product of matrix exponentials. Geometrically this could be visualised as taking any path from $I$ to $U$ on $\text{SU}(2^n)$. Randomly generating data may give two different decompositions for a $U$, and so $E$ is not one to one. To ensure the training data is unique, we propose that these paths should be chosen to be, at least approximately, minimal normal subRiemannian geodesics.

The choice of using geodesics is not particularly special. Other types of curves could be used as long as it uniquely joins $I$ and $U$. This is so $E^{-1}$ is well defined. Generating random geodesics can be done simply by generating random initial conditions. However the geodesics must also be minimal. The first way to try and ensure they are minimal is to bound the norms of the initial conditions.
From Chapter (3), the normal subRiemannian geodesic equations can be written as

\[ \dot{x} = ux, \]
\[ \dot{\Lambda} = [\Lambda, u], \]
\[ u = \text{proj}_\Delta(\Lambda), \]

where \( \Lambda : [0, 1] \to \mathfrak{su}(2^n), \ u : [0, 1] \to \Delta \subset \mathfrak{su}(2^n) \) and \( \text{proj}_\Delta \) is projection onto \( \Delta \).

This can be re-written as the single equation

\[ \dot{x} = \text{proj}_\Delta(x\Lambda_0 x^T)x, \tag{5.2} \]

where \( \Lambda_0 = \Lambda(0) \). Choosing the \( \Lambda_0 \) completely determines the geodesic. To generate the training data for the \( U_j \), first randomly choose a \( \Lambda_0 \). The \( U_j \) are then matrices which forward solve the geodesic equations

\[ x(t_{j+1}) \approx U_j x(t_j), \]

where \([0, 1]\) has been divided into \( N \) segments of width \( h \). For this paper we utilised the simple first order integrator

\[ U_j = \exp\left( h \text{proj}_\Delta(x_j\Lambda_0 x^T_j)\right), \]

since approximating the geodesic is sufficient. There are infinitely many bi-invariant Riemannian geodesics joining \( I \) and \( U \), for the different branches of \( \log(U) \). Sub-Riemannian geodesics are similarly behaved, but it varies on the norm of \( \Lambda_0 \). To generate the training data we bounded the norms by \( \dim(\Delta) = O(n^2) \), to try and ensure the geodesics are unique.

Further, the norm \( \|\text{proj}_\Delta(\Lambda_0)\| = \|u_0\| \) determines the distance between \( I \) and a \( U \). Nielsen showed that the distance can be thought of as approximately the complexity to implement \( U \). Lemma (3) in [1] shows that a \( U \) further away from \( I \) requires more gates. The distance however is likely to scale exponentially. By bounding the norm by a polynomial, this ensures the training data only contains \( U \) which are reachable with a polynomial number of quantum gates.

Naively, a neural network could take all the real and complex entries of \( U \) as inputs. For most \( U \) this is halved by only taking the real or imaginary components. Let \( U = B + iC \). If \( B \) or \( C \) is given then it is likely that the other component can be recovered. Suppose \( B \) or \( C \) is given. Then the other can be found when the system of equations

\[ (B + iC)(B - iC)^T = I \]

\[ (5.3) \]
has a unique solution for $C$ or $B$. Taking the real and imaginary components,

$$BB^T + CC^T - I = 0$$
$$BC^T - CB^T = 0.$$ 

This is equivalent to asking if the map $\phi(B) = (BB^T + CC^T, BC^T - CB^T)$ has full rank. Differentiating,

$$d\phi_B(W) = (WB^T + BW^T, WC^T - CW^T),$$

and suppose $d\phi_B(W) = (S, A)$, where $S$ is symmetric, and $A$ is anti-symmetric. Then either

$$W = \frac{1}{2}S(BB^T)^{-1}B \quad (5.4)$$

or

$$W = \frac{1}{2}A(CC^T)^{-1}C \quad (5.5)$$

So given that $BB^T$ or $CC^T$ is invertible, then solutions will likely exist. This then halves the total number of inputs.

