2009

The Classification of Simple Lie Algebras in Maple

D. Russell Sadler
Utah State University

Follow this and additional works at: https://digitalcommons.usu.edu/gradreports

Part of the Mathematics Commons

Recommended Citation
https://digitalcommons.usu.edu/gradreports/1261

This Report is brought to you for free and open access by the Graduate Studies at DigitalCommons@USU. It has been accepted for inclusion in All Graduate Plan B and other Reports by an authorized administrator of DigitalCommons@USU. For more information, please contact digitalcommons@usu.edu.
THE CLASSIFICATION OF SIMPLE LIE ALGEBRAS
IN MAPLE

by

D. Russell Sadler

A report submitted in partial fulfillment
of the requirements for the degree
of
MASTER OF SCIENCE
in
Mathematics

UTAH STATE UNIVERSITY
Logan, Utah
2009
ABSTRACT

The Classification of Semisimple Lie Algebras in Maple

by

D. Russell Sadler, Master of Science

Utah State University, 2009

Major Professor: Dr. Mark Fels
Department: Mathematics and Statistics

Lie algebras are invaluable tools in mathematics and physics as they enable us to study certain geometric objects such as Lie groups and differentiable manifolds. The computer algebra system Maple has several tools in its LieAlgebras package to work with Lie algebras and Lie groups. The purpose of this paper is to supplement the existing software with tools that are essential for the classification of simple Lie algebras over $\mathbb{C}$.

In particular, we use a method to find a Cartan subalgebra of a Lie algebra in polynomial time. From the Cartan subalgebra we can compute the corresponding root system. This allows us to develop a command to compute the Cartan Matrix of a semisimple Lie algebra. From the Cartan Matrix we can construct the corresponding Dynkin diagram and determine the structure of the Lie algebra. We use the Cartan subalgebra and Cartan matrix to classify the simple Lie algebras over $\mathbb{C}$.

We will also set out to define commands to initialize the classical Lie algebras of all dimensions in Maple. These commands will give us the tools needed to verify our results.

(71 pages)
For Ginger, Denton, and Amy. You’re the reason why I kept going.
ACKNOWLEDGMENTS

I would like to thank Mark Fels for his support and presenting me with an interesting problem. I would like to acknowledge Ian Anderson, Dariusz Wilczynski, and Zhi Qiang Wang for the help and instruction I’ve received from them. I would also like to thank my wife for her infinite patience and encouragement. I would probably still be working on this project without her.

Russell Sadler
CONTENTS

ABSTRACT ................................................................. iii

ACKNOWLEDGMENTS ...................................................... v

1. PRELIMINARIES ......................................................... 1
   1.1 Lie Algebras .................................................. 1
       1.1.1 Types of Lie Algebras .................................. 5
   1.2 The LieAlgebra Package ....................................... 7

2. ROOT SPACE DECOMPOSITIONS AND CARTAN SUBALGEBRAS .......... 10
   2.1 Root Space Decomposition .................................... 10
       2.1.1 The RootSpaceDecomposition Command .................... 14
   2.2 Cartan Subalgebras ........................................... 16
       2.2.1 Regular Elements ....................................... 16
       2.2.2 The FittingNull and NonNilp Commands .................. 19
       2.2.3 The CSA Command ....................................... 20

3. EXAMPLES ............................................................. 23
   3.1 FittingNull and NonNilp ..................................... 23
   3.2 CSA ........................................................... 29
   3.3 RootSpaceDecomposition ....................................... 32

4. SIMPLE LIE ALGEBRA STRUCTURE ................................... 38
   4.1 Root Systems ................................................ 38
       4.1.1 Positive and Simple Roots ............................... 38
   4.2 The Killing Form ............................................ 40
   4.3 Computing the Cartan Matrix ................................ 41
       4.3.1 The CartanMatrix Command ............................. 43
   4.4 Dynkin Diagrams ............................................. 44
   4.5 Cartan Matrices and Dynkin diagrams of the Complex Simple Lie Algebras 46
   4.6 Classification of Simple Lie Algebras ......................... 49

5. EXAMPLES ............................................................. 54
   5.1 CartanMatrix ................................................ 54
   5.2 Classify ...................................................... 56

REFERENCES .............................................................. 58

APPENDICES .............................................................. 59

A. Classical Lie Algebras ............................................ 60
LIST OF TABLES

1.1 Classical Lie Algebras ........................................... 6
4.1 Cartan Matrices and Dynkin diagrams of the Simple Lie Algebras .......................... 48
4.2 Dimension of the Complex Simple Lie Algebras ........................................ 49
1. PRELIMINARIES

1.1 Lie Algebras

Here we present the reader with an introduction to the theory of Lie algebras, which will be essential for the purposes of this report.

Definition 1.1: Let $K$ be a field. A Lie algebra over $K$ is a vector space $g$ over $K$ together with a binary operation $[\cdot, \cdot] : g \times g \rightarrow g$, called the Lie bracket, which satisfies the following for any $X, Y, Z \in g$ and $a, b \in K$:

i. Bilinear

(a) $[aX + bY, Z] = a[X, Z] + b[Y, Z],$

(b) $[X, aY + bZ] = a[X, Y] + b[X, Z].$

ii. Skew-Symmetric

$[X, Y] = -[Y, X] \Rightarrow [X, X] = 0.$

iii. Jacobi Identity

$[X, [Y, Z]] + [Y, [Z, X]] + [Z, [X, Y]] = 0.$

In this report we consider real and complex Lie algebras, however much of the discussion may be extended to Lie algebras over fields of characteristic 0.

Example 1.1: The real vector space $M_n(\mathbb{R})$ consisting of real $n \times n$ matrices together with the commutator $[A, B] = AB - BA$ is a Lie algebra.

Example 1.2: More generally, let $V$ be a vector space over a field $K$, $K = \mathbb{C}$ or $\mathbb{R}$, and let $\mathfrak{gl}(V)$ denote the vector space consisting of the endomorphisms of $V$. Then $\mathfrak{gl}(V)$ with the bracket operation $[S, T] = S \circ T - T \circ S$ is a Lie algebra.
Definition 1.2: Let \( g \) be a Lie algebra and let \( e_1, e_2, \ldots, e_n \) be a basis of \( g \). The structure constants \( a_{jk}^i \) are given by

\[
[e_j, e_k] = \sum_{i=1}^{n} a_{jk}^i e_i.
\]

These satisfy

\[
a_{jk}^i + a_{kj}^i = 0, \quad \text{and}
\]

\[
\sum_{j=1}^{n} a_{ji}^j a_{km}^j + a_{jm}^j a_{ik}^j + a_{jk}^i a_{mi}^j = 0.
\]

These follow from the skew-symmetric property and from the Jacobi identity of Lie algebras given in Definition 1.1.

From the structure constants we can compute \([X, Y]\) for any \( X, Y \in g \) using bilinearity. Conversely, given constants \( \{a_{jk}^i\} \), \( 1 \leq i, j, k \leq n \), which satisfy (1.2) we can define \([\cdot, \cdot]\) : \( V \times V \to V \) on a basis by (1.1) and extend \([\cdot, \cdot]\) over \( V \) using bilinearity. In this case, \( V \) together with \([\cdot, \cdot]\) is a Lie algebra.

We turn our attention to representations which is another important concept when considering Lie algebras.

Definition 1.3: A representation of a Lie algebra \( g \) on a vector space \( V \) is a homomorphism \( \rho : g \to \mathfrak{gl}(V) \).

The mapping \( \rho : g \to \mathfrak{gl}(V) \) is a Lie algebra homomorphism if

\[
\rho([X, Y]) = [\rho(X), \rho(Y)] = \rho(X) \circ \rho(Y) - \rho(Y) \circ \rho(X).
\]

In other words, a Lie algebra homomorphism is a mapping which preserves the bracket operation.

Example 1.3: If \( g \) is a Lie algebra over \( K \), with \( X \in g \), then we define the adjoint action of \( X \) on \( g \) to be the linear transformation \( \text{ad}(X) \in \mathfrak{gl}(g) \) defined by

\[
\text{ad}(X)(Y) = [X, Y].
\]
In particular,
\[
\text{ad} ([X, Y])(Z) = [X, Y], Z
\]
\[
= -[Z, [X, Y]]
\]
\[
= [X, [Y, Z]] + [Y, [Z, X]]
\]
\[
= [X, [Y, Z]] - [Y, [X, Z]]
\]
\[
= [X, \text{ad} (Y)(Z)] - [Y, \text{ad} (X)(Z)]
\]
\[
= \text{ad} (X)(\text{ad} (Y)(Z)) - \text{ad} (Y)(\text{ad} (X)(Z))
\]
\[
= (\text{ad} (X) \circ \text{ad} (Y))(Z) - (\text{ad} (Y) \circ \text{ad} (X))(Z)
\]
\[
= [\text{ad} (X), \text{ad} (Y)](Z).
\]
Thus \( \text{ad} ([X, Y]) = [\text{ad} (X), \text{ad} (Y)] \), where the second bracket is as in Example 1.2 where \( V = \mathfrak{g} \).

The mapping \( X \to \text{ad} (X) \) is a representation of \( \mathfrak{g} \) on the vector space \( \mathfrak{g} \). This representation is called the *adjoint representation* of \( \mathfrak{g} \).

If \( \mathfrak{h} \) is a subspace of \( \mathfrak{g} \), then \( \mathfrak{h} \) is a *subalgebra* of \( \mathfrak{g} \) if \( \mathfrak{h} \) is closed under the Lie bracket operation, that is if \( [X, Y] \in \mathfrak{h} \) for all \( X, Y \in \mathfrak{h} \). A subalgebra \( \mathfrak{h} \) of \( \mathfrak{g} \) is an *ideal* if \( [H, X] \in \mathfrak{h} \) for every \( H \in \mathfrak{h} \) and \( X \in \mathfrak{g} \).

If \( \mathfrak{h} \) is a subalgebra of \( \mathfrak{g} \), we can construct three important subalgebras.

**Definition 1.4:** The *centralizer* of a set \( S \) in \( \mathfrak{g} \), denoted \( C_\mathfrak{g}(S) \), is given by
\[
C_\mathfrak{g}(S) = \{ Y \in \mathfrak{g} : [X, Y] = 0 \text{ for all } X \in S \}
\]

**Theorem 1.1:** \( C_\mathfrak{g}(S) \) is a subalgebra of \( \mathfrak{g} \).

**Proof:** Let \( X \in S \) and \( Y, Z \in C_\mathfrak{g}(S) \) and let \( a \in \mathbb{C} \). Then \( [X, aY + Z] = a[X, Y] + [X, Z] = a \cdot 0 + 0 = 0 \). Thus \( aY + Z \in C_\mathfrak{g}(S) \) and \( C_\mathfrak{g}(S) \) is a subspace of \( \mathfrak{g} \).

By the Jacobi identity in Definition 1.1, \( [X, [Y, Z]] + [Y, [Z, X]] + [Z, [X, Y]] = 0 \). As \( Y, Z \in C_\mathfrak{g}(S) \) and \( X \in S \), \( [Z, X] = [X, Y] = 0 \). Thus \( [Y, [Z, X]] = [Z, [X, Y]] = 0 \). Therefore \( [X, [Y, Z]] = 0 \), i.e. \( [Y, Z] \in C_\mathfrak{g}(S) \), which implies \( C_\mathfrak{g}(S) \) is closed under the bracket operation. \( \square \)

If \( [X, Y] = 0 \), then we say that \( X \) and \( Y \) commute. Therefore the centralizer of \( S \) in \( \mathfrak{g} \) is the subalgebra of vectors in \( \mathfrak{g} \) which commute with every vector in \( S \).
Definition 1.5: The normalizer of the subalgebra \( \mathfrak{h} \) in a subalgebra \( \mathfrak{k} \) of \( \mathfrak{g} \), denoted \( \mathfrak{n}_\mathfrak{k}(\mathfrak{h}) \), is given by

\[
\mathfrak{n}_\mathfrak{k}(\mathfrak{h}) = \{ X \in \mathfrak{k} : [X, H] \in \mathfrak{h} \text{ for all } H \in \mathfrak{h} \}.
\]

Theorem 1.2: \( \mathfrak{n}_\mathfrak{k}(\mathfrak{h}) \) is a subalgebra of \( \mathfrak{k} \), and subsequently a subalgebra of \( \mathfrak{g} \).

Proof: Let \( X, Y \in \mathfrak{n}_\mathfrak{k}(\mathfrak{h}) \), \( H \in \mathfrak{h} \), and \( a \in \mathbb{C} \). Then \( [aX + Y, H] = a[X, H] + [Y, H] \). As \( [X, H], [Y, H] \in \mathfrak{h} \) by hypothesis, \( a[X, H] + [Y, H] \in \mathfrak{h} \) as \( \mathfrak{h} \) is a subalgebra. Therefore \( aX + Y \in \mathfrak{n}_\mathfrak{k}(\mathfrak{h}) \), which implies \( \mathfrak{n}_\mathfrak{k}(\mathfrak{h}) \) is a subspace of \( \mathfrak{k} \).

The normaizer of \( \mathfrak{h} \) is the largest subalgebra of \( \mathfrak{k} \) which contains \( \mathfrak{h} \) as an ideal.

Definition 1.6: The center of the Lie algebra \( \mathfrak{g} \), denoted \( Z(\mathfrak{g}) \), is given by

\[
Z(\mathfrak{g}) = \{ X \in \mathfrak{g} : [X, Y] = 0 \text{ for all } Y \in \mathfrak{g} \}.
\]

In particular, \( Z(\mathfrak{g}) = C_\mathfrak{g}(\mathfrak{g}) \). If the center of \( \mathfrak{g} \) is \( \mathfrak{g} \), then \( \mathfrak{g} \) is said to be abelian.

Theorem 1.3: \( Z(\mathfrak{g}) \) is an ideal of \( \mathfrak{g} \).

Proof: By Theorem 1.1, \( Z(\mathfrak{g}) \) is a subalgebra of \( \mathfrak{g} \). Let \( X \in Z(\mathfrak{g}) \) and \( Y \in \mathfrak{g} \). By definition of \( Z(\mathfrak{g}) \), \( [X, Y] = 0 \). As \( [0, Z] = 0 \) for any \( Z \in \mathfrak{g} \), \( 0 \in Z(\mathfrak{g}) \). Therefore \( Z(\mathfrak{g}) \) is an ideal of \( \mathfrak{g} \).
1.1.1 Types of Lie Algebras

Our attention will be focused specifically on simple Lie algebras.

