ALGORITMOS PARA ÁLGEBRAS ASSOCIATIVAS E DE LIE

Ricardo Miranda Martins

Monografia apresentada ao IMPA - Instituto Nacional de Matemática Pura e Aplicada, com o objetivo de propor apresentação de trabalho nas **II Jornadas de Iniciação Científica no IMPA**.

SETEMBRO/2005

Este trabalho foi apresentado ao CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico, na condição de Relatório Final do projeto de Iniciação Científica "A Complexidade Computacional dos Algoritmos para Álgebras Associativas e de Lie", financiado pelo órgão supracitado, e desenvolvido na Universidade Federal de Viçosa, pelo autor e sob a orientação da Profa. Dra. Marinês Guerreiro e do Prof. Rogério Picanço, no período de agosto de 2004 a julho de 2005.

RESUMO

ALGORITMOS PARA ÁLGEBRAS ASSOCIATIVAS E DE LIE

O estudo da complexidade computacional e o desenvolvimento de bons algoritmos para cálculos em álgebras de Lie teve início por volta de 1975. Os melhores resultados que foram obtidos para álgebras de Lie de dimensão finita sobre corpos de característica p, $p > 3$, ou de característica 0, se devem ao trabalho de classificação que foi feito com tais álgebras. Nestes casos, as implementações computacionais já são uma realidade, principalmente nos softwares de computação algébrica GAP e MAGMA.

No presente trabalho é estudada a teoria de álgebras associativas e de Lie de um ponto de vista computacional. Concomitantemente ao estudo teórico, foi utilizado o software GAP para efetuar cálculos com as álgebras e suas estruturas.

Ainda no que tange ao GAP, foi feito um estudo dos algoritmos nele presentes, inclusive com análise da complexidade dos mesmos. Este trabalho foi baseado principalmente nos artigos de membros do *CIRCA - Centre for Interdisciplinary Research in Computational Algebra*, na Universidade de St. Andrews, Escócia, e do *Johann Radon Institute for Computational and Applied Mathematics*, na Áustria, que estão na linha de frente da pesquisa de algoritmos para álgebras.

Após o estudo acima descrito, foram produzidos dois algoritmos para álgebras de Lie, um para o cálculo do diagrama de Dynkin de uma álgebra de Lie simples e outro para a definição de uma representação irredutível da álgebra de Lie $\mathbf{sl}(2, \mathbb{F})$, onde $\mathbb{F}$ é um corpo qualquer.

ABSTRACT

ALGORITHMS FOR ASSOCIATIVE AND LIE ALGEBRAS

The study of computational complexity and the development of good algorithms for calculations on Lie algebras began in the 70's. The best results were obtained for finite dimensional Lie Algebras over fields of prime characteristic (greater than 3) or 0, thanks to the well known classification of these algebras. These algorithms are successfully implemented in algebraic computational softwares like GAP and MAGMA.

In this work we studied the theory of associative and Lie algebras from a computational point of view. Together with the theoretical study, the software GAP was used to perform calculations with the algebras and their structure.

Also in the GAP scenario, a study about the implemented algorithms was conducted, including an analysis of their complexity. The main references of this work are papers of researchers from *CIRCA - Centre for Interdisciplinary Research in Computational Algebra*, from St. Andrews University, Scotland, and from *Johann Radon Institute for Computational and Applied Mathematics*, Austria, who actually lead the research on algorithms for algebras.

The last part of this work includes a construction of two algorithms for Lie algebras: one for drawing the Dynkin's diagram of a simple Lie algebra and other dealing with irreducible representations of the Lie algebra $\mathbf{sl}(2, \mathbb{F})$ over a field $\mathbb{F}$.

We (he and Halmos) share a philosophy about linear algebra: we think basis-free, we write basis-free, but when the chips are down we close the office door and compute with matrices like fury.
- Irving Kaplansky

Sumário

Introdução

Um pouco de história

As álgebras de Lie formam o aparato básico da teoria conhecida por "Teoria de Lie". Esse nome é devido ao fato de que as álgebras de Lie surgiram na década de 1870 com as pesquisas do matemático norueguês *Sophus Lie* (1842-1899) dentro de seu programa de estender às equações diferenciais a Teoria de *Galois* existente para as equações algébricas.[1]

Em torno de 1860 a teoria dos grupos de permutações de um conjunto finito estava se desenvolvendo e começava a ser usada (*Serret, Kronecker, Mathieu, Jordan*). Por outro lado, a teoria dos invariantes avançava e alguns matemáticos investigavam certos conjuntos infinitos de transformações geométricas, como as transformações lineares e projetivas. Por volta de 1869, Lie recebe a contribuição de *Felix Klein* (1849-1925), após descobrir que eles tinham interesses matemáticos em comum. Neste período, Lie concebeu uma de suas idéias mais originais, a introdução da noção de invariante na Análise e Geometria Diferencial. Ele e Klein produziram trabalhos motivados por idéias semelhantes, onde era possível perceber que esses pesquisadores foram profundamente influenciados pelas teorias de *Galois* e *Jordan*.