5.3 Network Design - SU(8)

5.3.1 Global decomposition

The neural network for the global decomposition takes an input of $U$ and returns a list of $U_j$. To do this $U$ is decomposed into rows of length $2^n$. This makes $2^n$ real vectors. Each row is treated as a single timestep in the GRU layer. The output $U_j$ are also decomposed into their rows and these rows are treated as timesteps in the output. This gives $2^nN$ output vectors of length $2^n$. In particular we examined the $n = 3$ qubit case. For SU(8) we found 10 stacked GRU layers was sufficient to give reasonable results.

In SU(8) we chose $N = 10$ segments, so there were 8 input vectors of length 8 and 80 output vectors of length 8. The network was implemented in the Keras Python library with the TensorFlow backend, on a Nvidia GTX 1080.

5.3.2 Local decomposition

For SU(8) a network with 2 fully connected dense hidden layers of 2000 neurons, with the ReLU activation function was found to be sufficient. The input layer took a vectorised $U_j$, and outputted $\dim(\Delta)$ values. The network was implemented in the Keras Python library with the TensorFlow backend, on a Nvidia GTX 1080.
5.4 Results - SU(8)

5.4.1 Global decomposition

The global decomposition network was trained on $U_j$ taken from 5000 randomly generated geodesics in SU(8). 500 were used for validation data. The loss function used was the standard Euclidean distance between the output vector and the desired output. The Euclidean distance is simply a rescaling of the standard mean squared error. However it performed better, as it made the local minima more distinguishable. The rescaling After 1500 training epochs the validation loss reached $\sim 0.9$ and did not decrease. This was found to be sufficient to generate $U_j$ close to the training data.

Figure (5.3) shows the validation and training loss. Figure (5.4(a)) and figure (5.4(b)) shows a randomly chosen $U_j$ from a list of $U_j$ generated by the network, and from the training data respectively for some random $U$. Most $U_j$ appeared to be very similar. Figure (5.5(a)) and figure (5.5(b)) show the same entry in consecutive $U_i$ for validation data. Again the network was able to output values very close to the values in the validation dataset. This similarity was typical. This shows the network is able to reasonably approximate the $U_j$.

![Global Decomposition Loss](image)

Figure 5.3: The loss and validation loss from training the global decomposition.
5.4. Results - SU(8)

Figure 5.4: A known $U_j$ from the validation data and the $U_j$ generated by the NN in SU(8) for global decomposition. Each $U_j$ is close to the identity matrix. The shading from blue to orange represents $[-1, 1]$

Figure 5.5: Real entries of validation $U_i$ vs the $U_i$ generated by the NN. Recall the $U_i$ are not constant, and solve equation (5.2). The behaviour displayed here was typical in other entries.

5.4.2 Local decomposition

The network to implement the local decomposition was trained on $U_j$ generated by choosing a random $m$-vector of the coefficients $c^j_i$, where each $c^j_i$ was order $1/N$. In total there were 5000 pairs in the training set, and 500 in the validation set. Figure (5.6) shows the validation and training loss. After 500 epochs the network was able to sufficiently compute the local decomposition to reasonable error (on average 0.16). Figures (5.7(a)) and (5.7(b)) show a matrix generated by the neural network and the target matrix.
Figure 5.6: The loss and the validation loss from training the local decomposition. There was no significant improvement after 500 epochs.

Figure 5.7: A known $U_j$ from the validation data and the $U_j$ generated by the NN in SU(8). These figures are for the local decomposition network. The shading from blue to orange represents $[-1, 1]$

5.5 Discussion

Training two neural networks to together decompose $U$ into $c^i_j$ via a two-step approach (global decomposition followed by local decomposition) was found to be successful, when restricting the set of training data generated to paths which approximate minimal normal subRiemannian geodesics. This restriction limited the training data pairs to ones which were one-to-one, eliminating redundancy. For the global decomposition, using a neural network consisting of stacked GRU layers allowed for efficient training of the network, with the validation loss of the network approaching its minimum at 500 epochs for SU(8). A simple dense network with two hidden layers proved sufficient for the local decomposition. In SU(8), the networks
were small enough that both networks were able to be trained on a desktop machine with a single Nvidia GTX 1080 GPU. The two stage decomposition proved more successful than single-stage attempts to form a solution, with the decomposition of a given $U$ into $U_j$ being crucial for this increase in effectiveness. This approach to the solution for this problem demonstrates a novel use of neural networks.