**Definition 1.7:** Let $\mathfrak{g}$ be a Lie algebra. The *derived series* of subalgebras $\mathcal{D}^k\mathfrak{g}$ is defined inductively by

$$\mathcal{D}^1\mathfrak{g} = [\mathfrak{g}, \mathfrak{g}] = \text{Span}\{X \in \mathfrak{g} : X = [Y, Z] \text{ for some } Y, Z \in \mathfrak{g}\},$$

and

$$\mathcal{D}^k\mathfrak{g} = [\mathcal{D}^{k-1}\mathfrak{g}, \mathcal{D}^{k-1}\mathfrak{g}].$$

A Lie algebra is *solvable* if $\mathcal{D}^k\mathfrak{g} = 0$ for some $k$.

**Theorem 1.4:** Let $\mathfrak{g}$ be a Lie algebra, then $[\mathfrak{g}, \mathfrak{g}]$ is an ideal of $\mathfrak{g}$.

*Proof:* Let $X, Y \in [\mathfrak{g}, \mathfrak{g}]$, which implies that $X, Y \in \mathfrak{g}$. By definition, $[X, Y] \in [\mathfrak{g}, \mathfrak{g}]$, which implies that $[\mathfrak{g}, \mathfrak{g}]$ is a subalgebra of $\mathfrak{g}$. Let $Y \in \mathfrak{g}$ and $X \in [\mathfrak{g}, \mathfrak{g}]$. As $[\mathfrak{g}, \mathfrak{g}]$ is a subspace of $\mathfrak{g}$, $X \in \mathfrak{g}$. Thus, $[X, Y] \in [\mathfrak{g}, \mathfrak{g}]$ by definition of $[\mathfrak{g}, \mathfrak{g}]$. \qed

**Definition 1.8:** A Lie algebra is *semisimple* if it has no nonzero solvable ideals.

**Definition 1.9:** A Lie algebra is called *simple* if it is semisimple and has no ideals except $0$ and itself.

**Theorem 1.5:** A semisimple Lie algebra is the direct sum of simple ideals.

*Proof:* See [6], pages 131-132. \qed

A convenient way to determine if a Lie algebra is semisimple is to consider the Killing form of the Lie algebra.

**Definition 1.10:** Let $\mathfrak{g}$ be a Lie algebra and let $X, Y \in \mathfrak{g}$. The *Killing form* is the bilinear form $B : \mathfrak{g} \times \mathfrak{g} \to K$ defined by

$$B(X, Y) = \text{Tr}(\text{ad}(X) \circ \text{ad}(Y)).$$

Where $\text{ad}(X) \in \mathfrak{gl}(\mathfrak{g})$ is defined in (1.3).
Theorem 1.6: A complex Lie algebra is semisimple if and only if its Killing form is nondegenerate.\footnote{This is true for a Lie algebra over a field of characteristic 0.}

Proof: See [7], page 22.

The finite-dimensional complex simple Lie algebras were classified by Élie Cartan building on the work of Wilhelm Killing. These Lie algebras are the classical Lie algebras $\mathfrak{a}_n$, $\mathfrak{b}_n$, $\mathfrak{c}_n$, and $\mathfrak{d}_n$ along with the five exceptional Lie algebras $\mathfrak{e}_6$, $\mathfrak{e}_7$, $\mathfrak{e}_8$, $\mathfrak{f}_4$, and $\mathfrak{g}_2$.

<table>
<thead>
<tr>
<th>Tab. 1.1: Classical Lie Algebras</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathfrak{a}_n$ $n \geq 1$</td>
</tr>
<tr>
<td>$\mathfrak{b}_n$ $n \geq 2$</td>
</tr>
<tr>
<td>$\mathfrak{c}_n$ $n \geq 3$</td>
</tr>
<tr>
<td>$\mathfrak{d}_n$ $n \geq 4$</td>
</tr>
</tbody>
</table>

A concept related to simple Lie algebras that will play an important role in Chapter 2 is that of a nilpotent Lie algebra.

Definition 1.11: Let $\mathfrak{g}$ be a Lie algebra. The lower central series of subalgebras $\mathcal{D}_k \mathfrak{g}$ is defined inductively by

$$\mathcal{D}_1 \mathfrak{g} = [\mathfrak{g}, \mathfrak{g}],$$

and

$$\mathcal{D}_k \mathfrak{g} = [\mathfrak{g}, \mathcal{D}_{k-1} \mathfrak{g}].$$

A Lie algebra is nilpotent if $\mathcal{D}_k \mathfrak{g} = 0$ for some $k$.

Definition 1.12: A linear operator $T \in \mathfrak{gl}(V)$ is said to be nilpotent if $T^k \equiv 0$ for some integer $k$.

Theorem 1.7: (Engel's Theorem) A Lie algebra $\mathfrak{g}$ is nilpotent if and only if $\text{ad}(X)$ is nilpotent for every $X$ in $\mathfrak{g}$.

Proof: See [6], page 160.
1.2 The LieAlgebra Package

As Lie algebras are essential tools in differential geometry the LieAlgebras package in Maple is coupled to the DifferentialGeometry package. One of the main features of the package is the ability to initialize a Lie algebra in Maple, that is, the package allows us to define a Lie algebra in a basis by equations (1.1).

To use the package, we type:

```maple
> with(DifferentialGeometry):with(LieAlgebras):
```

This allows us to use the short form of the commands in the package. To demonstrate the capabilities of the LieAlgebras package we will consider the complex Lie algebra $\mathfrak{s}_2 \mathbb{C}$, which is the Lie algebra consisting of complex $2 \times 2$ traceless matrices. A basis for $\mathfrak{s}_2 \mathbb{C}$ is:

\begin{equation}
\beta = \{ \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \}.
\end{equation}

We can enter this basis into Maple as follows:

```maple
> beta:=<Matrix(2, 2, [[1, 0], [0, -1]]), Matrix(2, 2, [[0, 1], [0, 0]]), Matrix(2, 2, [[0, 0], [1, 0]]):
```

The DGsetup command is used to initialize the Lie algebra. The first argument of the DGsetup command is a Lie algebra data structure. We can use the LieAlgebraData command to convert the basis of $\mathfrak{s}_2 \mathbb{C}$ into a data structure that the DifferentialGeometry package in Maple recognizes as a Lie algebra. We will call the data structure L1.

```maple
> L1:=LieAlgebraData(beta,s12):
```

Maple has computed the matrix commutators for the matrices in $\beta$ in (1.4) and has defined the Lie bracket operation accordingly, as indicated by the output. In particular, the program has determined the structure constants (1.1), which define the bracket operation. It is precisely these relations that provide the data structure for the Lie algebra. Maple has also given the basis vectors the names $e_1$, $e_2$, $e_3$, which will be useful for practical reasons.
The second argument of the LieAlgebraData command is the name Maple assigns the Lie algebra after it has been initialized.

```maple
> DGsetup(L1);

Lie algebra: sl2
```

Once the Lie algebra has been initialized we can work within the Lie algebra. The Query command can be used to check if the algebra has certain properties. For example, the command Query("Semisimple") yields true if the algebra is semisimple and false if it is not.

```maple
> Query("Semisimple");

true
```

We can compute \( \text{ad}(X) \) for any \( X \) in \( sl_2 \mathbb{C} \). Adjoint\((X)\) computes the matrix representation of \( \text{ad}(X) \) in the ordered basis \( \beta \). For example, \( \text{ad}(e_1) \) is computed as follows:

```maple
> Adjoint(e1);
```

If the Adjoint command does not receive an argument, the output is a list of the matrix representations of \( \text{ad}(X_i) \) for each \( X_i \) in the basis of the Lie algebra.

```maple
> Adjoint();
```

We can also compute the Killing form of \( sl_2 \mathbb{C} \). The command Killing without any arguments computes the matrix representation of the Killing form with respect to the ordered basis \( \beta \), while the Killing command with two arguments evaluates the Killing form on the two vectors.

```maple
> Killing();
> Killing(e1-e2,e1+e2);
```
The calling sequence to find a basis of the centralizer of a subset $S$ in a subalgebra $\mathfrak{h}$ is \texttt{Centralizer}(S, h), where $h$ is a basis for $\mathfrak{h}$. The second argument $h$ is optional. If $h$ is not specified, \textit{Maple} computes the centralizer of $S$ in the entire Lie algebra.

\begin{verbatim}
> Centralizer([el]);
[el]
\end{verbatim}

The output indicates that a basis for the centralizer of $S = \{e1\}$ in $\mathfrak{g}$ is $\{e1\}$.

Now, to find a basis for the normalizer of a subalgebra $\mathfrak{h}$ in a subalgebra $\mathfrak{t}$, where $\mathfrak{h}$ is contained in $\mathfrak{t}$, we must find a basis for $\mathfrak{h}$ and $\mathfrak{t}$, say $h$ and $k$ respectively. Once we have found $h$ and $k$, the calling sequence for the normalizer of $\mathfrak{h}$ in $\mathfrak{t}$ is \texttt{SubalgebraNormalizer}(h, k). If $k$ is omitted, the command computes the normalizer of $h$ in the initialized Lie algebra. For example, we can find the normalizer of the subalgebra whose basis is $\{e1\}$ in $\mathfrak{sl}_2\mathbb{C}$. First we check that this set defines a basis for a subalgebra.

\begin{verbatim}
> Query([el], "Subalgebra");
true
\end{verbatim}

Now we can find a basis for the normalizer of this subalgebra in $\mathfrak{sl}_2\mathbb{C}$.

\begin{verbatim}
> SubalgebraNormalizer([el]);
[el]
\end{verbatim}

Thus a basis for the normalizer of the subalgebra in $\mathfrak{sl}_2\mathbb{C}$ is $\{e1\}$. We conclude that the subalgebra is its own normalizer in $\mathfrak{sl}_2\mathbb{C}$. In Section 2.1 a more substantial example is given.
2. ROOT SPACE DECOMPOSITIONS AND CARTAN SUBALGEBRAS

2.1 Root Space Decomposition

An object of central importance in the classification of complex Lie algebras is the Cartan subalgebra. The Cartan subalgebra was named after Élie Cartan, who introduced them in his doctoral thesis in which he completed the classification of the complex simple Lie algebras. A Cartan subalgebra is required to produce the root system in order to define the Cartan Matrix of a simple Lie algebra, which in turn provides the tools necessary for classification.

Definition 2.1: A Cartan subalgebra of a Lie algebra $\mathfrak{g}$ is a subalgebra that is:

i. nilpotent, and

ii. its own normalizer in $\mathfrak{g}$.

Theorem 2.1: Given any two Cartan subalgebras $\mathfrak{h}_1$ and $\mathfrak{h}_2$ of a semisimple Lie algebra $\mathfrak{g}$ over $\mathbb{C}$, there exists an automorphism $\sigma$ of $\mathfrak{g}$ such that $\sigma(\mathfrak{h}_1) = \mathfrak{h}_2$.

Proof: See [6], page 249. \qed

Definition 2.2: By Theorem 2.1, $\dim \mathfrak{h}_1 = \dim \mathfrak{h}_2$ for any Cartan subalgebras $\mathfrak{h}_1$ and $\mathfrak{h}_2$ of a semisimple Lie algebra $\mathfrak{g}$ over $\mathbb{C}$. The dimension of a Cartan subalgebra of $\mathfrak{g}$ is called the rank of $\mathfrak{g}$.

Definition 2.3: Let $\mathfrak{g}$ be a complex Lie algebra and $\mathfrak{h}$ a Cartan subalgebra of $\mathfrak{g}$. Let $\alpha$ be a nonzero linear functional on the complex vector space $\mathfrak{h}$, and let $\mathfrak{g}^\alpha$ denote the linear subspace of $\mathfrak{g}$ given by

$\mathfrak{g}^\alpha = \{X \in \mathfrak{g} : [H, X] = \alpha(H)X \text{ for every } H \in \mathfrak{h}\}$

The linear functional $\alpha$ is called a root of $\mathfrak{g}$ with respect to $\mathfrak{h}$ if $\mathfrak{g}^\alpha \neq 0$. The subspace $\mathfrak{g}^\alpha$ is called a root space.
Note that the requirement \([H, X] = \alpha(H)X\) for every \(H \in \mathfrak{h}\) is equivalent to \(\text{ad}(H)(X) = \alpha(H)X\) for every \(H \in \mathfrak{h}\), i.e. \(\alpha(H)\) is an eigenvalue of \(\text{ad}(H)\) and \(X\) is the corresponding eigenvector for every \(H \in \mathfrak{h}\). If \(\{h_1, \ldots, h_r\}\) is a basis for \(\mathfrak{h}\), and if \(H\) is an element of \(\mathfrak{h}\), then \(H = \sum t_i h_i\) for some \(\{t_i\} \in \mathbb{C}\).

**Algorithm 2.1: Roots**

Let \(\mathfrak{h}\) be a Cartan subalgebra of a Lie algebra \(\mathfrak{g}\), and let \(\{h_1, \ldots, h_r\}\) be a basis of \(\mathfrak{h}\).

1. Let \(H := \sum t_i h_i\), where each \(t_i\) is an indeterminate.

2. Find \(\text{ad}(H)\).

3. Find the eigenvalues of \(\text{ad}(H)\), which are linear functionals in \(t\). These functionals are the roots of \(\mathfrak{g}\) with respect to \(\mathfrak{h}\).

4. Find the corresponding eigenspaces. These are the root spaces.

**Example 2.1:** In the case of \(\mathfrak{sl}_2 \mathbb{C}\), we saw in Section 1.2 that the subalgebra \(\mathfrak{h} = \text{Span}\{e_1\}\) is self-normalizing. To show that this subalgebra is a Cartan subalgebra it is left to show that it is nilpotent. Fortunately, we can check this property in Maple with the \texttt{Query} command.

\[
> \text{Query([e1],"Nilpotent");}
\]

\texttt{true}

Therefore \(\mathfrak{h}\) is nilpotent. It follows that \(\mathfrak{h}\) is a Cartan subalgebra of \(\mathfrak{sl}_2 \mathbb{C}\). Now that we have a Cartan subalgebra we can compute a corresponding root system. Let \(H\) be an arbitrary element of \(\mathfrak{h}\). Since the basis for \(\mathfrak{h}\) is \(\{e_1\}\), \(H = t_1 e_1\) for some complex number \(t_1\). Recall that \([H, X] = \text{ad}(H)(X)\), thus we use maple to compute \(\text{Adjoint}(t_1 e_1)\), which is the matrix representation of \(\text{ad}(H)\).

\[
> \text{Adjoint}(t[1]*e1);
\]

\[
\begin{bmatrix}
0 & 0 & 0 \\
0 & 2t_1 & 0 \\
0 & 0 & -2t_1
\end{bmatrix}
\]

Now, the requirement \([H, X] = \alpha(H)X\) for every \(H \in \mathfrak{h}\) is equivalent to \(\text{ad}(H)(X) = \alpha(H)X\), or

\[
\begin{bmatrix}
0 & 0 & 0 \\
0 & 2t_1 & 0 \\
0 & 0 & -2t_1
\end{bmatrix}X = \alpha(H)X
\]
This is true if and only if \( X \) is an eigenvector of \( \text{ad} \,(H) \) and \( \alpha(H) \) is the corresponding eigenvalue.