Posteriormente, Lie dedicou-se ao estudo das transformações infinitesimais e dos grupos contínuos e, assim, suas pesquisas concentravam-se no campo das equações diferenciais. Para Lie, a teoria dos grupos de transformações era uma ferramenta para as equações diferenciais que desempenhava papel análogo ao grupo de Galois para as equações algébricas.

A idéia inicial era estender às equações diferenciais a Teoria de Galois. Devido às suas descobertas, Lie abandona seu objetivo inicial voltado para as equações diferenciais e passa a pesquisar as álgebras de Lie. O vasto campo desta teoria desenvolveu-se a partir do conhecimento matemático dos "grupos infinitesimais", descobertos por Lie.

Os resultados pioneiros desta nova teoria, posteriormente denominados de *Teoremas de Lie*, estabelecem a relação entre grupos de transformações, hoje denominados grupos de Lie, e as álgebras de Lie, através da aplicação exponencial.

Fazendo analogia com a teoria dos grupos de permutações, Lie introduziu os conceitos de subgrupos, subgrupos normais, homomorfismos sobrejetivos e mostrou que eles correspondem às noções de subálgebra, ideais e homomorfismos sobrejetivos de álgebras de Lie, respectivamente. Outras noções foram detalhadas em trabalhos escritos juntamente com *F. Engel* e foram introduzidos os grupos derivados e solúveis e a relação entre os produtos nas álgebras e comutadores

[1] Estamos seguindo o exposto na introdução de [SILVA], que foi adaptado do interessantíssimo artigo [FRITZSCHE].

nos grupos. Assim, as álgebras de Lie aparecem como objetos infinitesimais associados aos grupos de transformações com o produto da álgebra correspondendo ao comutador do grupo.

Desta forma, a teoria recebe uma consideração mais algébrica. Os grupos de Lie têm natureza geométrica enquanto que as álgebras de Lie são naturalmente objetos algébricos. O termo "álgebra de Lie" foi introduzido a partir da década de 1920 por *Hermann Weyl* (por sugestão de *Nathan Jacobson*), em substituição à expressão "grupo infinitesimal".

Finalmente, podemos dizer que resultados interessantes foram surgindo com as pesquisas de *Engel*, *Killing* e *Cartan*, dentre outros. Atualmente a Teoria de Lie forma um vasto e coerente campo do conhecimento matemático e possui ramificações nas diversas áreas da Matemática e de suas aplicações.

Motivação deste trabalho

Como em toda teoria matemática, após a sistematização dos resultados básicos, começa o surgimento das aplicações em outras áreas. Neste sentido, poucas subáreas na Álgebra têm sido tão aplicadas a outras áreas da Matemática e das Ciências Físicas quanto as Álgebras de Lie. Estas aplicações envolvem cálculos longos e complicados, cálculos estes que, como bem disse Leibnitz (ainda que acerca de *outros* cálculos), *"poderiam ser relegados, com segurança, a qualquer um que usasse uma máquina"*. E é isto que vem sendo feito ultimamente.

Os grupos de pesquisa pioneiros em algoritmos para Teoria de Grupos se voltaram para as Álgebras Associativas e de Lie e uma abordagem semelhante parece funcionar. Os algoritmos para tais estruturas já resolvem os problemas básicos da teoria e as heurísticas existentes também têm obtido muito sucesso.

Objetivos

O objetivo deste trabalho é analisar os algoritmos e heurísticas existentes para realizar cálculos em estruturas algébricas, tanto do ponto de vista das Álgebras Associativas e de Lie, justificando cada passo dos algoritmos, quanto do ponto de vista computacional, criticando sua complexidade.

Metodologia

A fim de cumprir os objetivos citados anteriormente, num primeiro momento foi feito um estudo da teoria necessária para compreender os conceitos envolvidos nos desenvolvimentos das Álgebras Associativas e de Lie, baseando nas referências bibliográficas clássicas a cerca do assunto, como [JACOBSON] e [HUMPHREYS], além de [SanMARTIN], este último principalmente no que se refere aos pré-requisitos como Representações e Produto Tensorial.