Although this approach works well for systems with small numbers of qubits (such as the SU(8) case used as an example), the approach does not scale well with increasing number of qubits. This is because the size of the network scales by the number of entries in matrices in SU($2^n$). Although this is not a significant problem for currently realisable quantum computers, or those in the near future, it will increasingly become problematic as quantum computing continues to advance. To somewhat counteract this, the complexity of the problem can be decreased by restricting the set of $U$ on which the neural network is trained. For example if the $U$ are sparse, some savings in the size of the network may be made. Investigating this will be increasingly significant, as it will increase the practical usefulness of this approach.

As noted in section 5.2, the choice of using geodesics to restrict the training data is fairly arbitrary, and as such, there may be different ways of restricting the training data which, while still ensuring the input/output is one-to-one, may produce a better dataset, improving the accuracy of the networks. This is heavily related to the nature of $A_0$ which is currently not fully understood. Exploring this problem is a possible future avenue of investigation, which may improve the effectiveness of the approach described in this paper.

Finally note that training the network is the most computationally expensive part of this approach. Once the network is trained, propagating an input through the network is much more efficient than the conventional optimisation techniques for compiling $U$. 
6. Summary

The ability to efficiently compile specified $U$ into elementary gates is an assumption made in many quantum algorithms. Before Nielsen et al. [1–5], quantum compilation was treated as an algebraic problem. Trying to describe an optimal circuit using geometry is a novel approach. Previous work [1–8] in this direction focused on computing geodesics on a Riemannian manifold with a penalty metric where the penalty is made large. This added additional complexity as the penalty had to gradually be made large numerically. Recasting it as a problem in Subriemannian geometry removes the need for a penalty.

Subriemannian geodesics and cubics, while interesting mathematical objects, still have limited practicality in constructing quantum circuits. This is due to the difficulty in solving the boundary value problem in large dimensions. To significantly reduce the computational cost, we numerically calculated discretised cubics and geodesics. In SU(8) the discrete approach gave double or several times speedup, and with a much smaller memory footprint.

Naively computing geodesics also ignores information about the $U$ commonly required. Often the $U$ belong to a low dimensional subset of SU($2^n$). Solving the continuous boundary value problem, as it stands, requires adjusting an exponential number of parameters which is computationally prohibitive for larger $n$. The discrete methods can better account for the low dimensional $U$ by only taking some polynomial sized number of products. Future work could try to relate the sparse nature of $U$ to the number of parameters needed in a more rigorous way.

Another caveat with the geodesic approach is that there was no consideration for error correction. While the CNOT, $H$ and $Y$ gates can be implemented in an error correcting way, the phase gate is not. It must further be decomposed into a product of $H$, and $T$ gates which does add additional complexity. However this is unavoidable. The geodesic approach requires the continuously variable parameters in the phase gates so a continuous optimisation method can be used. This could then be considered an advantage, as a search algorithm dealing with the discrete objects would be even more computationally expensive as it must check permutations of discrete gates.

The neural network approach offers an interesting practical tool. For three qubits the neural network was able to generate circuits for arbitrary $U$, with bounded
complexity, with minimal error. Additionally, once the network had been trained, this was significantly faster than performing an optimisation procedure. Future work will extend this approach to higher dimensions. All data and programs used to produce this work can be found at https://github.com/Swaddle/nnQcompiler.
Bibliography


Appendix A

6.1 Commutation Relations

6.1.1 Commutation relations in \(\text{su}(8)\)

The Pauli basis has several useful properties

\[
\begin{align*}
[v_1, v_1] &\subset v_1 \\
[v_1, v_2] &\subset v_2 \\
[v_1, v_3] &\subset v_3 \\
[v_2, v_3] &\subset v_2.
\end{align*}
\] (6.1)

where \(v_1 = \text{span}\{i\sigma_i^1\}\), \(v_2 = \text{span}\{i\sigma_i^2\sigma_j^2\}\), \(v_3 = \text{span}\{i\sigma_i^2\sigma_j^2\sigma_k^2\}\). This can be verified by computing Lie brackets of basis elements of these subspaces. First for elements in \(v_3\) and \(v_2\),

