Imperial College London

BACHELOR OF SCIENCE

FINAL REPORT

Lie Algebras and Markovian Arrival Processes

Author:
Ashmi Mehta

Supervisor:
Prof. Peter Harrison

Second-Supervisor:
Dr. Giuliano Casale

Abstract

Lie algebra is a mathematical concept that involves various algebraic theories, but it has some very strong links with other areas of mathematics. Lie algebra and matrix Lie groups can be have an extensive use in the real world today, especially when solving problems related to Markovian arrival processes (MAPs). From its name, we can detect that MAPs have a connection with Markov chains and the Markov property. In this report, we will give a formal definition of MAPs and how Markov chains relate to it. MAPs are widely used in today's world as we can not only apply this model to scientific cases but also in daily-life situations. For instance, when jobs arrive in a queue (say on a network) then this can be modeled using MAPs or another example would be modeling the number of people arriving in a queue.

This report will introduce the different matrix Lie groups and its corresponding Lie algebra, which will form the basis of this project. We will then use two of these groups, SL $(2, \mathbb{R})$ and SU (2), and model them using MAPs, i.e. generate the rate matrices using Lie algebra and its key properties. Hence, in that section we will break down the steps on how to solve such a problem as well as include a further section where we apply these rate matrices to define a new concept. We will also give a verification of the solutions using MATLAB. The advantage of applying the commutator relation in order to solve the problem is to obtain a method that one could solve by hand (for simple cases) rather then relying only on computers.

In reality we know that not everything is perfect, hence in the case of MAPs we may not always have all the information. Therefore, we may need to estimate the unobserved data using the estimation-maximisation(EM) algorithm. A small discussion of the advantages and disadvantages of a second approach that is based on moments is also included. Finally, we will explore some applications of the EM algorithm as well as discover where and how we can apply it.

A further investigation will be carried out to identify if we can find the rate matrices for matrix Lie groups that have higher dimensions, such as 3 x 3. However, this is mostly left as future work. Other extensions also include, modeling MAPs on other matrix Lie groups other than the two already mentioned; and comparing outcomes of the EM algorithm and the moment based approach when applied to real data.

Acknowledgement

I would like to thank my supervisor, Prof. Peter Harrison for his endless support, inspiration and enthusiasm throughout this project. He always remained patient and confident in my ability to deliver results.

I would also like to thank Dr. Giuliano Casale, for his advice and constructive feedback during the interim report submission.

I would also like to take this opportunity to thank my personal tutor, Dr. Alessandra Russo, who has guided me throughout my three years at Imperial College.

Finally, I would like to thank my friends and my family for their encouragement and blessings throughout my three years at Imperial College.

Contents

1 Introduction

In this chapter, we will look at the history of Lie Groups and its unique features. We will also highlight the problems that can be solved with matrix Lie groups and Lie algebra while also discussing some of the applications of this theory. This chapter is divided into four sections, where Section 1.1 will focus on the motivation of the project and a short history of the topic. Section 1.2 outlines the applications of the theory and what the objective is. Finally, section 1.3 will provide a summary of how this report is organised.

1.1 Motivation

There are a lot of theorems and proofs involved in mathematics and the beauty about this subject is its applications in all different fields ranging from physics to economics. Furthermore, over the years we can find new ways of applying these century old mathematical concepts and see the world around us advance. Matrix Lie groups and Lie algebra have been researched extensively, but understanding its role in matrix exponentiation with applications to continuous time Markov chains is very interesting and useful in reality as it has many applications.

Lie Groups are named after Sophius Lie, who laid the foundations of the continuous transformation group theory. As Lie groups are smooth manifolds they can be studied using differential calculus and the key idea of this theory is to replace the global object (the group) with a *local/linearised* version of the group. Lie groups were initially introduced to help solve and simplify ordinary as well as partial differential equations.[1] The most beautiful and useful feature about Lie groups is that it can be applied in two of broadest mathematical divisions, algebra and geometry. Their algebraic properties are derived from the group axioms, while their geometric properties are identified from group operations with points in a topological space.

The study of matrix Lie groups can be facilitated by linearising the group in the neighbourhood of its identity and this results in a structure called a **Lie algebra**. The Lie algebra retains most of the properties of the original matrix Lie group, but not necessarily all. In addition, the majority of the Lie group properties can be recovered by the inverse of the linearisation operation, which is known as *exponential mapping*. Since a Lie algebra is a linear vector space, we can study it by using all the standard tools defined for linear vector spaces.[1] In particular, we can define the inner products and make standard choices of the basis vectors. These tools such as the inner product, can then be extended over the entire group manifold using the group multiplication operation.

This area of mathematics has a clear application within the computer science field i.e. the theory of Lie algebra and Markovian arrival processes will be used to show how we can solve queueing system problems. This project will allow us to further explore Markovian arrival processes (MAP) and understand the differences when various underlying distributions are used. The main process that this project will focus on is the Poisson process/distribution as well as the continuous-time Markov process and we will discuss examples that highlight the link between Lie algebra and MAPs.

1.2 Objectives

This project aims to explore Lie algebra and matrix Lie groups in terms of matrix exponential as well as apply the theory to applications such as continuous-time Markov chains (CTMC). More specifically, we will go in depth into some of the theory and apply Lie algebra and matrix Lie groups to the class of Markovian arrival processes (MAP).

One of the aims at the end of this project is to understand the link between MAPs and Lie algebra. In addition, we would like to discuss how we could use matrix exponentials to simplify some of the equations that appear in examples involving MAPs. We will be working with two different matrix Lie groups and these will be discussed further in the next chapter.

Another aim is to understand the method of solving problems related to MAPs i.e. finding the two rate matrices C and D as these are very important elements of a MAP. Further explanations about the rate matrices and their definitions will follow in later chapters. We will also use MATLAB to generate these matrices and apply them to different situations, such as the estimation-maximisation algorithm.

We will also briefly discuss the estimation and maximisation (EM) algorithm and how we can apply that to MAPs. In other words, we will answer questions such as why should we use this algorithm and in what cases is it useful? We we will also discuss the two different ways of applying the EM algorithm, i.e. either using differential equations or uniformisation.

The final aim is to look into the EM algorithm and apply it to MAPs. We will break down each step of the algorithm and see how to use the equations in order to solve a specific problem.

1.3 Layout of the report

This report is organised as follows:

- Chapter 2 outlines the relevant work in Lie algebra that will form the basis of this project. It focuses on matrix Lie groups, and some of these will be discussed in further detail. In addition, we discuss and prove some of the key theorems and propositions related to Lie algebra.
- Chapter 3 introduces the concept of Markovian arrival processes (MAP). We will introduce the method of solving MAPs as well as provide two examples based on matrix Lie groups. Finally, we will investigate how we could potentially solve higher dimension matrices, namely 3 x 3.
- Chapter 4 is closely linked with the previous chapter as we take those two examples and validate the results in MATLAB.
- Chapter 5 introduces a relatively new topic, the estimation and maximisation algorithm for MAPs. We will give an overview of the algorithm and describe its importance within MAPs.
- Chapter 6 is where we will evaluate our results and check if we have successfully achieved our aims that we outlined at the onset of this report.
- Chapter 7 summarises the contributions of this project and will be a formal conclusion to this report. We will discuss any limitations we encountered along with suggestions for how to develop this project in the future.

2 Background

In this chapter, we will explore in more detail the theory behind matrix Lie groups and Lie algebra. We will also prove some key theorems and propositions as shown in [2] and these will be applied in the project. We will define a matrix Lie group and give many examples that highlight the importance of this mathematical concept. Furthermore, we will understand the link between matrix Lie groups and its respective Lie algebra.

There is emphasis on all the different matrix Lie groups as each one portrays unique properties. However, this project will focus on the special linear group and the special unitary group.

We end this chapter by proving the Baker-Hausdorff-Campbell Lemma, which is important and heavily used in the study of Markovian arrival processes..

2.1 Matrix Lie Groups

In order to understand the other matrix Lie groups in depth, one must be familiar with the concept of **general linear groups**.

Definition 2.1.1.
A general linear group over the real numbers is denoted by:

$$\mathrm{GL}\,(n, \mathbb{R})$$

This is the group of all n x n invertible matrices with real entries.

Definition 2.1.2.
A general linear group over the complex numbers is denoted by:

$$\mathrm{GL}\,(n, \mathbb{C})$$

This is the group of all n x n invertible matrices with complex entries.

We can easily show that the general linear groups are a group under the matrix multiplication operation as:

1. The product of two invertible matrices is an invertible matrix
2. Matrix multiplication is associative
3. The identity of any matrix is the identity of the group
4. An invertible matrix has by definition an inverse

Definition 2.1.3.
Let A_m be a sequence of complex matrices in $M_n(\mathbb{C})$, where $M_n(\mathbb{C})$ is the space of all n x n matrices with complex entries.
Hence, we can say that A_m converges to a matrix A if every corresponding entry in A_m converges to its respective entry in A as $m \to \infty$.
For instance, as $m \to \infty$ then $(A_m)_{kl}$ converges A_{kl} $\quad \forall 1 \leq k, l \leq n$

Now, we can combine Definition 2.1.2. and Definition 2.1.3. and write a new definition for a concept called **matrix Lie groups**.

Definition 2.1.4.
A matrix Lie group is a closed subgroup, say G, of GL $(n, \mathbb{C})$ with the following property:

- If A_m is any sequence of matrices in G, and A_m converges to a matrix A then there can only be two possibilities:
  - $A \in G$ or
  - A is not invertible

[**Note:** G is a closed subset of GL $(n, \mathbb{C})$, but this does not imply that G is closed in $M_n(\mathbb{C})$. Nonetheless, there are many examples of matrix Lie groups which are also closed in $M_n(\mathbb{C})$.]

A simple counterexample, which is not closed and hence is not a matrix Lie group is the set of all invertible, n x n, matrices with every entry being real and rational. We know that such a matrix is a subgroup of GL $(n, \mathbb{C})$, however it is not closed as we can show that a sequence of invertible matrices with rational entries can converge to an invertible matrix with some irrational entries.

2.2 Examples Of Matrix Lie Groups

Example 1.

The general linear group: GL $(n, \mathbb{R})$ and GL $(n, \mathbb{C})$

The most straightforward example is the general linear groups, GL $(n, \mathbb{R})$ and GL $(n, \mathbb{C})$, which are also matrix Lie groups and can be proved just by using the definition.

[**Note:** From group theory we know that a group is also a subgroup of itself.]

Hence, GL $(n, \mathbb{R})$, is by definition a subgroup of itself. So, if A_m is a sequence of matrices in GL $(n, \mathbb{R})$ and A_m converges to A then either $A \in$ GL $(n, \mathbb{R})$ or A is not invertible.

In addition, GL $(n, \mathbb{C})$, is also subgroup of itself. Therefore, just like the previous case if A_m is a sequence of matrices in GL $(n, \mathbb{C})$ and A_m converges to A then the entries of A are real, hence either $A \in$ GL $(n, \mathbb{C})$ or A is not invertible.

Example 2.

The special linear group: SL $(n, \mathbb{R})$ and SL $(n, \mathbb{C})$

The special linear group is a group of n x n matrices over $\mathbb{R}$ or $\mathbb{C}$ with real or complex entries and each matrix has a determinant equal to one.

Clearly, **SL** $(n, \mathbb{R})$ and **SL** $(n, \mathbb{C})$ are subgroups of the general linear groups **GL** $(n, \mathbb{R})$ and **GL** $(n, \mathbb{C})$, respectively. Moreover, if A_m is a sequence of matrices, all with determinant one, and if A_m converges to A then A will also have determinant one. This is true because the determinant of the product of two square matrices say X and Y is equal to the product of their individual determinants i.e.

$$det(XY) = det(X)\, det(Y)$$

and this can be extended to a finite number of matrices. So, in this case as the determinant of each matrix in the sequence of A_m is one then as A_m converges to A it implies that A also has determinant one.

Example 3.

The orthogonal group: O (n) and the special orthogonal group: SO (n)

There are a few different ways to define an orthogonal matrix and each definition is important in order to show that O (n) is a matrix Lie group:

Definition 2.2.1.
A n x n matrix is orthogonal if its column vectors are orthonormal i.e. if A is an orthogonal matrix then the columns of A would be orthonormal and defined as:

$$\begin{aligned} \sum_{i=1}^{n} A_{ij}A_{ik} &= \delta_{jk} \\ 1 \leq \quad j,k \leq \quad n \end{aligned}$$

where δ_{jk} is defined as the *Kronecker delta:*
if $j = k$ then

$$\delta_{jk} = 1$$

if $j \neq k$ then

$$\delta_{jk} = 0$$

Proposition 2.2.2.
A is an orthogonal, n x n, matrix if it preserves the inner product. The inner product is defined as:

$$\begin{aligned} \langle x, y \rangle &= x^T y \\ &= \sum_{i=1}^{n} x_i y_i \end{aligned}$$

where $x, y \in \mathbb{R}^n$ and x^T is the transpose of x.
So, A preserves the inner product if:

$$\langle Ax, Ay \rangle = \langle x, y \rangle$$

Proposition 2.2.3.
A matrix, A, is orthogonal if

$$\begin{aligned} A^T A &= AA^T \\ &= I \end{aligned}$$

In other words, $A^T = A^{-1}$ where A^T is the transpose of A i.e.

$$A_{ij} = \left(A^T\right)_{ji}$$

and the transpose of A is equal to the inverse of A.

So, from the above definition we can easily prove the resulting two propositions.

So, if A is an orthogonal matrix then using proposition 2.2.3. we know that $AA^T = A^T A = I$ and from the definitions of determinants we know that $det\,(A) = det\left(A^T\right)$, so now we can write:

$$\begin{aligned} det\left(AA^T\right) &= det\,(A)\,det\left(A^T\right) \\ &= \left(det\,(A)\right)^2 \\ &= det\,(I) \\ &= 1 \end{aligned}$$

Therefore, $det\,(A) = \pm 1$ for all orthogonal matrices, A.

This clearly indicates that every orthogonal matrix, A, is invertible. From this we can also prove that an inverse of an orthogonal matrix is also orthogonal. This can be done using proposition 2.2.2:

$$\begin{aligned} \left\langle A^{-1}x, A^{-1}y\right\rangle &= \left\langle A\left(A^{-1}x\right), A\left(A^{-1}y\right)\right\rangle \\ &= \langle x, y\rangle \end{aligned}$$

In addition, the product of an orthogonal matrix, A, and another orthogonal matrix, B, will also result in an orthogonal matrix i.e. AB is orthogonal because both A and B preserve the inner product definition, so AB must also preserve it. Hence, this is a set of orthogonal matrices that forms into a group.

So, the set of all n x n orthogonal matrices can be defined as the orthogonal group, O (n), which is a subgroup of GL $(n, \mathbb{C})$. The limit of a sequence of orthogonal matrices is also orthogonal, because under limits the relation $A^T A = I$ is preserved. Therefore, by definition, O (n) is a matrix Lie group.

SO (n) is called the special orthogonal group, as we define it to be the set of all n x n matrices that have determinant one. This clearly implies that SO (n) is a subgroup of O (n), and as O (n) is the group with matrices that has determinant ± 1 then SO (n) is the set of all matrices with determinant, $+1$.

[**Note:** This also means that number of elements (size of) of SO (n) is exactly half of O (n).]

Hence, as SO (n) is a subgroup of O (n) and O (n) is a subgroup of GL $(n, \mathbb{C})$ this implies that SO (n) is a subgroup of GL $(n, \mathbb{C})$. Furthermore, the limit of a sequence of orthogonal

matrices preserves the property of orthogonality and determinants being equal to one. Therefore, again, by definition this is a matrix Lie group.

Example 4

The unitary group: U (n) and the special unitary group: SU (n)

The complex n x n matrix, A, is unitary if the columns of A are orthonormal i.e.

$$\sum_{i=1}^{n} \overline{A_{ij}} A_{ik} \quad = \quad \delta_{jk}$$

where A_{ik} is the the ik^{th} element of matrix A and this implies $\overline{A_{ij}}$ is the conjugate of the element at the ij^{th} position (i.e. $a + ib$ has the complex conjugate $a - ib$, only negate imaginary parts).

δ_{jk} is the Kronecker delta which is defined earlier.

In addition, A is unitary if it preserves the inner product i.e.

$$\langle x, y \rangle \quad = \quad \langle Ax, Ay \rangle$$

where $x, y \in \mathbb{C}^n$.

However, the inner product for complex vectors in $\mathbb{C}^n$ are defined differently:

$$\langle x, y \rangle \quad = \quad \sum_{k} \overline{x_k} y_k$$

where it is clear that instead of using the transpose of x we are using the complex conjugate of, x.

Moreover, we can also define A to be unitary by using the adjoint of A. The adjoint of A is defined as A^* where A^* is obtained by taking the transpose of A and then taking the complex conjugate of each element in A. So, by this definition we can write:

$$(A^*)_{jk} \quad = \quad \overline{A_{kj}}$$

So, if:

$$\begin{aligned} A^* A \quad &= \quad I \\ \implies A^* \quad &= \quad A^{-1} \end{aligned}$$

and thus A is unitary.

Using the same argument as we did for orthogonal groups, we can say that:

$$\begin{aligned} det\,(A^*) &= det\,(A) \\ \implies det\,(AA^*) &= det\,(A)\,det\,(A^*) \\ &= |det\,(A)|^2 \\ &= det\,(I) \\ &= 1 \end{aligned}$$

and this is true for all unitary matrices, A.

Just like for orthogonal matrices, this shows that all unitary matrices are invertible and hence the set of unitary matrices form a group.

Now we can easily apply the definition of matrix Lie groups and prove that $\mathrm{U}\,(n)$ is also a matrix Lie group, this is shown below:

We can write that the set of all n x n unitary matrices is the unitary group, $\mathrm{U}\,(n)$, and is clearly a subgroup of $\mathrm{GL}\,(n, \mathbb{C})$. The limit of any sequence of unitary matrices is also unitary, hence it is a matrix Lie group.

Just as the case for orthogonal groups, there is a special case for unitary groups which is called the special unitary group, $\mathrm{SU}\,(n)$. This group is a subgroup of $\mathrm{U}\,(n)$, as it includes all the matrices that have determinant $+1$, hence $\mathrm{SU}\,(n)$ is also a subgroup of $\mathrm{GL}\,(n, \mathbb{C})$. The limit of the sequence of matrices that have determinant one will converge to a matrix within the group $\mathrm{SU}\,(n)$ as the property unitary and determinants are preserved. Hence, $\mathrm{SU}\,(n)$ is also matrix Lie group.

Example 5.

The complex orthogonal group: $\mathrm{O}\,(n, \mathbb{C})$ and the complex special orthogonal group: $\mathrm{SO}\,(n, \mathbb{C})$

Define $(\cdot, \cdot)$ as the bilinear form on $\mathbb{C}^n$ such that $(x, y) = \sum_k x_k y_k$. The set of all n x n matrices, A, which preserve this form (i.e. $(Ax, Ay) = (x, y)\ \forall x, y \in \mathbb{C}^n$) is called the complex orthogonal group, $\mathrm{O}\,(n, \mathbb{C})$ and it is clearly a subgroup of $\mathrm{GL}\,(n, \mathbb{C})$.

We can see that this group is similar to $\mathrm{O}\,(n)$, except we have complex entries, so we can use the same argument and say that any n x n complex matrix, A, is in $\mathrm{O}\,(n, \mathbb{C})$ if and only if $A^T A = I$ and if the $det\,(A) = \pm 1$, $\forall A \in \mathrm{O}\,(n, \mathbb{C})$. Hence, the complex orthogonal group is a matrix Lie group.

Similarly, $\mathrm{SO}\,(n, \mathbb{C})$ is defined as the subgroup of $\mathrm{O}\,(n, \mathbb{C})$ (and this further implies that it is a subgroup of $\mathrm{GL}\,(n, \mathbb{C})$) with all its elements, A, having determinant $+1$. Again using a similar argument as for $\mathrm{SO}\,(n)$, we know that $\mathrm{SO}\,(n, \mathbb{C})$ is also a matrix Lie group.

A simple example of a $\mathrm{SO}\,(2, \mathbb{C})$ are matrices of the form:

$$A = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

Any matrix of the form A is orthogonal, which is a standard definition and the determinant of A is also going to always be one as it is the following trigonometric identity:

$$\cos^2(\theta) + \sin^2(\theta) = 1$$

Example 6.

The generalised orthogonal group: O (n,k)

Let n and k be positive integers, such that we are working in $\mathbb{R}^{n+k}$. Define a symmetric bilinear form $[\cdot,\cdot]_{n,k}$ on $\mathbb{R}^{n+k}$ by the formula:

$$[x,y]_{n,k} \quad = \quad x_1y_1 + \cdots + x_ny_n - x_{n+1}y_{n+1} - \cdots - x_{n+k}y_{n+k}$$

The set of $(n+k)$ x $(n+k)$ real matrices, A, which preserve this form:

$$[Ax,Ay]_{n,k} \quad = \quad [x,y]_{n,k}$$

for all $x,y \in \mathbb{R}^{n+k}$ are called the generalised orthogonal group, O (n,k). It is a subgroup of GL $(n+k,\mathbb{R})$ and hence a matrix Lie group.

Example 7.