Já num segundo momento foi feita uma pesquisa de artigos atuais e/ou que tenham importância histórica e prática, a fim de ter uma visão do estado da arte da área, o que possibilitou um contato com os tipos de algoritmos existentes para efetuar os cálculos necessários nestas estruturas algébricas. Concomitantemente, o software GAP foi usado para efetuar cálculos nestas estruturas algébricas.

Após adquirida uma bagagem teórica sobre álgebras associativas e de Lie, além de prática com o software GAP estudamos alguns tópicos de Complexidade de Algoritmos, para poder, finalmente, analisar o código-fonte dos algoritmos, compreendendo seu funcionamento e sua complexidade.

Preliminares Algébricas

Neste capítulo introduziremos alguns conceitos que serão usados no decorrer do texto.

Álgebras

A palavra "álgebra" é uma distorção do título de um tratado arábico de al-Khwarizmi sobre métodos algébricos. Atualmente, "álgebra" tem vários significados.

Em um nível elementar, álgebra pode designar o estudo de polinômios e das propriedades das operações fundamentais. Já em textos mais avançados, a palavra "álgebra" é usada para designar o estudo abstrato dos sistemas numéricos e de suas operações, incluindo tópicos como grupos, anéis e cohomologia. Este é o sentido mais utilizado pelos matemáticos para a palavra "álgebra". Para evitar ambigüidades, os tópicos acima são referidos muitas vezes como "álgebra abstrata". Para uma discussão mais filosófica a cerca do tema, veja [POINCARÉ]. Para nós, a palavra álgebra será para um tipo particular de estrutura algébrica.

Definição 0.1 *Uma **álgebra** é um espaço vetorial $\mathcal{V}$ sobre um corpo $\mathbb{F}$ com uma multiplicação interna. Esta multiplicação deve ser distributiva e, para cada $\alpha \in \mathbb{F}$ e $x, y \in \mathcal{V}$ deve satisfazer $\alpha(xy) = (\alpha x)y = x(\alpha y)$.*

Uma álgebra é dita **associativa** se, para quaisquer elementos $x, y, z \in \mathcal{V}$, vale $x(yz) = (xy)z$.

Exemplo 0.2 *Os números reais constituem uma álgebra, onde a multiplicação é a usual. Da mesma maneira, qualquer corpo $\mathbb{F}$ constitui uma álgebra, já que $\mathbb{F}$ pode ser visto como espaço vetorial sobre si próprio.*

Uma álgebra tem **dimensão finita** quando é finitamente gerada como espaço vetorial. Se $\mathcal{A}$ é uma álgebra associativa, dizemos que $\mathcal{A}$ é **comutativa** quando $xy = yx, \forall x, y \in \mathcal{A}$. Álgebras comutativas raramente serão consideradas neste trabalho, mas são fundamentais em outras áreas da matemática, como a *Geometria Algébrica*.

Exemplo 0.3 *Seja $\mathbb{G}$ um grupo multiplicativo finito*[1] *e vamos considerar a **álgebra de grupo** $\mathbb{F}\mathbb{G} = \{\sum \alpha_i \cdot g_i \mid \alpha_i \in \mathbb{F},\ g_i \in \mathbb{G}\}$, onde $\cdot$ é um "produto formal" e a soma é finita. Os elementos $g \in \mathbb{G}$ formam uma base de $\mathbb{F}\mathbb{G}$ como álgebra sobre $\mathbb{F}$ e, portanto, $\mathbb{F}\mathbb{G}$ tem dimensão finita.*

Dado $x \in \mathbb{F}\mathbb{G}$, $x = \sum \alpha_i g_i$, com $\alpha_i \in \mathbb{F}$ e $g_i \in \mathbb{G}$. É óbvio que os g_i's são linearmente independentes. Assim, para $x = \sum \alpha_i g_i$ e $y = \sum \beta_j g_j$ em $\mathbb{F}\mathbb{G}$, definimos:

$$x + y := \sum \alpha_i g_i + \sum \beta_i g_i = \sum (\alpha_i + \beta_i) g_i$$

[1] Nada impede que o grupo seja infinito, porém, como as álgebras de grupo que teremos a oportunidade de lidar neste trabalho são todas de dimensão finita, restringiremos nossa abordagem a elas.

$$x * y := \sum \alpha_i g_i * \sum \beta_j g_j = \sum_i \sum_j \alpha_i * \beta_j * g_i * g_j$$

Desta forma, $\mathbb{F}\mathbb{G}$ se torna uma álgebra. Como vale $a(bc) = (ab)c$ para $a, b, c \in \mathbb{G}$, $\mathbb{F}\mathbb{G}$ é uma álgebra associativa. Se $\mathbb{G}$ for abeliano, então $\mathbb{F}\mathbb{G}$ é comutativa.