\[
i^2[\sigma_i \otimes \sigma_j \otimes I, \sigma_r \otimes \sigma_s \otimes \sigma_t] = -i\epsilon_{irj} \theta \otimes \sigma_t - \sigma_r \otimes \sigma_s \otimes \sigma_j \otimes \sigma_t
\]

\[
= 2i\epsilon_{irj} \theta \otimes \delta_{sj} I \otimes \sigma_t + 2i\delta_{ir} \otimes \epsilon_{jsk} \sigma_r \otimes \sigma_t \in v_2.
\]

where \(\sigma_i \sigma_j = i\epsilon_{ijr} \theta \delta + \delta_{ij} I\). Other choice of \(m\) and \(n\) in \(\sigma^m_i \sigma^n_j\) will follow similarly, so \([v_2, v_3] \subset v_2\), as the \(v_i\) are closed under addition. Now checking Lie brackets of terms such as \(i\sigma^m_i \otimes i\sigma^n_j \otimes i\sigma^p_k\), to examine \([v_1, v_2]\),

\[
[i\sigma_i \otimes I \otimes I, i\sigma_j \otimes \sigma_k \otimes I] = [i\sigma_i, i\sigma_j] \otimes \sigma_k \otimes I
\]

which is at least contained in \(v_2\). Alternatively if there is no overlap of Pauli matrices in the Lie bracket, for example \([I \otimes I \otimes \sigma_i, \sigma_j \otimes \sigma_k \otimes I] = 0\), the Lie bracket is still in \(v_2\). Hence we conclude \([v_1, v_2] \subset v_2\). The \([v_1, v_3] \subset v_3\) case follows similarly.

These relationships can be used to separate the normal equations in \(\text{su}(8)\). Given \(S \in v_1\), \(D \in v_2\), \(T \in v_3\), it is clear that their derivatives are confined to the same subspaces \(S \in v_1\), \(D \in v_2\), \(T \in v_3\). By using the previous commutation relations, the normal equations in \(\text{su}(8)\) must separate into the form presented in the main body of the text.

6.1.2 Commutation relations in \(\text{su}(16)\)

The basis for \(\text{su}(16)\) is spanned by matrices of the form \(i\sigma_i^1, i\sigma_i^2 \sigma_j^2, i\sigma_i^2 \sigma_j^2 \sigma_k^2\) and \(i\sigma_i^2 \sigma_j^2 \sigma_k^2 \sigma_k^1\). These form bases for the four orthogonal subspaces respectively denoted
\( \mathfrak{v}_1, \mathfrak{v}_2, \mathfrak{v}_3 \) and \( \mathfrak{v}_4 \). In the normal geodesic equations, \( S \in \mathfrak{v}_1, D \in \mathfrak{v}_2, T \in \mathfrak{v}_3 \) and \( F \in \mathfrak{v}_4 \). Examining the commutation relationships of these subspaces can help separate the normal equations. First for basis elements in \( \mathfrak{v}_4 \) and \( \mathfrak{v}_2 \).

\[
[i\sigma_r \otimes \sigma_s \otimes \sigma_t \otimes \sigma_v, i\sigma_j \otimes \sigma_k \otimes I \otimes I]
\]

\[
= -\left( (\varepsilon_{rjm}i\sigma_m + \delta_{rj}I) \otimes (\varepsilon_{skn}i\sigma_n + \delta_{sk}I) \right.
\]

\[
- (-\varepsilon_{rjm}i\sigma_m + \delta_{rj}I) \otimes (-\varepsilon_{skn}i\sigma_n + \delta_{sk}I) \otimes \sigma_t \otimes \sigma_v
\]

\[
= -\left( 2\varepsilon_{rjm}i\delta_{sk}i\sigma_m \otimes I + 2\varepsilon_{skn}\delta_{rj}I \otimes I \right) \otimes \sigma_t \otimes I,
\]