The symplectic groups: Sp $(n,\mathbb{R})$, Sp $(n,\mathbb{C})$, Sp (n)

Sp $(n,\mathbb{R})$, Sp $(n,\mathbb{C})$, Sp (n) involve skew-symmetric bilinear forms rather than symmetric bilinear forms.

Define the skew-symmetric bilinear form, B, on $\mathbb{R}^{2n}$ as:

$$B[x,y] \quad = \quad \sum_{k=1}^{n} x_ky_{n+k} - x_{n+k}y_k$$

The set of all $2n$ x $2n$ matrices, A, which preserve B such that:

$$B[Ax,Ay] \quad = \quad B[x,y]$$

$\forall x,y \in \mathbb{R}^{2n}$ is called the **real symplectic group**, Sp $(n,\mathbb{R})$, and it is a subgroup of GL $(2n,\mathbb{R})$ and hence it is also a matrix Lie group.

If J is a $2n$ x $2n$ matrix such that:

$$J \quad = \quad \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix}$$

then $B[x,y] = \langle x, Jy \rangle$ and a $2n$ x $2n$ real matrix, A, is in Sp $(n,\mathbb{R})$ if and only if $A^TJA = J$. Taking the determinant of this gives $(det\,(A))^2\, det\,(J) = det\,(J)$ as $(det\,(A))^2 = 1$. Hence, $det\,(A) = \pm 1 \qquad \forall A \in$ Sp $(n,\mathbb{R})$. However, $det\,(A) = 1 \qquad \forall A \in$ Sp $(n,\mathbb{R})$ but this is not very obvious.

In addition, we can define a similar bilinear form as above on $\mathbb{C}^{2n}$ which does not involve any complex conjugates. The set of $2n$ x $2n$ complex matrices which preserve this form are called the **complex symplectic group**, Sp $(n,\mathbb{C})$.

A $2n$ x $2n$ complex matrix, A, is in $\mathsf{Sp}(n,\mathbb{C})$ if and only if $A^T J A = J$. Hence, like the real case the $det(A) = \pm 1 \qquad \forall A \in \mathsf{Sp}(n,\mathbb{C})$. However, again we have $det(A) = 1 \qquad \forall A \in \mathsf{Sp}(n,\mathbb{C})$.

The final case is called the **compact symplectic group**, $\mathsf{Sp}(n)$, defined as:

$$\mathsf{Sp}(n) \quad = \quad \mathsf{Sp}(n,\mathbb{C}) \cup \mathsf{U}(2n)$$

Example 8.

Heisenberg group: H

The set of all 3 x 3 real matrices, A, of the form:

$$A \quad = \quad \begin{pmatrix} 1 & a & b \\ 0 & 1 & c \\ 0 & 0 & 1 \end{pmatrix} \tag{1}$$

where $a, b, c \in \mathbb{R}$. This set of matrices is called the **Heisenberg group**, H. The product of two matrices of the form 1 is also of the form 1 and clearly if $a = b = c = 0$ then we have the identity, I.

Furthermore, if A is of the form as in equation 1then it is clear that A^{-1} is:

$$A^{-1} \quad = \quad \begin{pmatrix} 1 & -a & ac-b \\ 0 & 1 & -c \\ 0 & 0 & 1 \end{pmatrix}$$

Thus, H is a subgroup of $\mathsf{GL}(3,\mathbb{R})$ and clearly the limit of such matrices remains in the form of equation 1 and this implies that H is a matrix Lie group.

Example 9.

Some groups are <u>not</u> naturally a group of matrices, these include:

- $\mathbb{R}^*$ is a group of non-zero real numbers and under multiplication it is isomorphic to $\mathsf{GL}(1,\mathbb{R})$.
- $\mathbb{C}^*$ is a group of non-zero complex numbers and under multiplication it is isomorphic to $\mathsf{GL}(1,\mathbb{C})$.
- S^1 is defined as a group of complex numbers with absolute value one and that is isomorphic to $U(1)$.

Since we have already proved that the general cases of $\mathsf{GL}(1,\mathbb{R})$, $\mathsf{GL}(1,\mathbb{C})$,$\mathsf{U}(1)$ are matrix Lie groups, we can say that $\mathbb{R}^*, \mathbb{C}^*, S^1$ are also defined as matrix Lie groups as they are isomorphic to $\mathsf{GL}(1,\mathbb{R})$, $\mathsf{GL}(1,\mathbb{C})$, $\mathsf{U}(1)$, respectively.

Another example is taking the group $\mathbb{R}$ under addition, and this is isomorphic to $\mathsf{GL}(1,\mathbb{R})^+$ i.e all 1 x 1 real matrices with positive determinants. This isomorphism is achieved using the map $x \to e^x$.

On the other hand, if we have the group $\mathbb{R}^n$ under vector addition then it is isomorphic to the group of diagonal real matrices with positive entries, via a map:

$$(x_1, \ldots, x_n) \quad \rightarrow \quad \begin{pmatrix} e^{x_1} & & 0 \\ & \ddots & \\ 0 & & e^{x_n} \end{pmatrix}$$

Example 10.

The **Euclidean group**: $\mathsf{E}(n)$ and **Poincare group**: $\mathsf{P}(n,1)$

Definition 2.2.4.
The Euclidean group, $\mathsf{E}(n)$, is the group of all one-to-one distance-preserving maps of $\mathbb{R}^n$ to itself, $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$
where $d(f(x), f(y)) = d(x,y) \quad \forall x, y \in \mathbb{R}^n$ and d is defined as the usual distance formula $d(x,y) = |x - y|$.

In this example, f does not have to be linear hence it is clear that the orthogonal group, $O(n)$, is a subgroup of $E(n)$, since $O(n)$ is the group of *all linear* distance-preserving maps of $\mathbb{R}^n$ to itself.

In addition, for $x \in \mathbb{R}^n$, we can define the translation by $T_x(y) = x + y$ and the set of these translations is also a subgroup of $\mathsf{E}(n)$.

The Poincare group, $\mathsf{P}(1,n)$, is isomorphic to the group of $(n+2)$ x $(n+2)$ matrices with the form, $M = \begin{pmatrix} & & x_1 \\ & A & \vdots \\ & & x_{n+1} \\ 0 & \ldots & 0 \quad 1 \end{pmatrix}$, with $A \in \mathsf{O}(n,1)$. Hence, the set of matrices of the form, M, is a matrix Lie group.

2.3 Compactness

Definition 2.3.1.
A matrix Lie group is **compact** if the following two conditions are satisfied:

1. If A_m is any sequence of matrices in G and A_m converges to a matrix, A, then A is in G.
2. There exists a constant, C, such that $\forall A \in G$, $|A_{ij}| \leq C \quad \forall i, j$ such that $1 \leq i, j \leq n$.

The set $M_n(\mathbb{C})$ of all n x n complex matrices can be thought of as $\mathbb{C}^{n^2}$. So, the above definition says that G is compact if it is a *closed and bounded* subset of $\mathbb{C}^{n^2}$. Heine-Borel theorem states that a subset of $\mathbb{C}^{n^2}$ is compact if and only if it is *closed and bounded.[3]*

All the examples from section 2 except GL $(n, \mathbb{R})$ and GL $(n, \mathbb{C})$ satisfy the first condition of definition 2.3.1, hence the second property (boundedness) is the most important to check.

Examples of Compact Groups.

The orthogonal group, O (n), is compact because it satisfies the first condition as the limit of orthogonal matrices is orthogonal. The second condition is satisfied because if a matrix, A, is orthogonal then the column vectors of A have norm one, hence one is the constant C (i.e. $C = 1$) with $|A_{ij}| \leq 1$, $\forall i, j$ such that $1 \leq i, j \leq n$.

Similarly, the special orthogonal group, SO (n) is compact because it satisfies the first condition as the limit of orthogonal matrices is orthogonal and the limit of matrices with determinant one is also determinant one. The second condition is satisfied by the same logic as above.

Further examples include U (n), SU (n), Sp (n) and using a similar argument as above we can show that they are compact too.[4]

Examples of Non-Compact Groups.

The remaining examples of matrix Lie groups that were discussed in section 2.2 are non-compact groups. The groups GL $(n, \mathbb{R})$ and GL $(n, \mathbb{C})$ are non-compact because they violate property one as the limit of invertible matrices can be non-invertible.

The remaining examples all violate condition two, hence they are all non-compact.

2.4 Connectedness

Definition 2.4.1.
A matrix Lie group, G, is **connected** if there exists a continuous path between any two matrices, $A, B \in G$ where the continuous path is defined as $A(t)$, $a \leq t \leq b$ lying in G such that $A(a) = A$ and $A(b) = B$.

In general, *connected* is also known as *path-connected.* **A matrix Lie group is connected if and only if it is path-connected.**

A matrix Lie group is *not connected* if it can be uniquely decomposed as a union of *components,* such that two elements of the same component can be joined by a continuous path, while two elements of different components cannot.

Proposition 2.4.2.
If G is a matrix Lie group, then the component of G containing the identity is a subgroup of G.

Proof.

A and B are, two elements, both in the component containing the identity such that there exists continuous paths $A(t)$ and $B(t)$ with $A(0) = B(0) = I$, $A(1) = A$, $B(1) = B$.

So, $A(t)B(t)$ is a continuous path, which starts at I and ends at AB. Hence, the product of two elements of the identity component is also in the identity component.

$A(t)^{-1}$ is a continuous path starting at I and ends at A^{-1}, hence the inverse of any element of the identity component is again in the identity component.

[**Note:** matrix multiplication and matrix inversion are continuous on GL $(n, \mathbb{C})$ it follows that if $A(t)$ and $B(t)$ are continuous, then so are $A(t)B(t)$ and $A(t)^{-1}$.]

QED

Proposition 2.4.3.
The group GL $(n, \mathbb{C})$ is connected $\forall n \geq 1$.

Proof.

Case 1: $n = 1$

A 1 x 1 invertible complex matrix, A, is of the form $A = [\lambda]$ with $\lambda \in C^*$ (i.e. the set of non-zero complex numbers). Given any two non-zero complex numbers there exists a continuous path, which connects them and does not intersect at zero.

Case 2: $n \geq 2$

In order to prove this case, we need to show that any element of GL $(n, \mathbb{C})$ can be connected to the identity by a continuous path lying in GL $(n, \mathbb{C})$. Then by taking any two elements, A and B, of GL $(n, \mathbb{C})$ they can be connected by a path going from A to the identity and then from the identity to B.

Thus, to complete the proof we need to use the algebraic property which states that every matrix is similar to an upper triangular matrix, i.e. given any n x n complex matrix, A, there exists an invertible n x n complex matrix, C, such that

$$A = CBC^{-1}$$

where B is an upper triangular matrix:

$$B = \begin{pmatrix} \lambda_1 & & * \\ & \ddots & \\ 0 & & \lambda_n \end{pmatrix}$$

Assume that A is invertible, then all the λ_i's must be non-zero, since from the properties of upper triangular matrices we know that $det(A) = det(B) = \lambda_1 \ldots \lambda_n$.

Now, let $B(t)$ be obtained by multiplying $(1 - t)$ to the elements that are above the diagonal in B, for $0 \leq t \leq 1$ and let $A(t) = CB(t)C^{-1}$.

Then, $A(t)$ is a continuous path which starts at A and ends at CDC^{-1}, where D is the diagonal matrix:

$$D = \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{pmatrix}$$

This path lies in GL $(n, \mathbb{C})$ as $det(A(t)) = \lambda_1 \ldots \lambda_n = det(A) \qquad \forall t$

Similar to case 1, we can define $\lambda_i(t)$ which connects each λ_i to 1 in C^* as t goes from 1 to 2.

Define $A(t)$ on the interval $1 \leq t \leq 2$ by:

$$A(t) = C \begin{pmatrix} \lambda_1(t) & & 0 \\ & \ddots & \\ 0 & & \lambda_n(t) \end{pmatrix} C^{-1}$$

This is a continuous path, which starts at CDC^{-1} when $t = 1$ and ends at $I\left(= CIC^{-1}\right)$ when $t = 2$. Since, the $\lambda_k(t)$'s are always non-zero, $A(t)$ lies in GL $(n, \mathbb{C})$.

Hence, it is clear that every matrix A in GL $(n, \mathbb{C})$ can be connected to the identity by a continuous path lying in GL $(n, \mathbb{C})$.

QED

Proposition 2.4.4.
The group GL $(n, \mathbb{R})$ is not connected, but has two components GL $(n, \mathbb{R})^+$ (n x n real matrices with positive determinant) and GL $(n, \mathbb{R})^-$ (n x n real matrices with negative determinant).

Proof.

GL $(n, \mathbb{R})$ cannot be connected because if $\det(A) > 0$ and $\det(B) < 0$ then any continuous path connecting A to B would have to include a matrix that has determinant zero which is clearly outside the group, GL $(n, \mathbb{R})$.

QED

There are ways of proving if other matrix Lie groups are connected or not, and they are straightforward to solve provided you use the definition and the key conditions related to each group.

Below is a table listing the different matrix Lie groups and indicating if the group is connected and how many components it has:

Groups	Connected?	Components
GL $(n, \mathbb{R})$	No	2
GL $(n, \mathbb{C})$	Yes	1
SL $(n, \mathbb{R})$	Yes	1
SL $(n, \mathbb{C})$	Yes	1
O (n)	No	2
SO (n)	Yes	1
U (n)	Yes	1
SU (n)	Yes	1
O $(n, 1)$	No	4
SO $(n, 1)$	No	2
Heisenberg	Yes	1
E (n)	No	2
P $(n, 1)$	No	4

2.5 Simple Connectedness

Definition 2.5.1.
A matrix Lie group is **simply connected** if it is connected *and* every loop in G can be shrunk continuously to a point in G.
In other words, assume that the group, G, is connected. So, G is simply connected if given any continuous path, $A(t)$, $0 \leq t \leq 1$ lying in G with $A(0) = A(1)$ then there exists a continuous function $A(s,t)$, $0 \leq s,t \leq 1$ that takes the values in G. We can think of $A(t)$ as a loop and $A(s,t)$ as a family of loops, parameterised by the variable, s, which shrinks $A(t)$ to a single point. This has the following properties:
(I) $A(s,0) = A(s,1) \qquad \forall s$
Condition (I) is saying that for each value of the parameter, s, we have a loop.
(II) $A(0,t) = A(t) \qquad \forall t$
Condition (II) is saying that if $s = 0$, then the loop is the specific loop, $A(t)$.
(III)$A(1,t) = A(1,0) \qquad \forall t$
Condition (III) is saying that if $s = 1$, then the loop is a point.

2.6 Homomorphisms And Isomorphisms

Definition 2.6.1.
Let G and H be matrix Lie groups. A map Φ from G to H is a **Lie group homomorphism** if the following conditions hold:
(I) Φ is a group homomorphism
(II) Φ is continuous
Furthermore, if Φ is a one-to-one relation and the inverse map, Φ^{-1}, is continuous then Φ is **Lie group isomorphism**.

[**Note:** it is very difficult to give an example of a group homomorphism between two matrix Lie groups which is *not* continuous, so this condition is just stated for completeness.]

A simple matrix Lie group homomorphism is: $\Phi : \mathbb{R} \to \mathrm{SO}(2)$ i.e.

$$\Phi(\theta) = \begin{pmatrix} \cos(\theta) & \text{-}\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$

The above map is clearly continuous and using standard trigonometry we can show that it is also a homomorphism.

2.7 The Polar Decomposition

Polar decomposition can be used to prove the connectedness of $\mathrm{SL}(n,\mathbb{R})$ and $\mathrm{SL}(n,\mathbb{C})$ as well as show that $\mathrm{SL}(n,\mathbb{R})$ and $\mathrm{SL}(n,\mathbb{C})$ are the same as $\mathrm{SO}(n)$ and $\mathrm{SU}(n)$, respectively. This decomposition is similar to the decomposition of a complex number, z, where $z = up$ such that $|u| = 1$ and p is real as well as positive.

Definition 2.7.1.
A real symmetric matrix, P i.e $P = P^T$, is defined as **positive** if $\langle x, Px\rangle > 0 \quad \forall$non-zero $x \in \mathbb{R}^n$.
In addition, a symmetric matrix is **positive** if all its eigenvalues are positive.
Given a symmetric matrix, P, there exists an orthogonal matrix, R, such that:

$$P = RDR^{-1}$$

where D is the diagonal matrix with positive diagonal entries of eigenvalues, $\lambda_1, \ldots, \lambda_n$. If we take the orthogonal basis $v_1, \ldots, v_n$ of the matrix, P, then $v_1, \ldots, v_n$ are the columns of R.

2.8 Lie Groups

Definition 2.8.1.
A Lie group is a differentiable manifold, G, which is also a group such that the group product $G \times G \to G$ and the inverse map $g \to g^{-1}$ are differentiable.

The simplest example would be $G = \mathbb{R}^n$ with the product map given by $(x, y) \to x + y$

A more interesting example is the following:

Let

$$\begin{aligned} G &= \mathbb{R} \times \mathbb{R} \times S^1 \\ &= \{(x, y, u) \,|x \in \mathbb{R}, y \in \mathbb{R}, u \in S^1 \subset \mathbb{C}\} \end{aligned}$$

Now, define the group product $G \times G \to G$ by

$$(x_1, y_1, u_1) \cdot (x_2, y_2, u_2) = \left(x_1 + x_2, y_1 + y_2, e^{ix_1y_2}u_1u_2\right)$$

Checking if the operation makes G a group:

Associative:

$$\begin{aligned} [(x_1, y_1, u_1) \cdot (x_2, y_2, u_2)] \cdot (x_3, y_3, u_3) &= \left[\left(x_1 + x_2, y_1 + y_2, e^{ix_1y_2}u_1u_2\right)\right] \cdot (x_3, y_3, u_3) \\ &= \left(x_1 + x_2 + x_3, y_1 + y_2 + y_3, e^{i(x_1y_2 + x_1y_3 + x_2y_3)}u_1u_2u_3\right) \end{aligned}$$

$$\begin{aligned} (x_1, y_1, u_1) \cdot [(x_2, y_2, u_2) \cdot (x_3, y_3, u_3)] &= (x_1, y_1, u_1) \cdot \left[\left(x_2 + x_3, y_2 + y_3, e^{ix_2y_3}u_2u_3\right)\right] \\ &= \left(x_1 + x_2 + x_3, y_1 + y_2 + y_3, e^{i(x_1y_2 + x_1y_3 + x_2y_3)}u_1u_2u_3\right) \end{aligned}$$

So, the above equations prove associativity.

There also exists an identity element in the group, G i.e. $e = (0, 0, 1)$.

The inverse of each element, (x, y, u), is $\left(-x, -y, e^{ixy}u^{-1}\right)$.

Hence, clearly G is a group. In addition, both the group product and the map that sends each element to its inverse is smooth, and so G is also a Lie group. However, this example shows that not all Lie groups are matrix Lie groups as there is no continuous injective homomorphism of G into any GL $(n, \mathbb{C})$.

2.9 The Matrix Exponential and Computing The Exponential of a Matrix

Definition 2.9.1.
Let X be a real or complex n x n matrix, so define the exponential of X denoted as e^X by the usual power series:

$$e^X = \sum_{m=0}^{\infty} \frac{X^m}{m!}$$

There are three standard ways of exponentiating general matrices and they are described below.

Case 1: X is diagonalisable

Suppose that X is an n x n real or complex matrix and X is diagonalisable over $\mathbb{C}$. This means that there exists an invertible matrix C such that $X = CDC^{-1}$ where D is the diagonal matrix with the eigenvalues of X as the diagonal elements. In other words, D would be a n x n matrix with the following form:

$$D = \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{pmatrix}$$

It is also true that e^D is the diagonal matrix with eigenvalues $e^{\lambda_1} \dots e^{\lambda_n}$ and hence from the theory of differential equations and algebra, it is clear that

$$e^X = Ce^DC^{-1}$$

Thus, if X is diagonalisable then we can compute e^X.

[**Note:** X may be real, however the values in the matrix C and the $\lambda_k's$ may be complex. Nonetheless, e^X must be real as the terms in the power series of e^X are always real.]

Case 2: X is nilpotent

A n x n matrix, X, is defined as nilpotent if $X^m = 0$ for some positive integer m. Clearly, if $X^m = 0$, then $X^l = 0 \qquad \forall l > m$. So, the power series that defines e^X terminates after the first m terms.

For example, if $X = \begin{pmatrix} 0 & a & c \\ 0 & 0 & b \\ 0 & 0 & 0 \end{pmatrix}$ then

$$\begin{aligned} X^2 &= \begin{pmatrix} 0 & 0 & ac \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\ X^3 &= 0 \end{aligned}$$

Hence, $e^X = \begin{pmatrix} 1 & a & b + \frac{1}{2}ac \\ 0 & 1 & c \\ 0 & 0 & 1 \end{pmatrix}$

Case 3: X is arbitrary

A matrix, X, can be neither diagonalisable nor nilpotent, however every X can be written uniquely as a sum of a diagonal and nilpotent matrix i.e. $X = D + N$, where D is a diagonal matrix and N is a nilpotent matrix.

In addition, $DN = ND$ so since D and N commute $e^X = e^{D+N} = e^D e^N$. Hence, from Case 1 and Case 2 we can calculate e^N and e^D.