The nonzero eigenvalues of \( \text{ad} \,(H) \) are \( 2t_1 \) and \( -2t_1 \). Therefore, the linear functionals \( \alpha_1, \alpha_2 : \mathfrak{h} \rightarrow \mathbb{C} \),
given by \( \alpha_1(t_1e_1) = 2t_1 \) and \( \alpha_2(t_1e_1) = -2t_1 \), are roots. The corresponding root spaces are the spaces spanned by \( e_2 \) and \( e_3 \) respectively.

Take particular note that the direct sum of \( \mathfrak{h} \) and the root spaces is \( \mathfrak{sl}_2 \mathbb{C} \).

**Theorem 2.2:** Let \( \mathfrak{g} \) be a Lie algebra, \( \mathfrak{h} \) a Cartan subalgebra of \( \mathfrak{g} \), and \( \Delta \) denote the set of all nonzero roots.

i. \( \mathfrak{g} = \mathfrak{h} \oplus (\bigoplus_{\alpha \in \Delta} \mathfrak{g}^\alpha) \).

ii. \( \dim \mathfrak{g}^\alpha = 1 \) for each \( \alpha \in \Delta \),

iii. If \( \alpha \in \Delta \), then \( -\alpha \in \Delta \).

See [6], pages 166 - 168.

**Definition 2.4:** The decomposition of \( \mathfrak{g} \) given by \( \mathfrak{h} \oplus (\bigoplus_{\alpha \in \Delta} \mathfrak{g}^\alpha) \) is called the root space decomposition or the Cartan decomposition. The set \( \Delta \) is the root system of \( \mathfrak{g} \) with respect to \( \mathfrak{h} \).

**Example 2.2:** \( \mathfrak{sl}_3 \mathbb{C} \) is the Lie algebra consisting of complex \( 3 \times 3 \) traceless matrices. A basis for this Lie algebra is

\[
\begin{bmatrix}
1 & 0 & 0 \\
0 & -1 & 0 \\
0 & 0 & 0
\end{bmatrix},
\begin{bmatrix}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & -1
\end{bmatrix},
\begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix},
\begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix},
\begin{bmatrix}
0 & 0 & 1 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix},
\begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 1 & 0
\end{bmatrix}
\]

(2.1)

Let us initialize the Lie algebra in *Maple*.

> beta := Matrix(3,3,{(1, 1) = 1, (2, 2) = -1}), Matrix(3,3,{(1, 2) = 1, (3, 3) = -1}), Matrix(3,3,{(1, 1) = 1}), Matrix(3,3,{(2, 2) = 1, (3, 3) = -1});
> beta := Matrix(3,3,{(1, 1) = 1}), Matrix(3,3,{(2, 2) = 1}), Matrix(3,3,{(1, 2) = 1});

\[
\beta := \begin{bmatrix}
1 & 0 & 0 \\
0 & -1 & 0 \\
0 & 0 & 0
\end{bmatrix},
\begin{bmatrix}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & -1
\end{bmatrix},
\begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix},
\begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix},
\begin{bmatrix}
0 & 0 & 1 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix},
\begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 1 & 0
\end{bmatrix}
\]
> L2:=LieAlgebraData(beta, s13);

\[
L2 \triangleq \begin{align*}
[e_6, e_8] &= -e_7
\end{align*}
\]

> DGsetup(L2);

\[
\text{Lie algebra : } sl_3
\]

We need to find a Cartan subalgebra \( h \) of \( sl_3 \) to find the corresponding root system, just as we did for \( sl_2 \). The subspace spanned by \( \{e_1, e_2\} \) is a Cartan subalgebra.

> h:=[e1, e2];

\[
h \triangleq [e_1, e_2]
\]

> Query(h,"Nilpotent");

true

> Tools:-DGequal(h,SubalgebraNormalizer(h));

true

This shows that the span of \( \{e_1, e_2\} \), which we call \( h \), is nilpotent. The \texttt{DGequal} command is part of the tools package in the \texttt{DifferentialGeometry} package. It determines whether two subspaces are equal. Since this command returned \texttt{true}, we conclude that \( h \) is its own normalizer in \( sl_3 \). Thus \( h \) is a Cartan Subalgebra of \( sl_3 \).

Now that we have found a Cartan subalgebra of \( sl_3 \), we can find the corresponding root space decomposition. We do this the same way we did in Example 2.1. Note that an arbitrary element \( H \) of \( h \) has the form \( t_1 e_1 + t_2 e_2 \). We are looking for linear functionals \( \alpha_i : H \rightarrow \mathbb{C} \) such that \( [H, X] = \alpha_i(H)X \), or \( \text{ad}(H)(X) = \alpha_i(H)X \), for every \( H \in h \) and some \( X \in \mathfrak{g} \). In particular, we are looking for the eigenvalues of \( \text{ad}(H) \) and the corresponding eigenvectors. We use the \texttt{Adjoint} command to compute \( \text{ad}(H) \).

> Adjoint(evalDG(t[1]*e1+t[2]*e2));
We see that the roots corresponding to the Cartan Subalgebra $h$ are the functionals defined by

$$
a_i(t_1 e_1 + t_2 e_2) = \begin{cases} 2t_1 - t_2, & i = 1 \\ t_1 + t_2, & i = 2 \\ -t_1 + 2t_2, & i = 3 \\ -2t_1 + t_2, & i = 4 \\ -t_1 - t_2, & i = 5 \\ t_1 - 2t_2, & i = 6 \\ t_1 - 2t_2, & i = 7 \\ t_1 - 2t_2, & i = 8 \end{cases}
$$

(2.2)

The corresponding eigenspaces, or root spaces, are the spaces spanned by $e_3, e_4, e_5, e_6, e_7$ and $e_8$, respectively. The direct sum of these root spaces and $h$ is a root space decomposition of $\mathfrak{sl}_3 \mathbb{C}$.

However, this decomposition is not unique (see Example 3.10).

2.1.1 The RootSpaceDecomposition Command

Using the method in Algorithm 2.1 in Examples 2.1 and 2.2, we can write a function to find the root space decomposition of a Lie algebra $g$ with respect to a Cartan subalgebra, $h$. Upon entry of a basis for $h$, our algorithm will:

1. verify $h$ is a Cartan subalgebra with the Query and DGequal command;

2. compute $\text{ad}(H)$ for a generic element $H \in h$;

3. compute the eigenvalues and eigenspaces of $\text{ad}(H)$; and

4. list the eigenspaces corresponding to the nonzero eigenvalues, i.e. the root spaces of the roots.
RootSpaceDecomposition := proc(h)
local g0, n, rank, adH, Roots, RootVectors, RootSpaces;
if DifferentialGeometry:-LieAlgebras:-Query(h,"Nilpotent")=false or
DifferentialGeometry:-Tools:-DGequal(h,DifferentialGeometry:-
LieAlgebras:-SubalgebraNormalizer(h))=false
then error "expected argument to be a valid Cartan subalgebra"
end if;
g0:=DifferentialGeometry:-Tools:-DGinfo("FrameBaseVectors");
n:=DifferentialGeometry:-Tools:-DGinfo("FrameBaseDimension");
rank:=nops(h);
adH:=DifferentialGeometry:-LieAlgebras:-Adjoint(DifferentialGeometry:-
evalDG(add(h[i]*t[i],i=l .. rank)));
(Roots,RootVectors):=LinearAlgebra:-Eigenvectors(adH);
RootSpaces:=[seq('if'(Roots[j]=0,NULL,DifferentialGeometry:-
evalDG(add(evalc(evala(RootVectors[i,j])*g0[i],i=1 .. n))),j=1 .. n)),j=1 .. n)];
h,RootSpaces;
end proc:

Example 2.3: If we want to find the root space decomposition of sl3C with respect to the Cartan subalgebra h as in Example 2.2, then we can use the RootSpaceDecomposition command as follows:

RootSpaceDecomposition([e1, e2]);

[e1, e2], [e5, e7, e3, e4, e6, e8]

Observe that the first list given is a basis for the Cartan subalgebra and that each entry of the second list is a basis for the root space of a root in the root system with respect to the given Cartan subalgebra. Finally, observe that the root space decomposition of sl3C found by the command agrees with the root space decomposition we found for this Lie algebra in Example 2.2.
2.2 Cartan Subalgebras

2.2.1 Regular Elements

As we have seen in the previous section, the key to finding the roots and root spaces of a Lie algebra is to find a Cartan subalgebra. The key to finding a Cartan subalgebra will be to find a regular element.

Definition 2.5: Let $g$ be a Lie algebra, $H$ be any non-zero element in $g$, and let $0 = \lambda_0, \lambda_1, \ldots, \lambda_r$ be the different eigenvalues of $\text{ad}(H)$. For each $\lambda$, consider the subspace

$$g(H, \lambda) = \{ X \in g : (\text{ad}(H) - \lambda I)^k(X) = 0 \text{ for some } k \}.$$  

This subspace is the general eigenspace for $\text{ad}(H)$ corresponding to $\lambda$. In the case that $\lambda = 0$, this space is called the Fitting null component of $\text{ad}(H)$, (see [8]).

Definition 2.6: The element $H \in g$ is called regular if

$$\dim g(H, 0) = \min_{X \in g} (\dim g(X, 0)).$$

Theorem 2.3: For any $X \in g$, $g(X, 0)$ is self-normalizing.

Proof: See [7], page 79.

Theorem 2.4: Let $g$ be a semisimple Lie algebra. If $H$ is regular in $g$, then $g(H, 0)$ is a Cartan subalgebra of $g$. In particular, Cartan subalgebras exist.

Proof: See [6], pages 163-165.

The task of finding a Cartan subalgebra in polynomial time reduces to the task of finding a regular element and computing the Fitting null component of that element.

We will use an algorithmic approach to find a regular element in a semisimple Lie algebra, $g$, so that we can find a Cartan subalgebra in polynomial time. This is a variation of the approach used in [2].
Algorithm 2.2: Regular Element

1. We begin by finding a non-nilpotent element, $X$, of $\mathfrak{g}$. We compute $\mathfrak{g}(X,0)$. From Theorem 2.3, it follows that if $\mathfrak{g}(X,0)$ is nilpotent, then $\mathfrak{g}(X,0)$ is a Cartan subalgebra.

2. If $\mathfrak{g}(X,0)$ is nilpotent, then return $\mathfrak{g}(X,0)$. Otherwise, by Engel's theorem, $\mathfrak{g}(X,0)$ contains a non-nilpotent element. Find a non-nilpotent element, $Y$, of this subalgebra.

3. Let $\Omega = \{1, 2, \ldots, n+1\}$, where $n = \dim \mathfrak{g}$. Choose an element $Z$ from the set $\{X + \beta(Y-X) : \beta \in \Omega\}$ for which $\mathfrak{g}(Z,0)$ is properly contained in $\mathfrak{g}(X,0)$.

4. Let $X := Z$.

5. Go back to step (2).

Upon completion of Algorithm 2.2, $\mathfrak{g}(X,0)$ is nilpotent. As $\mathfrak{g}(X,0)$ is self-normalizing by Theorem 2.3, $\mathfrak{g}(X,0)$ is a Cartan subalgebra. In step 1 of Algorithm 2.2 we need to find $\mathfrak{g}(X,0)$. To compute the Fitting null component of an element we use a result from linear algebra given in Theorem 2.5 and Corollary 2.6 below. Also, in Theorem 2.7 below we show how to find a non-nilpotent element of a Lie algebra for steps 1 and 2. Step 3 guarantees that $\mathfrak{g}(X,0)$ reduces dimension each time the loop is repeated, thus the loop is repeated at most $n$ times. At step 3, the algorithm computes at most the image of $n+1$ transformations. Theorem 2.8 and Proposition 2.9 below show that an element $Z$ can be chosen as in step 3 of Algorithm 2.2.

Theorem 2.5: Let $\mathfrak{g}$ be a Lie algebra, $X \in \mathfrak{g}$, and let $\lambda$ be an eigenvalue of $\text{ad}(X)$ with multiplicity $m$. Then

$$\mathfrak{g}(X,\lambda) = \{Y \in \mathfrak{g} : (\text{ad}(X) - \lambda I)^m(Y) = 0\},$$

and

$$\dim \mathfrak{g}(X,\lambda) = m.$$  

Proof: See [3], page 486 - 487.

Corollary 2.6: Let $\mathfrak{g}$ be a Lie algebra, $X \in \mathfrak{g}$, and let $\dim \mathfrak{g} = n < \infty$. Then

$$\mathfrak{g}(X,0) = \{Y \in \mathfrak{g} : (\text{ad}(X))^n(Y) = 0\}.$$
Proof: Clearly \( \{ Y \in g : (\text{ad}(X))^n(Y) = 0 \} \subseteq g(X,0) \). Let \( m \) be the multiplicity of 0 as an eigenvalue of \( \text{ad}(X) \). By Theorem 2.3, \( g(X,0) = \{ Y \in g : (\text{ad}(X))^m(Y) = 0 \} \). Let \( Z \in g(X,0) \). Then \( (\text{ad}(X))^m(Z) = 0 \) and \( (\text{ad}(X))^n(Z) = (\text{ad}(X))^{n-m}((\text{ad}(X))^m(Z)) = (\text{ad}(X))^{n-m}(0) = 0 \). Thus \( g(X,0) \subseteq \{ Y \in g : (\text{ad}(X))^n(Y) = 0 \} \). \( \square \)

Theorem 2.7: Let \( g \) be a finite-dimensional Lie algebra over a field of characteristic 0. Let \( e_1, e_2, \ldots, e_n \) be a basis for \( g \). If \( g \) is not a nilpotent Lie algebra, then the set \( \{ e_1, \ldots, e_n \} \cup \{ e_i + e_j : 1 \leq i < j \leq n \} \) contains at least one non-nilpotent element of \( g \).

Proof: See [2], page 343. \( \square \)

Theorem 2.8: Let \( S \) and \( T \) be linear transformations \( S, T : V \rightarrow V \). Let

\[
K_0(S) = \{ v \in V : S^k(v) = 0 \text{ for some positive integer } k \},
\]

and suppose \( K_0(S) \) is \( T \)-invariant. Suppose also that \( T \) is an isomorphism on \( V/K_0(S) \) and not nilpotent on \( K_0(S) \). Then \( K_0(T) \) is a proper subset of \( K_0(S) \).