Definição 0.4 *Um subespaço $\mathcal{B} \subseteq \mathcal{A}$ é dito uma **subálgebra** de $\mathcal{A}$ se $\mathcal{B}$ é fechado para a multiplicação.*

Exemplo 0.5 *O conjunto $\mathbb{F}^{n\times n}$ das matrizes $n \times n$ sobre um corpo $\mathbb{F}$ é uma álgebra sobre $\mathbb{F}$ que tem dimensão n^2. Esta álgebra é gerada pelas matrizes elementares e_{ij}, que têm 1 na posição (i, j) e 0 nas demais, para $1 \leq i, j \leq n$. As subálgebras de $\mathbb{F}^{n\times n}$ são seus subespaços fechados para a multiplicação matricial.*

Exemplo 0.6 *Um importante exemplo de subálgebra é o **centro** $Z(\mathcal{A}) = \{x \in \mathcal{A} \mid xy = yx \ \forall y \in \mathcal{A}\}$, onde $\mathcal{A}$ é uma álgebra. Se existir um elemento neutro, $1_\mathcal{A}$, para a multiplicação em $\mathcal{A}$, então temos que $\mathbb{F} = 1_\mathcal{A}\mathbb{F} \leq Z(\mathcal{A})$, onde $\leq$ denota subálgebra. Caso $Z(\mathcal{A}) = \mathbb{F}$, a álgebra $\mathcal{A}$ será dita **central**.*

Generalidades Algébricas

Um subespaço $\mathcal{I}$ de $\mathcal{A}$ é um **ideal à esquerda** de $\mathcal{A}$ se $yx \in \mathcal{A}$, para todo $x \in \mathcal{I}$ e $y \in \mathcal{A}$. Analogamente se define **ideal à direita** de $\mathcal{A}$. Um $\mathbb{F}$-subespaço $\mathcal{I}$ é um **ideal** de $\mathcal{A}$ se $\mathcal{I}$ é um **ideal bilateral** (ideal à esquerda e à direita). Dado $\mathcal{I}$ um ideal de $\mathcal{A}$, podemos considerar o quociente $\mathcal{A}/\mathcal{I}$, que também será uma álgebra, com a multiplicação induzida pela de $\mathcal{A}$.

Os homomorfismos de álgebras são definidos da forma usual e, assim como para espaços vetoriais, valem o *Teorema do Núcleo e da Imagem* e o *Primeiro Teorema do Isomorfismo*.

Dois elementos $x, y \in \mathcal{A}$, ambos não nulos, são ditos **um par de divisores de zero** se $xy = 0$. Um elemento $e \in \mathcal{A}$ é **idempotente** se $e^2 = e$.

A álgebra $\mathcal{A}$ é dita **simples** se não tem ideais além dos triviais $\langle 0\rangle$ e $\mathcal{A}$ e se $\mathcal{A}\mathcal{A} \neq \langle 0\rangle$, onde $\mathcal{A}\mathcal{A}$ denota o conjunto $\{\sum xy \mid x, y \in \mathcal{A}\}$. Dizemos que $\mathcal{A}$ é **soma direta de seus ideais** $\mathcal{A}_1, ..., \mathcal{A}_k$ e denotamos por $\mathcal{A} = \mathcal{A}_1 \oplus \mathcal{A}_2 \oplus ... \oplus \mathcal{A}_k$ ou $\mathcal{A} = \coprod \mathcal{A}_i$ quando o espaço vetorial $\mathcal{A}$ puder ser assim decomposto.