For example, if $X = \begin{pmatrix} a & b \\ 0 & a \end{pmatrix}$ then

$$X = \underbrace{\begin{pmatrix} a & 0 \\ 0 & a \end{pmatrix}}_{Diagonal} + \underbrace{\begin{pmatrix} 0 & b \\ 0 & 0 \end{pmatrix}}_{Nilpotent}$$

The above two terms commute because the diagonal matrix is a multiple of the identity, hence

$$\begin{aligned} e^X &= \begin{pmatrix} e^a & 0 \\ 0 & e^a \end{pmatrix} \begin{pmatrix} 1 & b \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} e^a & e^a b \\ 0 & e^a \end{pmatrix} \end{aligned}$$

2.10 The Matrix Logarithm and More Properties of the Matrix Exponential

Definition 2.10.1.
The **matrix logarithm** is the inverse function to the matrix exponential. The function for the logarithm of complex numbers can be defined as a power series such that:

$$\log z = \sum_{m=1}^{\infty} (-1)^{m+1} \frac{(z-1)^m}{m}$$

Similarly, for any n x n matrix, A, $\log(A)$ can be defined as:

$$\log A = \sum_{m=1}^{\infty} (-1)^{m+1} \frac{(A-I)^m}{m}$$

when the series converges.

A key theorem that is essential in the study of Lie algebras is the **Lie Product Formula**.

Theorem 2.10.2. (Lie Product Formula)
Let X and Y be n x n complex matrices then

$$e^{X+Y} = \lim_{m\to\infty} \left(e^{\frac{X}{m}} e^{\frac{Y}{m}}\right)^m$$

Proof.

Writing out the power series for $e^{\frac{X}{m}}$ and $e^{\frac{Y}{m}}$ i.e.

$$\begin{aligned} e^{\frac{X}{m}} &= I + \frac{X}{m} + \frac{\left(\frac{X}{m}\right)^2}{2} + \frac{\left(\frac{X}{m}\right)^3}{3!} + \dots \\ &= I + \frac{X}{m} + \frac{X^2}{2m^2} + \frac{X^3}{6m^3} + \dots \\ e^{\frac{Y}{m}} &= I + \frac{Y}{m} + \frac{\left(\frac{Y}{m}\right)^2}{2} + \frac{\left(\frac{Y}{m}\right)^3}{3!} + \dots \\ &= I + \frac{Y}{m} + \frac{Y^2}{2m^2} + \frac{Y^3}{6m^3} + \dots \end{aligned}$$

So, multiplying the above two power series it is clear that except three terms the rest include higher powers of $\frac{1}{m}$:

$$\begin{aligned} e^{\frac{X}{m}} e^{\frac{Y}{m}} &= \left(I + \frac{X}{m} + \frac{X^2}{2m^2} + \frac{X^3}{6m^3} + \dots\right)\left(I + \frac{Y}{m} + \frac{Y^2}{2m^2} + \frac{Y^3}{6m^3} + \dots\right) \\ &= I + \frac{X}{m} + \frac{Y}{m} + O\left(\frac{1}{m^2}\right) \end{aligned}$$

Hence, it is clear that $e^{\frac{X}{m}}e^{\frac{Y}{m}} \to I$ as $m \to \infty$ and $e^{\frac{X}{m}}e^{\frac{Y}{m}}$ is the domain for the logarithm for large values of m.

A proposition about matrix exponentials states that, $\forall n \times n$ matrices, B, $\log(I+B) = B + O\left(\|B\|^2\right)$ and this proposition can be used to prove this theorem.

$$\log\left(e^{\frac{X}{m}}e^{\frac{Y}{m}}\right) = \log\left(I + \frac{X}{m} + \frac{Y}{m} + O\left(\frac{1}{m^2}\right)\right)$$

The above log equation matches the form of the log equation given in the proposition, so $\log\left(e^{\frac{X}{m}}e^{\frac{Y}{m}}\right)$ can also be written as:

$$\begin{aligned}\log\left(e^{\frac{X}{m}}e^{\frac{Y}{m}}\right) &= \frac{X}{m} + \frac{Y}{m} + O\left(\left\|\frac{X}{m} + \frac{Y}{m} + O\left(\frac{1}{m^2}\right)\right\|^2\right)\\ &= \frac{X}{m} + \frac{Y}{m} + O\left(\frac{1}{m^2}\right)\end{aligned}$$

By exponentiating the above logarithm we get

$$\begin{aligned}e^{\frac{X}{m}}e^{\frac{Y}{m}} &= \exp\left(\frac{X}{m} + \frac{Y}{m} + O\left(\frac{1}{m^2}\right)\right)\\ \left(e^{\frac{X}{m}}e^{\frac{Y}{m}}\right)^m &= \exp\left(X + Y + O\left(\frac{1}{m}\right)\right)\end{aligned}$$

Finally, by the continuity of exponentials, we can say

$$\lim_{m\to\infty}\left(e^{\frac{X}{m}}e^{\frac{Y}{m}}\right)^m = \exp(X+Y)$$

which is the **Lie Product Formula.**

QED

Another key theorem is the following:

Theorem 2.10.3.
For any $n \times n$ complex matrix, X:

$$\det\left(e^X\right) = e^{\text{trace}(X)}$$

Proof.

Earlier, we defined three ways of exponentiating general matrices and to prove 2.10.3. we will break it down into the three different cases.

Case 1: X is diagonalisable

Suppose there is complex invertible matrix, C, such that

$$\begin{aligned} X &= C \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{pmatrix} C^{-1} \\ \implies e^X &= C \begin{pmatrix} e^{\lambda_1} & & 0 \\ & \ddots & \\ 0 & & e^{\lambda_n} \end{pmatrix} C^{-1} \end{aligned}$$

So, the trace of X is

$$\begin{aligned} \text{trace}\,(X) &= \sum \lambda_i \\ \implies e^{\text{trace}(X)} &= e^{\sum \lambda_i} \end{aligned}$$

And the determinant of e^X is

$$\begin{aligned} \det\left(e^X\right) &= \Pi e^{\lambda_i} \\ &= e^{\sum \lambda_i} \end{aligned}$$

Hence, $\text{trace}\,(X) = \det\left(e^X\right)$.

Case 2: X is nilpotent

If X is nilpotent then there exists an invertible matrix C such that

$$X = C \begin{pmatrix} 0 & & * \\ & \ddots & \\ 0 & & 0 \end{pmatrix} C^{-1}$$

In this case, it is evident that e^X will also be upper triangular, with 1's on the diagonal i.e.

$$X = C \begin{pmatrix} 1 & & * \\ & \ddots & \\ 0 & & 1 \end{pmatrix} C^{-1}$$

So, the trace of X is

$$\begin{aligned} \text{trace}\,(X) &= 0 \\ \implies e^{\text{trace}(X)} &= 1 \end{aligned}$$

And the determinant of e^X is the product of the diagonal, which are all 1's, so

$$\det\,(X) = 1$$

Hence, trace $(X) = \det\left(e^X\right)$.

Case 3: X is arbitrary

From the previous section, we showed that any matrix, X, can be written as a sum of two commuting matrices D (diagonalisable matrix over $\mathbb{C}$) and N (nilpotent matrix). Since, $DN = ND$ i.e. they commute we can write

$$e^X = e^D e^N$$

So, from the previous two cases we know that

$$\begin{aligned} \det\left(e^X\right) &= \det\left(e^D e^N\right) \\ &= \det\left(e^D\right)\det\left(e^N\right) \\ &= e^{\text{trace}(D)} e^{\text{trace}(N)} \\ &= e^{\text{trace}(X)} \end{aligned}$$

[**Note:** From the Case 1 and Case 2, we know that trace $(N) = 0$ and trace $(D) =$ trace (X)]

QED

Another important theory, which highlights the relationship between Lie groups and Lie algebra is the *one-parameter subgroups*. One-parameter subgroups are of basic importance in the theory of Lie groups, as every element of the associated Lie algebra defines a homomorphism to the exponential map. And in the case of matrix groups it is defined by the matrix exponential.

Definition 2.10.4.
A function $A : \mathbb{R} \to \text{GL}(n, \mathbb{C})$ is called the **One-Parameter Subgroup** of GL $(n, \mathbb{C})$ if
(I) A is continuous
(II) $A(0) = I$
(III)$A(t+s) = A(t)A(s) \quad \forall t, s \in \mathbb{R}$

In addition, if A is a one-parameter subgroup of GL $(n, \mathbb{C})$ then there exists a unique $n \times n$ complex matrix, X, such that $A(t) = e^{tX}$ (One-Parameter Subgroups Theorem).

2.11 The Lie Algebra of a Matrix Lie Group

Lie algebra is a linear space and can be understood by linear algebra, hence it is simpler to solve than matrix Lie groups. So, many questions about matrix Lie groups can be solved by considering a similar but easier Lie algebra problem.

Definition 2.11.1.
Let G be a matrix Lie group, then the **Lie algebra** of G is denoted by, $\mathfrak{g}$, and it is the set of all matrices, X, such that e^{tX} is in $G \quad \forall$ real numbers t.
In other words, X is in $\mathfrak{g}$ if and only if the one-parameter subgroup generated by X lies in G.
[**Note 1:** Even if G is a subgroup of GL $(n, \mathbb{C})$ (not necessarily GL $(n, \mathbb{R})$), we do not require e^{tX} to be in $G \, \forall$ complex numbers t, but only $\forall$ real numbers t.]
[**Note 2:** It is not enough to have $e^X \in G$. It is easy to give an example of an X and a G such that $e^X \in G$, but $e^{tX} \notin G$ for some real values of t. Such an X is not in the Lie algebra of G.]

2.11.1 The General Linear Group

If X is a n x n complex matrix, then we know that the exponential of the matrix is invertible, i.e. e^{tX} is invertible. Hence the Lie algebra of GL $(n, \mathbb{C})$, defined as $\mathfrak{gl}(n, \mathbb{C})$, is the space of all n x n complex matrices.

If X is a n x n real matrix, then e^{tX} is invertible and real. In addition, if e^{tX} is real ($\forall$real t) then $X = \frac{d}{dt} e^{tX}|_{t=0}$ will also be real. Therefore the Lie algebra of GL $(n, \mathbb{R})$, defined as $\mathfrak{gl}(n, \mathbb{R})$, is the space of all n x n real matrices.

[**Note:** The above argument shows that if G is a subgroup of GL $(n, \mathbb{R})$, then the Lie algebra of G must be consist entirely of real matrices.]

2.11.2 The Special Linear Group

Recall, that the special linear group are n x n complex matrices, X that have complex entries and a determinant of one.

Also, theorem 2.10.3 states that $\det\left(e^X\right) = e^{\text{trace}(X)}$ therefore if

$$\begin{aligned} & \text{trace}\,(X) &=& \ 0 \\ \implies & \det\left(e^{tX}\right) &=& \ 1 \ \forall \text{real } t \end{aligned}$$

So, if X is any n x n matrix such that $\det\left(e^{tX}\right) = 1 \ \forall t$ then $e^{t\,\text{trace}(X)} = 1 \ \forall t$. This implies that $t \cdot \text{trace}\,(X)$ is an integer multiple of $2\pi i \ \forall t$, but this is only possible if $\text{trace}\,(X) = 0$. Thus, the Lie algebra of SL $(n, \mathbb{C})$ is the space of all n x n complex matrices with trace zero, written as $\mathfrak{sl}(n, \mathbb{C})$.

Similarly, the Lie algebra of SL $(n, \mathbb{R})$ is the space of all n x n real matrices with trace zero, written as $\mathfrak{sl}(n, \mathbb{R})$.

2.11.3 The Unitary Group

Recall, a matrix U is unitary if and only if $U^* = U^{-1}$. Hence, e^{tX} is unitary if and only if

$$\begin{aligned} \left(e^{tX}\right)^* &= \left(e^{tX}\right)^{-1} \\ &= e^{-tX} \end{aligned}$$

There exists a proposition, which states that $\left(e^{tX}\right)^* = e^{tX^*}$ and using this proposition the above equation becomes

$$e^{tX^*} = e^{-tX} \tag{2}$$

Therefore, a sufficient condition for 2 to hold is $-X = X^*$. However, if 2 holds $\forall t$ then by differentiating at $t = 0$ we will get a necessary condition, $-X = X^*$.

Hence, the Lie algebra of U (n) is the space of all n x n complex matrices, X, such that $X^* = -X$. The Lie algebra of U (n) is denoted as $\mathfrak{u}(n)$.

Similarly, by using the Lie algebra definition of the unitary group and the special linear group we can define the Lie algebra of special unitary group, SU (n). The Lie algebra of SU (n) is the space of all n x n complex matrices, X, such that $X^* = -X$ and trace $(X) = 0$ and is written as $\mathfrak{su}(n)$.

2.11.4 The Orthogonal Group

The identity component of $\mathrm{O}(n)$ is simply $\mathrm{SO}(n)$. Furthermore, another proposition states that the exponential of a matrix in the Lie algebra is automatically in the identity component. So, this means that the Lie algebra of $\mathrm{O}(n)$ is the same as the Lie algebra of $\mathrm{SO}(n)$.

Recall, a n x n real matrix, R, is orthogonal if and only if $R^T = R^{-1}$. So, given a n x n matrix, X, e^{tX} is orthogonal if and only if

$$\begin{aligned} \left(e^{tX}\right)^T &= \left(e^{tX}\right)^{-1} \\ \implies e^{tX^T} &= e^{-tX} \end{aligned} \tag{3}$$

Therefore, a sufficient condition for 3 to hold is $X^T = -X$. However, if 3 holds $\forall t$ then by differentiating at $t = 0$ we will get the necessary condition, $X^T = -X$.

Hence, the Lie algebra of $\mathrm{O}(n)$ and of $\mathrm{SO}(n)$ is the space of all n x n real matrices X with $X^T = -X$ and it is denoted as $\mathfrak{so}(n)$. The condition $X^T = -X$ forces the diagonal of X to be zero, which implies that the trace of X is zero.

Similarly, the Lie algebra of $\mathrm{SO}(n, \mathbb{C})$ is the space of n x n complex matrices with $X^T = -X$ and written as $\mathfrak{so}(n, \mathbb{C})$.

2.11.5 The Generalised Orthogonal Group

A matrix, A, is in $\mathrm{O}(n, k)$ if and only if $A^T g A = g$, where g is the $(n + k)$ x $(n + k)$ diagonal matrix with the first n diagonal entries equal to one and the last k entries equal to negative one. This is equivalent to saying $g^{-1} A^T g = A^{-1}$ and since $g^{-1} = g$, we can write, $g A^T g = A^{-1}$.

So, if X is a $(n + k)$ x $(n + k)$ real matrix, then e^{tX} is $\mathrm{O}(n, k)$ if and only if

$$\begin{aligned} g e^{tX^T} g &= e^{tgX^T g} \\ &= e^{-tX} \end{aligned}$$

This condition holds for all real t if and only if $g X^T g = -X$. Therefore, the Lie algebra of $\mathrm{O}(n, k)$, which is also the same as the Lie algebra of $\mathrm{SO}(n, k)$ (same argument as the orthogonal group) is the space of all $(n + k)$ x $(n + k)$ real matrices X with $g X^T g = -X$ and it is denoted by $\mathfrak{so}(n, k)$.

2.11.6 The Symplectic Group

The calculation of these Lie algebras are similar to generalised orthogonal groups. So, let J be the matrix defined in section 2.2 (example 7) then the Lie algebra for $\mathrm{Sp}(n, \mathbb{R})$ is the space of $2n$ x $2n$ real matrices X with $J X^T J = X$ and it is denoted by $\mathfrak{sp}(n, \mathbb{R})$.

Similarly, the Lie algebra for $\mathrm{Sp}(n, \mathbb{C})$ is the space of $2n$ x $2n$ real matrices X with $J X^T J = X$ and it is denoted by $\mathfrak{sp}(n, \mathbb{C})$.

2.11.7 The Heisenberg Group

Recall, the Heisenberg group, H, is the group of all 3 x 3 real matrices, A, which are of the form:

$$A = \begin{pmatrix} 1 & a & b \\ 0 & 1 & c \\ 0 & 0 & 1 \end{pmatrix}$$

with $a, b, c \in \mathbb{R}$.

Furthermore, earlier we computed the exponential of a nilpotent matrix, X, which is of the form

$$X = \begin{pmatrix} 0 & a & b \\ 0 & 0 & c \\ 0 & 0 & 0 \end{pmatrix}$$

and we discovered that e^X is in H as

$$e^X = \begin{pmatrix} 1 & a & b + \frac{1}{2}ac \\ 0 & 1 & c \\ 0 & 0 & 1 \end{pmatrix}$$

However, if X is any matrix such that e^{tX} is of the form A, then all the entries of $X = \frac{d}{dt}e^{tX}|_{t=0}$, which are on or below the diagonal must be zero in order for X to be in the form of a nilpotent matrix.

Therefore, the Lie algebra of a Heisenberg group is the space of all 3 x 3 real matrices that are upper triangular.

2.12 Properties of Lie Algebra

Basic properties of the Lie algebra of a matrix Lie group:

Proposition 2.12.1.
Let G be a matrix Lie group, and X an element of its Lie algebra. Then e^X is an element of the identity component of G.

Proof.

From the definition of Lie algebra, e^{tX} lies in G $\forall t$. So, as t varies from 0 to 1, e^{tX} is a continuous path connecting the identity to e^X.

QED

Proposition 2.12.2.
Let G be a matrix Lie group with a Lie algebra, $\mathfrak{g}$. Let $X \in \mathfrak{g}$ and $A \in G$ then $AXA^{-1} \in \mathfrak{g}$.

Proof.

From the theory about exponentials we know that if C is an invertible matrix then:

$$e^{CXC^{-1}} = Ce^{X}C^{-1}$$

By using this result we can write:

$$e^{t\left(AXA^{-1}\right)} = Ae^{tX}A^{-1}$$

thus $Ae^{tX}A^{-1} \in G \quad \forall t$.

QED

A list of a few other properties that we will not prove in this project:

Let G be a matrix Lie group, $\mathfrak{g}$ defined as its Lie algebra, and $X, Y \in \mathfrak{g}$, then this following properties hold:

1. $sX \in \mathfrak{g}$ $\forall$real numbers s
2. $X + Y \in \mathfrak{g}$
3. $XY - YX \in \mathfrak{g}$

The following definitions and properties are going to be useful, for proving the Baker-Campbell-Hausdorff Formula.

Definition 2.12.3.
Given two n x n matrices A and B, the **bracket/commutator** of A and B is defined as

$$[A, B] = AB - BA$$

From Property 3 (above), we know that the Lie algebra of any matrix Lie group is closed under brackets.

A simple example of using the commutator is show below:

$$\begin{aligned}
A &= \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\
B &= \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\
[A, B] &= AB - BA \\
&= \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} - \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\
&= 0 - 0 \\
&= 0
\end{aligned}$$

This is a very useful shorthand to simplify some complex matrix equations.

Definition 2.12.4.
A matrix Lie group, G, is **complex** if its Lie algebra,$\mathfrak{g}$, is a complex subspace of $M_n(\mathbb{C})$ i.e. if $iX \in \mathfrak{g} \quad \forall X \in \mathfrak{g}$
Examples of complex groups are:
$\mathsf{GL}(n,\mathbb{C})$, $\mathsf{SL}(n,\mathbb{C})$, $\mathsf{SO}(n,\mathbb{C})$, and $\mathsf{Sp}(n,\mathbb{C})$

Definition 2.12.5.
Let G be a matrix Lie group, with Lie algebra $\mathfrak{g}$. Then for each $A \in G$, a linear map $\mathsf{Ad}_A : \mathfrak{g} \to \mathfrak{g}$ is defined using:

$$\mathsf{Ad}_A(X) = AXA^{-1}$$

Properties 2.12.6.

1. There is an associated real linear map $X \to \mathsf{ad}_X$ from the Lie algebra of G to the Lie algebra of $\mathsf{GL}(\mathfrak{g})$ i.e. linear map from $\mathfrak{g}$ to $\mathsf{gl}(\mathfrak{g})$ with a property that states

   $$e^{\mathsf{ad}_X} = \mathsf{Ad}\left(e^X\right)$$

   Note: $\mathsf{GL}(\mathfrak{g})$ is quite the same as $\mathsf{GL}(k,\mathbb{R})$ as $\mathfrak{g}$ is essentially a real vector space with some dimension k, hence $\mathsf{GL}(\mathfrak{g})$ can be regarded as a matrix Lie group.
   Note: $\mathbf{gl}(\mathfrak{g})$ is the Lie algebra of the matrix Lie group, $\mathsf{GL}(\mathfrak{g})$ i.e. the space of all linear maps of $\mathfrak{g}$ to itself.

2. For any $X \in M_n(\mathbb{C})$, let $\mathsf{ad}_X : M_n(\mathbb{C}) \to M_n(\mathbb{C})$ be defined as

   $$\begin{aligned} e^{\mathsf{ad}_X} Y &= \mathsf{Ad}_{e^X} Y \\ &= e^X Y e^{-X} \end{aligned}$$

   This property can easily be proved using property 1 (above) and definition 2.12.5.