Proof: As \( K_0(S) \) is \( T \)-invariant, \( T \) induces a well-defined linear transformation \( \tilde{T} : V/K_0(S) \rightarrow V/K_0(S) \), defined by \( \tilde{T}(v + K_0(S)) = T(v) + K_0(S) \). Let \( v_0 \in K_0(T) \), then \( T^k(v_0) = 0 \) which implies \( \tilde{T}^k(v_0 + K_0(S)) = \tilde{T}^k(v_0) + K_0(S) = 0 + K_0(S) = K_0(S) \). Thus \( v_0 \) is in the null space of \( \tilde{T}^k \). By hypothesis, \( \tilde{T} \) is an isomorphism, which implies that \( \tilde{T}^k \) is an isomorphism for any positive integer \( k \). Therefore the null space of \( \tilde{T}^k \) is \( 0 + K_0(S) = K_0(S) \), and we can conclude that \( v_0 \in K_0(S) \) which implies \( K_0(T) \subseteq K_0(S) \).

The hypothesis that the restriction of \( T \) to \( K_0(S) \), \( T_{K_0(S)} \), is not nilpotent means \( T_{K_0(S)}^k \neq 0 \) for any positive integer \( k \). This implies that there exists \( w \in K_0(S) \) such that \( T_{K_0(S)}^k(w) \neq 0 \) for any positive integer \( k \), i.e. \( w \not\in K_0(T) \). Therefore \( K_0(T) \) is properly contained in \( K_0(S) \). \( \square \)

Proposition 2.9: We can select \( Z \) as in step 3 of Algorithm 2.2.

Proof: Let \( \text{ad}(X + \beta(Y - X)) = T_\beta \) and note that \( g(X,0) = K_0(\text{ad}(X)) \). By construction, \( K_0(\text{ad}(X)) \) is \( T_\beta \)-invariant for any \( \beta \in \mathbb{C} \). The determinant of \( T_\beta \) in \( g/K_0(\text{ad}(X)) \) is a polynomial of \( \beta \) of degree less than or equal to \( n - \dim K_0(\text{ad}(X)) \). The polynomial is not identically 0 as the determinant of \( \text{ad}(a) = T_0 \) in \( g/K_0(\text{ad}(X)) \) is nonzero. Thus, there are at
most \( n - \dim K_0(\text{ad}(X)) \) roots of the polynomial and there exists a subset \( \Omega' \subset \Omega \) such that 
\[
|\Omega'| \geq (n + 1) - (n - \dim K_0(\text{ad}(X))) = \dim K_0(\text{ad}(X)) + 1,
\]
and for every \( \beta \in \Omega' \), \( \text{ad}(X + \beta(Y - X)) \)
is nonsingular in \( \mathfrak{g}/K_0(\text{ad}(X)) \).

Let
\[
(2.5) \quad h(\lambda) = \lambda^n + h_1(\beta)\lambda^{n-1} + h_2(\beta)\lambda^{n-2} + \ldots + h_{n-r}(\beta)\lambda^r
\]
be the characteristic polynomial of \( T_\beta \) on \( K_0(\text{ad}(X)) \), where \( h_i \) is a polynomial in \( \beta \) of degree \( i \) if \( h_i \neq 0 \) and \( \deg h_i \leq \dim K_0(\text{ad}(X)) \). If \( T_\beta \) is nilpotent on \( K_0(\text{ad}(X)) \), then \( h_i(\beta) = 0 \) for each \( i \). As \( Y = X + l(Y - X) \) is non-nilpotent on \( \mathfrak{g}(X,0) \) by step 2 of Algorithm 2.2, the polynomials \( h_i \) can not all be identically zero. Thus there are at most \( \dim K_0(\text{ad}(X)) \) zeros common to each \( h_i \). As \( |\Omega'| = \dim K_0(\text{ad}(X)) + 1 > \dim K_0(\text{ad}(X)) \), there exists \( \beta_0 \in \Omega' \subset \Omega \) such that \( T_{\beta_0} \) is not nilpotent.

Let \( Z = X + \beta_0(Y - X) \). By Theorem 2.8, \( \mathfrak{g}(Z,0) \) is properly contained in \( \mathfrak{g}(X,0) \).

\[\square\]

### 2.2.2 The FittingNull and NonNilp Commands

In order to implement Algorithm 2.2 to find a regular element and the subsequent Cartan subalgebra we need to be able to compute the Fitting null component of an element and find a non-nilpotent element of a Lie algebra. Using Corollary 2.6, we create a function in Maple to compute the Fitting null component of an element in a Lie algebra \( \mathfrak{g} \).

```maple
> FittingNull:=proc(A)
> local gO, n, adA, GenNullSpace;
> adA:=DifferentialGeometry:-LieAlgebras:-Adjoint(A);
> gO:=DifferentialGeometry:-Tools:-DGinfo("FrameBaseVectors");
> n:=DifferentialGeometry:-Tools:-DGinfo("FrameBaseDimension");
> GenNullSpace:=LinearAlgebra:-NullSpace(adA-n);
> [seq(DifferentialGeometry:-DGzip(convert(GenNullSpace[i],list),gO,"plus"),i=1..nops(GenNullSpace))];
> end proc:
```
We have named our function \texttt{FittingNull}. Upon entry of an element $A$ in the initialized Lie algebra, the function computes the matrix representation of $\text{ad}(A)$, named $\text{ad}A$ in our function. At this point, \texttt{FittingNull} computes the null space of $\text{ad}A^n$. The vectors in the basis of the null space of $\text{ad}A$ are then converted to a basis for the null space of $(\text{ad}(A))^n$, which is the Fitting null component of $\text{ad}(A)$ by Corollary 2.6.

Now we create a function to find a non-nilpotent element of a Lie algebra using Theorem 2.7 which we will call \texttt{NonNilp}.

\begin{verbatim}
> NonNilp:=proc(h)
> local V, a, i;
> V:=[op(h),seq(seq(h[i]+h[j],j=i+1..nops(h)),i=1..nops(h)-1)];
> a:=V[1];
> for i from 1 to nops(V)-1 while
> nops(LinearAlgebra:-NullSpace(DifferentialGeometry:-LieAlgebras:-
> Adjoint(a,h)-(nops(h))))=nops(h) do
> a:=V[i+1]
> end do;
> if
> nops(LinearAlgebra:-NullSpace(DifferentialGeometry:-LieAlgebras:-
> Adjoint(a,h)-(nops(h))))=nops(h) then {} else a end if;
> end proc:
\end{verbatim}

The function compares at most $n + (n - 1) + \ldots + 2 + 1 = \frac{n(n+1)}{2}$ elements and returns a non-nilpotent element of the Lie algebra. As the \texttt{NullSpace} and matrix multiplication commands run in polynomial time, we conclude that \texttt{NonNilp} runs in polynomial time.

2.2.3 The CSA Command

Based on Algorithm 2.2 which finds a regular element in a Lie algebra, we can create a function to find a Cartan subalgebra of a Lie algebra. We will also use the \texttt{NonNilp} and \texttt{FittingNull}
commands of the previous section. As these commands run in polynomial time, and the loop is executed at most \( n \) times, the \texttt{CSA} command will run in polynomial time. As \texttt{CSA} is a common abbreviation for a Cartan subalgebra, we will use this notation for our command to find a Cartan subalgebra. The command is as follows:

```plaintext
> CSA:=proc()
> local n, gO, X, Y, Z, h, h0, i, j;
> if nargs > 1 then
> error "expected 0 to 1 arguments."
> elif nargs = 1 then
> if
> DifferentialGeometry:-LieAlgebras:-Query(args[1], "Semisimple")=false
> then
> error "expected first argument to be a semisimple Lie algebra."
> Received \%i", args[1] else
> DifferentialGeometry:-ChangeFrame(args[1]);
> end if;
> end if;
> g0:=DifferentialGeometry:-Tools:-DGinfo("FrameBaseVectors");
> n:=DifferentialGeometry:-Tools:-DGinfo("FrameBaseDimension");
> X:=NonNilp(g0);
> h:=FittingNull(X);
> for i from 1 while
> DifferentialGeometry:-LieAlgebras:-Query(DifferentialGeometry:-
> LieAlgebras:-SubalgebraNormalizer(h),"Nilpotent")=false do
> Y:=NonNilp(h);
> h0:=FittingNull(Y);
```
for j from 1 while
  DifferentialGeometry:-Tools:-DGequal(DifferentialGeometry:-
  IntersectSubspaces([h0,h]),h0)=false or
  nops(h)<=nops(h0) do
  Z := DifferentialGeometry:-evalDG(X+j*(Y-X));
  h0 := FittingNull(Z);
end do;
X := Z;
h := h0;
end do;
h;
end proc:

Note that we have given ourselves the option to specify the Lie algebra whose Cartan subalgebra
we want to find. If no Lie algebra is specified, the command finds a Cartan subalgebra of the current
initialized Lie algebra.
3. EXAMPLES

Examples are provided in this chapter illustrating the execution of the functions defined in Chapter 2.

3.1 FittingNull and NonNilp

Example 3.1: We will use the `FittingNull` command to find the Fitting null component of an element in $\mathfrak{sl}_3 \mathbb{C} = \mathfrak{sl}_3$. We can initialize this Lie algebra using the `Initialize` command found in Appendix A.

```maple
> Initialize(A, 2);

Lie algebra: sl3
```

The dimension of $\mathfrak{sl}_3 \mathbb{C}$ is 8. The basis elements are $e_1, \ldots, e_8$. We use `FittingNull` to find the Fitting null component of $e_1$.

```maple
> trace(FittingNull);

FittingNull
```

```maple
> FittingNull(e1);

\{--\> enter FittingNull, args = \_DG(["vector", s13, [1], [[[1], 1]]])

\[
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

\[adA := \]

\[g0 := [e1, e2, e3, e4, e5, e6, e7, e8]\n
\[n := 8\]
The Fitting null component is \( \text{Span}\{e_1, e_2\} \). Note that this is the Cartan subalgebra used in Example 2.2. As the Fitting null component of \( e_1 \) is a Cartan subalgebra, \( e_1 \) is a regular element.

**Example 3.2:** We now find the Fitting null component of \( e_1 + e_3 \in \mathfrak{sl}_3 \mathbb{C} \) and call it \( \mathfrak{h} \).

\[
\begin{align*}
\mathfrak{h} &:= \text{FittingNull}(e_1 + e_3); \\
\{ \cdots \} &\Rightarrow \text{enter FittingNull, args} = \text{DG}([[\text{"vector"}, \mathfrak{s}13, []], \{[[2], 1]\}]), \text{DG}([[\text{"vector"}, \mathfrak{s}13, []], \{[[1], 1]\}])
\end{align*}
\]

\[
\begin{align*}
adA &:=
\begin{bmatrix}
0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-2 & 1 & 2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 1
\end{bmatrix} \\
g_0 &:= [e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8]
\end{align*}
\]

\[
n := 8
\]

\[
\begin{align*}
\text{GenNullSpace} &:= \\
\begin{bmatrix}
1 & 0 \\
0 & 1 \\
0 & 0 \\
0 & 0 \\
0 & 0 \\
0 & 0
\end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
\{ \cdots \} &\Rightarrow \text{exit FittingNull (now at top level)} = \text{DG}([[\text{"vector"}, \mathfrak{s}13, []], \{[[1], 1/2], [[2], 1]\}]), \text{DG}([[\text{"vector"}, \mathfrak{s}13, []], \{[[1], 1], [[3], 1]\}])
\end{align*}
\]
Thus \( \mathfrak{h} = \text{Span}\{\frac{e_1}{2} + e_2, e_1 + e_3\} \). Note that the dimension of \( \mathfrak{h} \) is 2. As the dimension of the Fitting null component of \( e_1 \) is 2 and, by Definition 2.6, the Fitting null component of a regular element is minimal, we conclude that \( e_1 + e_3 \) is a regular element. We use the \texttt{Query} command to verify that this subalgebra is nilpotent, and therefore a Cartan subalgebra.

\[
\text{DifferentialGeometry:-LieAlgebras:-Query(h, "Nilpotent");}
\]
\[
\text{true}
\]

In Example 3.10 we will compute the root space decomposition of \( \mathfrak{s}_3 \mathbb{C} \) with respect to this Cartan subalgebra.

Now we compute the Fitting null component of \( e_3 \in \mathfrak{s}_3 \mathbb{C} \).

\[
\text{FittingNull(e3);}
\]
\[
\text{true}
\]

\[
\text{GenNullSpace :=}\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

\[
\text{n := 8}
\]

\[
\text{[e_8, e_7, e_6, e_5, e_4, e_3, e_2, e_1]}
\]
\[ \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{pmatrix} \]

\[ \text{adA} := \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \]

g0 := [e1, e2, e3, e4, e5, e6, e7, e8]

\[
\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}
\]

n := 8

\[
\begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}
\]

The Fitting null component of \(e3\) is all of \(s_{13}\). In particular, the \(g(e3,0)\) is not minimal, which implies that \(e3\) is not a regular element.

\section*{Example 3.3:} We now use NonNilp to find a non-nilpotent element in \(s_{13}\). Again, we will trace the function to visualize the execution.

\[ \text{trace(NonNilp);} \]

NonNilp

\[ \text{Initialize(A,2);} \]
Lie algebra: \( \mathfrak{sl}3 \)

\[ g0 := \text{DifferentialGeometry:-Tools:-DGinfo("FrameBaseVectors")}; \]

\[ g0 := \{e1, e2, e3, e4, e5, e6, e7, e8\} \]

\[ \text{NonNilp}(g0); \]

\[ \{\text{-} \rightarrow \text{enter NonNilp, args = \{._DG(["vector", s13, [], [[[[1], 1]]]), _DG(["vector", s13, [], [[[[2], 1]]]), _DG(["vector", s13, [], [[[[3], 1]]]), _DG(["vector", s13, [], [[[[4], 1]]]), _DG(["vector", s13, [], [[[[5], 1]]]), _DG(["vector", s13, [], [[[[6], 1]]]), _DG(["vector", s13, [], [[[[7], 1]]]), _DG(["vector", s13, [], [[[[8], 1]]])\}}\} \]

\[ V := \{e1, e2, e3, e4, e5, e6, e7, e8, e1 + e2, e1 + e3, e1 + e4, e1 + e5, e1 + e6, e1 + e7, e1 + e8, e2 + e3, e2 + e4, e2 + e5, e2 + e6, e2 + e7, e2 + e8, e3 + e4, e3 + e5, e3 + e6, e3 + e7, e3 + e8, e4 + e5, e4 + e6, e4 + e7, e4 + e8, e5 + e6, e5 + e7, e5 + e8, e6 + e7, e6 + e8, e7 + e8\} \]

\[ a := e1 \]

\[ c1 \]

\[ \langle- \text{exit NonNilp (now at top level) = \{_DG(["vector", s13, [], [[[[1], 1]]])\}} \]

By Theorem 2.7, the list \( V \) contains a non-nilpotent element. The command verified that the first element of the list, \( e1 \), is not a nilpotent element of \( \mathfrak{sl}3 \mathfrak{C} \). That is to say, \( \text{ad}(e1) \) is not nilpotent. To view an example where the first element in the list fails to be non-nilpotent, we can reorder the basis.

\[ \text{NonNilp([e3, e4, e5, e6, e7, e8, e1, e2]);} \]

\[ \{\text{-} \rightarrow \text{enter NonNilp, args = \{._DG(["vector", s13, [], [[[[3], 1]]]), _DG(["vector", s13, [], [[[[4], 1]]]), _DG(["vector", s13, [], [[[[5], 1]]]), _DG(["vector", s13, [], [[[[6], 1]]]), _DG(["vector", s13, [], [[[[7], 1]]]), _DG(["vector", s13, [], [[[[8], 1]]])\}}\} \]

\[ V := \{e3, e4, e5, e6, e7, e8, e1, e2, e3 + e4, e3 + e5, e3 + e6, e3 + e7, e3 + e8, e3 + e1, e3 + e2, e4 + e5, e4 + e6, e4 + e7, e4 + e8, e4 + e1, e4 + e2, e5 + e6, e5 + e7, e5 + e8, e5 + e1, e6 + e2, e6 + e7, e6 + e8, e6 + e1, e6 + e2, e7 + e8, e7 + e1, e7 + e2, e8 + e1, e8 + e2, e1 + e2\} \]

\[ a := e3 \]

\[ a := e4 \]

\[ a := e5 \]
We see that $e_3, e_4, e_5, e_6, e_7$, and $e_8$ are nilpotent elements, and that $e_1$ is non-nilpotent just as before.

Example 3.4: We can also use the *NonNilp* to find a non-nilpotent element of a subalgebra of $sl_3\mathbb{C}$. Let us use the command to find a non-nilpotent element of the subalgebra spanned by $e_1$ and $e_2$.