Exemplo 0.7 *Se $\mathbb{K} \supseteq \mathbb{F}$ é uma extensão finita (e algébrica) de $\mathbb{F}$, então $\mathbb{K}$ é uma álgebra simples (e comutativa) sobre $\mathbb{F}$.*

Com efeito, é fácil ver que $\mathbb{K}$ é uma álgebra sobre $\mathbb{F}$. Resta mostrar que $\mathbb{K}$ é simples. Seja $\mathcal{I}$ um ideal não-nulo de $\mathbb{K}$. Dado $a \in \mathcal{I}$, tomando $a^{-1} \in \mathbb{K}$ (cuja existência está assegurada por $\mathbb{K}$ ser corpo), $1 = aa^{-1} \in \mathcal{I}$. Assim, $\mathbb{K} = 1\mathbb{K} \subseteq \mathcal{I}$, ou seja, $\mathcal{I} = \mathbb{K}$. Logo, os únicos ideais de $\mathbb{K}$ são os triviais.

Exemplo 0.8 *A álgebra $\mathbb{F}^{d\times d}$ das matrizes $d \times d$ sobre $\mathbb{F}$ é uma álgebra simples sobre $\mathbb{F}$.*
Já vimos que $\mathbb{F}^{d\times d}$ é uma álgebra sobre $\mathbb{F}$. Seja $\mathcal{I}'$ um ideal não-nulo de $\mathbb{F}^{n\times n}$. A partir de $\mathcal{I}'$, podemos construir um ideal unidimensional $\mathcal{I} \subseteq \mathcal{I}'$, tomando um elemento qualquer não-nulo de $\mathcal{I}'$ como gerador de $\mathcal{I}$. Assim, $\mathcal{I} = [x]$, onde

$$x = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nn} \end{pmatrix},$$

ou, em notação que nos será útil,

$$x = \sum_{i,j=1}^{n} x_{ij} e_{ij}.$$

O produto entre duas matrizes elementares e_{ij} e e_{kl} pode ser descrito como

$$e_{ij} e_{kl} = \delta_{jk} e_{il},$$

onde δ_{jk} é o delta de Kronecker. Como $\mathcal{I}$ é fechado para a multiplicação (em ambos os lados) por qualquer elemento da álgebra, os produtos $e_{kl}x$, xe_{kl} e $e_{kl}xe_{kl}$ pertencem ao ideal $\mathcal{I}$ para quaisquer $k, l \in \{1, ..., n\}$. De

$$xe_{kl} = \sum_{i,j=1}^{n} x_{ij} e_{ij} e_{kl} = \sum_{i,j=1}^{n} x_{ij} \delta_{jk} e_{il},$$

segue que

$$e_{kl} x e_{kl} = e_{kl} \sum_{i,j=1}^{n} x_{ij} e_{ij} e_{kl} = \sum_{i,j=1}^{n} x_{ij} \delta_{jk} e_{kl} e_{il} = \sum_{i,j=1}^{n} x_{ij} \delta_{jk} \delta_{li} e_{kl}.$$

Note que um termo da soma acima só é não-nulo quando $j = k$ e $i = l$. Mas quando isso acontece o somatório se reduz a uma única parcela: $x_{lk}e_{kl}$. Assim, temos a igualdade

$$e_{kl} x e_{kl} = x_{lk} e_{kl},$$

ou seja, $e_{kl} \in \mathcal{I}$, para quaisquer $k, l \in \{1, ..., n\}$. Logo, $\mathcal{I} = \mathbb{F}^{d\times d}$, e temos o resultado.

Exemplo 0.9 *Se $\mathcal{A}$ é uma álgebra de dimensão n, podemos estender $\mathcal{A}$ a uma álgebra com unidade e de dimensão $n + 1$. Definimos a* ***extensão de Dorroh*** *$\mathcal{A}^*$* ***de*** *$\mathcal{A}$ como sendo o conjunto dos pares (a, λ), com $a \in \mathcal{A}$ e $\lambda \in \mathbb{F}$. A adição é feita da maneira usual e a multiplicação em $\mathcal{A}^*$, denotada aqui por $*$, é definida por:*

$$(a_1, \lambda_1) * (a_2, \lambda_2) = (a_1 a_2 + \lambda_1 a_2 + \lambda_2 a_1, \lambda_1 \lambda_2)$$

Desta forma, $\mathcal{A}^*$ é uma álgebra com unidade sobre $\mathbb{F}$, já que:

$$(0, 1) * (a, \lambda) = (0a + 1a + \lambda 0, \lambda) = (a, \lambda)$$

Mais ainda, $\mathcal{A}$ é isomorfa a um ideal de $\mathcal{A}^*$, a saber, o ideal $\mathcal{A} \cong \{(a, 0) \mid a \in \mathcal{A}\} \subseteq \mathcal{A}^*$. Note ainda que $dim(\mathcal{A}^*) = dim(\mathcal{A}) + dim(\mathbb{F}) = dim(\mathcal{A}) + 1$.