2.13 Lie Algebra

Proposition 2.13.1.
A **finite-dimensional real or complex Lie algebra** is a finite-dimensional real or complex vector space, $\mathfrak{g}$, together with a map $[\cdot,\cdot]$ from $\mathfrak{g}$ x $\mathfrak{g}$ into $\mathfrak{g}$ and holds the following properties:

1. $[\cdot,\cdot]$ is bilinear
2. $[X,Y] = -[Y,X]\ \ \forall X,Y \in \mathfrak{g}$ (Skew-symmetry)
   (a) Property 2 implies that $[X,X] = -[X,X] = 0\ \forall X \in \mathfrak{g}$
3. $[X,[Y,Z]] + [Y\,[Z,X]] + [Z,[X,Y]] = 0\ \forall X,Y,Z \in \mathfrak{g}$ (Jacobi identity)

Using the properties defined in section 2.12, we can prove proposition 2.13.1. Now we can define two simple examples for Property 2 and 3 that will allow us to understand how the commutator relation works. Assume for both examples that X and Y are 3 x 3 Heisenberg matrices.

Example (Property 2):

$$\begin{aligned} X &= \begin{pmatrix} 1 & 1 & 2 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{pmatrix} \\ Y &= \begin{pmatrix} 1 & 4 & 5 \\ 0 & 1 & 6 \\ 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

$$\begin{aligned} [X,Y] &= XY - YX \\ &= \begin{pmatrix} 1 & 1 & 2 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 4 & 5 \\ 0 & 1 & 6 \\ 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 1 & 4 & 5 \\ 0 & 1 & 6 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 1 & 2 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 0 & -6 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{aligned}$$

$$\begin{aligned} [Y,X] &= YX - XY \\ &= \begin{pmatrix} 1 & 4 & 5 \\ 0 & 1 & 6 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 1 & 2 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 1 & 1 & 2 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 4 & 5 \\ 0 & 1 & 6 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 0 & 6 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\ &= -[X,Y] \end{aligned}$$

Example (Property 3):

$$
\begin{aligned}
X &= \begin{pmatrix} 1 & 1 & 2 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{pmatrix} \\
Y &= \begin{pmatrix} 1 & 4 & 5 \\ 0 & 1 & 6 \\ 0 & 0 & 1 \end{pmatrix} \\
Z &= \begin{pmatrix} 1 & 7 & 8 \\ 0 & 1 & 9 \\ 0 & 0 & 1 \end{pmatrix}
\end{aligned}
$$

$$
\begin{aligned}
& [X,[Y,Z]] + [Y\,[Z,X]] + [Z,[X,Y]] \\
= \; & \left[\begin{pmatrix} 1 & 1 & 2 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{pmatrix}, \left[\begin{pmatrix} 1 & 4 & 5 \\ 0 & 1 & 6 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 7 & 8 \\ 0 & 1 & 9 \\ 0 & 0 & 1 \end{pmatrix}\right]\right] \\
& + \left[\begin{pmatrix} 1 & 4 & 5 \\ 0 & 1 & 6 \\ 0 & 0 & 1 \end{pmatrix} \left[\begin{pmatrix} 1 & 7 & 8 \\ 0 & 1 & 9 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 & 2 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{pmatrix}\right]\right] \\
& + \left[\begin{pmatrix} 1 & 7 & 8 \\ 0 & 1 & 9 \\ 0 & 0 & 1 \end{pmatrix}, \left[\begin{pmatrix} 1 & 1 & 2 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 4 & 5 \\ 0 & 1 & 6 \\ 0 & 0 & 1 \end{pmatrix}\right]\right] \\
= \; & \begin{pmatrix} 0 & 0 & -6 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 12 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & -6 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\
= \; & \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \\
= \; & 0
\end{aligned}
$$

Definition 2.13.2.
Let $\mathfrak{g}$ be a Lie algebra, then for $X \in \mathfrak{g}$ we can define a linear map $\mathrm{ad}_X : \mathfrak{g} \to \mathfrak{g}$ by

$$\mathrm{ad}_X(Y) = [X,Y]$$

Note: This notation can be useful in situations where we have the following expression and we can simplify it using "ad":

$$[X,[X,[X,Y]]] = (\mathrm{ad}_X)^3(Y)$$

2.14 The Baker-Campbell-Hausdorff Formula

This is a very important formula as it defines $\log\left(e^X e^Y\right)$ in terms of brackets of X and Y, brackets of brackets of X and Y etc. This clearly indicates that for elements of the form e^X,

where X is small, the group product for a matrix Lie group, G, is completely expressible in terms of its Lie algebra. This is true because one form the Baker-Campbell-Hausdorff formula states that if X and Y are sufficiently small, then:

$$\log\left(e^X e^Y\right) = X + Y + \frac{1}{2}[X,Y] + \frac{1}{12}[X,[X,Y]] - \frac{1}{12}[Y[X,Y]] + \dots$$

So $\log\left(e^X e^Y\right)$ and, hence, $e^X e^Y$ can be computed in *Lie-algebraic* terms.

Rather than proving the above series of the Baker-Campbell-Hausdorff formula, we will show the proof of the general (i.e. the integral) part of the formula.

Below is the integral form of the Baker-Campbell-Hausdorff formula and its proof.

Theorem 2.14.1.
For all n x n complex matrices X and Y with $\|X\|$ and $\|Y\|$ sufficiently small, then

$$\log\left(e^X e^Y\right) = X + \int_0^1 g\left(e^{\mathsf{ad}_X} e^{t\mathsf{ad}_Y}\right)(Y)\cdot \mathsf{dt}$$

Proof.

Define:

$$Z(t) = \log\left(e^X e^{tY}\right)$$

So, if X and Y are sufficiently small then $Z(t)$ can be defined for $0 \le t \le 1$ (Note: $Z(t)$ is smooth). Hence, the aim is to compute $Z(1)$ and then the proof will be complete.

Clearly, using the above definition of $Z(t)$ we can write:

$$e^{Z(t)} = e^X e^{tY}$$

so that

$$\begin{aligned} e^{-Z(t)}\frac{d}{dt}e^{Z(t)} &= \left(e^X e^{tY}\right)^{-1} e^X e^{tY} Y \\ &= Y \end{aligned} \tag{4}$$

On the other hand, equation 4 can be re-written using the **Derivative of Exponential Theorem**, which states:

$$\begin{aligned} \frac{d}{dt}e^{X+tY}|_{t=0} &= e^X\left\{\frac{I - e^{-\mathsf{ad}_X}}{\mathsf{ad}_X}(Y)\right\} \\ &= e^X\left\{Y - \frac{[X,Y]}{2!} + \frac{[X,[X,Y]]}{3!} - \dots\right\} \end{aligned}$$

where X and Y are complex matrices.

And if $X(t)$ is a smooth matrix valued function (like equation 4) then the following is true:

$$\frac{d}{dt}e^{X(t)}|_{t=0} = e^{X(t)}\left\{\frac{I-e^{-\mathsf{ad}_{X(t)}}}{\mathsf{ad}_{X(t)}}\left(\frac{dX}{dt}\right)\right\}$$

Now, using the above theorem, equation 4 can be written as:

$$\begin{aligned} e^{-Z(t)}\frac{d}{dt}e^{Z(t)} &= \left\{\frac{I-e^{-\mathsf{ad}_{Z(t)}}}{\mathsf{ad}_{Z(t)}}\right\}\left(\frac{dZ}{dt}\right) \\ \implies \left\{\frac{I-e^{-\mathsf{ad}_{Z(t)}}}{\mathsf{ad}_{Z(t)}}\right\}\left(\frac{dZ}{dt}\right) &= Y \end{aligned}$$

If X and Y are small, then $Z(t)$ will also be small which implies that $\left[I-e^{-\mathsf{ad}_{Z(t)}}\right]$ will be close to the identity and hence invertible.

Thus, the following would hold:

$$\frac{dZ}{dt} = \left\{\frac{I-e^{-\mathsf{ad}_{Z(t)}}}{\mathsf{ad}_{Z(t)}}\right\}^{-1}(Y) \tag{5}$$

Since, we have defined $e^{Z(t)} = e^{X}e^{tY}$ then by applying the homomorphism "Ad" we get:

$$\mathsf{Ad}_{e^{Z(t)}} = \mathsf{Ad}_{e^{X}}\mathsf{Ad}_{e^{tY}} \tag{6}$$

Property 2 mentioned in Properties 2.12.6 describes the relationship between "Ad" and "ad", hence equation 6 becomes:

$$\begin{aligned} e^{\mathsf{ad}_{Z(t)}} &= e^{\mathsf{ad}_{X}}e^{t\mathsf{ad}_{Y}} \\ \implies \mathsf{ad}_{Z(t)} &= \log\left(e^{\mathsf{ad}_{X}}e^{t\mathsf{ad}_{Y}}\right) \end{aligned} \tag{7}$$

So, we can substitute equation 7 into equation 5 and this results in:

$$\frac{dZ}{dt} = \left\{\frac{I-\left(e^{\mathsf{ad}_{X}}e^{t\mathsf{ad}_{Y}}\right)^{-1}}{\log\left(e^{\mathsf{ad}_{X}}e^{t\mathsf{ad}_{Y}}\right)}\right\}^{-1}(Y) \tag{8}$$

Note, we can define:

$$g(z) = \left\{\frac{1-z^{-1}}{\log(z)}\right\}^{-1}$$

Thus, equation 8 can be re-written as:

$$\frac{dZ}{dt} = g\left(e^{\mathsf{ad}_X} e^{t\mathsf{ad}_Y}\right)(Y) \tag{9}$$

Now, if we take $Z(0) = X$ and integrate equation 9 then we get:

$$Z(1) = X + \int_0^1 g\left(e^{\mathsf{ad}_X} e^{t\mathsf{ad}_Y}\right)(Y) \cdot dt$$

which is what we needed as $Z(1) = \log\left(e^X e^Y\right)$, and this ends the proof.

QED

3 Markovian Arrival Process (MAP)

3.1 Overview

Markovian arrival processes (MAP) is defined as a random sequence of *events* and we may think of it as a *point process.* There are many different types of MAPs, for instance the most common ones are *Poisson process, PH renewal process, etc. [5]*

Markovian arrival processes have two processes; one of them is a *continuous-time Markov chain (also known as a continuous-time Markov process process),* $J(t)$, which is a Markov process generated by a *rate matrix/generator,* Q. The second process is a counting process $N(t)$, which has state space $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$. Every time there is a transition in $J(t)$, that is marked, it increases $N(t)$. [6]

Now, we can further understand the first process that is required in a MAP: *continuous-time Markov process.* This is a stochastic process that satisfies the Markov property and takes values from a set called the state space.

Below are the definitions of the underlined terms:

Definition 3.1.1. (Markov Property)[9]:
The Markov property states that for any t within, $0 < t < s$, the conditional probability distribution of the process at s (given the history of the whole process up to and including the state of the process at time, t) depends only on the state of the process at time, t.
In other words, the conditional probability distribution of the future states in the process (i.e. say at time s) is only dependent on the current state (say at time t) and is not affected by the events of the states that occur before time, t.

Definition 3.1.2. (Finite State Space)[10]:
A **finite state space** can be finite with the transition probability distribution represented as a matrix. This matrix is defined as the transition matrix, say P, with the (i, j)th element of P equal to:

$$p_{ij} = \mathrm{P}(X_{n+1} = j | X_n = i)$$

So, each row of P sums to one and all elements are non-negative. Thus, P is a right stochastic matrix.

Now, we explore the second process that is required to define a MAP. For this project we will mainly be using the *Poisson process (distribution).*

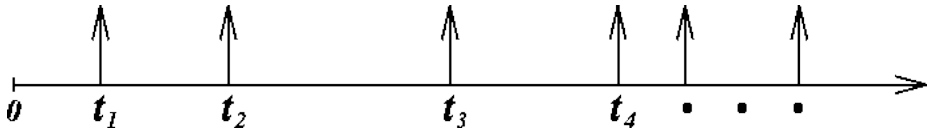
Definition 3.1.3 (Poisson Process)[7]:
As mentioned one of the counting processes, which is frequently used in MAPs is the *Poisson process*. This is a continuous-time counting process, $\{N(t), t \geq 0\}$, and the following properties hold:

1. $N(0) = 0$
2. The number of changes in disjoint intervals are independent of each other.
3. The probability distribution of the number of occurrences in any time interval is dependent on the length of the interval.
   (a) The probability of *exactly one change* in a relatively small interval, $h = \frac{1}{n}$ is $P = v \cdot h \equiv \frac{v}{n}$ where v is the probability of one change and n the number of trials.

The results of these properties are summarised below:

1. The probability distribution of this process, $N(t)$, is a *Poisson distribution.*
2. But, the probability distribution of the waiting time between the next occurrence is an *exponential distribution.*
3. The total number of occurrences, $N(t)$, has a Poisson distribution over $(0, t]$, while any individual occurrence, $t \in (a, b]$ is a uniform distribution.

So, in a Poisson process the sequence of events are randomly spaced over time and that can be shown using a diagram:



Examples of real-life situations where we may estimate the probability of events using a Poisson process include:

- Customers arriving into a bank (day-to-day life)
- Geiger-Muller counter clicks: detects the emission of radiation (science related)
- Packets arriving into a buffer on a network (computer-science)

So the rate, λ, of a Poisson process is the average number of events per unit time (calculated over a long period of time). Thus, probability of n arrivals in t units of time is:

$$\mathrm{P}_n(t) = \frac{(\lambda t)^n}{n!} e^{-\lambda t}$$

Looking at the two different components of a MAP, we can now further explore the properties of a MAP and see its unique features. So, now we can give a more formal definition of a MAP:

Definition 3.1.4. (Markov Arrival Process):
A MAP is a stochastic process defined by the continuous-time bivariate Markov chain $\{N(t), J(t)\}$. $N(t)$ is the number of arrivals in $[0, t)$ and $J(t)$, $1 \leq J(t) \leq m$, where $J(t)$ is the state of the Markov chain (phase at time, t).
Now, the m-state MAP can be written using a Markov process $\{N(t), J(t)\}$ on the state space $\{(i, j) | i \geq 0, 1 \leq j \leq m\}$ with an infinitesimal generator, Q^* with the following structure:

$$Q^* = \begin{bmatrix} C & D & 0 & 0 & \dots \\ 0 & C & D & 0 & \dots \\ 0 & 0 & C & D & \dots \\ \vdots & \vdots & \vdots & \ddots & \ddots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

where both C and D are m x m matrices. C is the matrix that defines transitions without arrival events and all its diagonal entries are negative while its off-diagonal elements are non-negative. Matrix D has only non-negative elements and represents transitions with arrival events.
Thus, matrix Q defined as

$$Q = C + D$$

is an irreducible infinitesimal generator of the underlying continuous-time Markov chain (CTMC), $\{J(t)\}$. **[11]**
In other words, if we define Q as:

$$Q = \begin{bmatrix} -\gamma_{1,1} & \gamma_{1,2} & \cdots & \gamma_{1,m} \\ \gamma_{2,1,} & -\gamma_{2,2} & \cdots & \gamma_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{m,1} & \gamma_{m,2} & \cdots & -\gamma_{m,m} \end{bmatrix}$$

then by the definition of a matrix generator $\gamma_{i,i} = \sum_{j=1, j \neq i}^{m} \gamma_{i,j}$ i.e. the rows of Q must sum to 0. [12]

In addition, we have C on the diagonal of Q^* as it is the block matrix that defines state transitions with no arrivals and has negative diagonal entries. So, if we have an arrival then we move to the next element in Q^*, which is defined by the rate matrix, D. This process continues as we proceed down the rows of Q^*.

3.2 Example: SL $(2, \mathbb{R})$

This example is based on the information and data given in [8]. We build on this theory and expand the steps defined in that paper.

Given the Lie algebra of the special linear group, SL $(2, \mathbb{R})$, as $\mathfrak{sl}(2, \mathbb{R})$ i.e. the group of 2 x 2 matrices with determinant one we can denote $E_+^{(c)}$, $E_-^{(c)}$, and $E_3^{(c)}$ as $(c+1)$ x $(c+1)$ $c \in \mathbb{R}$ matrix representations of the standard generators $\hat{e}_+$, $\hat{e}_-$, $\hat{e}_3$. We define the commutator relations directly from it definition: $[A, B] = AB - BA$ i.e:

$$\begin{aligned} [\hat{e}_+, \hat{e}_-] &= \hat{e}_3 \\ [\hat{e}_3, \hat{e}_+] &= 2\hat{e}_+ \\ [\hat{e}_3, \hat{e}_-] &= -2\hat{e}_- \end{aligned}$$

so, $\left[\hat{a}, \hat{b}\right] = \hat{a}\hat{b} - \hat{b}\hat{a}$ for any operator $\hat{a}, \hat{b}$

For 2 x 2 matrices there are generally four generators, however since there is one condition (trace $= 0$) on the Lie algebra of SL $(2, \mathbb{R})$ we subtract one, thus leaving us with only three generators i.e. $2^2 - 1 = 3$.

From Lie algebra we know that there exists infinitely many matrix representations that satisfy the commutator, and we can also conclude that any $(c+1)$ x $(c+1)$ matrix representations of $\hat{e}_+, \hat{e}_-, \hat{e}_3$ can be defined by:

$$\begin{aligned} \left(E_+^{(c)}\right)_{i,j} &= (c-i)\,\delta_{i,j-1} \\ \left(E_-^{(c)}\right)_{i,j} &= i\delta_{i,j+1} \\ \left(E_3^{(c)}\right)_{i,j} &= (c-2i)\,\delta_{i,j} \end{aligned}$$

where $\delta_{i,j}$ is the Kroncker delta that we defined earlier.

Further investigations will be demonstrated on the class of MAPs that have the following matrices, C and D:

$$\begin{aligned} C &= c_+E_+^{(c)} + c_-E_-^{(c)} - \frac{\alpha}{2}E_3^{(c)} - \frac{\beta}{2}I^{(c)} \\ D &= d_+E_+^{(c)} + d_-E_-^{(c)} + \frac{\lambda_+ - \lambda_-}{2}E_3^{(c)} + \frac{\lambda_+ + \lambda_-}{2}I^{(c)} \end{aligned}$$

where $\alpha = (c_+ + d_+ + \lambda_+) - (c_- + d_- + \lambda_-)$ and $\beta = (c_+ + d_+ + \lambda_+) + (c_- + d_- + \lambda_-)$. $I^{(c)} = cI_{(c+1)\text{ x }(c+1)}$ i.e. c times the $(c+1)$ x $(c+1)$ identity matrix, $I_{(c+1)\text{ x }(c+1)}$. Now, we can show that C and D satisfy the conditions for representing matrices of a MAP if the following hold:

- c_+, c_- are non-negative real numbers.
- $d_+, d_-, \lambda_+, \lambda_-$ are also all non-negative real numbers while at least one of them is strictly positive.

3.2.1 Solving for C and D

We can now use matrix Lie algebra on $E_+^{(c)}, E_-^{(c)}, E_3^{(c)}$ as the commutator relations from above hold. Hence, the following is true:

$$\begin{aligned}[C,D] &= \left[c_+E_+^{(c)} + c_-E_-^{(c)} - \frac{\alpha}{2}E_3^{(c)} - \frac{\beta}{2}I^{(c)}, d_+E_+^{(c)} + d_-E_-^{(c)} + \frac{\lambda_+ - \lambda_-}{2}E_3^{(c)} + \frac{\lambda_+ + \lambda_-}{2}I^{(c)}\right] \\ &= -\left[\alpha d_+ + c_+\left(\lambda_+ - \lambda_-\right)\right]E_+^{(c)} + \left[\alpha d_- + c_-\left(\lambda_+ - \lambda_-\right)\right]E_-^{(c)} + \left(c_+d_- - c_-d_+\right)E_3^{(c)}\end{aligned}$$

$$\begin{aligned}[C,[C,D]] &= \left\{\alpha\left[\alpha d_+ + c_+\left(\lambda_+ - \lambda_-\right)\right] - 2c_+\left(c_+d_- - c_-d_+\right)\right\}E_+^{(c)} \\ &\quad + \left\{\alpha\left[\alpha d_- + c_-\left(\lambda_+ - \lambda_-\right)\right] + 2c_-\left(c_+d_- - c_-d_+\right)\right\}E_-^{(c)} \\ &\quad + \left\{c_+\left[\alpha d_- + c_-\left(\lambda_+ - \lambda_-\right)\right] + c_-\left[\alpha d_+ + c_+\left(\lambda_+ - \lambda_-\right)\right]\right\}E_3^{(c)}\end{aligned}$$

To further simplify this we can use the definition of the linear map ad as mentioned in Definition 2.13.2. So, we can write ad_C and repeatedly apply this operation to D which results in:

$$\begin{aligned}\mathrm{ad}_C^3 D &= [C,[C,[C,D]]] \\ &= a^2\,[C,D]\end{aligned}$$

Therefore, we can write the general case as:

$$\mathrm{ad}_C^n D = \begin{cases} a^{2k}\,[C,D] & n = 2k+1 \\ a^{2k}\,[C,[C,D]] & n = 2k+2 \end{cases}$$

for $k \in \mathbb{Z}_+$ where $a^2 = \alpha^2 + 4c_+c_-$.