```plaintext
> NonNilp([e1,e2]);

V := [e1, e2, e1 + e2]

The command shows that the list $V$ does not contain a non-nilpotent element in the subalgebra. By Theorem 2.7, this subalgebra does not contain a non-nilpotent element, i.e. every element of this subalgebra is nilpotent. Recall from Example 2.2 that this subalgebra is a Cartan subalgebra, in particular it is nilpotent. By Engel’s theorem, every element of a nilpotent Lie algebra is nilpotent. The command confirms that there are no non-nilpotent elements in the subalgebra.
3.2 CSA

Example 3.5: We will use the **CSA command** to find a Cartan subalgebra of $sl_4 \mathbb{C}$. Note that $sl_4 \mathbb{C}$ is the Lie algebra $\mathfrak{sl}_4$.

```wolfram
> Initialize(A,3);
```

*Lie algebra: sl4*

We want to be able to see the execution of each step of the algorithm. We will use the `trace` function for this.

```wolfram
> trace(CSA);
```

**CSA**

Now we can use the **CSA command** to compute a Cartan subalgebra.

```wolfram
> CSA();
```

```wolfram
<-- enter CSA, args =

$ g0 := [e1, e2, e3, e4, e5, e6, e7, e8, e9, e10, e11, e12, e13, e14, e15]$

$ n := 15$

$ X := e1$

$ h := [e15, e9, e3, e2, e1]$

$ Y := e3$

$ h0 := [e3, e4, e10, e1, e2]$

$ Z := e3$

$ h0 := [e1, e2, e3, e4, e10]$

$ Z := -e1 + 2 e3$

$ h0 := [e3, e2, e1]$

$ X := -e1 + 2 e3$

$ h := [e3, e2, e1]$

$ [e3, e2, e1]$

<-- exit CSA (now at top level) = [_DG(["vector", sl4, []], [[[3], 1]]), _DG(["vector", sl4, []], [[[2], 1]]), _DG(["vector", sl4, []], [[[1], 1]]])
```
The algorithm found that $X = e_1$ was a non-nilpotent element of the Lie algebra. It then computed $g(X, 0) = \text{Span}\{e_{15}, e_9, e_3, e_2, e_1\}$. As indicated by the index, $\mathfrak{sl}_4 \mathbb{C} = e_3$ has rank 3, thus $g(X, 0)$ is not a Cartan subalgebra. The algorithm proceeds to find an element $Y = e_3$ which is non-nilpotent in $g(X, 0)$. It then finds an element $Z = -e_1 + 2e_3$ such that $g(Z, 0) \subset g(X, 0)$, as in Algorithm 2.2. The function then sets $X$ equal to $Z$. As the Fitting null component of the newly defined $X$ is nilpotent, the loop ends and $\mathfrak{h} = \text{Span}\{e_3, e_2, e_1\}$ is a Cartan subalgebra of $\mathfrak{sl}_4 \mathbb{C}$.

**Example 3.6:** The next example, $b_3 = \mathfrak{so}_7 \mathbb{C}$, illustrates the execution of the loop twice.

```
> Initialize(B,3);

Lie algebra: so7

> CSA();

{--> enter CSA, args =

$g_0 := [e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{12}, e_{13}, e_{14}, e_{15}, e_{16}, e_{17}, e_{18}, e_{19}, e_{20}, e_{21}]$

$n := 21$

$X := e_1$

$h := [e_{21}, e_{17}, e_{18}, e_{20}, e_{15}, e_9, e_{12}, e_6, e_3, e_2, e_1]$

$Y := e_3$

$h_0 := [e_2, e_1, e_{20}, e_{17}, e_{19}, e_{16}, e_{13}, e_{10}, e_4, e_7, e_9]$

$Z := e_3$

$h_0 := [e_4, e_7, e_{10}, e_{13}, e_{20}, e_{16}, e_{17}, e_{19}, e_1, e_2, e_9]$

$Z := -e_1 + 2e_3$

$h_0 := [e_3, e_1, e_{20}, e_{17}, e_2]$

$X := -e_1 + 2e_3$

$h := [e_3, e_1, e_{20}, e_{17}, e_2]$

$Y := e_2$

$h_0 := [e_2, e_{14}, e_{21}, e_{16}, e_{11}, e_8, e_5, e_3, e_1, e_{19}, e_{18}]$
```
\[
Z := e2
\]
\[
h0 := [e18, e19, e21, e2, e1, e3, e8, e5, e14, e11, e16]
\]
\[
Z := e1 + 2 e2 - 2 e3
\]
\[
h0 := [e3, e12, e1, e2, e15]
\]
\[
Z := 2 e1 + 3 e2 - 4 e3
\]
\[
h0 := [e2, e1, e3]
\]
\[
X := 2 e1 + 3 e2 - 4 e3
\]
\[
h := [e2, e1, e3]
\]
\[
[e2, e1, e3]
\]
\[
←← exit CSA (now at top level) = \{DG(["vector", so7, [1]], [[2], [1]]), DG(["vector", so7, [3]], [[1], [1]]), DG(["vector", so7, [1]], [[3], [1]])\}
\]
\[
[e2, e1, e3]
\]

Example 3.7: This final example, \( \mathfrak{d}_4 = so_8 \mathbb{C} \), illustrates the execution of the loop twice and a failure to find an element \( Z \) as in step 3 of Algorithm 2.2 on the first attempt.

> Initialize(D,4);

\[
\text{Lie algebra: } so8
\]

> CSA();

←← enter CSA, args =

\[
\]
\[
n := 28
\]
\[
X := e1
\]
\[
\]
\[
Y := e2
\]
\[
\]
\[
Z := e2
\]
\[
\]
\[
Z := -e1 + 2 e2
\]
\[ h_0 := [e_{28}, e_{10}, e_1, e_2, e_3, e_4, e_{16}, e_{22}] \]
\[ X := -e_1 + 2e_2 \]
\[ h := [e_{28}, e_{10}, e_1, e_2, e_3, e_4, e_{16}, e_{22}] \]
\[ Y := e_3 \]
\[ h_0 := [e_{20}, e_{11}, e_1, e_{21}, e_2, e_3, e_4, e_5, e_{23}, e_8, e_9, e_{13}, e_{25}, e_{27}, e_{15}, e_{17}] \]
\[ Z := e_3 \]
\[ h_0 := [e_2, e_3, e_4, e_5, e_8, e_9, e_{11}, e_{13}, e_{15}, e_{17}, e_{20}, e_{21}, e_{23}, e_{25}, e_{27}, e_{11}] \]
\[ Z := e_1 - 2e_2 + 2e_3 \]
\[ h_0 := [e_1, e_2, e_3, e_4, e_{19}, e_{26}] \]
\[ Z := 2e_1 - 4e_2 + 3e_3 \]
\[ h_0 := [e_3, e_1, e_2, e_4] \]
\[ X := 2e_1 - 4e_2 + 3e_3 \]
\[ h := [e_3, e_1, e_2, e_4] \]
\[ [e_3, e_1, e_2, e_4] \]

3.3 RootSpaceDecomposition

Now that we have an effective CSA command, we can use the RootSpaceDecomposition command.

Example 3.8: We find the root space decomposition of \( a_2 = \mathfrak{sl}_3 \mathbb{C} \) with respect to the Cartan subalgebra spanned by \( \{e_1, e_2\} \).

> Initialize(A,2);

\textit{Lie algebra: \mathfrak{sl}3} 

> trace(RootSpaceDecomposition);
RootSpaceDecomposition

> RootSpaceDecomposition([e1,e2]);

$$\left\langle \text{RootSpaceDecomposition, \text{args} = \left[ \_DG(["vector", s13, []], [[1], 1])), _DG(["vector", s13, []], [[2], 1])]) \right\rangle$$

g0 := [e1, e2, e3, e4, e5, e6, e7, e8]

$$n := 8$$

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 2t_1 - t_2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & t_1 + t_2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -t_1 + 2t_2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -2t_1 + t_2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & t_1 - t_2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & t_1 - 2t_2
\end{bmatrix}$$

\[\text{Roots, RootVectors} := \left[\begin{array}{c}
t_1 - 2t_2 \\
t_1 + t_2 \\
-t_1 + 2t_2 \\
2t_1 - t_2 \\
-t_1 - t_2 \\
-2t_1 + t_2 \\
0 \\
0
\end{array}\right], \left[\begin{array}{c}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right]\]

\[\text{RootSpaces} := [e_8, e_4, e_5, e_3, e_7, e_6]

\[\left[\begin{array}{c}
e_1, e_2 \end{array}\right], \left[\begin{array}{c}
e_8, e_4, e_5, e_3, e_7, e_6\end{array}\right]\]

Note that this root space decomposition is the same as the one computed in Example 2.2.

Example 3.9: For sl_4 \mathbb{C} = a_4, we will use the Cartan subalgebra found in Example 3.5.

> Initialize(A,3);

\[\text{Lie algebra: } sl_4\]

> RootSpaceDecomposition([e1,e2,e3]);
```plaintext
(→→ enter RootSpaceDecomposition, args = [_DG(["vector", sl4, []]), [[[1], 1]])], _DG(["vector", sl4, []], [[[2], 1]])], _DG(["vector", sl4, []], [[[3], 1]])])]

g0 := [e1, e2, e3, e4, e5, e6, e7, e8, e9, e10, e11, e12, e13, e14, e15]
n := 15
rank := 3

adH :=
[15 Element Column Vector
Data Type : algebraic
Storage : rectangular
Order : Fortran_order]

Roots, RootVectors :=
[15 x 15 Matrix
Data Type : anything
Storage : rectangular
Order : Fortran_order]

RootSpaces := [e6, e9, e12, e7, e5, e11, e15, e14, e8, e13, e4, e10]

Example 3.10: In this example we find the root space decomposition of sl₃C with respect to the Cartan subalgebra ℎ in Example 3.2.

> RootSpaceDecomposition(h);

(→→ enter RootSpaceDecomposition, args = [_DG(["vector", sl3, []]), [[[1], 1/2]], [[[2], 1]])], _DG(["vector", sl3, []], [[[1], 1]], [[[3], 1]])])]

g0 := [e1, e2, e3, e4, e5, e6, e7, e8]
n := 8
rank := 2
```
Note that this root space decomposition of $\mathfrak{sl}_3 \mathbb{C}$ is different than the one found in Example 2.2. The root space decomposition of a Lie algebra is not unique. It is dependent on the Cartan subalgebra chosen.

Example 3.11: In this example we make a change of basis to $\mathfrak{sl}_3 \mathbb{C}$.

```maple
> Initialize(A,2);

Lie algebra: sl3
```

\[ L2 := \left[ [e1, e2] = -e2 - \frac{2e5}{3} - \frac{17e6}{2} + \frac{4e7}{3} + \frac{193e8}{3}, \right. \]
\[ [e1, e3] = \frac{3e1}{2} + \frac{11e2}{2} + \frac{7e3}{2} - \frac{3e4}{2} + \frac{9e5}{2} - \frac{21e6}{4} - \frac{9e7}{2} + 46e8, \]
\[ [e1, e4] = \frac{-2e1 - 11e2 - \frac{2e3}{3} - \frac{e4}{3} - \frac{2e5}{9} + \frac{25e6}{2} + \frac{14e7}{3} - \frac{1022e8}{9}, \]
\[ [e1, e5] = -4e2 - 8e5 + 3e6 - e7 - 45e8, \]
\[ [e1, e6] = 8e2 - 19e6 + 10e7 + 234e8, \]
\[ [e1, e7] = -e2 + 2e6 - 25e8, \]
\[ [e1, e8] = e2 - \frac{3e6}{2} + e7 + 22e8, \]
\[ [e2, e3] = -\frac{9e1}{2} + \frac{29e2}{2} - e3 + \frac{e5}{2} + \frac{39e6}{2} + \frac{9e7}{2} - \frac{313e8}{9}, \]
\[ [e2, e4] = \frac{2e2}{3} + \frac{e4}{3} + \frac{20e5}{9} + \frac{4e7}{3} + \frac{59e8}{9}, \]
\[ [e2, e5] = 3e2 + 2e5 - \frac{9e6}{2} + 3e7 + 46e8, [e2, e6] = e6 - 26e7 - 50e8, \]
\[ [e2, e7] = 2e7 + 4e8, [e2, e8] = -3e7 - 5e8, \]
\[ [e3, e4] = \frac{e1}{2} + \frac{e2}{2} - \frac{e5}{6} - \frac{e6}{2} - e7 + \frac{17e8}{6}, \]
\[ [e3, e5] = e2 + \frac{9e4}{2} - \frac{21e6}{4} + \frac{95e8}{2}, [e3, e6] = 3e1 + e2 - e5 - 6e7 - 9e8, \]
\[ [e3, e7] = e2 - 2e6 + 14e8, [e3, e8] = -e2 + \frac{3e6}{2} - 13e8, \]
\[ [e4, e5] = -3e2 - \frac{2e3}{3} - \frac{2e5}{9} - \frac{e6}{2} - \frac{5e7}{3} + \frac{364e8}{9}, \]
\[ [e4, e6] = \frac{16e3}{3} + \frac{4e5}{9} - \frac{46e7}{3} - \frac{134e8}{9}, \]
\[ [e4, e7] = e1 + 5e2 - \frac{2e3}{3} - \frac{5e5}{9} - 7e6 - \frac{e7}{3} + \frac{532e8}{9}, \]
\[ [e4, e8] = \frac{2e3}{3} + \frac{2e5}{9} - \frac{5e7}{3} - \frac{13e8}{9}, [e5, e6] = 24e2 - 36e6 - 2e7 + 310e8, \]
\[ [e5, e7] = -3e2 + 6e6 - 51e8, [e5, e8] = 3e2 - \frac{9e6}{2} + 39e8, \]
\[ [e6, e7] = 2e7 + 2e8, [e6, e8] = -2e7 - 2e8 \]

\[ > \text{DifferentialGeometry:-DGsetup}(L2); \]

\[ \text{Lie algebra : sl3e2} \]