Exemplo 0.10 *Os subespaços de $\mathbb{F}^{d\times d}$ fechados para a multiplicação matricial são álgebras.*

Teoremas de Estrutura

O último exemplo da seção anterior é dos mais importantes, já que temos o:

Teorema 0.11 (Teorema da Representação) *Seja $\mathcal{A}$ uma álgebra sobre $\mathbb{F}$ tal que $dim_{\mathbb{F}}\mathcal{A} = n$. Então $\mathcal{A}$ é isomorfa a uma subálgebra de $\mathbb{F}^{(n+1)\times(n+1)}$.*

Prova. *Vamos usar a **representação regular**. Para cada $x \in \mathcal{A}$ defina a função $R_x : \mathcal{A} \to \mathcal{A}$ por $R_x(y) = xy$, para $y \in A$. Assim, temos a representação $\Psi : \mathcal{A} \to End(\mathcal{A})$ ($\cong F^{n\times n}$) tal que $\Psi(x) = R_x : \mathcal{A} \to \mathcal{A}$. Suponha que exista $1_{\mathcal{A}}$ (neutro multiplicativo). Logo, se $\Psi(x) = \Psi(y)$, então $R_x(z) = R_y(z)$, $\forall z \in A$, ou seja, $xz = yz$, $\forall z \in A$. Em especial, para $z = 1_{\mathcal{A}}$, temos $x = y$ e, portanto, Ψ é injetora. Logo, $\mathcal{A} \cong Im(\Psi) \subseteq F^{n\times n}$.*

Vamos tratar agora o caso em que A não possui unidade, usando o conceito de "extensão de Dorroh", definido no Exemplo 0.9. Pela primeira parte da demonstração, se $\mathcal{A}^*$ é a extensão de Dorroh de $\mathcal{A}$, então $\mathcal{A}^* \cong F^{(n+1)\times(n+1)}$, pois $\mathcal{A}^*$ tem unidade. Como $\mathcal{A} \subseteq \mathcal{A}^*$ (isomorficamente), temos que $\mathcal{A} \subseteq F^{(n+1)\times(n+1)}$. ■

Um elemento $x \in \mathcal{A}$ é dito **nilpotente** se $x^k = 0$ para algum k inteiro positivo. Um elemento $x \in \mathcal{A}$ é **fortemente nilpotente** se xy é nilpotente para todo $y \in \mathcal{A}$. O **radical** (de Jacobson) de $\mathcal{A}$, $Rad(\mathcal{A})$, é o conjunto dos elementos fortemente nilpotentes de $\mathcal{A}$. É rotina checar que $Rad(\mathcal{A})$ é um ideal de A.

Por essa caracterização do radical, a álgebra $\mathcal{A}/Rad(\mathcal{A})$ não tem elementos fortemente nilpotentes. Na verdade, $Rad(\mathcal{A})$ é um ideal nilpotente, isto é, existe k tal que $x_1 \cdot ... \cdot x_k = 0$ para todos $x_i \in Rad(\mathcal{A})$.

Uma álgebra $\mathcal{A}$ é dita **semi-simples** se $\mid \mathcal{A} \mid \geq 2$ e $Rad(\mathcal{A}) = \{0\}$, onde $\mid X \mid$ denota o número de elementos do conjunto X. A semi-simplicidade tem conseqüências interessantes, e uma delas é o famoso:

Teorema 0.12 (Teorema de Wedderburn) *Se A é uma álgebra associativa semi-simples de dimensão finita sobre o corpo F, então A pode ser expressa como $A = A_1 \oplus A_2 \oplus ... \oplus A_k$, onde os $A_i's$ são exatamente os ideais minimais[2] de A. Mais ainda, cada A_i é isomorfo a uma álgebra de matrizes, $F_i^{n_i\times n_i}$, onde F_i é um anel de divisão que contém o corpo $\mathbb{F}$.*

Exemplo 0.13 *Vamos calcular a decomposição dada pelo Teorema de Wedderburn para uma álgebra de matrizes.*

Seja $\mathcal{S}_3 \leq \mathbb{F}^{3\times3}$ o subespaço das matrizes cuja soma dos elementos de cada linha e de cada coluna é a mesma, ou seja, se $x = (x_{ij})_{3\times3} \in \mathcal{S}_3$, então as somas $\sum_{i=1}^3 x_{ik}$ e $\sum_{j=1}^3 x_{kj}$ são todas iguais, para $k = 1, 2, 3$.