We can choose a 2 x 2 matrix representation of $\hat{e}_+, \hat{e}_-, \hat{e}_3$:

$$\begin{aligned}E_+^{(1)} &= \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \\ E_-^{(1)} &= \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \\ E_3^{(1)} &= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\end{aligned}$$

So, now we can write C and D using the definitions given above i.e.

$$\begin{aligned}C &= c_+E_+^{(c)} + c_-E_-^{(c)} - \frac{\alpha}{2}E_3^{(c)} - \frac{\beta}{2}I^{(c)} \\ D &= d_+E_+^{(c)} + d_-E_-^{(c)} + \frac{\lambda_+ - \lambda_-}{2}E_3^{(c)} + \frac{\lambda_+ + \lambda_-}{2}I^{(c)}\end{aligned}$$

Hence,

$$
\begin{aligned}
C &= c_+E_+^{(1)} + c_-E_-^{(1)} - \frac{\alpha}{2}E_3^{(1)} - \frac{\beta}{2}I^{(1)} \\
\implies C &= c_+\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} + c_-\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} - \frac{\alpha}{2}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} - \frac{\beta}{2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\
\implies C &= \begin{pmatrix} 0 & c_+ \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ c_- & 0 \end{pmatrix} - \begin{pmatrix} \frac{\alpha}{2} & 0 \\ 0 & -\frac{\alpha}{2} \end{pmatrix} - \begin{pmatrix} \frac{\beta}{2} & 0 \\ 0 & \frac{\beta}{2} \end{pmatrix} \\
\implies C &= \begin{pmatrix} 0 & c_+ \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ c_- & 0 \end{pmatrix} - \begin{pmatrix} \frac{(c_++d_++\lambda_+)-(c_-+d_-+\lambda_-)}{2} & 0 \\ 0 & -\frac{(c_++d_++\lambda_+)-(c_-+d_-+\lambda_-)}{2} \end{pmatrix} \\
&\quad - \begin{pmatrix} \frac{(c_++d_++\lambda_+)+(c_-+d_-+\lambda_-)}{2} & 0 \\ 0 & \frac{(c_++d_++\lambda_+)+(c_-+d_-+\lambda_-)}{2} \end{pmatrix} \\
\implies C &= \begin{pmatrix} -c_+ - d_+ - \lambda_+ & c_+ \\ c_- & -c_- - d_- - \lambda_- \end{pmatrix}
\end{aligned}
$$

$$
\begin{aligned}
D &= d_+E_+^{(1)} + d_-E_-^{(1)} + \frac{\lambda_+ - \lambda_-}{2}E_3^{(1)} + \frac{\lambda_+ + \lambda_-}{2}I^{(1)} \\
\implies D &= d_+\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} + d_-\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} + \frac{\lambda_+ - \lambda_-}{2}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} + \frac{\lambda_+ + \lambda_-}{2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\
\implies D &= \begin{pmatrix} 0 & d_+ \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ d_- & 0 \end{pmatrix} + \begin{pmatrix} \frac{\lambda_+-\lambda_-}{2} & 0 \\ 0 & -\left(\frac{\lambda_+-\lambda_-}{2}\right) \end{pmatrix} + \begin{pmatrix} \frac{\lambda_++\lambda_-}{2} & 0 \\ 0 & \frac{\lambda_++\lambda_-}{2} \end{pmatrix} \\
\implies D &= \begin{pmatrix} \lambda_+ & d_+ \\ d_- & \lambda_- \end{pmatrix}
\end{aligned}
$$

These are standard representing matrices of a two-state MAP that has ON (subscript '+') and OFF (subscript '-'). There are many other examples that can be described by the MAP with representing matrices C and D, both in the case when $c = 1$ and in the general case.

3.2.2 Implementing C and D

We can also construct a solution of $P(n,t)$ for a MAP $\{N(t), J(t)\}$ with representing matrices C and D that do not necessarily have to be expanded in terms of E_+, E_-, E_3, I where $n \in \mathbb{Z}_+$ and $t \in \mathbb{R}_+$. Thus, the MAP can be defined by the matrix with the i,jth element written as:

$$
\left(P(n,t)\right)_{i,j} = \mathrm{P}\left(N(t) = n, J(t) = j | N(0) = 0, J(0) = i\right)
$$

We know that matrix C defines the transitions without arrival events, while matrix D defines state transitions with arrival events. So, we can write the i,jth element of $\exp(Ct)$ and it is defined as the probability that there is no arrival event in the time, $(0,t]$. In addition, given that the state of the Markov chain is i at time 0 then at time t the state of the Markov chain is j. This implies that when $n = 0$ (i.e. no arrivals in time, t) then $P(0,t) = \exp(Ct)$.

Now, let t_i be the time when the ith arrival event occurs for $i = 1, 2, \ldots, n$ where $0 < t_1 < t_2 < \cdots < t_n < t$.

Hence, if the following conditions hold:

1. Markov property and the probability density that an arrival event occurs at time, t
2. The state of the Markov chain at time t is j given that the state of the Markov chain at time 0 is i

then we can write the i, jth element of $\exp[Ct]\, D$.

This is defined as the probability that there are n consecutive arrival events in $(0, t]$ and occur at time $t_1, t_2, \ldots, t_n$. Finally, condition 2 can be deduced as the i, jth element of the following matrix:

$$e^{Ct_1} D e^{C(t_2 - t_1)} D \ldots e^{C(t_n - t_{n-1})} D e^{C(t - t_n)} \mathrm{d}t_1 \mathrm{d}t_2 \ldots \mathrm{d}t_n$$

Note: the above expression come directly from the matrix integral expression for $P(n, t)$ for $n \geq 1$, which is defined as:

$$P(n, t) = \int_{\Omega_{n,t}} e^{Ct_1} D e^{C(t_2 - t_1)} D \ldots e^{C(t_n - t_{n-1})} D e^{C(t - t_n)} \mathrm{d}t_1 \mathrm{d}t_2 \ldots \mathrm{d}t_n$$

where $\Omega_{n,t} = \left\{(t_1, t_2, \ldots, t_n) \in \mathbb{R}_+^n | 0 < t_1 < \cdots < t_n < t\right\}$

Furthermore, by keeping in mind the above equation and using the idea of $e^{C(x-y)} = e^{Cx} e^{-Cy} = e^{-Cy} e^{Cx}$ we can define a matrix, $M(n, t)$, for $t \in \mathbb{R}_+$ and $n \geq 1$ as:

$$M(n, t) = \int_{\Omega_{n,t}} \left(e^{Ct_1} D e^{-Ct_1}\right) \left(e^{Ct_2} D e^{-Ct_2}\right) \ldots \left(e^{Ct_n} D e^{-Ct_n}\right) \mathrm{d}t_1 \mathrm{d}t_2 \ldots \mathrm{d}t_n$$

Also, $M(0, t) = I_{(c+1) \times (c+1)}$.

Now, clearly from above we can see that $P(n, t)$ can be decomposed using $M(n, t)$ and this results in:

$$P(n, t) = M(n, t)\, e^{Ct}$$

for $n \in \mathbb{Z}_+$. The initial conditions are

$$\begin{aligned} P(0, 0) &= I_{(c+1) \times (c+1)} \\ P(n, 0) &= O \end{aligned}$$

where O is the zero matrix. Similarly, the initial conditions for $M(n, t)$ are:

$$\begin{aligned} M(0, 0) &= I_{(c+1) \times (c+1)} \\ M(n, 0) &= O \end{aligned}$$

where $n \geq 1$

Therefore, we can see that a solution for $P(n,t)$ can be obtained if $M(n,t)$ and e^{Ct} can be explicitly derived.

Recalling, the Baker-Hausdorff lemma we can write, $e^{Xt}Ye^{-Xt}$, for a given $t \in \mathbb{R}$ and two square matrices X and Y as:

$$\begin{aligned} e^{Xt}Ye^{-Xt} &= Y + [X,Y] + \frac{t}{2!}[X,[X,Y]] + \frac{t}{3!}[X,[X,[X,Y]]] + \dots \\ &=_{\text{defn 13.2}} Y + \mathsf{ad}_X Y + \frac{t}{2!}\mathsf{ad}_X^2 Y + \frac{t}{3!}\mathsf{ad}_X^3 Y + \dots \\ & \quad Y + \sum_{n=1}^{\infty} \frac{t^n}{n!}\mathsf{ad}_X^n Y \end{aligned}$$

So, by using the above lemma and the general case of the ad definition i.e.

$$\mathsf{ad}_C^n D = \begin{cases} a^{2k}[C,D] & n = 2k+1 \\ a^{2k}[C,[C,D]] & n = 2k+2 \end{cases} \tag{10}$$

and

$$\mathsf{ad}_C(D) = [C,D] \tag{11}$$

we can deduce the following:

$$\begin{aligned} e^{Ct}De^{-Ct} &= D + t\mathsf{ad}_C D + \frac{t^2}{2!}\mathsf{ad}_C^2 D + \dots + \frac{t^n}{n!}\mathsf{ad}_C^n D + \dots \qquad \text{(definition of B-H lemma)} \\ &= D + \left(t + \frac{t^3}{3!}a^2 + \frac{t^5}{5!}a^4 + \dots\right)[C,D] + \left(\frac{t^2}{2!} + \frac{t^4}{4!}a^2 + \frac{t^6}{6!}a^4 + \dots\right)[C,[C,D]] \qquad \text{(from 10)} \\ &= D + \left(t + \frac{t^3}{3!}a^2 + \frac{t^5}{5!}a^4 + \dots\right)\mathsf{ad}_C D + \left(\frac{t^2}{2!} + \frac{t^4}{4!}a^2 + \frac{t^6}{6!}a^4 + \dots\right)\mathsf{ad}_C^2 D \qquad \text{(from 11)} \\ &= D + \left(\frac{e^{at} - e^{-at}}{2a}\right)\mathsf{ad}_C D + \left(\frac{e^{at} + e^{-at}}{2a^2} - \frac{1}{a^2}\right)\mathsf{ad}_C^2 D \\ &= D + (\sinh(at))\,\mathsf{ad}_C D + \left(\frac{1}{a^2}(\cosh(at) - 1)\right)\mathsf{ad}_C^2 D \end{aligned}$$

So, we have obtained a solution for $e^{Ct}De^{-Ct}$ when $a \neq 0$ and we simplified it using Lie algebra and the Baker-Hausdorff Lemma.

If $a = 0$ (when $c_+ = c_- = 0$ and $\alpha = 0$) then the above equation simplifies to:

$$\begin{aligned} & D + \left(t + \frac{t^3}{3!}a^2 + \frac{t^5}{5!}a^4 + \dots\right)\mathsf{ad}_C D + \left(\frac{t^2}{2!} + \frac{t^4}{4!}a^2 + \frac{t^6}{6!}a^4 + \dots\right)\mathsf{ad}_C^2 D \\ \text{for } a &= 0 \\ \implies & D + (t)\,\mathsf{ad}_C D + \left(\frac{t^2}{2!}\right)\mathsf{ad}_C^2 D \\ &= D + t\mathsf{ad}_C D + \frac{t^2}{2}\mathsf{ad}_C^2 D \end{aligned}$$

In addition, from the above equivalence we can substitute $e^{Ct}De^{-Ct}$ into the definition of $M(n,t)$ and hence we can rewrite $P(n,t)$.

$$\begin{aligned} M(n,t) &= \int_{\Omega_{n,t}} \left(D + (\sinh(at_1))\,\mathrm{ad}_C D + \left(\frac{1}{a^2}(\cosh(at_1) - 1)\right)\mathrm{ad}_C^2 D\right) \\ &\quad \cdot \left(D + (\sinh(at_2))\,\mathrm{ad}_C D + \left(\frac{1}{a^2}(\cosh(at_2) - 1)\right)\mathrm{ad}_C^2 D\right) \\ &\quad \ldots \left(D + (\sinh(at_n))\,\mathrm{ad}_C D + \left(\frac{1}{a^2}(\cosh(at_n) - 1)\right)\mathrm{ad}_C^2 D\right) \mathrm{d}t_1 \mathrm{d}t_2 \ldots \mathrm{d}t_n \end{aligned}$$

Thus,

$$\begin{aligned} P(n,t) &= M(n,t)\, e^{Ct} \\ &= e^{Ct} \int_{\Omega_{n,t}} \left(D + (\sinh(at_1))\,\mathrm{ad}_C D + \left(\frac{1}{a^2}(\cosh(at_1) - 1)\right)\mathrm{ad}_C^2 D\right) \\ &\quad \cdot \left(D + (\sinh(at_2))\,\mathrm{ad}_C D + \left(\frac{1}{a^2}(\cosh(at_2) - 1)\right)\mathrm{ad}_C^2 D\right) \\ &\quad \ldots \left(D + (\sinh(at_n))\,\mathrm{ad}_C D + \left(\frac{1}{a^2}(\cosh(at_n) - 1)\right)\mathrm{ad}_C^2 D\right) \mathrm{d}t_1 \mathrm{d}t_2 \ldots \mathrm{d}t_n \end{aligned}$$

3.3 Example: SU(2)

Taking some inspiration from [14] and the idea of how to specify a general case of the Lie algebra, $\mathfrak{su}(2)$, we have been able to define a second example as shown below.

We can define a MAP for the Lie algebra of the special unitary group, SU(2), as $\mathfrak{su}(2)$ i.e. the group of complex 2 x 2 matrices, say X, such that $X^* = X^{-1}$ and trace$(X) = 0$. We can define a general element in $\mathfrak{su}(2)$ by:

$$X = \begin{pmatrix} ix & -\bar{a} \\ a & -ix \end{pmatrix}$$

where $x \in \mathbb{R}$ and $a \in \mathbb{C}$.

We can denote $U_+^{(c)}$, $U_-^{(c)}$, and $U_3^{(c)}$ as $(c+1)$ x $(c+1)$ matrix representations of the standard generators $\hat{u}_+$, $\hat{u}_-$, $\hat{u}_3$ of $\mathfrak{su}(2)$.

One possible set of standard generators are:

$$\begin{aligned} \hat{u}_+ &= \begin{pmatrix} 0 & i \\ i & 0 \end{pmatrix} \\ \hat{u}_- &= \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \\ \hat{u}_3 &= \begin{pmatrix} i & 0 \\ 0 & -i \end{pmatrix} \end{aligned}$$

These are a form of the Pauli matrices[15].

Hence, we can define the commutator relations as:

$$\begin{aligned}
[\hat{u}_+, \hat{u}_-] &= \begin{pmatrix} 0 & i \\ i & 0 \end{pmatrix}\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} - \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 0 & i \\ i & 0 \end{pmatrix} \\
&= \begin{pmatrix} i & 0 \\ 0 & -i \end{pmatrix} - \begin{pmatrix} -i & 0 \\ 0 & i \end{pmatrix} \\
&= \begin{pmatrix} 2i & 0 \\ 0 & -2i \end{pmatrix} \\
&= 2\begin{pmatrix} i & 0 \\ 0 & -i \end{pmatrix} \\
&= 2\hat{u}_3
\end{aligned}$$

$$\begin{aligned}
[\hat{u}_3, \hat{u}_+] &= \begin{pmatrix} i & 0 \\ 0 & -i \end{pmatrix}\begin{pmatrix} 0 & i \\ i & 0 \end{pmatrix} - \begin{pmatrix} 0 & i \\ i & 0 \end{pmatrix}\begin{pmatrix} i & 0 \\ 0 & -i \end{pmatrix} \\
&= \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} - \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \\
&= \begin{pmatrix} 0 & -2 \\ 2 & 0 \end{pmatrix} \\
&= 2\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \\
&= 2\hat{u}_-
\end{aligned}$$

$$\begin{aligned}
[\hat{u}_-, \hat{u}_3] &= \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} i & 0 \\ 0 & -i \end{pmatrix} - \begin{pmatrix} i & 0 \\ 0 & -i \end{pmatrix}\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \\
&= \begin{pmatrix} 0 & i \\ i & 0 \end{pmatrix} - \begin{pmatrix} 0 & -i \\ -i & 0 \end{pmatrix} \\
&= \begin{pmatrix} 0 & 2i \\ 2i & 0 \end{pmatrix} \\
&= 2\begin{pmatrix} 0 & i \\ i & 0 \end{pmatrix} \\
&= 2\hat{u}_+
\end{aligned}$$

In addition, one of the many 2 x 2 matrix representations of $\hat{u}_+, \hat{u}_-, \hat{u}_3$ can be defined as below:

$$
\begin{aligned}
U_+^{(1)} &= \frac{1}{2}\left(\hat{u}_+ + i\hat{u}_-\right) \\
&= \begin{pmatrix} 0 & 0 \\ i & 0 \end{pmatrix} \\
U_-^{(1)} &= -\frac{1}{2}\left(\hat{u}_+ - i\hat{u}_-\right) \\
&= \begin{pmatrix} 0 & i \\ 0 & 0 \end{pmatrix} \\
U_3^{(1)} &= \frac{1}{2}\hat{u}_3 \\
&= \begin{pmatrix} \frac{i}{2} & 0 \\ 0 & -\frac{i}{2} \end{pmatrix}
\end{aligned}
$$

Hence, we can use the earlier definitions of C and D and replace it with the above matrix representations:

$$
\begin{aligned}
C &= c_+ U_+^{(c)} + c_- U_-^{(c)} - \frac{\alpha}{2} U_3^{(c)} - \frac{\beta}{2} I^{(c)} \\
D &= d_+ U_+^{(c)} + d_- U_-^{(c)} + \frac{\lambda_+ - \lambda_-}{2} U_3^{(c)} + \frac{\lambda_+ + \lambda_-}{2} I^{(c)}
\end{aligned}
$$

where $\alpha = (c_+ + d_+ + \lambda_+) - (c_- + d_- + \lambda_-)$ and $\beta = (c_+ + d_+ + \lambda_+) + (c_- + d_- + \lambda_-)$. As well as c_+, c_- are non-negative real numbers and $d_+, d_-, \lambda_+, \lambda_-$ are also all non-negative real numbers while at least one of them is strictly positive.

Thus, by using the matrix representations of $U_+^{(1)}, U_-^{(1)}, U_3^{(1)}$ we can denote C and D in terms of it:

$$
\begin{aligned}
C &= c_+U_+^{(1)} + c_-U_-^{(1)} - \frac{\alpha}{2}U_3^{(1)} - \frac{\beta}{2}I^{(1)} \\
&= c_+\begin{pmatrix} 0 & 0 \\ i & 0 \end{pmatrix} + c_-\begin{pmatrix} 0 & i \\ 0 & 0 \end{pmatrix} - \frac{\alpha}{2}\begin{pmatrix} \frac{i}{2} & 0 \\ 0 & -\frac{i}{2} \end{pmatrix} - \frac{\beta}{2}I \\
&= \begin{pmatrix} 0 & 0 \\ c_+i & 0 \end{pmatrix} + \begin{pmatrix} 0 & c_-i \\ 0 & 0 \end{pmatrix} - \frac{\alpha}{2}\begin{pmatrix} \frac{i}{2} & 0 \\ 0 & -\frac{i}{2} \end{pmatrix} - \frac{\beta}{2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} 0 & c_-i \\ c_+i & 0 \end{pmatrix} - \left(\frac{(c_+ + d_+ + \lambda_+) - (c_- + d_- + \lambda_-)}{2}\right)\begin{pmatrix} \frac{i}{2} & 0 \\ 0 & -\frac{i}{2} \end{pmatrix} \\
&\quad - \left(\frac{(c_+ + d_+ + \lambda_+) + (c_- + d_- + \lambda_-)}{2}\right) I \\
&= \begin{pmatrix} 0 & c_-i \\ c_+i & 0 \end{pmatrix} - \begin{pmatrix} \frac{((c_+ + d_+ + \lambda_+) - (c_- + d_- + \lambda_-))i}{4} & 0 \\ 0 & -\frac{((c_+ + d_+ + \lambda_+) - (c_- + d_- + \lambda_-))i}{4} \end{pmatrix} \\
&\quad - \begin{pmatrix} \frac{(c_+ + d_+ + \lambda_+) + (c_- + d_- + \lambda_-)}{2} & 0 \\ 0 & \frac{(c_+ + d_+ + \lambda_+) + (c_- + d_- + \lambda_-)}{2} \end{pmatrix} \\
&= \begin{pmatrix} -\frac{((c_+ + d_+ + \lambda_+) - (c_- + d_- + \lambda_-))i}{4} - \frac{(c_+ + d_+ + \lambda_+) + (c_- + d_- + \lambda_-)}{2} & c_-i \\ c_+i & -\frac{((c_+ + d_+ + \lambda_+) - (c_- + d_- + \lambda_-))i}{4} - \frac{(c_+ + d_+ + \lambda_+) + (c_- + d_- + \lambda_-)}{2} \end{pmatrix}
\end{aligned}
$$

$$
\begin{aligned}
D &= d_+U_+^{(1)} + d_-U_-^{(1)} + \frac{\lambda_+ - \lambda_-}{2}U_3^{(1)} + \frac{\lambda_+ + \lambda_-}{2}I^{(1)} \\
&= d_+\begin{pmatrix} 0 & 0 \\ i & 0 \end{pmatrix} + d_-\begin{pmatrix} 0 & i \\ 0 & 0 \end{pmatrix} + \frac{\lambda_+ - \lambda_-}{2}\begin{pmatrix} \frac{i}{2} & 0 \\ 0 & -\frac{i}{2} \end{pmatrix} + \frac{\lambda_+ + \lambda_-}{2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} 0 & 0 \\ d_+i & 0 \end{pmatrix} + \begin{pmatrix} 0 & d_-i \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} \frac{(\lambda_+ - \lambda_-)i}{4} & 0 \\ 0 & -\frac{(\lambda_+ - \lambda_-)i}{4} \end{pmatrix} + \begin{pmatrix} \frac{\lambda_+ + \lambda_-}{2} & 0 \\ 0 & \frac{\lambda_+ + \lambda_-}{2} \end{pmatrix} \\
&= \begin{pmatrix} 0 & d_-i \\ d_+i & 0 \end{pmatrix} + \begin{pmatrix} \frac{(\lambda_+ - \lambda_-)i}{4} + \frac{2\lambda_+ + 2\lambda_-}{4} & 0 \\ 0 & -\frac{(\lambda_+ - \lambda_-)i}{4} + \frac{2\lambda_+ + 2\lambda_-}{4} \end{pmatrix} \\
&= \begin{pmatrix} \frac{(\lambda_+ - \lambda_-)i}{4} + \frac{2\lambda_+ + 2\lambda_-}{4} & d_-i \\ d_+i & -\frac{(\lambda_+ - \lambda_-)i}{4} + \frac{2\lambda_+ + 2\lambda_-}{4} \end{pmatrix} \\
&= \begin{pmatrix} \frac{(\lambda_+ i + 2\lambda_+) + (-\lambda_- i + 2\lambda_-)}{4} & d_-i \\ d_+i & \frac{(-\lambda_+ i + 2\lambda_+) + (\lambda_- i + 2\lambda_-)}{4} \end{pmatrix} \\
&= \begin{pmatrix} \frac{(2+i)\lambda_+ + (2-i)\lambda_-}{4} & d_-i \\ d_+i & \frac{(2-i)\lambda_+ + (2+i)\lambda_-}{4} \end{pmatrix}
\end{aligned}
$$

If we compare the C and D matrices of $\mathfrak{sl}(2,\mathbb{R})$ to the new ones above we can see that the that $\mathfrak{su}(2)$ matrices are more complicated. This is clear as we have both real and complex entries.