Next we use CSA to find a Cartan subalgebra in this basis.

\[ > \text{h:=CSA}(); \]
\[ h := [e1, \frac{96e2}{1063} - \frac{8e5}{1063} - \frac{291e6}{2126} - \frac{165e7}{1063} + e8] \]

Now we can find the root space decomposition of sl3C in this basis with respect to the Cartan subalgebra.

\[ > \text{RootSpaceDecomposition}(h); \]

\[ > \]
\[
[e_1, \frac{96e_2}{1063} - \frac{8e_5}{1063} - \frac{291e_6}{2126} - \frac{165e_7}{1063} + e_8], \left(\frac{669}{63401} - \frac{543\sqrt{17}}{63401}\right) e_2
\]
\[
+ \left(\frac{2185}{63401} - \frac{501\sqrt{17}}{63401}\right) e_5 + \left(-\frac{6204}{63401} - \frac{1017\sqrt{17}}{63401}\right) e_6 + \left(\frac{3378}{63401} - \frac{183\sqrt{17}}{63401}\right) e_7
\]
\[
+ e_8, \left(\frac{669}{63401} - \frac{543\sqrt{17}}{63401}\right) e_2 + \left(\frac{501\sqrt{17}}{63401} - 2185\right) e_5 + \left(-\frac{6204}{63401} + \frac{1017\sqrt{17}}{63401}\right) e_6
\]
\[
+ \left(\frac{3378}{63401} + \frac{183\sqrt{17}}{63401}\right) e_7 + e_8, \quad \frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8,
\]
\[
\frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8, \quad \frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8,
\]
\[
\frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8, \quad \frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8,
\]
\[
\frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8, \quad \frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8,
\]
\[
\frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8, \quad \frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8,
\]
\[
\frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8, \quad \frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8,
\]
\[
\frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8, \quad \frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8,
\]
\[
\frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8, \quad \frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8,
\]
\[
\frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8, \quad \frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8,
\]
\[
\frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8, \quad \frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8,
\]
\[
\frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8, \quad \frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8,
\]
\[
\frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8, \quad \frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8,
\]
\[
\frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8, \quad \frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8,
\]
\[
\frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8, \quad \frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8,
\]
\[
\frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8, \quad \frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8,
\]
\[
\frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8, \quad \frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8,
\]
\[
\frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8, \quad \frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8,
\]
\[
\frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8, \quad \frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8,
\]
\[
\frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8, \quad \frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8,
\]
\[
\frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8, \quad \frac{\sqrt{17} e_6}{34} + (1 + \frac{4\sqrt{17}}{17}) e_7 + e_8,
4. SIMPLE LIE ALGEBRA STRUCTURE

4.1 Root Systems

In the previous chapter, we investigated the root systems of $\mathfrak{sl}_2 \mathbb{C}$ and $\mathfrak{sl}_3 \mathbb{C}$. We should observe that if $\Delta = \{\alpha_1, \alpha_2, \ldots\}$ is the root system with respect to the Cartan subalgebra $\mathfrak{h}$, then $\Delta \subset \mathfrak{h}^*$, where $\mathfrak{h}^*$ represents the dual space of $\mathfrak{h}$. Jacobson [8] calls the real vector space spanned by the roots $\mathfrak{h}_0^*$. For each $\alpha_i \in \Delta$ we can choose a vector $e_{\alpha_i} \in \mathfrak{g}^{\alpha_i}$, where $e_{\alpha_i}$ generates $\mathfrak{g}^{\alpha_i}$. Recall that dim $\mathfrak{h}$ is called the rank of the Lie algebra. Suppose dim $\mathfrak{g} = n$. As $\mathfrak{g} = \mathfrak{h} \oplus (\bigoplus_{\alpha_i \in \Delta} \mathfrak{g}^{\alpha_i})$, if $\{h_1, h_2, \ldots, h_r\}$ is a basis for $\mathfrak{h}$, then we can choose as a basis for $\mathfrak{g}$ the set $\{h_1, \ldots, h_r, e_{\alpha_1}, \ldots, e_{\alpha_{n-r}}\}$. In this basis, the adjoint of an arbitrary element $H \in \mathfrak{h}$ takes on the form:

\[
\text{ad} (H) = \begin{pmatrix}
0 & \cdots & 0 \\
\cdots & \ddots & \cdots \\
0 & \cdots & \alpha_1(H) \\
\alpha_{n-r}(H) & \cdots & \alpha_{n-r}(H)
\end{pmatrix}.
\]

There are several advantages of this basis. We see that the first $r$ elements form a Cartan subalgebra. Also, if we take $H$ to be the arbitrary element $H = \sum_{i=1}^{r} t_i h_i$, we can see that the the last $n - r$ diagonal entries are the roots $\alpha_i$ evaluated at $H$.

4.1.1 Positive and Simple Roots

The roots span a real vector space $\mathfrak{h}_0^*$. Jacobson [8] shows that dim $\mathfrak{h}_0^* = \text{dim} \mathfrak{h} = r = \text{rank} \mathfrak{g}$. Thus we can choose $r$ elements from $\Delta$ which form a basis for $\mathfrak{h}_0^*$.

**Definition 4.1:** Let $\gamma = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ be a basis of roots for $\mathfrak{h}_0^*$. We call $\rho = \sum_{i=1}^{r} \mu_i \alpha_i$ positive if the first nonzero $\mu_i$ is positive. We call $\rho$ negative if the first nonzero $\mu_i$ is negative.
By Theorem 2.2, if $\alpha \in \Delta$ then $-\alpha \in \Delta$. Thus we can split $\Delta$ into two equal subsets of positive and negative roots. From the positive roots we can sort out the "simple" roots.

**Definition 4.2:** A positive root is said to be *simple* if it is not the sum of two positive roots.

**Example 4.1:** Let us find the simple roots of the root system, $\Delta$, of $s\mathfrak{sl}_3 \mathbb{C}$ with respect to the Cartan subalgebra, $\mathfrak{h}$, spanned by $e1, e2$. The roots are listed in (2.2).

First, we must find a basis for $\mathfrak{h}^*$. One can easily check that $\{\alpha_1, \alpha_2\}$ forms such a basis. Now we must find the positive roots. To do this, we write each root as a linear combination of $\alpha_1$ and $\alpha_2$:

\[
\begin{align*}
\alpha_1 &= 1 \cdot \alpha_1 + 0 \cdot \alpha_2, \\
\alpha_2 &= 0 \cdot \alpha_1 + 1 \cdot \alpha_2, \\
\alpha_3 &= -1 \cdot \alpha_1 + 1 \cdot \alpha_2, \\
\alpha_4 &= -1 \cdot \alpha_1 + 0 \cdot \alpha_2, \\
\alpha_5 &= 0 \cdot \alpha_1 - 1 \cdot \alpha_2, \\
\alpha_6 &= 1 \cdot \alpha_1 - 1 \cdot \alpha_2.
\end{align*}
\]  

(4.2)

We can see that $\alpha_1$, $\alpha_2$, and $\alpha_6$ are all positive roots. Finally, we check our set of positive roots for roots that are the sum of two other positive roots. We observe that $\alpha_2 + \alpha_6 = \alpha_1$, thus $\alpha_1$ is not simple. Since $\alpha_1 + \alpha_2$ and $\alpha_1 + \alpha_6$ are not roots, we conclude that $\alpha_2$ and $\alpha_6$ are simple roots relative to the basis chosen. Note that if we choose another basis for $\mathfrak{h}^*$, we could end up with a different set of simple elements.

**Theorem 4.1:** If $\alpha$ and $\beta$ are two simple roots, then $\beta - \alpha$ is not a root.

*Proof:* Suppose $\gamma = \beta - \alpha$ is a root. Then either $\gamma > 0$ or $\gamma < 0$. If $\gamma > 0$, then $\gamma + \alpha = \beta - \alpha + \alpha = \beta$, which contradicts simplicity of $\beta$. Likewise, if $\gamma < 0$, then $-\gamma > 0$ and $-\gamma + \beta = -\beta + \alpha + \beta = \alpha$, which contradicts simplicity of $\alpha$. \hfill $\square$

**Theorem 4.2:** Let $\{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ be the set of simple roots of the root space $\Delta$ with respect to a Cartan subalgebra $\mathfrak{h}$ of a Lie algebra $\mathfrak{g}$. Then $\{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ forms a basis for $\mathfrak{h}^*$ and for each $\alpha \in \Delta$, $\alpha = \sum_{i=1}^{r} \mu_i \alpha_i$ where each $\mu_i$ is positive [resp. negative] if $\alpha$ is positive [resp. negative].

*Proof:* See [6] and [7].
4.2 The Killing Form

Recall Definition 1.10 of the Killing form.

Theorem 4.3: The restriction of the Killing form to the Cartan subalgebra \( \mathfrak{h} \) is nondegenerate.

Proof: See [7], page 37.

By Theorem 4.3, for each \( \alpha \in \mathfrak{h}^* \), there exists a unique vector \( H_\alpha \in \mathfrak{h} \) such that

\[
B(H, H_\alpha) = \alpha(H), \quad \text{for all } H \in \mathfrak{h}.
\]

If \( h_1, h_2, \ldots, h_r \) is a basis for \( \mathfrak{h} \), then \( H = \sum_{i=1}^r \lambda_i h_i, \quad H_\alpha = \sum_{i=1}^r \mu_i h_i, \) and \( \alpha(H) = \sum_{i=1}^r c_i \lambda_i \), for some \( \lambda_i, \mu_i, c_i \in \mathbb{C} \). Let \( [B_\mathfrak{h}] \) be the matrix representation of the Killing form restricted to \( \mathfrak{h} \). Then

\[
B(H, H_\alpha) = [\lambda_1 \lambda_2 \cdots \lambda_r \mu_1 \mu_2 \cdots \mu_r],
\]

which implies

\[
[\lambda_1 \lambda_2 \cdots \lambda_r \mu_1 \mu_2 \cdots \mu_r] [B_\mathfrak{h}] = \alpha(H) = \sum_{i=1}^r c_i \lambda_i = [\lambda_1 \lambda_2 \cdots \lambda_r] [c_1 c_2 \cdots c_r].
\]

If

\[
[B_\mathfrak{h}] = \begin{bmatrix}
\mu_1 \\
\mu_2 \\
\vdots \\
\mu_r
\end{bmatrix},
\]

then (4.5) holds. Since the Killing form restricted to \( \mathfrak{h} \) is nondegenerate, \( [B_\mathfrak{h}] \) is invertible. Therefore,

\[
\begin{bmatrix}
\mu_1 \\
\mu_2 \\
\vdots \\
\mu_r
\end{bmatrix} = [B_\mathfrak{h}]^{-1} \begin{bmatrix}
c_1 \\
c_2 \\
\vdots \\
c_r
\end{bmatrix}.
\]

Since \( H_\alpha \) is unique, (4.7) provides an algorithm to find \( H_\alpha \). In particular, we see that the mapping \( \alpha \rightarrow H_\alpha \) is a bijective map.

Example 4.2: Let us compute the vectors \( H_\alpha \), for each \( \alpha \) in the root system for \( \mathfrak{sl}_3 \mathbb{C} \), given by (2.2). Recall that a basis for \( \mathfrak{h} \) was \( e1, e2 \). If \( [B_\mathfrak{h}] \) is the matrix representation of the Killing form
restricted to $\mathfrak{h}$, then $[B_{\mathfrak{h}}]$ is a $2 \times 2$ matrix and $[B_{\mathfrak{h}}]_{i,j} = B(e_i, e_j)$. We see that

$$[B_{\mathfrak{h}}] = \begin{pmatrix} 12 & -6 \\ -6 & 12 \end{pmatrix} \Rightarrow [B_{\mathfrak{h}}]^{-1} = \begin{pmatrix} \frac{1}{9} & \frac{1}{18} \\ \frac{1}{18} & \frac{1}{9} \end{pmatrix},$$

and

$$\begin{pmatrix} \frac{1}{9} & \frac{1}{18} \\ \frac{1}{18} & \frac{1}{9} \end{pmatrix} \begin{pmatrix} 2 \\ -1 \end{pmatrix} = \begin{pmatrix} \frac{1}{6} \\ 0 \end{pmatrix} \begin{pmatrix} \frac{1}{9} & \frac{1}{18} \\ \frac{1}{18} & \frac{1}{9} \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{6} \\ 0 \end{pmatrix}.$$
is called the Cartan Matrix of $\Delta$.

For example, if $\mathfrak{g} = sl_3 \mathbb{C}$ and we order the simple roots $\alpha_2, \alpha_6$, then the Cartan Matrix is

$$(4.11) \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$$

The matrix will depend on the ordering of the simple roots, however it is independent of our choice of basis for $\Delta$. In fact, the Cartan Matrix determines $\Delta$ up to isomorphism, and by extension $\mathfrak{h}$ and $\mathfrak{g}$ (see [7], page 55).

The outline for computing these matrices is as follows:

1. **Use the CSA command to find a Cartan subalgebra of the Lie algebra.**

2. **Find the root system with respect to the Cartan subalgebra found in the first step.** We can find the adjoint of a generic element $H = \sum t_i h_i$. The roots $\alpha_i(H)$ are the nonzero eigenvalues of $\text{ad}(H)$.

3. **Find the simple roots with respect to some basis for the root system.** First we need to find a basis for the root system. We do this by computing a vector representation for the root system and using the Basis command of Maple's LinearAlgebra package. We then proceed to write each root as a linear combination of the roots in this basis. To find the positive roots, we are only interested in the nonzero coefficients of this linear combination, as such we eliminate all of the 0 coefficients. The first nonzero coefficient indicates whether the root is positive or negative.

By Definition 4.2, the simple roots are the positive roots that cannot be written as the sum of two other positive roots. As such, a positive root $\alpha$ is simple if and only if the difference of $\alpha$ and $\beta$ is not a positive root for any positive root $\beta$.