É imediato que se $a \in \mathcal{S}_3$ e $b \in \mathcal{S}_3$ com $\sum a = \alpha$ e $\sum b = \beta$, onde $\sum$ denota a soma dos elementos das linhas/colunas, então $\sum ab = \alpha\beta$. Assim, $\mathcal{S}_3$ é fechado para a multiplicação matricial e é, portanto, uma subálgebra de $\mathbb{F}^{3\times3}$.

Vamos calcular a dimensão de $\mathcal{S}_3$. Se uma matriz $\tau \in \mathcal{S}_3$ então

$$\tau = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix},$$

[2]Um ideal $\mathcal{I}$ de uma álgebra $\mathcal{A}$ é dito **minimal** se é um ideal próprio de $\mathcal{A}$ e sempre que $\mathcal{J}$ for outro ideal de $\mathcal{A}$ tal que $\mathcal{J} \subseteq \mathcal{I}$, então ou $\mathcal{J} = \mathcal{I}$ ou $\mathcal{J} = \{0\}$.

com $a, b, c, d, e, f, g, h, i \in F$.
Seja $M = \sum \tau$. *Assim, temos o seguinte sistema:*

$$S : \begin{cases} a+b+c = M \\ d+e+f = M \\ g+h+i = M \\ a+d+g = M \\ b+e+h = M \\ c+f+i = M \end{cases}$$

Após alguns cálculos simples com o sistema S, *e considerando as igualdades acima, encontramos que*

$$\begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} = \begin{pmatrix} M-b-c & b & c \\ b+h-f & M-h-b & f \\ c+f-h & h & M-c-f \end{pmatrix}$$

Esta última matriz pode ser escrita como

$$M\mathbf{I} + b\lambda_1 + c\lambda_2 + h\lambda_3 + f\lambda_4,$$

onde $\mathbf{I}$ *é a matriz identidade e os* $\lambda_i's$ *são dados por:*

$\lambda_1 = -e_{11} - e_{22} + e_{12} + e_{21}$
$\lambda_2 = -e_{11} + e_{13} + e_{31} - e_{33}$
$\lambda_3 = e_{21} - e_{22} - e_{31}$
$\lambda_4 = -e_{21} + e_{23} + e_{31} - e_{33}$

Como os $\lambda_i's$ *acima são linearmente independentes, segue que* $dim_F \mathcal{S}_3 = 5$.

Vamos agora calcular uma decomposição da álgebra $\mathcal{S}_3$ *em subálgebras simples. Seja* $\mathbb{F}^3$ *com a base canônica* $\{e_1, e_2, e_3\}$. *Seja* $\mathcal{U} \leq F^3$ *o subespaço gerado pelo vetor* $e_1 + e_2 + e_3 = (1,1,1)$ *e* $\mathcal{U}' \leq F^3$ *o subespaço dos vetores tais que a soma de suas coordenadas é igual a 0,* $\mathcal{U}' = \{(x,y,z) \mid x+y+z = 0\}$. *Da forma usual, prova-se que* $\mathbb{F}^3 = \mathcal{U} \oplus \mathcal{U}'$ *e que* $\mathcal{U}$ *e* $\mathcal{U}'$ *são* $\mathcal{S}_3$*-invariantes, no sentido de que* $\mathcal{S}_3\mathcal{U} \subseteq \mathcal{U}$ *e* $\mathcal{S}_3\mathcal{U}' \subseteq \mathcal{U}'$.

Seja $\mathcal{B} \leq \mathbb{F}^{3\times 3}$ *a subálgebra de todas as matrizes b tais que* $b\mathcal{U} \subseteq \mathcal{U}$ *e* $b\mathcal{U}' \subseteq \mathcal{U}'$. *Se* $b = (b_{ij})_{3\times 3}$, *como* $\mathcal{U} = \langle (1,1,1) \rangle$ *e* $\mathcal{U}' = \langle (1,0,-1), (0,1,-1) \rangle$, *resolvendo o sistema*

$$\begin{cases} b(1,1,1) = (\alpha, \alpha, \alpha) \\ b(1,0,-1) = (\beta, 0, -\beta) \\ b(0,1,-1) = (0, \gamma, -\gamma) \end{cases}$$

teremos os geradores para $\mathcal{B}$, *que são exatamente os* $\lambda_i's$ *encontrados para base de* $\mathcal{S}_3$. *Logo,* $\mathcal{B} = \mathcal{S}_3$.