This is contained in \( \mathfrak{v}_3 \), and will contribute to \( \text{proj}_{\mathfrak{v}_3} \) terms. Next there are two possible outcomes for \( \mathfrak{v}_2 \) and \( \mathfrak{v}_4 \). Commutators. First there can be two Pauli matrices overlapping

\[
[i\sigma_r \otimes \sigma_s \otimes \sigma_t \otimes I, i\sigma_j \otimes \sigma_k \otimes I \otimes I]
\]

\[
= -\left( (\varepsilon_{rjm}i\sigma_m + \delta_{rj}I) \otimes (\varepsilon_{skn}i\sigma_n + \delta_{sk}I) \right.
\]

\[
- (-\varepsilon_{rjm}i\sigma_m + \delta_{rj}I) \otimes (-\varepsilon_{skn}i\sigma_n + \delta_{sk}I) \otimes \sigma_t \otimes I
\]

\[
= -\left( 2\varepsilon_{rjm}i\delta_{sk}i\sigma_m \otimes I + 2\varepsilon_{skn}\delta_{rj}I \otimes I \right) \otimes \sigma_t \otimes I,
\]

which is in \( \mathfrak{v}_2 \). So this will only be nonzero under \( \text{proj}_{\mathfrak{v}_2} \). Alternatively if there is only a single Pauli matrix overlapping,

\[
[i\sigma_r \otimes \sigma_s \otimes \sigma_t \otimes I, i\sigma_j \otimes I \otimes I \otimes \sigma_k]
\]

\[
= -[\sigma_r, \sigma_j] \otimes \sigma_s \otimes \sigma_t \otimes \sigma_k
\]

\[
= -2i\varepsilon_{rjm}i\sigma_m \otimes \sigma_s \otimes \sigma_t \otimes \sigma_k,
\]

which will only be nonzero under \( \text{proj}_{\mathfrak{v}_4} \). Summarising these results, using the linearity of the Lie bracket for linear combinations of matrices,

\[
[\mathfrak{v}_1, \mathfrak{v}_j] \subset \mathfrak{v}_j \quad (6.5)
\]

\[
[\mathfrak{v}_2, \mathfrak{v}_4] \subset \mathfrak{v}_3 \quad (6.6)
\]

\[
[\mathfrak{v}_2, \mathfrak{v}_3] \subset \mathfrak{v}_4 + \mathfrak{v}_2 \quad (6.7)
\]

Now consider the Lie bracket of terms like \( \sigma_r^i \sigma_j^k \sigma^l \) and \( \sigma_m^i \sigma^j \sigma_n^k \). The first term in the Lie bracket can be written as

\[
(i\varepsilon_{rj}i\sigma_\theta + \delta_{rj}I) \otimes (i\varepsilon_{sk}i\sigma_\phi + \delta_{sk}I) \otimes (i\varepsilon_{tm}i\sigma_\mu + \delta_{tm}I) \otimes \sigma_n.
\]
6.1. Commutation Relations

Note this will be at least contained in $v_2 + v_3 + v_4$. However all the terms in $v_1$ will be canceled by the second term in the Lie bracket as $\delta$ is symmetric in the indices. Components in $v_3$ will have coefficients containing two Levi-Cevita symbols. However because of the Levi-Cevita symbol, if two indices are swapped in both Levi-Cevita symbols, the sign remains unchanged. Hence terms in $v_3$ will vanish from the Lie bracket. Terms in $v_2 + v_4$ however will contain three or one Levi-Cevita symbols. Swapping one index in all of them, results in a change of the sign, and will not vanish from the Lie bracket. Therefore

$$[v_3, v_4] \subset v_2 + v_4. \quad (6.8)$$

The two body couplings seen in SU(16) naturally extend to higher dimensional $su(2^n)$. Commutators of a two body with a $k$ body produce terms in $k+1$ and $k-1$ directions.

$$[v_k, v_2] \subset v_{k-1} + v_{k+1}.$$  

This is because products of the form $\ldots \sigma_i \sigma_j \otimes \sigma_k \sigma_l \ldots$ in the Lie bracket will leave a term like $\ldots I \otimes \sigma_s \ldots$, reducing the order. If a two body and a $k$ body only has a single overlapping product of Pauli matrices, this will increase the order as seen previously.

Also in SU($2^n$) the single body terms preserve the body order

$$[v_k, v_1] \subset v_k.$$  

The normal equations in SU($2^n$) become highly coupled, as the Lie brackets do not neatly belong to one of the different subspaces. Hence this does not give enough information to solve the equations for the normal geodesics in terms of the different subspaces.