4. **Compute the Cartan matrix.** To accomplish this task we find $H_\alpha$, for each simple root $\alpha_i$. We do this by computing the inverse of the matrix of the Killing form restricted to $\mathfrak{h}$. The Killing command in Maple's LieAlgebras package can do this given a basis a basis for $\mathfrak{h}$ as the argument. After we have found this matrix, we can compute $H_\alpha$, as given by (4.7).
Now we can compute the Cartan Matrix, $A$, where

$$A_{ij} = \frac{2\langle \alpha_i, \alpha_j \rangle}{\langle \alpha_j, \alpha_j \rangle} = 2\frac{B(H_{\alpha_i}, H_{\alpha_j})}{B(H_{\alpha_j}, H_{\alpha_j})}$$

### 4.3.1 The CartanMatrix Command

```maple
> CartanMatrix:=proc ()
local Alg1, n, h, rank, L2, H, adH, Delta, v, beta, B, NonzeroCoefs,
PosRoots, SimpRoots, V, M, h_alpha, CMatrix;
description "Find the Cartan Matrix of a semisimple Lie Algebra";
if nargs > 1 then error "expected 0 to 1 arguments." end if;
if nargs = 1 then
if
member(args[1],DifferentialGeometry:-Tools:-DGinfo("FrameNames"))=
false then
error "expected first argument to be a valid Lie algebra name.
Received %1",args[1]
else DifferentialGeometry:-ChangeFrame(args[1]);
end if;
end if;
n:=DifferentialGeometry:-Tools:-DGinfo("FrameBaseDimension");
h:=CSA();
r:=nops(h);
H:=DifferentialGeometry:-evalDG(add(h[i]*t[i],i=1..rank));
adH:=DifferentialGeometry:-LieAlgebras:-Adjoint(H);
Delta:=convert(convert(LinearAlgebra:-Eigenvalues(adH,output='list'),
set) minus {0},list);
v:=[seq(Vector([seq(coeff(Delta[j],t[i]),i=1..rank)]),j=1..(n-rank))];
```

The CartanMatrix command without an argument computes the Cartan matrix of the root system of the current Lie algebra initialized in Maple. Whereas the command can have a single argument where the argument is a Lie algebra that has been initialized previously in Maple.

4.4 Dynkin Diagrams

The Cartan matrix, $A$, is a useful tool in the classification of Lie algebras. As stated, the root system of a Lie algebra is completely determined by $A$. 
Theorem 4.4: Regardless of the ordering of the simple roots, the diagonal entries of a Cartan Matrix $A$ are always 2.

Proof: This is immediate from Definition 4.4 of $A$. \qed

Theorem 4.5: If $A$ is a Cartan matrix, then $A_{ij} = 0$ if and only if $A_{ji} = 0$.

Proof: If $A_{ij} = 0$, then by Definition 4.4, $B(H_{\alpha_i}, H_{\alpha_j}) = 0$, which implies that

$$\text{Tr} (\text{ad} (H_{\alpha_i}) \circ \text{ad} (H_{\alpha_j})) = \text{Tr} (\text{ad} (H_{\alpha_j}) \circ \text{ad} (H_{\alpha_i})) = 0. \quad (4.13)$$

Thus,

$$B(H_{\alpha_j}, H_{\alpha_i}) = B(H_{\alpha_i}, H_{\alpha_j}) = 0, \quad (4.14)$$

and $A_{ji} = 0$.

Similarly, if $A_{ji} = 0$, then $A_{ij} = 0$. \qed

Theorem 4.6: The only possible values for the non-diagonal entries are 0, $-1$, $-2$, and $-3$. If $A_{ij} = -2$ or $A_{ij} = -3$, then $A_{ji} = -1$.

Proof: See [8], pages 117-121. \qed

We can associate the Cartan matrix, $A$, of a root system with a figure known as a Dynkin diagram. To construct the Dynkin diagram of a root system we consider the Coxeter graph of the root system. This graph assigns a vertex to each simple root $\alpha_1, \alpha_2, \ldots, \alpha_r$ in the root system. The number of edges between the vertices assigned to $\alpha_i$ and $\alpha_j$ is equal to the product $A_{ij}A_{ji}$. To this graph, we assign a weight to the vertex for $\alpha_i$ proportional to $\langle \alpha_i, \alpha_i \rangle$. If $\mathfrak{g}$ is a simple Lie algebra, there are at most two different weights. We darken the vertices of the smaller roots. The resulting figure is called a Dynkin Diagram.

Recall that the Cartan matrix associated with $\mathfrak{sl}_3\mathbb{C}$ was

$$A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \quad (4.15)$$

Since there are two simple roots corresponding to this root system, $\alpha_2$ and $\alpha_6$, the Coxeter graph has two vertices. The number of edges between the two vertices is equal to $A_{12}A_{21} = (-1)(-1) = 1$. 

The graph is

(4.16) \[
\begin{array}{c}
\ast_2 \\
\ast_5
\end{array}
\]

As \(\langle \alpha_2, \alpha_2 \rangle = \langle \alpha_5, \alpha_5 \rangle = \frac{1}{2}\), the vertices have equal weight. Thus (4.16) is the Dynkin diagram for this root system. It should be noted that if two vertices of the Dynkin diagram share a single edge, then the two corresponding roots have equal length.

It is advantageous to consider the Dynkin diagram of a root system as the Cartan matrix depends on the ordering of the simple roots. The Dynkin diagram, however, is unique (up to isomorphism). From the Dynkin diagram it is not hard to construct a somewhat canonical Cartan matrix of the root system of the Lie algebra. Thus, from the Dynkin diagram we can reconstruct the root system, and by extension the Lie algebra itself.

4.5 Cartan Matrices and Dynkin diagrams of the Complex Simple Lie Algebras

Using the commands in Appendix A, let us initialize the Lie algebras \(\mathfrak{sl}_4\mathbb{C} = \mathfrak{a}_3\). As indicated by the subscript of \(\mathfrak{a}\), we expect the rank of this Lie algebra to be 3.

> Initialize(A,3);

\textit{Lie algebra}: \(\mathfrak{sl}_4\)

There are three possibilities for the Cartan matrix.

> CartanMatrix();

\[
\begin{pmatrix}
2 & 0 & -1 \\
0 & 2 & -1 \\
-1 & -1 & 2
\end{pmatrix}
\]

> CartanMatrix();

\[
\begin{pmatrix}
2 & -1 & -1 \\
-1 & 2 & 0 \\
-1 & 0 & 2
\end{pmatrix}
\]

> CartanMatrix();

\[
\begin{pmatrix}
2 & -1 & 0 \\
-1 & 2 & -1 \\
0 & -1 & 2
\end{pmatrix}
\]

Although the matrices are different, the corresponding Dynkin diagrams are isomorphic. Let us investigate each. First, as there are three simple roots, let us label them \(\alpha_1, \alpha_2, \alpha_3\), in that order.
The matrix

\[
\begin{pmatrix}
2 & 0 & -1 \\
0 & 2 & -1 \\
-1 & -1 & 2
\end{pmatrix}
\]

indicates that the vertices for \(\alpha_1\) and \(\alpha_2\) each share a single edge with \(\alpha_3\).

As a single edge connects each, the weights of the vertices are equal.

Now let \(\beta_1, \beta_2,\) and \(\beta_3\) be another ordering of the simple roots corresponding to the Cartan matrix

\[
\begin{pmatrix}
2 & -1 & -1 \\
-1 & 2 & 0 \\
-1 & 0 & 2
\end{pmatrix}
\]

Thus the vertices for \(\beta_2\) and \(\beta_3\) each share a single edge with the vertex for \(\beta_1\). The respective Dynkin diagram is

Finally, if \(\gamma_1, \gamma_2,\) and \(\gamma_3\) is the ordering of the simple roots corresponding to the Cartan matrix

\[
\begin{pmatrix}
2 & -1 & 0 \\
-1 & 2 & -1 \\
0 & -1 & 2
\end{pmatrix}
\]

Then the corresponding Dynkin diagram is

We see that each of the diagrams are isomorphic, i.e. they have the same number of vertices and edges and respective weights.

Table 4.1 provides an exhaustive list of the complex simple Lie algebras and the corresponding Dynkin diagram, up to isomorphism (see [7], pages 57-58). The table also lists the Cartan matrices of the Lie algebras, as constructed from the Dynkin diagram.
Tab. 4.1: Cartan Matrices and Dynkin diagrams of the Simple Lie Algebras

| $\alpha_n$, $n \geq 1$ | \[ \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & \cdot \\ \cdot & \cdot & -1 & -1 & 0 \\ \cdot & \cdot & \cdot & -1 & -1 \\ 0 & \cdot & \cdot & 0 & -1 \end{pmatrix} \] | [ ]
| $\beta_n$, $n \geq 2$ | \[ \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & \cdot \\ \cdot & \cdot & -1 & -1 & 0 \\ \cdot & \cdot & \cdot & -1 & -1 \\ 0 & \cdot & \cdot & 0 & -1 \end{pmatrix} \] | [ ]
| $\gamma_n$, $n \geq 3$ | \[ \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & \cdot \\ \cdot & \cdot & -1 & -1 & 0 \\ \cdot & \cdot & \cdot & -1 & -1 \\ 0 & \cdot & \cdot & 0 & -1 \end{pmatrix} \] | [ ]
| $\delta_n$, $n \geq 4$ | \[ \begin{pmatrix} 2 & 0 & -1 & 0 & 0 & 0 \\ 0 & 2 & 0 & -1 & 0 & 0 \\ -1 & 0 & 2 & -1 & 0 & 0 \\ 0 & -1 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & -1 & 2 \end{pmatrix} \] | [ ]
| $\varepsilon_6$ | \[ \begin{pmatrix} 2 & 0 & -1 & 0 & 0 & 0 \\ 0 & 2 & 0 & -1 & 0 & 0 \\ -1 & 0 & 2 & -1 & 0 & 0 \\ 0 & -1 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & -1 & 2 \end{pmatrix} \] | [ ]
| $\varepsilon_7$ | \[ \begin{pmatrix} 2 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 2 & -1 & 0 & 0 & 0 \\ 0 & -1 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 \end{pmatrix} \] | [ ]
| $\varepsilon_8$ | \[ \begin{pmatrix} 2 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & -1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \end{pmatrix} \] | [ ]
| $\mathfrak{f}_4$ | \[ \begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix} \] | [ ]
| $\mathfrak{g}_2$ | \[ \begin{pmatrix} 2 & -3 \\ -1 & 2 \end{pmatrix} \] | [ ]
4.6 Classification of Simple Lie Algebras

Given a simple Lie algebra $\mathfrak{g}$, our goal is to classify $\mathfrak{g}$ as one of the Lie algebras in Table 4.1. Luckily, most of this classification can be made by dimensional analysis. The following table shows the dimension of the simple Lie algebras through rank 10.

<table>
<thead>
<tr>
<th>$r$</th>
<th>$\mathfrak{a}_r$</th>
<th>$\mathfrak{b}_r$</th>
<th>$\mathfrak{c}_r$</th>
<th>$\mathfrak{d}_r$</th>
<th>$\mathfrak{e}_r$</th>
<th>$\mathfrak{f}_r$</th>
<th>$\mathfrak{g}_r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>8 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>15 21 21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>24 36 36 28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>52</td>
</tr>
<tr>
<td>5</td>
<td>35 55 55 45</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>48 78 78 66 78</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>63 105 105 91 133</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>80 136 136 120 248</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>99 171 171 153</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>120 210 210 190</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

From the table, we can see that the dimensions of the simple Lie algebras are almost unique. For example, $\mathfrak{g}_2$ is the only simple Lie algebra of dimension 14. Likewise, the dimension of $\mathfrak{a}_r$ is unique for each $r$, with the exception of $\mathfrak{a}_{10}$ which has the same dimension as $\mathfrak{d}_8$. We can use the CSA command to determine the rank of the algebra, and therefore, after a dimensional analysis, we can compare the rank to determine if the Lie algebra is isomorphic to $\mathfrak{a}_{10}$ or $\mathfrak{d}_8$.

Note that $\dim \mathfrak{b}_r = \dim \mathfrak{c}_r$ for every $r$. To determine if a given complex simple Lie algebra is isomorphic to $\mathfrak{b}_r$ or $\mathfrak{c}_r$, we can compare the Cartan matrix of the given Lie algebra to the Cartan matrices of $\mathfrak{b}_r$ and $\mathfrak{c}_r$. Note that in the Dynkin diagram of $\mathfrak{b}_r$ that there is a string of vertices at the end of which is a double edge to a smaller vertex. In the Dynkin diagram of $\mathfrak{c}_r$ the weights are reversed. Therefore, for comparison, we generate the Cartan matrix of the Lie algebra.
As there is a double edge, there is a row which contains \(-2\). The simple root corresponding to this row has a greater weight than the simple root in the respective column. If there is a \(-1\) in the row containing \(-2\), then this vertex is connected to one of equal weight. It follows that the Dynkin diagram will have a string of vertices of equal weight, at the end of which is a double edge to a smaller vertex, and so the Lie algebra is isomorphic to \(\mathfrak{b}_r\). Otherwise, the Lie algebra is isomorphic to \(\mathfrak{c}_r\).

Finally, there is the case where \(\dim \mathfrak{b}_6 = \dim \mathfrak{c}_6 = \dim \mathfrak{e}_6 = 78\). Again, we can compare the dimension of a 78-dimensional simple Lie algebra to the Cartan matrices of these Lie algebras. Note that the Dynkin diagram of \(\mathfrak{e}_6\) only has single edges, therefore \(-2\) is not contained in any row. If \(-2\) is contained in the Cartan matrix, then the Lie algebra is isomorphic to \(\mathfrak{b}_6\) or \(\mathfrak{c}_6\) and we can do a comparison as explained previously.