Unfortunately, we can see that $\mathfrak{su}(2)$ does not work as sum of the matrices C and D should result in a matrix Q such that:

$$Q = C + D$$

and the the rows of Q must sum to zero. In other words, the diagonal entries of Q must be

equal to the sum of the other elements in the same row. Nonetheless, further research can be done as it is very likely to find the correct rate matrices, C and D. We generated very simple and straightforward commutator relations, so there is definitely scope in generating the required rate matrices.

So, just like the earlier example we can define $P(n,t)$ for the new C and D matrices and similarly we can reconstruct that integral using Lie algebra and rewrite $P(n,t)$ in terms of $M(n,t)$ and e^{Ct} as shown below.

$$P(n,t) = e^{Ct}M(n,t)$$

3.3.1 Investigate SU(3)

We define the Lie algebra of SU(3) as $\mathfrak{su}(3)$ and just as the case for $\mathfrak{su}(2)$ we have to define the generators. However, as we are dealing with 3 x 3 matrices there will be more generators to take into account. In fact, technically, we should have 9 generators, however one condition is imposed on $\mathfrak{su}(3)$ therefore we only have $3^2 - 1 = 8$ generators. This itself makes the process more complicated and it becomes harder to find the right commutator relations in order to find C and D.

Below are the useful standard generators that we would use to obtain our matrix representations:

$$\hat{u}_1 = \begin{bmatrix} 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\hat{u}_2 = \begin{bmatrix} 0 & -\frac{i}{2} & 0 \\ \frac{i}{2} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\hat{u}_3 = \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ 0 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\hat{u}_4 = \begin{bmatrix} 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 \end{bmatrix}$$

$$\hat{u}_5 = \begin{bmatrix} 0 & 0 & -\frac{i}{2} \\ 0 & 0 & 0 \\ \frac{i}{2} & 0 & 0 \end{bmatrix}$$

$$\hat{u}_6 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & 0 \end{bmatrix}$$

$$\hat{u}_7 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -\frac{i}{2} \\ 0 & \frac{i}{2} & 0 \end{bmatrix}$$

$$\hat{u}_8 = \frac{1}{\sqrt{3}} \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

And from this we can define a matrix representation, which are also known as the Gell-Mann matrices[13]:

$$
\begin{aligned}
U_1^{(2)} &= \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\
U_2^{(2)} &= \begin{bmatrix} 0 & -i & 0 \\ i & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\
U_3^{(2)} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\
U_4^{(2)} &= \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \\
U_5^{(2)} &= \begin{bmatrix} 0 & 0 & -i \\ 0 & 0 & 0 \\ i & 0 & 0 \end{bmatrix} \\
U_6^{(2)} &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \\
U_7^{(2)} &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -i \\ 0 & i & 0 \end{bmatrix} \\
U_8^{(2)} &= \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -2 \end{bmatrix}
\end{aligned}
$$

So, we can see that the basic matrix representations originate from the 8 generators and using the relation: $\hat{u}_i = \frac{U_i^{(2)}}{2} \qquad \forall i \in \{1, \ldots, 8\}$.

Furthermore, the generators satisfy the following general commutator relation:

(We return only the final solution as the steps are identical to those in the $\mathfrak{su}(2)$ case)

$$
\begin{aligned}
[\hat{u}_1, \hat{u}_2] &= i \begin{bmatrix} \frac{1}{2} & 0 & 0 \\ 0 & -\frac{1}{2} & 0 \\ 0 & 0 & 0 \end{bmatrix} \\
&= i\hat{u}_3
\end{aligned}
$$

$$
\begin{aligned}
[\hat{u}_1, \hat{u}_3] &= \begin{bmatrix} 0 & -\frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\
&= -i\hat{u}_2
\end{aligned}
$$

$$
\begin{aligned}
[\hat{u}_1, \hat{u}_4] &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} \\ 0 & -\frac{1}{4} & 0 \end{bmatrix} \\
&= \frac{i}{2}\hat{u}_7
\end{aligned}
$$

$$
\begin{aligned}
[\hat{u}_1, \hat{u}_5] &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -\frac{i}{4} \\ 0 & -\frac{i}{4} & 0 \end{bmatrix} \\
&= -\frac{i}{2}\hat{u}_6
\end{aligned}
$$

$$
\begin{aligned}
[\hat{u}_1, \hat{u}_6] &= \begin{bmatrix} 0 & 0 & \frac{1}{4} \\ 0 & 0 & 0 \\ -\frac{1}{4} & 0 & 0 \end{bmatrix} \\
&= \frac{i}{2}\hat{u}_5
\end{aligned}
$$

$$
\begin{aligned}
[\hat{u}_1, \hat{u}_7] &= \begin{bmatrix} 0 & 0 & -\frac{1}{4} \\ 0 & 0 & 0 \\ -\frac{1}{4} & 0 & 0 \end{bmatrix} \\
&= -\frac{1}{2}\hat{u}_4
\end{aligned}
$$

$$
\begin{aligned}
[\hat{u}_1, \hat{u}_8] &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\
&= 0
\end{aligned}
$$

$$
\begin{aligned}
[\hat{u}_2, \hat{u}_3] &= \begin{bmatrix} 0 & \frac{i}{2} & 0 \\ \frac{i}{2} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\
&= i\hat{u}_1
\end{aligned}
$$

$$
\begin{aligned}
[\hat{u}_2, \hat{u}_4] &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} \\ 0 & \frac{1}{4} & 0 \end{bmatrix} \\
&= \frac{1}{2}\hat{u}_6
\end{aligned}
$$

$$
\begin{aligned}
[\hat{u}_2, \hat{u}_5] &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} \\ 0 & -\frac{1}{4} & 0 \end{bmatrix} \\
&= \frac{i}{2}\hat{u}_7
\end{aligned}
$$

$$
\begin{aligned}
[\hat{u}_2, \hat{u}_6] &= \begin{bmatrix} 0 & 0 & -\frac{i}{4} \\ 0 & 0 & 0 \\ -\frac{i}{4} & 0 & 0 \end{bmatrix} \\
&= -\frac{i}{2}\hat{u}_4
\end{aligned}
$$

These are not all the commutator relations, however we can see that they are straightforward to compute and the output from the brackets are useful results. Therefore, now as future work we can complete the list and solve for the rate matrices C and D of a MAP. In order to find useful equations for C and D we will need to generate a lot of test cases in order to find the correct combination of matrix representations.

4 Coded Examples in MATLAB

4.1 Why use MATLAB?

MATLAB is a very useful mathematical programming tool that allows us to easily and quickly compute large calculations that would take very long to compute by hand. Like MATLAB there were other mathematical programming options that we considered such as Mathematica. MATLAB is a straightforward programming tool that has a lot of the functions required to compute integrals, matrices etc. This can help us simplify a lot of the code and allow other readers to understand the concept in simple terms. The two examples that were defined in the previous section were coded into MATLAB in order to see if we could generate the correct output when given the correct input. As mentioned earlier there were two Lie algebra cases that we looked into namely, $\mathfrak{sl}(2,\mathbb{R})$ and $\mathfrak{su}(2)$.

Let us look into some of the functionalities that will be useful when computing the two examples. The key functionality that we will heavily use is the idea of accessing individual elements within a matrix. This is done using the following manner:

First define a 2 x 2 matrix A as

$$\begin{aligned} \mathsf{A} &= [1,2;3,4] \\ &\equiv \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \end{aligned}$$

Then we can easily access the first element of the matrix by typing $\mathsf{A}(1,1)$, the second element by typing $\mathsf{A}(1,2)$, and so on. So, it is clear that $\mathsf{A}(x,y)$ returns the element in the x^{th} row and y^{th} column. This is particularly useful for the first example where we have a general equation that creates the generating matrices $E_+^{(c)}, E_-^{(c)}, E_3^{(c)}$ as it is dependent on what the user enters as c. In addition, the use of for loops and recursion will simplify a lot of the code and we can find solutions for general cases too as we do not have to hard code any data.

Furthermore, simplify is a function that takes any symbolic function (i.e. with variables) and reduces it into its simplest form. Sometimes, equations can become very complex and reducing them by hand can become very difficult. Thus, this functionality will allow us to re-write complicated expressions into those that are understandable.

The Kronecker Delta is not defined by MATLAB, hence we had to define it using a for loop, which would return 1 when $i = j$ and 0 otherwise i.e. return 1 for the diagonal entries of a matrix. This was a simple function to code and we will need to call the Kroncker Delta function in the examples.

4.2 Example: $\mathfrak{sl}(2,\mathbb{R})$

In order to compute this example, we clearly have to create the three matrices $E_+^{(c)}, E_-^{(c)}, E_3^{(c)}$ that have a specific definitions. Below is a reminder of those definitions:

$$\begin{aligned}\left(E_+^{(c)}\right)_{i,j} &= (c-i)\,\delta_{i,j-1}\\ \left(E_-^{(c)}\right)_{i,j} &= i\delta_{i,j+1}\\ \left(E_3^{(c)}\right)_{i,j} &= (c-2i)\,\delta_{i,j}\end{aligned}$$

This can be expressed in MATLAB in a very straightforward manner. The user inputs a value C, which is the size of the output matrix say $C = 2$ then we will return a 2×2 matrix for $E_+^{(c)}, E_-^{(c)}, E_3^{(c)}$, and in this case c will be defined as $c = C - 1$. Therefore, 2×2 matrices correspond to $E_+^{(1)}, E_-^{(1)}, E_3^{(1)}$ which are generated through MATLAB and these are shown below:

```
Ep =

     0     1
     0     0


Em =

     0     0
     1     0


E3 =

     1     0
     0    -1
```

When we compare this with the results we get from the computation we did by hand, it is clear that our MATLAB output gives us the identical result for all three matrices.

From this output, we can define matrix C (state transitions that have no arrivals) and D (state transitions that have arrivals) by using the definition that we mentioned in the previous chapter. Hence, it is quite straightforward to code the algorithm once we have the Ep, Em, E3 matrices defined and computed. By substituting the Ep, Em, E3 matrices into the definition of C and D, we get the following results:

```
C =

[ - cp - dp - xp,                cp]
[                cm, - cm - dm - xm]


D =

[ -xp,   dp]
[   dm, -xm]
```

Note that c_+=cp, c_-=cm, d_+=dp, d_-=dm, λ_+=xp, λ_-=xm and these have been created as symbolic variables that can be replaced with appropriate values to return matrices, C and D.

Now, we can define a solution of $P(n,t)$ for a MAP $\{N(t), J(t)\}$ with the representing matrices C and D that we have defined above. Calculating this by hand can be very tedious and there are more chances of making errors, hence coding the integral algorithm into MATLAB will allow us to generate a solution more efficiently. There are various ways to integrate expressions in MATLAB, the two functions already defined are described below:

1. The int() function takes in a symbolic expression and if there are multiple symbolic variables then the user can define which variable we will integrate with respect to. If applicable, we can also define the limits to the integral.

   (a) Unfortunately, the integral that we are working with is a bit more complicated and it is not a simple expression that can be evaluated with int() as there are multiple recursions and different parameters that determine the outcome of the integral.

2. The second function is quad(), this can numerically evaluate an integral using the Simpson adaptive quadrature method can handle more complex expressions. Specifically, it can evaluate the integral of functions by using the approximated quadrature rules. These quadrature rules break the integral into subintervals and then approximate it as a sum that can be solved. So, it uses a method of recursion to break the integral into smaller integrals that we can approximate and then sum up to get the final result.

The quad () function is the best option, but rather than utilising the in built function it seemed more clear and appropriate to implement a new function that would cater to the functionality of MAPs and the result we hope to acquire. The aim of this MATLAB code is to simplify the matrices that we compute for $P(n,t)$ and see if we can find a commutator relation i.e. $[A,B] = AB - BA$ relation for any two matrices A, B. This is the key connection between Lie algebra and MAPs, thus with this link we can easily simplify the integral and write higher order terms in relation to the lower order one as there would be a pattern emerging.

The algorithm to calculate the integral is broken down and explained below:

There are four parameters that are required in order to calculate $P(n,t)$, these include:

- MAP - the Markovian arrival process (MAP) that we will be using i.e. the matrix that has no arrivals, C and the matrix that has a state of arrivals, D.
- n - the number of arrivals
- t - the time frame within which the arrivals should occur
- tsteps - this variable determines how many times, t, has to be *split* into as the integral calculates each state and then computes the result for the next state based on the preceded one.

As seen in the previous chapter we can write $P(n,t)$ as $P(n,t) = e^{Ct}M(n,t)$ where $M(n,t) = \int_{\Omega_{n,t}} \left(e^{Ct_1}De^{-Ct_1}\right)\left(e^{Ct_2}De^{-Ct_2}\right)\ldots\left(e^{Ct_n}De^{-Ct_n}\right) \mathrm{d}t_1\mathrm{d}t_2\ldots\mathrm{d}t_n$ and this is easier to compute into MATLAB as we can use recursion. We will compute $M(n,t)$ and then multiply it with e^{Ct} at the end in order to get the final result, $P(n,t)$. The integration will take a very long time to compute as each step is based on the previous result. Hence, in this report we have focused on 2 x 2 matrices as this will allow us to work with smaller but useful examples.

We have already calculated our C and D matrices that will be used as input for the variable MAP. In addition, we define the base case for n as $n = 0$ where it would return only e^{Ct} as

there are no arrivals. Thus, in every case we would have e^{Ct} in the solution so we need to define a separate case for when $n = 1$ and make that our base case to ensure that we return the correct solution. In our model we would like to try and get results for when there are arrivals i.e. when $n \neq 0$ hence we make our base case $n = 1$ to ensure that we get our results to include cases with matrix, D. With higher levels of n the program is going to be much slower as the recursion is based on n and we must ensure to calculate the entire probability of n arrivals within the given time, t.

The time variable, t, is also user defined because it allows us to specify a the time frame for which we must calculate the probability of arrivals. However, with large time values the program runs slow as the probability is harder to calculate due to the higher chances of arrival events. Hence, the matrix becomes more complex and as a result so is the integral.

And finally the tsteps variable is key as it breaks down the time, t, into segments that are used to calculate each $\left(e^{Ct_i} D e^{-Ct_i}\right) \quad \forall i \in \{1 \dots n\}$ and it is used to determine the next time frame.

Note: that there are only n segments so once we reach $n = 1$, we have hit the base case. This is the final time that we will 'cut' in order to calculate the result.

The way the final program works is it takes the entire time, t, and starts to segment it during every for loop iteration this way it gets smaller and smaller until it reaches $n = 1$ where we have the segment t_1 which will be the smallest time frame. In some sense, we are working backwards in order to calculate the final integral.

In this case, we can have two separate parameters for matrix C and D, or we can use the $\{\}$ to *extract* the different matrices C and D from the MAP definition. Below is a sample of how we would separate the C and D matrices:

$$C = \begin{pmatrix} c_+ - d_+ - \lambda_+ & c_+ \\ c_- & -c_- - d_- - \lambda_- \end{pmatrix}$$
$$D = \begin{pmatrix} \lambda_+ & d_+ \\ d_- & \lambda_- \end{pmatrix}$$

```
1  C =
2
3  [ - cp - dp - xp,                cp]
4  [               cm, - cm - dm - xm]
5
6
7  D =
8
9  [ -xp,   dp]
10 [   dm, -xm]
11
12 MAP = {(cp - dp - xp, cp ; cm, - cm - dm - xm) (- xp, dp ; dm, -
       xm)}
13
14 % Extract the relevenat matrices to use for the integral
15 C = MAP{1}
16 D = MAP{2}
```

This format of defining gives us the opportunity to reduce one parameter and it gives a more clear understanding as the MAP contains both C and D. Hence, coupling them together and then separating them allows us to concisely use them for the integral. In addition, MATLAB allows us to call results from other functions hence the output of C and D can be called without hard coding it. Therefore, even if we replace C and D then it will automatically change those matrices in our calculation of $P(n,t)$.

However, the output for the small cases for n are also very large even before simplification. We did a few runs using the following parameters:

```
Run 1:

MAP = {C,D}; t = 3; n = 1; tsteps = 2;

Run 2:

MAP = {C,D}; t = 3; n = 5; tsteps = 2;
```

These are just two of the test runs that we performed on our code, however the simplification was still very long and MATLAB was unable to output the entire result. This limitation in MATLAB does not allow us to display our output, however if the code is run the user can clearly see that once the entire matrix is simplified we can generate the correct output as suggested in [8].

In the future, it would be suitable to try and code this section using a different programming language in order to take advantage of other simplification options. Nonetheless, even though the matrices are large we must still do some of the simplification by hand, i.e. defining the matrices that the solution must be broken down into.

5 Estimation-Maximisation (EM) For MAPs

(**Note:** Estimation-Maximisation algorithm is also known as the Expectation-Maximisation algorithm)

As mentioned earlier, MAPs are widely used in communication systems and measuring network traffic. The key issue is to estimate the parameters for MAPs and other stochastic models. There are two different ways to approach this, one way is to use the moment-based approach and the second option is to use the likelihood-based approach. [12]

In the moment-based approach, we determine the model parameters that are associated with the MAP in order to fit the real moments from the observed data with the theoretical ones. The advantage of using moments rather than the maximum likelihood (ML) approach is to reduce computational costs. MAPs have many parameters, hence it is harder to find the ML from just the data. Hence, we require the ML estimation principle to maximise the likelihood that the observed data has occurred. However, a negative comment about ML estimates of MAPs is that it requires us to work with large and computationally intensive matrices.

5.1 General EM algorithm

In general, the EM algorithm consists of a set, $\mathbf{X}$, which includes all the observed data. Secondly, $\mathbf{Z}$ is the set of missing or unobserved data with a vector, θ, that holds information about unknown parameters.

Along with the above information we need the likelihood function such that $\mathcal{L}(\theta; \mathbf{X}, \mathbf{Z}) = P(\mathbf{X}, \mathbf{Z}|\theta)$. Using all this data we can define the ML estimation of the unknown parameters by the marginal likelihood of the observed data:

$$\begin{aligned} \mathcal{L}(\theta; \mathbf{X}) &= P(\mathbf{X}|\theta) \\ &= \sum_{\mathbf{Z}} P(\mathbf{X}, \mathbf{Z}|\theta) \end{aligned}$$

To find the ML estimation of the marginal likelihood we need to iteratively apply the following two steps:

Estimation (E step):

We must calculate the log likelihood of the function, with respect to $\mathbf{Z}$ given $\mathbf{X}$ which is under the current estimate of the parameters $\theta^{(t)}$. This is written as:

$$Q\left(\theta|\theta^{(t)}\right) = E_{\mathbf{Z}|\mathbf{X},\theta^{(t)}}\left(\log\mathcal{L}(\theta; \mathbf{X}, \mathbf{Z})\right)$$

Maximisation (M step):

We must find the θ (unknown parameter) that maximises the above expectation step. This is defined as:

$$\theta^{(t+1)} = \arg\max_{\theta} Q\left(\theta|\theta^{(t)}\right)$$

where $\theta^{(t)}$ is the estimate of the parameters at time or iteration, t.[16]

5.2 EM and MAPs

As seen in the previous section, we discussed the EM algorithm and the two key steps in calculating it for various situations. Hence, now we can apply this EM algorithm to MAPs in order to generate a solution when the phases are not observable i.e. the sample used is not complete.

So, to apply the EM algorithm in order to get the ML estimation we must first define some parameters. Using the same variables as in section 5.1 we can define $\mathbf{X}$ as the observed data and $\mathbf{Z}$ as the unobserved data. Hence, we want to estimate the set of parameters, θ, using the observed data, $\mathbf{X}$. We will be following and examining the algorithm mentioned in [12] as well as taking some further information from [17]. Below, we will break down the estimation and maximisation steps in order to use it for MAPs:

1. Calculating the estimated log-likelihood function for the data pair $(\mathbf{X}, \mathbf{Z})$ given that $\mathbf{X}$ is the only observed data.
2. Computing θ (parameters) that maximises the log-likelihood function.