6.1.3 Adjoint action in SU(8)

In SU(8), $e^{Sot} T_0 e^{-Sot} \in g/\Delta$. This follows from (6.1.1) as $[v_4, v_3] \subset v_3 = g/\Delta$, and the well known property $e^X Y e^{-X} = Y + [X, Y] + \frac{1}{2} [X, [X, Y]] + \frac{1}{6} [X, [X, [X, Y]]] + \ldots$. Clearly nested Lie brackets of the form $[v_1, \ldots [v_4, v_3]]$ will always be contained in $v_3$, and as $v_3$ is closed under addition, this gives the desired result. This same argument applies when the adjoint action of elements in $\mathfrak{g} = v_1 + v_2$ is taken by $e^{Sot}$. 

Appendix B

6.2 Gradients for discrete geodesics and cubics

6.2.1 Trace error gradient

The gradient of the error function err,
\[
\frac{\partial}{\partial \xi_i} \text{err}(Y, \xi) = -\frac{1}{2n} \frac{\partial}{\partial \xi_j} \text{tr}(U^\dagger e^{\sum_{j=1}^m \xi_j^j} \ldots e^{\sum_{j=1}^m \xi_j^N})
\]

6.2.2 Discrete cubics gradient

To apply Newton’s method we need the gradient of the discrete cubic functional.
\[
S_A[\xi] = \sum_{j=1}^N \frac{\hbar}{2} (\langle \dot{u}(t_{j+1}), \dot{u}(t_{j+1}) \rangle + \langle \dot{u}(t_j), \dot{u}(t_j) \rangle)
\]

We will perform the minimization with respect to the scalar coefficients, \(c_i^j\) in \(u(t_j) = c_i^j \tau_i\), and where \(\dot{u}\) has been approximated with finite difference.
\[
\frac{\partial S}{\partial c_i^j} = \frac{\partial}{\partial c_i^j} \left( \sum_{j=1}^N \frac{1}{2} \left( ||c_i^{j+1} \tau_i - c_i^{k} \tau_i||^2 + ||c_i^{k} \tau_i - c_i^{k-1} \tau_i||^2 \right) \right)
\]

Clearly the only non zero terms which will remain after the partial derivative will be when \(j = k - 2, k - 1, k\).
\[
\frac{\partial S}{\partial c_i^k} = \frac{\partial}{\partial c_i^k} \left( ||c_i^{j+1} \tau_i - c_i^{k} \tau_i||^2 + ||c_i^{k} \tau_i - c_i^{k-1} \tau_i||^2 \right),
\]

where the norm is calculated by (repeated indices are summed)
\[
||c_i^{j+1} \tau_i - c_i^{k} \tau_i||^2 = \text{tr} \left( (c_i^{j+1} \tau_i - c_i^{j} \tau_i) (c_i^{j+1} \tau_i - c_i^{j} \tau_i)^\dagger \right)
\]
\[
= \text{tr} \left( -c_i^{j+1} \tau_i^2 + c_i^{j+1} c_i^j \tau_i \tau_i + c_i^{j+1} c_i^{j+1} \tau_i \tau_i - (c_i^j \tau_i)^2 \right)
\]

In the normalised Pauli basis \(\text{tr}(\tau_i \tau_j) = -\delta_{ij}\). Taking the trace,
\[
||c_i^{j+1} \tau_i - c_i^{k} \tau_i||^2 = c_i^{j+1} + 2c_i^{j+1} c_i^j + c_i^j c_i^j
\]

Now it is easy to calculate the partial derivative,
\[
\frac{\partial S_A}{\partial c_i^k} = \frac{\partial}{\partial c_i^k} \left( c_i^{k+1} c_i^k - 2c_i^{k+1} c_i^k + 2c_i^k c_i^k - 2c_i^k c_i^{k-1} + c_i^{k-1} c_i^{k-1} \right)
\]
\[
= -2c_i^{k+1} + 4c_i^k - 2c_i^{k-1}
\]
If \(k = 1\), we have no \(k - 1\) terms. Remember there are \(N + 1\), \(c_i^k\) vectors.