All other simple Lie algebras under rank 10 have unique dimension. For rank greater than 10, we can compute the rank \(r\) of the Lie algebra. If the degree of the Lie algebra is \(n\), and \(r = \sqrt{n+1} - 1\), then the Lie algebra is isomorphic to \(\mathfrak{a}_r\). If \(r = (1 + \sqrt{8n+1})/4\), then the Lie algebra is isomorphic to \(\mathfrak{v}_r\). Otherwise, the Lie algebra is isomorphic to \(\mathfrak{b}_r\) or \(\mathfrak{c}_r\) and we can investigate the Cartan matrix as explained. We can now write a function to classify a given complex simple Lie algebra.

```maple
> Classify:=proc()
> local n,h,CM,V,r;unassign('A', 'B', 'C', 'E', 'F', 'G');
> if nargs > 1 then
> error "expected 0 to 1 arguments."
> elif nargs = 1 then
> DifferentialGeometry:-ChangeFrame(args[1]);
> end if;
> n:=DifferentialGeometry:-Tools:-DGinfo("FrameBaseDimension");
> if member(n,[3,8,15,24,35,48,63,80,99]) then
> A[sqrt(n+1)-1]
> elif member(n,[28,45,66,91,153,190]) then
```
```plaintext
> D((1+sqrt(8*n+1))/4)
> elif n=133 then
>  E[7]
>  elif n=248 then
>  E[8]
>  elif n=14 then
>  G[2]
>  elif n=52 then
>  F[4]
>  elif n=120 then
>  h:=CSA();
>  r:=nops(h);
>  if r=10 then
>    A[10] else
>    D[8]
>  end if;
>  elif n=78 then
>  CM:=CartanMatrix();
>  if member(-2,convert(CM,set))=false then
>    E[6]
>  else
>    V:=seq('if'(member(-2,convert(CM,listlist)[i])=true,convert(CM,
>    listlist)[i],NULL),i=1..6);
>    if member(-1,V)=true then
>    B[6] else
>    C[6]
```
> end if

> end if

> elif member(n, [10, 21, 36, 55, 105, 136, 171, 210]) then

> r:=-1+sqrt(8*n+1))/4;

> CM:=CartanMatrix();

> V:=seq('if'(member(-2, convert(CM, listlist)[i])=true, convert(CM, listlist)[i], NULL), i=1..r);

> if member(-1, V)=true then

> B[r] else

> C[r]

> end if;

> else

> h:=CSA();

> r:=nops(h);

> if evalb(r = sqrt(n+1)-1)=true then A[r]

> elif evalb(r = (1+sqrt(8*n+1))/4)=true then D[r]

> elif evalb(r = (-1+sqrt(8*n+1))/4)=true then

> CM:=CartanMatrix();

> V:=seq('if'(member(-2, convert(CM, listlist)[i])=true, convert(CM, listlist)[i], NULL), i=1..r);

> if member(-1, V)=true then

> B[r] else

> C[r]

> end if;

> end if;
Examples of the execution of this function are given in Chapter 5. To reduce computation time, the function does not verify if the Lie algebra is simple. The downside is that any 3-dimensional Lie algebra is classified as \( \mathfrak{sl}_2 \mathbb{C} \), for example. This verification can easily be done. By using the Query command we can verify if a function is semisimple and indecomposable. If so, the Lie algebra is simple.
5. EXAMPLES

5.1 CartanMatrix

Example 5.1: Here we compute the Cartan matrix of $\mathfrak{sl}_3\mathbb{C}$ using the CartanMatrix command.

First, we initialize $\mathfrak{sl}_3\mathbb{C} = \mathfrak{a}_2$.

$$\text{Initialize}(A, 2);$$

**Lie algebra**: $\mathfrak{sl}_3$

Now we will use the trace feature to see the execution of the command.

$$\text{trace}(\text{CartanMatrix});$$

CartanMatrix

$$\text{CartanMatrix}();$$

{--enter CartanMatrix, args =

\begin{align*}
&n := 8 \\
&h := \{e_2, e_1\} \\
&\text{rank} := 2
\end{align*}

$$H := t_2 e_1 + t_1 e_2$$

$$adH :=
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 2 t_2 - t_1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & t_2 + t_1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -t_2 + 2 t_1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -2 t_2 + t_1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -t_2 - t_1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -t_2 - 2 t_1 & 0
\end{bmatrix}$$

$$\Delta := \{t_2 + t_1, -t_2 + 2 t_1, -2 t_2 + t_1, -t_2 - t_1, 2 t_2 - t_1, t_2 - 2 t_1\}$$

$$v := \{1, 2, -1, -2, -1, -2, -1, 2\}$$

$$\beta := \left\{\begin{bmatrix}1 \\ 1\end{bmatrix}, \begin{bmatrix}2 \\ -1\end{bmatrix}\right\}$$
NonzeroCoeffs := [[1], [1], [-1, 1], [1, -1], [-1, -1], [1]]

PosRoots := \{t_2 + t_1, -t_2 + 2t_1, 2t_2 - t_1\}

SimpRoots := \{-t_2 + 2t_1, 2t_2 - t_1\}

V := \begin{bmatrix}
2 \\
-1
\end{bmatrix}, \begin{bmatrix}
-1 \\
2
\end{bmatrix}

B := \begin{bmatrix}
12 & -6 \\
-6 & 12
\end{bmatrix}

h_{alpha} := \begin{bmatrix}
e_2/6 \\
e_1/6
\end{bmatrix}

CMatrix := \begin{bmatrix}
2 & -1 \\
-1 & 2
\end{bmatrix}

The function first found a Cartan subalgebra \( \mathfrak{h} \) of \( \mathfrak{sl}_3 \mathbb{C} \), which is the same as the one in Example 2.2. The function then computed \( \text{ad}(H) \) for a generic element \( H \in \mathfrak{h} \). From there, it computed the root system \( \Delta \) of \( \mathfrak{sl}_3 \mathbb{C} \) with respect to \( \mathfrak{h} \). Now \( \nu \) is a vector representation of this root system and \( \beta \) is a vector representation of a basis. Using this basis, we were able to compute positive and simple roots of \( \Delta \). \( V \) is a vector representation of the simple roots. The function then computed the Killing form \( B \) restricted to \( \mathfrak{h} \). It then computed \( H_{\alpha} \) for each of the simple roots and computed the Cartan matrix.

Example 5.2: In this example we find the Cartan matrix of \( \mathfrak{g}_2 \) and then make a change of basis and show that the Cartan matrix is independent of basis. The \texttt{trace} feature has been turned off.

\begin{verbatim}
> Initialize(G,2);

Lie algebra: G2

> CartanMatrix();

\begin{bmatrix}
2 & -3 \\
-1 & 2
\end{bmatrix}
\end{verbatim}

We can verify that this is the Cartan matrix in Table 4.5. We now make a change of basis to \( \mathfrak{g}_2 \) and call the Lie algebra \( G2v2 \) in \texttt{Maple}.
> L2:=DifferentialGeometry:-LieAlgebras:-LieAlgebraData([e1-3*e3+5*e5+e6,e2-e7+e8,e3+3*e4+8*e5,e4-2*e6+e8,e5-e7+2*e8,e6-e8,e7+e8,2*e8,3*e9-2*e14,e10-e12+e14,e11-e13,e12+e13,e13-2*e14,2*e14],G2v2):
> DifferentialGeometry:-DGsetup(L2);

\[
\text{Lie algebra: } G2v2
\]

We now compute the Cartan matrix of \( g_2 \) in the new basis.

> CartanMatrix();

\[
\begin{bmatrix}
2 & -3 \\
-1 & 2
\end{bmatrix}
\]

We see that we get the same Cartan Matrix as before.

### 5.2 Classify

**Example 5.3:** We use the `Classify` function to classify \( \text{so}_5 \mathbb{C} \). Note \( \text{so}_5 \mathbb{C} = b_2 \).

> Initialize(B,2);

\[
\text{Lie algebra: } \text{so}_5
\]

> trace(Classify);

\[
\text{Classify}
\]

> Classify();

\[
\{
\text{--> enter Classify, args =}
\]

\[
\]

\[
\{n := 10, r := 2, CM := \begin{bmatrix} 2 & -1 \\ -2 & 2 \end{bmatrix}, V := [-2, 2] \}
\]

\[
\text{C}_2
\]

\[
\text{C}_2
\]

Thus \( \text{so}_5 \mathbb{C} \) was classified as \( c_2 \). This is due to the isomorphism \( b_2 \cong c_2 \).

**Example 5.4:** To verify that the function differentiates between \( b_r \) and \( c_r \), we classify \( \text{so}_7 \mathbb{C} = b_3 \).

> Initialize(B,3);

\[
\text{Classify}
\]

\[
\{n := 7, r := 3, CM := \begin{bmatrix} 2 & -1 & 0 \\ -2 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}, V := [-2, 2, 2] \}
\]

\[
\text{C}_3
\]

\[
\text{C}_3
\]

Thus \( \text{so}_7 \mathbb{C} \) was classified as \( c_3 \). This is due to the isomorphism \( b_3 \cong c_3 \).
Lie algebra: \textit{so7}

\texttt{Classify();}
\{\texttt--> enter Classify, args =}

\texttt{n := 21}
\texttt{r := 3}
\texttt{CM := \begin{bmatrix} 2 & 0 & -1 \\ 0 & 2 & -1 \\ -2 & -1 & 2 \end{bmatrix}}
\texttt{V := [-2, -1, 2]}

\texttt{B3}
\texttt{\textit{<-- exit Classify (now at top level) = B[3]}}
\texttt{B3}

The function classifies \textit{so7} C as \textit{b3}, as expected.
REFERENCES


APPENDICES
A. CLASSICAL LIE ALGEBRAS

In this section we define commands in Maple to initialize the classical Lie algebras. We use the bases found in Helgason [6], pages 186-189.

1. \( a_n = \mathfrak{sl}_{n+1} \mathbb{C} \)

\[ \text{sl} := \text{proc} (n::\text{posint}) \text{ local } A, i, j; \]
\[ \text{description} \quad \text{"initializes the algebra } \mathfrak{sl}(n)\"; \]
\[ A := [\text{seq} (\text{Matrix}(n, \{(i, i) = 1, (i+1, i+1) = -1\}), i = 1 .. n-1), \text{seq}(\text{seq}( \]
\[ \text{seq}(\text{seq}(\text{Matrix}(n, \{(i, j) = 1\}), i = j+1 .. n), j = 1 .. n), \]
\[ \text{DifferentialGeometry:-DGsetup(DifferentialGeometry:-LieAlgebras[} \]
\[ \text{LieAlgebraData}(A, \text{sl||ln}) \text{ end proc; } \]

2. \( b_n = \mathfrak{so}_{2n+1} \mathbb{C} \) and \( \mathfrak{b}_n = \mathfrak{so}_{2n} \mathbb{C} \)

\[ \text{so} := \text{proc} (n::\text{posint}) \]
\[ \text{local } B,D; \]
\[ \text{description} \quad \text{"initializes the algebra } \mathfrak{so}(n)\"; \]
\[ \text{if} \text{ type}(n, \text{odd}) \text{ then } \]
> B:=[seq(Matrix(n,[(2*j-1,2*j)=1,(2*j,2*j-1)=-1]),j=1..(n-1)/2),
> seq(seq(Matrix(n,
> [(2*j-1,2*k-1)=1,(2*j,2*k)=1,(2*j-1,2*k)=I,(2*j,2*k-1)=-I
> ],shape=antisymmetric),j=1..k-1),k=1..(n-1)/2),
> seq(seq(Matrix(n,
> [(2*j-1,2*k-1)=1,(2*j,2*k)=1,(2*j-1,2*k)=I,(2*j,2*k-1)=I
> ],shape=antisymmetric),j=1..k-1),k=1..(n-1)/2),
> seq(seq(Matrix(n,
> [(2*j-1,2*k-1)=1,(2*j,2*k)=-1,(2*j-1,2*k)=I,(2*j,2*k-1)=I
> ],shape=antisymmetric),j=1..k-1),k=1..(n-1)/2),
> seq(seq(Matrix(n,
> [(2*j-1,2*k-1)=1,(2*j,2*k)=-1,(2*j-1,2*k)=-I,(2*j,2*k-1)=-I
> ],shape=antisymmetric),j=1..k-1),k=1..(n-1)/2),
> seq(Matrix(n,[(2*j-1,n)=1,(2*j,n)=I
> ],shape=antisymmetric),j=1..(n-1)/2),
> seq(Matrix(n,[(2*j-1,n)=1,(2*j,n)=-I
> ],shape=antisymmetric),j=1..(n-1)/2)];

> DifferentialGeometry:-DGsetup(DifferentialGeometry:-LieAlgebras:-
> LieAlgebraData(B,sol In));

> else
D := [seq(Matrix(n, 
{(2*j-1,2*j)=1,(2*j,2*j-1)=-1}),j=1..(n/2)),
seq(seq(Matrix(n,
{(2*j-1,2*k-1)=1,(2*j,2*k)=1,(2*j-1,2*k)=I,(2*j,2*k-1)=-I
},shape=antisymmetric),j=1..k-1),k=1..(n/2)),
seq(seq(Matrix(n,
{(2*j-1,2*k-1)=1,(2*j,2*k)=1,(2*j-1,2*k)=I,(2*j,2*k-1)=-1
},shape=antisymmetric),j=k+1..(n/2)),k=1..(n/2)),
seq(seq(Matrix(n,
{(2*j-1,2*k-1)=1,(2*j,2*k)=1,(2*j-1,2*k)=I,(2*j,2*k-1)=I
},shape=antisymmetric),j=1..k-1),k=1..(n/2)),
seq(seq(Matrix(n,
{(2*j-1,2*k-1)=1,(2*j,2*k)=1,(2*j-1,2*k)=I,(2*j,2*k-1)=I
},shape=antisymmetric),j=k+1..(n/2)),k=1..(n/2))];
DifferentialGeometry:-DGsetup(DifferentialGeometry:-LieAlgebras:-
LieAlgebraData(D, so_{10}));
end if end proc;
3. \( c_n = sp_{10}C \)

sp := proc(n::posint)
local C;
description "initializes the algebra \( sp(n) \);"
> C:=
> seq(seq(Matrix(n-2,{{i,i}=1,({n+i,n+i})=-1}},i=1..n),
> seq(seq(Matrix(n-2,{{n+i,j}=1,({n+j,i})=1}},i=1..n),j=1..n),
> seq(seq(Matrix(n-2,{{i,n+j}=1,({j,n+i})=1}},i=1..n),j=1..n),
> seq(seq(Matrix(n-2,{{i,j}=1,({n+j,n+i})=-1}},i=1..n),j=1..n),
> seq(seq(Matrix(n-2,{{i,j}=1,({n+j,n+i})=-1}},i=j+1..n),j=1..n));
> DifferentialGeometry:-DGsetup(DifferentialGeometry:-LieAlgebras:-
> LieAlgebraData(C,sp||n));
> end proc;

The notation $a_n, b_n, c_n$, and $\vartheta_n$ is useful as it indicates the dimension of the root space. For example, the rank of $\mathfrak{sl}_{n+1}C$ is $n$. For convenience, we will create a command to initialize these simple Lie algebras. The command will also have the option to initialize the simple Lie algebra $\mathfrak{g}_2$, whose structure equations can be found in [4], page 346.

> Initialize:=proc(class,n);
> if class=A
> then sl(n+1)
> elif class=B
> then so(2*n+1)
> elif class=C
> then sp(n)
> elif class=D
> then so(2*n)
> elif (class,n)=(G,2)
> then
> unassign('H1','H2','X1','Y1','X2','Y2','X3','Y3','X4','Y4','X5','Y5',
> 'X6','Y6');
> end proc;
This command allows us to initialize the simple Lie algebra in any series \( a - d \) and \( g_2 \) by specifying the rank.