Combining the two steps (1 and 2) we can define the following equation:

$$\theta \quad := \quad \text{argmax}_\theta \mathsf{E}_{\mathbf{Z}} \left(\mathcal{L} \left(\theta | \mathbf{X}, \mathbf{Z} \right) | \mathbf{X} \right) \tag{12}$$

where $\mathsf{E}_{\mathbf{Z}}$ is the expectation of the unobserved data, $\mathbf{Z}$ and $\mathcal{L}$ is the log-likelihood function.

Therefore, we can compute the expected $\mathcal{L}$ by using an initial set of parameters, θ which is defined in equation 12. In addition, equation 12 updates the parameters, θ, based on the previously calculated parameters. In other words, this is a recursive function or an iterative method that continues to update the parameters until it converges towards a certain threshold value.

In order to apply the EM algorithm, it would be useful to introduce a generalised data format for the observed data that includes the arrival time of the data and other necessary information related to the data (usual group data). Information that we can gain from the observed data is the time within a certain period and the number of arrivals that are observed in the given period. An additional variable that we must add to the group is an indicator variable (a variable that takes a value of 0 or 1), which returns a 1 if an arrival occurs at the end of the observation period. We can concisely write this as:

$$\mathbf{X}_G \quad := \quad \{(t_1, x_1, a_1), \ldots, (t_K, x_K, a_K)\}$$

where each bracket includes the three main variables described above and the subscripts range from $1, \ldots, K$ which identify the observation period we are in. Hence, t_k is the time of the k^{th} period and x_k is the number of arrivals in the k^{th} period. Finally, a_k is the indicator variable for the event that an arrival occurs at the end of the k^{th} observation period.

From the above variables we can also calculate a few other facts that relate to the EM algorithm:

1. Define s_k as the sum of the times for the first k periods, i.e.

$$s_k \quad = \quad \sum_{i=1}^{k} t_i$$

2. The time intervals can be written in terms of the s_k's. For instance, x_k will be the number of arrivals within the time interval (s_{k-1}, s_k).
3. $a_k = 1 \implies$ an arrival occurred at time, s_k.
4. $a_k = 0 \implies$ no arrival occurred at time, s_k.
5. Total number of arrivals within the time interval $(s_{k-1}, s_k]$ is clearly $x_k + a_k$ (combining 2,3,4).
6. $a_k = 0 \quad \forall k \in \{1, \ldots, K\}$ implies we get the data for the usual data group.
7. $a_k = 1$ and $x_k = 0 \quad \forall k \in \{1, \ldots, K\}$ implies we get the data for the arrival date group.

Let us recall a few other facts from the definition of MAPs, which we will use as well as build on. MAPs is a counting process whose arrival rate is based on a continuous-time Markov chain (CTMC). Then we define C and D as m x m matrices such that C is the matrix that defines state transitions without arrival events and D represents state transitions with arrival events.

Denote matrices C and D as:

$$C = \begin{bmatrix} -\mu_{1,1} & \mu_{1,2} & \cdots & \mu_{1,m} \\ \mu_{2,1,} & -\mu_{2,2} & \cdots & \mu_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{m,1} & \mu_{m,2} & \cdots & -\mu_{m,m} \end{bmatrix}$$

$$D = \begin{bmatrix} \lambda_{1,1} & \lambda_{1,2} & \cdots & \lambda_{1,m} \\ \lambda_{2,1,} & \lambda_{2,2} & \cdots & \lambda_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{m,1} & \lambda_{m,2} & \cdots & \lambda_{m,m} \end{bmatrix}$$

C is also a generator of the underlying CTMC (when there are no arrivals), so $\mu_{i,i} = \sum_{j=1, j\neq i}^{m} \mu_{i,j} + \sum_{j=1}^{m} \lambda_{i,j}$ and D is a rate matrix.

In addition, MAPs include two stochastic processes $N(t)$ and $J(t)$. Just as a reminder these processes are defined as:

1. $\{N(t); t \geq 0\}$ is a stochastic process, which shows the number of arrivals during the time interval $[0, t)$.
2. $\{J(t); t \geq 0\}$ is a stochastic process, which indicates the state at time t.

From this definition of MAPs, we can further define π as the initial steady state (probability) vector, which determines the initial state of the underlying Markov chain, $\{J(t)\}$. We can write π as $(\pi_1, \ldots, \pi_m)$ such that $\sum_{i=1}^{m} \pi_i = 1$.

Moreover, from our example in chapter 3 we defined the following matrix:

$$(P(n,t))_{i,j} = \mathsf{P}(N(t) = n, J(t) = j | N(0) = 0, J(0) = i)$$

From this we can now write the differential-difference equations:

$$\begin{aligned} \tfrac{d}{dt}P(0,t) &= P(0,t)\,C \\ \tfrac{d}{dt}P(n,t) &= P(n,t)\,C + P(n-1,t)\,D \qquad \text{for } n = 1,2,\ldots \end{aligned} \tag{13}$$

where $t \in \mathbb{R}_+$ with initial conditions $P(0,0) = I$ and $P(n,0) = O$ with I is the identity and O is the zero matrix.

5.2.1 Formulas for the M-Step

We must first define a few new variables for the unobserved data:

- B_i: an indicator random variable for the event that the current state is i at time, $t = 0$
- $Y_{i,j}^{[k]}$: an indicator random variable for the event when an arrival has state transitions between i and j at time, s_k
- $Z_i^{[k]}$: total sojourn time (i.e. expected time spent within the state) for phase i during the time interval, (s_{k-1}, s_k)
- $M_{i,j}^{[k]}$: total number of state transitions between i and j during the time interval, (s_{k-1}, s_k) where there are no arrivals
- $W_{i,j}^{[k]}$: total number arrivals that lead to a state transition from i to j during the interval, (s_{k-1}, s_k)

Then we denote $I(\cdot)$ as an indicator function, i.e. a function defined on a set of elements X and if we take a subset of X, say A then the indicator function returns 1 for all the elements that are in A and 0 otherwise.

Hence, we can write the unknown data variables by using the definition of MAPs:

- $B_i = I(J(0) = i)$
- $Y_{i,j}^{[k]} = I\left(J\left(s_k^-\right) = i,\, J\left(s_k^+\right) = j\right)$
- $Z_i^{[k]} = \int_{s_{k-1}}^{s_k} I(J(\tau) = i) \cdot \mathrm{d}\tau$
- $M_{i,j}^{[k]} = \int_{s_{k-1}}^{s_k} I(J(\tau^-) = i,\, J(\tau^+) = j) \cdot \mathrm{d}\tau \qquad$ where $i \neq j$

where τ^- and τ^+ are the left and right limits respectively, i.e

$$I\left(N\left(\tau^-\right) = x,\, N\left(\tau^+\right) = y\right) = \lim_{\Delta t \to +0} I(N(\tau - \Delta t) = x,\, N(\tau + \Delta t) = y)$$

Let the parameter set be defined as:

$$\theta := \{\pi_i, \mu_{i,j}, \lambda_{i,j}\}$$

where $i, j = 1, \ldots, m$.

Denote the unobserved data using the new defined variables:

$$\mathbf{Z} \quad := \quad \left\{ B_i, Y_{i,j}^{[k]}, Z_i^{[k]}, M_{i,j}^{[k]}, W_{i,j}^{[k]} \right\}$$

where $i, j = 1, \ldots, m$ and $k = 1, \ldots, K$.

The variables $\mu_{i,j}$ and $\lambda_{i,j}$ are the rates of the exponential distributions that represent the state transitions and arrivals in a MAP. Therefore, if we have the complete date i.e. $(\mathbf{X}_G, \mathbf{Z})$ then we can write the ML estimation for the parameters defined in θ:

$$\begin{aligned} \hat{\pi_i} &= B_i \\ \hat{\mu_{i,j}} &= \frac{\sum_{k=1}^{K} M_{i,j}^{[k]}}{\sum_{k=1}^{K} Z_i^{[k]}} \\ \hat{\lambda_{i,j}} &= \frac{\sum_{k=1}^{K} \left(W_{i,j}^{[k]} + Y_{i,j}^{[k]} \right)}{\sum_{k=1}^{K} Z_i^{[k]}} \end{aligned}$$

Recall, $\lambda_{i,j}$ are variables that correspond to the rate matrix D i.e. the case where an arrival leads to a potential state change of the continuous-time Markov chain. So, we know that we can have an arrival within the time interval (s_{k-1}, s_k), which relates to $W_{i,j}^{[k]}$. However, we also have to take into account the case when an arrival occurs at time s_k and causes a state transition, which relates to the indicator random variable, $Y_{i,j}^{[k]}$. Therefore, we have two variables for the ML estimation of $\lambda_{i,j}$.

Now, taking the above ML estimations and equation 12 we can write the M-step formulas (the update equation for the parameter, θ, in order to find the maximum) of the EM algorithm for a MAP:

$$\begin{aligned} \pi_i &= E\left[B_i | \mathbf{X}_G\right] \\ \mu_{i,j} &= \frac{\sum_{k=1}^{K} E\left[M_{i,j}^{[k]} | \mathbf{X}_G\right]}{\sum_{k=1}^{K} E\left[Z_i^{[k]} | \mathbf{X}_G\right]} \\ \lambda_{i,j} &= \frac{\sum_{k=1}^{K} \left(E\left[W_{i,j}^{[k]} | \mathbf{X}_G\right] + E\left[Y_{i,j}^{[k]} | \mathbf{X}_G\right] \right)}{\sum_{k=1}^{K} E\left[Z_i^{[k]} | \mathbf{X}_G\right]} \end{aligned} \tag{14}$$

5.2.2 Formulas for the E-step

We denote the following indicator random variable:

$$\mathcal{A}_k \quad = \quad I\left(N\left(s_k^+\right) - N\left(s_k^-\right) = 1\right)$$

Then we can define the following:

- Forward event: $\mathcal{F}_k = \mathcal{A}_1, \ldots, \mathcal{A}_k$
- Backward event: $\mathcal{B}_k = \mathcal{A}_k, \ldots, \mathcal{A}_K$
- Overall event: $\mathcal{O}_k = \mathcal{A}_1, \ldots, \mathcal{A}_K$

Let A be any indicator random variable, then we can write $\mathsf{P}(A) = \mathsf{P}(A = 1)$ as the probability of the indicator variable, A.

Furthermore, we can let $\underline{f_k}(u)$ and $\underline{b_k}(u)$ be row and column vectors that represent the likelihoods from the forward and backward events, respectively, during the interval (s_{k-1}, s_k).

Now, the i^{th} element of both the vectors are written as:

$$\begin{aligned}\left[\underline{f_k}(u)\right]_i &= P\left(\mathcal{F}_{k-1}, N\left((s_{k-1}+u)^-\right) - N\left(s_{k-1}^+\right) = 0, J\left((s_{k-1}+u)^-\right) = i\right) \\ \left[\underline{b_k}(u)\right]_i &= P\left(N\left(s_k^-\right) - N\left((s_k-u)^+\right) = 0, \mathcal{A}_k, \mathcal{B}_{k+1} | J\left((s_k-u)^+\right) = i\right)\end{aligned}$$

Using the equations described in 14 as well as the indicator random variable, $\mathcal{O}$, then we can write:

$$\begin{aligned}\pi_i &= \frac{E\left[B_i \mathcal{O}\right]}{P(\mathcal{O})} \\ &= \frac{\pi_i \left[b_1(t_1)\right]_i}{\pi b_1(t_1)}\end{aligned}$$

It is more difficult to understand and compute, $E\left[M_{i,j}^{[k]} | \mathbf{X}_G\right]$, $E\left[Y_{i,j}^{[k]} | \mathbf{X}_G\right]$ and $E\left[Z_i^{[k]} | \mathbf{X}_G\right]$.

Let us begin by first computing $E\left[M_{i,j}^{[k]} | \mathbf{X}_G\right]$ as $E\left[W_{i,j}^{[k]} | \mathbf{X}\right]$ can then be computed in a similar manner.

So, in the same way as we wrote π_i, we can formulate $E\left[M_{i,j}^{[k]} | \mathbf{X}_G\right]$ as:

$$E\left[M_{i,j}^{[k]} | \mathbf{X}_G\right] = \frac{E\left[M_{i,j}^{[k]} \mathcal{O}\right]}{P(\mathcal{O})}$$

We can determine $E\left[M_{i,j}^{[k]} \mathcal{O}\right]$ if we are given the Markov process, $J(t)$ and the independent increments of $N(t)$ (i.e. the increments of the arrivals within the time interval) then we get:

$$\begin{aligned}E\left[M_{i,j}^{[k]} \mathcal{O}\right] &= \int_{s_{k-1}}^{s_k} P\left(J\left(\tau^-\right) = i, J\left(\tau^+\right) = j, N\left(\tau^+\right) - N\left(\tau^-\right) = 0, \mathcal{O}\right) \cdot \mathsf{d}\tau \quad (15) \\ &= \int_{s_{k-1}}^{s_k} P\left(\mathcal{F}_{k-1}, N\left(\tau^-\right) - N\left(s_{k-1}^+\right) = 0, J\left(\tau^-\right) = i\right) \\ &\quad \cdot P\left(J\left(\tau^+\right) = j, N\left(\tau^+\right) - N\left(\tau^-\right) = 0 | J\left(\tau^-\right) = i\right) \\ &\quad \cdot P\left(N\left(s_k^-\right) - N\left(\tau^+\right) = 0, \mathcal{A}_k, \mathcal{B}_{k+1} | J\left(\tau^+\right) = j\right) \cdot \mathsf{d}\tau\end{aligned}$$

We can write the above in terms of $\underline{f_k}(u)$ and $\underline{b_k}(u)$:

$$E\left[M_{i,j}^{[k]}\mathcal{O}\right] = \int_0^{t_k} \left[\underline{f_k}(\tau)\right]_i \mu_{i,j} \left[\underline{b_k}(t_k-\tau)\right]_j \cdot \mathrm{d}\tau \tag{16}$$

Furthermore, we can formulate the equation for $E\left[Z_i^{[k]}|\mathbf{X}\right]$ using 16 and substituting the j for i:

$$E\left[Z_i^{[k]}\mathcal{O}\right] = \int_0^{t_k} \left[\underline{f_k}(\tau)\right]_i \left[\underline{b_k}(t_k-\tau)\right]_i \cdot \mathrm{d}\tau$$

In a similar way, we can derive the expected value for $W_{i,j}^{[k]}$:

$$\begin{aligned} E\left[W_{i,j}^{[k]}\mathcal{O}\right] = \int_{s_{k-1}}^{s_k} & P\left(\mathcal{F}_{k-1}, N\left(\tau^-\right) - N\left(s_{k-1}^+\right) = 0, J\left(\tau^-\right) = i\right) \\ & \cdot P\left(J\left(\tau^+\right) = j, N\left(\tau^+\right) - N\left(\tau^-\right) = 1 | J\left(\tau^-\right) = i\right) \\ & \cdot P\left(N\left(s_k^-\right) - N\left(\tau^+\right) = 1, \mathcal{A}_k, \mathcal{B}_{k+1} | J\left(\tau^+\right) = j\right) \cdot \mathrm{d}\tau \end{aligned} \tag{17}$$

Hence, we can write the following:

$$E\left[W_{i,j}^{[k]}\mathcal{O}\right] = \int_0^{t_k} \left[\underline{f_k}(\tau)\right]_i \lambda_{i,j} \left[\underline{b_k}(t_k-\tau)\right]_j \cdot \mathrm{d}\tau$$

The key difference between equation 15 and 17 is a check to see if an arrival occurs at time, τ or not.

Hence, we also have the a $\mu_{i,j}$ in 15 and by definition $M_{i,j}^{[k]}$ includes those cases where there are no arrivals between state transitions.

(**Note:** $\mu_{i,j}$ is a variable of rate matrix, C, where there are no arrivals)

In contrast, we have $\lambda_{i,j}$ in 17 and by definition $W_{i,j}^{[k]}$ includes those cases where there are are arrivals causing possible state transitions.

(**Note:** $\lambda_{i,j}$ is a variable of rate matrix, D, where there are arrivals leading to possible state transitions)

Finally, we can also derive the expected value for $Y_{i,j}^{[k]}$:

$$\begin{aligned} E\left[Y_{i,j}^{[k]}\mathcal{O}\right] = & P\left(\mathcal{F}_{k-1}, N\left(s_k^-\right) - N\left(s_{k-1}^+\right) = 0, J\left(s_k^-\right) = i\right) \\ & \cdot P\left(J\left(s_k^+\right) = j, N\left(s_k^+\right) - N\left(s_k^-\right) = 1 | J\left(s_k^-\right) = i\right) \\ & \cdot P\left(\mathcal{B}_{k+1} | J\left(s_k^+\right) = j\right) \end{aligned}$$

Thus, we can simplify the above to:

$$E\left[Y_{i,j}^{[k]}\mathcal{O}\right] = \left[\underline{f_k}(t_k)\right]_i \lambda_{i,j} \left[\underline{b_{k+1}}(t_{k+1})\right]_j$$

This completes all the formulas for the E-step and we can now apply them.

5.2.3 Computing the EM algorithm

There are two distinct ways of computing the EM algorithm of MAPs, one way is to use differential equations and the second way is to use uniformisation. Each way has its advantages and disadvantages, but we will have a quick overview of the two algorithms as well as compare their differences.

First, we will understand how to apply EM algorithm using the **differential equations method**.

We need to compute the vectors $\underline{f_k}(u)$ and $\underline{b_k}(u)$ as well as their convolutions.

Thus, from their definitions we can express the vectors $\underline{f_k}(u)$ and $\underline{b_k}(u)$ as below:

$$\begin{aligned} \underline{f_k}(u) &= \pi \exp(Ct_1) D \times \cdots \times \exp(Ct_{k-1}) D \times \exp(Cu) \\ \underline{b_k}(u) &= \exp(Ct) D \times \cdots \times \exp(Ct_K) De \end{aligned} \tag{18}$$

The differential equations for $\underline{f_k}(u)$ and $\underline{b_k}(u)$ are then directly obtained from C and D. In order for us to derive these equations we need to rewrite $\underline{f_k}(u)$ and $\underline{b_k}(u)$ as $\underline{f_k}(n,u)$ and $\underline{b_k}(n,u)$, i.e. the general case and they are defined below:

$$\begin{aligned} \left[\underline{f_k}(n,u)\right]_i &= P\left(\mathcal{F}_{k-1}, N\left((s_{k-1}+u)^-\right) - N\left(s_{k-1}^+\right) = n, J\left((s_{k-1}+u)^-\right) = i\right) \\ \left[\underline{b_k}(n,u)\right]_i &= P\left(N\left(s_k^-\right) - N\left((s_k-u)^+\right) = n, \mathcal{A}_k, \mathcal{B}_{k+1} | J\left((s_k-u)^+\right) = i\right) \end{aligned}$$

Hence, if we have $\underline{f_k}(n,u)$ and $\underline{b_k}(n,u)$ we can then apply the same differential-difference method as we did for equation 13. As a result, we will have the following new equations for $\underline{f_k}(n,u)$:

$$\begin{aligned} \tfrac{d}{du}\underline{f_k}(0,u) &= \underline{f_k}(u) \\ &= \underline{f_k}(0,u) C \\ \tfrac{d}{du}\underline{f_k}(n,u) &= \underline{f_k}(n,u) C + \underline{f_k}(n-1,u) D \end{aligned} \tag{19}$$

where $n = 1, 2, \ldots, x_k$ and $x_k = N\left(s_k^-\right) - N\left(s_{k-1}^+\right)$.

Similarly, we have the following equations for $\underline{b_k}(n,u)$:

$$\begin{aligned} \tfrac{d}{du}\underline{b_k}(0,u) &= \underline{b_k}(u) \\ &= \underline{b_k}(0,u) C \\ \tfrac{d}{du}\underline{b_k}(n,u) &= \underline{b_k}(n,u) C + \underline{b_k}(n-1,u) D \end{aligned} \tag{20}$$

where $n = 1, 2, \ldots, x_k$ and $x_k = N\left(s_k^-\right) - N\left(s_{k-1}^+\right)$.

From the equations 19 and 20 we can see that the relevant equations are the first set of formulas in both $\underline{f_k}$ and $\underline{b_k}$.

And the initial conditions are:

1. $\underline{f_1}(0) = \underline{\pi}$
2. $\underline{f_k}(0) = \underline{f_{k-1}}(t_{k-1}) D$
3. $\underline{f_k}(0) = \underline{0} \quad$ for $k \geq 2$
4. $\underline{b_1}(0) = \underline{\pi}$
5. $\underline{b_k}(0) = \underline{b_{k-1}}(t_{k-1}) D$
6. $\underline{b_k}(0) = \underline{0} \quad$ for $k \geq 2$

Finally, based on the equations defined by 18 we can argue that $\underline{f_k}(u)$ and $\underline{b_k}(u)$ can be computed in a forward-backward manner. Therefore, another m x m matrix, $H_k(u)$ is defined as the convolution of $\underline{f_k}(u)$ and $\underline{b_k}(u)$. This matrix is needed to derive the computation method of the expected values. $H_k(u)$ is defined as:

$$H_k(u) = \int_0^u \underline{b_k}(u-\tau)\,\underline{f_k}(\tau)\,\mathrm{d}\tau$$

Just as we defined the general cases for $\underline{f_k}(u)$ and $\underline{b_k}(u)$, we do the same for $H_k(u)$ i.e. $H_k(n,u)$ and its differential equations are:

$$\begin{aligned} \tfrac{d}{du} H_k(0,u) &= C \cdot H_k(0,u) + \underline{b_k}(0,0)\,\underline{f_k}(0,u) \\ \tfrac{d}{du} H_k(n,u) &= C \cdot H_k(n,u) + D \cdot H_k(n-1,u) + \underline{b_k}(0,0)\,\underline{f_k}(n,u) \end{aligned}$$

where $n = 1, 2, \ldots, x_k$ and $x_k = N\left(s_k^-\right) - N\left(s_{k-1}^+\right)$.

The initial condition is $H_k(n,0) = O$ where O is the zero matrix and $n = 0, \ldots, x_k$.

Hence, this completes the first method that we can use to solve the EM algorithm for MAPs.

The second method we will investigate uses **uniformisation** to find the formulas for $\underline{f_k}(t)$ and $\underline{b_k}(t)$ as well as $H_k(t)$.

Just like earlier we define the $\underline{f_k}(t)$ and $\underline{b_k}(t)$ as:

$$\begin{aligned} \underline{f_k}(t) &= \pi \exp(Ct_1) D \times \cdots \times \exp(Ct_{k-1}) D \times \exp(Ct) \\ \underline{b_k}(t) &= \exp(Ct) D \times \cdots \times \exp(Ct_K) De \end{aligned}$$

Then we let q be a constant that is higher than the absolute value of the maximum diagonal element of matrix, C i.e. $q > \max_i |\mu_{i,i}|$. Then we can write:

$$\exp(Ct) = \sum_{y=0}^{\infty} e^{-qt} \frac{(qt)^y}{y!} \left(I + \frac{C}{q}\right) \tag{21}$$

where I is the m x m identity matrix.

In practice, equation 21 can be reduced by the Poisson probability mass function.

Below is the convolution integral that uses the uniformisation-based integration of matrix exponential method:

1. Compute $\underline{b_u}$ for $u = 1, \ldots, U$:

   (a) $\underline{b_u} := P\underline{b_{u-1}}, \qquad \underline{b_0} = \underline{b_k}(0)$

2. Compute $\underline{c_u}$ for $u = U-1, \ldots, 0$

   (a) $\underline{c_u} := c_{u+1}P + e^{-qt_k}\frac{(qt_k)^{u+1}}{(u+1)!}\underline{f_k}(0), \qquad c_U := e^{-qt_k}\frac{(qt_k)^{U+1}}{(U+1)!}\underline{f_k}(0)$

3. Compute $H_k = \left(\frac{1}{q}\right)\sum_{u=0}^{U}\underline{b_u}c_u$

where $q > |\max_i(\mu_{i,i})|$ and $P = I + \frac{C}{q}$

Furthermore, U is a right truncation point of uniformisation that satisfies the following condition:

$$\begin{aligned} \sum_{u=0}^{U} e^{-qt_k}\frac{(qt_k)^u}{u!} &\geq 1-\epsilon \\ \implies 1-\sum_{u=0}^{U} e^{-qt_k}\frac{(qt_k)^u}{u!} &\leq \epsilon \end{aligned}$$

where we define ϵ as the tolerance error.

It is important to have a truncation point as we are dealing with an infinite sum and so depending on the accuracy we require, we can reduce our sum to that many terms, say U.

After computing the above three steps, we can write the $(j,i)^{\text{th}}$ element of H_k as:

$$[H]_{j,i} = \int_0^{t_k} \left[\underline{f_k}(\tau)\right]_i \left[\underline{b_k}(t_k-\tau)\right]_j \mathrm{d}\tau$$

The time complexity of using the uniformisation method is much less then applying the differential equations approach. Therefore, MAPs with many state transitions can be solved using the above procedure.

However, in practice, there are certain situations where we find it difficult to find the expectations in the E-step. The most significant problem is the **stiffness** of the the underlying continuous-time Markov chain (CTMC). In simple terms, we define a stiff CTMC as the inclusion of both rapid and slow events that occur simultaneously. So, as we know that MAPs are composed of an underlying CTMC that have state transitions it poses a problem when estimating MAPs.

Hence, we leave it as an extension to further explore the possibilities and solutions in order to solve this stiffness problem. This is a recurring problem as many interesting models are stiff i.e. they have important events occurring at different time-scales (i.e. some slow and some fast). [18]

Now, let us summarise the algorithm into exactly 5 steps:

1. Determine initial set, $\theta := \{\pi_i, \mu_{i,j}, \lambda_{i,j}\}$

2. Compute $\underline{b_k}(u)$, $\underline{f_k}(u)$ and $\underline{H_k}(u)$ by using the differential equation approach or the uniformisation method.
3. Compute the expected values for $\mathcal{B}_i$, $Y_{i,j}^{[k]}$, $Z_i^{[k]}$, $M_{i,j}^{[k]}$, $W_{i,j}^{[k]}$ as described earlier.
4. Update parameters using the formulas given in equation 14.
5. If termination condition satisfied then we stop, otherwise loop back to 2.

The termination condition must be accurate as it determines the accuracy of the estimates. Determining the optimal number of state transitions is a key issue as the accuracy of fitting gets better as the number of transitions increase and this leads to an over-fitting problem. In order to avoid the over-fitting problem we can use an information criteria such as Akaike's information criterion (AIC)[19].

$$\text{AIC} \quad = \quad -2\,(\text{maximum log} - \text{likelihood}) + 2p$$

where p is the degrees of freedom. The number of free parameters for m transitions is generally, $2m^2 - 1$ or m^2. Therefore, by using AIC we can determine the number of state transitions that returns the lowest AIC.

5.2.4 Example using Uniformisation

Let us show an example of how we can apply the uniformisation method of the EM algorithm. This example will clearly highlight all the steps we stated in the earlier section.

We can define equation 21 using an iterative solution scheme[20] and the resulting formula is below:

$$\sum_{y=0}^{y_\epsilon} \Phi_y \beta_y$$

where $\Phi_0 = \theta$ (initial set), $\Phi_{y+1} = \Phi_y P$, $\beta_0 = e^{-qt}$, $\beta_{y+1} = \beta_y \cdot \frac{qt}{y+1}$ and y_ϵ is the truncation point.

Thus we can define our truncation point as well as our error, which are bound by:

$$\sum_{y=y_\epsilon+1}^{\infty} \beta_y \quad = \quad 1 - \sum_{y=0}^{y_\epsilon} \beta_y$$

Now, we can take an example with the initial set, θ, defined as:

$$\theta \quad := \quad \{0.5, 2, 1\}$$

i.e. $\pi = 0.5$, $\mu = 2$, and $\lambda = 1$

Next, we define our generator matrix, Q, of the underlying CTMC as:

$$Q = \begin{bmatrix} -3 & 2 & 1 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

(**Note:** Q is a generator matrix because the sum of its rows are 0 and the diagonal entries are negative or 0.)

Then we take $q = 3$ and define $P = I + \frac{Q}{q}$, which results in:

$$I + \frac{1}{3}\begin{bmatrix} -3 & 2 & 1 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \end{bmatrix} = I + \begin{bmatrix} -1 & \frac{2}{3} & \frac{1}{3} \\ 0 & -\frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 \end{bmatrix}$$

$$P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} -1 & \frac{2}{3} & \frac{1}{3} \\ 0 & -\frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 \end{bmatrix}$$

$$P = \begin{bmatrix} 0 & \frac{2}{3} & \frac{1}{3} \\ 0 & \frac{2}{3} & \frac{1}{3} \\ 0 & 0 & 1 \end{bmatrix}$$

Then using MATLAB or another programming language we can define the result. We have the output for each step outlined below:

initial: $\beta_0 = e^{-\frac{3}{2}} \approx 0.223$, $\Phi_0 = (1, 0, 0)$

```
y = 0
phi = 0    0.6667    0.3333
beta = 0.3347

y = 1
phi = 0    0.4444    0.5556
beta = 0.2510

y =
     2
phi = 0    0.2963    0.7037
beta = 0.1255

y = 3
phi = 0    0.1975    0.8025
beta = 0.0471

y = 4
phi = 0    0.1317    0.8683
beta = 0.0141

y = 5
phi = 0    0.0878    0.9122
beta = 0.0035

y = 6
```

```
phi = 0     0.0585     0.9415
beta = 7.5643e-04

y = 7
phi = 0     0.0390     0.9610
beta = 1.4183e-04

y = 8
phi = 0     0.0260     0.9740
beta = 2.3638e-05

y = 9
phi = 0     0.0173     0.9827
beta = 3.5457e-06

y =
    10
phi = 0     0.0116     0.9884
beta = 4.8351e-07

y = 11
phi = 0     0.0077     0.9923
beta = 6.0439e-08
```

We could continue running the code as we know this is an infinite sum, but because of our pre-defined truncation point we stop at $y = 11$. First, let us see the sum after $y = 2$ and $\pi = 1.5$ (as $0.5 * 3 = 1.5$ and there are three state transitions):

$$\begin{aligned}\sum_{y=0}^{2} \Phi_y \beta_y &= \Phi_0\beta_0 + \Phi_1\beta_1 + \Phi_2\beta_2 \\ &= (0.2231, 0.3347, 0.2510)\end{aligned}$$

And after $k = 11$ the error is less than our truncation point, which we defined as 10^{-7}. Therefore, we are left with our answer as an approximation at $\pi = 1.5$:

$$(0.2231, 0.383, 0.393)$$

6 Evaluation

We will evaluate the two distinct sections of this project and give critical justification on the parameters as well as the tools used. We will analyse the results that we generated in chapters 3, 4 and 5.

As this project is more theoretical based rather than practical, we evaluate our achievements differently and if we have succeeded in completing the goals set out at the inception. Throughout this report we have focused on the relationship between Lie algebra and Markovian arrival processes. There have already been some papers written about this topic and how they can be expanded in terms of matrix representations by the special linear group. As mentioned earlier this project is very closely linked with the paper written by Ken'ichi Kawanishi on *The Counting Process for a Markovian Arrival Process with an Application to a Queueing System[8].*

6.1 Result From Lie Algebra Examples

There were two matrix Lie groups that we discussed in detail, namely SL $(2, \mathbb{R})$ and SU (2). We defined the two groups and using their respective Lie algebra we were able to define their C and D matrices. The first example was introduced in *[8]* and in this project we expand those steps in order to prove the rate matrices for MAPs.

Let us recall the rate matrices derived in section 3:

$$\begin{aligned} C &= \begin{pmatrix} -c_+ - d_+ - \lambda_+ & c_+ \\ c_- & -c_- - d_- - \lambda_- \end{pmatrix} \\ D &= \begin{pmatrix} d_+ & \lambda_+ \\ \lambda_- & d_- \end{pmatrix} \end{aligned}$$

From the above C and D we can define the infinitesimal generator, Q such that:

$$\begin{aligned} Q &= C + D \\ &= \begin{pmatrix} -c_+ - d_+ - \lambda_+ & c_+ \\ c_- & -c_- - d_- - \lambda_- \end{pmatrix} + \begin{pmatrix} d_+ & \lambda_+ \\ \lambda_- & d_- \end{pmatrix} \\ &= \begin{pmatrix} -c_+ - \lambda_+ & c_+ + \lambda_+ \\ c_- + \lambda_- & -c_- - \lambda_- \end{pmatrix} \end{aligned}$$

So, by definition of generators we know that the sum of each row must be equivalent to zero i.e.

$$\begin{aligned} -c_+ - \lambda_+ + c_+ + \lambda_+ &= 0 \\ c_- + \lambda_- - c_- - \lambda_- &= 0 \end{aligned}$$

Hence, the first example works and in chapter 3 we discussed the uses of of these matrices in the real world. In addition, we coded these examples into MATLAB and in order to compute

$P(n,t)$ as well as verify that the rate matrices C and D are correct and their sum is a generator matrix.

We generated our own example for the matrix Lie group, SU (2) and performed further analysis into generating commutator relations for SU (3). As we realised the number of brackets increase very quickly and we will require more time to try and generate appropriate equations for C and D. Even though we used a well known set of matrices as our basis, the commutator relations were defined by us and we could continue to expand them as we further investigate the 3 x 3 case.

6.2 Efficiency of Estimation-Maximisation Algorithm

We have seen that there are two ways of solving the problem of incomplete data when modeling MAPs. One way was using moments, however we did not discuss or show examples of this method and the second approach involved the EM algorithm, which we discussed in detail.

Within the EM algorithm we also had two different ways of solving the problem. The first method uses the concept of differential equations, which is computationally intensive as there are will be a lot of recursion if the value of n is very large. We did not get to reach a stage to be able to compute our example using this algorithm as it is harder to implement and we did not get the time. However, the problem of stiffness can be addressed as we can apply solve stiff cases by using the implicit Runge-Kutta method.[17]

The second approach, uniformisation, is more novel and applicable in the real world as we can code it in a straightforward manner and it returns results in an instant. As seen in the computation of our example, we were able to put in the relevant data and get a clear set of output that shows us each step. This is a very simple algorithm that can prove to be very effective, but it can go on and get very complicated i.e. it can reach a point where the answer is not accurate. So, the tolerance error must be defined accurately and it should be the best value in order to estimate the results and generate the correct solution.

This section of the report was based on two different papers that we mentioned in section 5, however the implementation and example was derived in order to fit the theory. We also simplified some of the equations that were mentioned in the paper as well as expanded some of the situations where parameters were missing. For instance, we added an extra variable,$W_{i,j}^{[k]}$, that was not mentioned in [12]. However, even though it was included in [17] we simplified all those equations in order to prove the required result, i.e. the expectations of our variables which are used in the EM algorithm.

7 Conclusion & Future Work

7.1 Conclusion

To conclude this report, we summarise our achievements in terms of what we have learnt and if we have met our objectives. Our goal was to understand the link between Lie algebra and Markovian arrival processes (MAPs). Using the work from the following paper "On the Counting Process for a Class of Markovian Arrival Processes with an Application to a Queueing System", we have been able to define a strong link between some of the matrix Lie groups, their corresponding Lie algebra and MAPs. We were successfully able to demonstrate two different methods in order to find the rate matrices, C and D of a MAP and one of these approaches' requires us to use the commutator relation (i.e. Lie algebra). By computing the results in MATLAB, we were able to show that computationally integration with matrix exponentials as well as commutators are slow and require a lot of time. However, it was slightly faster to computer the rate matrices using Lie algebra and it is more readable as well as understandable because the equations become much smaller to work with and in some cases we can solve the problem by hand. While with integration it requires a lot more CPU time and it is very difficult to solve these problems by hand even for simple, 2 x 2 cases.

Furthermore, we went into depth and further understood the estimation-maximisation algorithm and how we can apply it to MAPs. As mentioned in the introduction, MAPs are widely used in network and queueing theory, so in these real-life scenarios not all the data arrives (incomplete data). Therefore, the EM algorithm can help estimate the missing information. In chapter 6, we discussed an alternative to the EM algorithm i.e. the moment-based approach. Even though we only briefly described the advantages and disadvantages, our main focus was the more recent and new idea of the EM algorithm and its use in solving MAPs.

7.2 Future Work

There are various extensions that can be built and explored from the above aims and those are left as future work. Some of these include:

7.2.1 Larger Dimension Matrices

Solving MAPs not just for 2 x 2 matrices, but also for 3 x 3 and larger dimensions. We have already tried to apply a method to solve the SU (3) case, however as we have not been successful in finding a solution we can continue to find different approaches in order to prove or disprove the claim. We would need to perform a similar analysis as we did for SL $(2, \mathbb{R})$ and using that as a basis find relations that we could use to generate the rate matrices.

7.2.2 Simplify Large Matrices

As seen earlier, MATLAB was not very helpful in simplifying very large matrices. Therefore, when we work on larger matrices and matrix exponentials it would be useful to code in a different mathematical language in order to ensure that we get the simplest form of our solution.

7.2.3 Other Matrix Lie Groups

Finding examples within the other matrix Lie groups i.e. other than SL $(2, \mathbb{R})$ and SU (2) . There are many different matrix Lie groups and we can attempt to prove the commutator relation for different cases. Therefore, as a result find the rate matrices C and D that could be used to further solve for $P(n, t)$.

7.2.4 Apply EM Algorithm

Applying the EM algorithm to a real situation where MAPs are used and comparing the solution with the moment-based method. Give more examples and compute them to existing problems.

7.2.5 Other Forms of MAPs

Further investigate into the different MAPs such as Markov-modulated Poisson process or Phase-type renewal process and see if we can apply Lie algebra and its commutator relations to such cases. In addition, we could apply these models to real networking situations in order to compare results and see if the EM algorithm can also be applied to other situations.

7.2.6 Stiff Matrices

While discussing the EM algorithm, we came across the concept of stiff matrices. As an extension we could look into ways of applying the uniformisation method in order to solve this problem.

References

[1] "Lie Groups, Physics, and Geometry." Lie Groups, Physics and Geometry. N.p., n.d. Web. 14 Mar. 2012. <http://einstein.drexel.edu/~bob/LieGroups.html>.

[2] Hall, Brian C. Lie Groups, Lie Algebras, and Representations: An Elementary Introduction. New York: Springer, 2003. 3-73. Print.

[3] Hanson, Robert. "A Proof of the Heine-Borel Theorem." N.p., n.d. Web. 20 June 2012. <http://www.math.utah.edu/~bobby/3210/heine-borel.pdf>.

[4] Sepanski, Mark R. Compact Lie Groups. New York, NY: Springer, 2007. PDF.

[5] "Examples of Markovian Arrival Processes." Examples of Markovian Arrival Processes. N.p., n.d. Web. 17 Feb. 2012. <http://www.cs.cmu.edu/~osogami/thesis/html/node72.html>.

[6] "Definition of MAP." Definition of MAP. N.p., n.d. Web. 17 Feb. 2012. <http://www.cs.cmu.edu/~osogami/thesis/html/node73.html>.

[7] "Poisson Process." – from Wolfram MathWorld. N.p., n.d. Web. 14 Mar. 2012. <http://mathworld.wolfram.com/PoissonProcess.html>.

[8] Kawanishi, Ken'ichi. "On the Counting Process for a Class of Markovian Arrival Processes with an Application to a Queueing System." Queueing Systems 49.2 (2005): 93-122. Print.

[9] "3.5 The Markov Property." 3.5 The Markov Property. N.p., n.d. Web. 13 Feb. 2012. <http://webdocs.cs.ualberta.ca/~sutton/book/ebook/node32.html>.

[10] "Introduction to Finite Markov Chains." N.p., n.d. Web. 13 Feb. 2012. <http://www.ams.org/bookstore/pspdf/mbk-58-prev.pdf>.

[11] Kang, Sang H., Yong Han Kin, Dan K. Sung, and Bong D. Choi. "An Application of Markovian Arrival Process (MAP) to Modeling Superposed ATM Cell Streams." IEEE Transactions on Communications 50.4 (2004): 633-44. PDF.

[12] Okamura, Hiroyuki, Yuya Kamahara, and Tadashi Dohi. "An EM Algorithm for a Superposition of Markovian Arrival Processes." N.p., 2008. Web. 27 May 2012. <http://www.kurims.kyoto-u.ac.jp/~kyodo/kokyuroku/contents/pdf/1589-28.pdf>.

[13] Haywood, Stephen. "Gell-Mann Matrices." N.p., n.d. Web. 18 June 2012. <http://hepwww.rl.ac.uk/Haywood/Group_Theory_Lectures/Lecture_4.pdf>.

[14] Turzillo, Alex. "Representations of Matrix Lie Algebras." N.p., Aug. 2010. Web. 25 Mar. 2012. <http://www.math.uchicago.edu/~may/VIGRE/VIGRE2010/REUPapers/Turzillo.pdf>.

[15] "Pauli Matrices." – from Wolfram MathWorld. N.p., n.d. Web. 18 June 2012. <http://mathworld.wolfram.com/PauliMatrices.html>.

[16] Borman, Sean. "The Expectation Maximization Algorithm." N.p., 09 Jan. 2009. Web. 25 May 2012. <http://www.seanborman.com/publications/EM_algorithm.pdf>

[17] Okamura, Hiroyuki, Tadashi Dohi, and Kishor S. Trivedi. "IEEE/ACM Transactions on Networking." Markovian Arrival Process Parameter Estimation With Group Data 17.4 (2009): 1328-330. PDF.

[18] Reibman, Andrew, Kishor Trivedi, Sanjaya Kumar, and Gianfranco Ciardo. "Analysis of Stiff Markov Chains." N.p., 1989. Web. 10 June 2012. <http://people.ee.duke.edu/~kst/markovpapers/paper05.pdf>.

[19] H. Akaike, B. N. Petrov and F. Csaki, Eds., "Information theory and an extension of the maximum likelihood principle," in Proc. 2nd Int. Symp. Inform. Theory, 1973, pp. 267–281.

[20] "Uniformisation." N.p., n.d. Web. 15 June 2012. <www.informatik.huberlin.de/~wolter/teaching/.../uniformisation.ps>.