A aplicação $\Lambda : \mathbb{F}^{3\times 3} \to \mathbb{F}^{2\times 2}$ *tal que* $\Lambda(\lambda_2) = e_{11}$, $\Lambda(\lambda_3) = e_{12}$, $\Lambda(\lambda_4) = e_{21}$, $\Lambda(\lambda_5) = e_{22}$ *é um isomorfismo de* $\langle \lambda_2, \lambda_3, \lambda_4, \lambda_5 \rangle$ *em* $\mathbb{F}^{2\times 2}$. *Como* $\mathbb{F} \cong \langle \lambda_1 \rangle$, *segue que*

$$\mathcal{B} \cong \mathbb{F} \oplus \mathbb{F}^{2\times 2},$$

e os termos dessa soma direta são simples, por exemplos anteriores. Assim, esta é a decomposição de Wedderburn para a álgebra $\mathcal{B} = \mathcal{S}_3$, *o que mostra que essa é uma álgebra semi-simples.*

Constantes de Estrutura

Seja $\mathcal{A}$ uma álgebra. Dada uma base $\{a_1, ..., a_n\}$ do espaço vetorial $\mathcal{A}$, a multiplicação na álgebra $\mathcal{A}$ fica definida pelas constantes (de estrutura) γ_{ij}^k que são os coeficientes dos elementos da base nos produtos $a_i a_j$, como pode-se ver abaixo:

$$a_i * a_j = \gamma_{ij}^1 a_1 + \gamma_{ij}^2 a_2 + ... + \gamma_{ij}^{n-1} a_{n-1} + \gamma_{ij}^n a_n$$

Exemplo 0.14 *A álgebras das matrizes $n \times n$ sobre um corpo $\mathbb{F}$ tem como base as matrizes e_{ij}. As multiplicações entre os elementos da base são dadas (veja Exemplo 0.8) por*

$$e_{ij} e_{kl} = \delta_{jk} e_{il}.$$

Assim, as constantes de estrutura são todas inteiras iguais a 1 ou zero.

Computacionalmente, definimos uma álgebra por meio de suas constantes de estrutura. Esta maneira nos permite trabalhar com álgebras de uma maneira bem abstrata, já que só precisamos das relações entre os geradores para conhecer totalmente como a álgebra se comporta.

Módulos e Representações

Nesta seção apresentaremos as definições básicas da Teoria de Módulos e daremos alguns exemplos. Estes conceitos são em parte necessários para estudar representações de álgebras.

De certo modo, um módulo é a generalização de um espaço vetorial, agora construído sobre um anel e não mais sobre um corpo. Porém, esta generalização causa "anomalias", quando olhadas sob o ponto de vista da Álgebra Linear. Fatos curiosos sobre módulos é que nem todos admitem uma base, e nem todo conjunto gerador de um módulo pode ser reduzido a uma base. Para referência, consulte [POLCINO].

Definição 0.15 *Um* ***módulo à esquerda*** *sobre um anel $\mathcal{R}$ com unidade, ou um $\mathcal{R}$-módulo à esquerda, é constituído de um grupo abeliano $(\mathcal{M}, +)$ e uma operação $\mathcal{R} \times \mathcal{M} \to \mathcal{M}$ tal que para quaisquer $r, s, \in \mathcal{R}$ e $x, y \in \mathcal{M}$, temos*

i) $r(x + y) = rx + ry$
ii) $(r + s)x = rx + sx$
iii) $(rs)x = r(sx)$
iv) $1x = x$

Analogamente definimos **$\mathcal{R}$-módulo à direita** sobre $\mathcal{R}$. Quando o anel $mathcalR$ é comutativo, os $\mathcal{R}$-módulos à esquerda e os $\mathcal{R}$-módulos à direita coincidem.

Exemplo 0.16 *Se $\mathbb{F}$ é um corpo, então todo espaço vetorial sobre $\mathbb{F}$ é um $\mathbb{F}$-módulo.*

Exemplo 0.17 *Todo grupo abeliano $\mathbb{G}$ é um $\mathbb{Z}$-módulo de maneira única: se $n \in \mathbb{Z}$ e $g \in \mathbb{G}$, definimos $ng = g + g + \ldots + g$ (n parcelas) para $n > 0$ e $ng = (-n)(-g)$ para $n < 0$.*

Exemplo 0.18 *Se $\mathcal{R}$ é o anel das matrizes $n \times n$ sobre $\mathbb{R}$, então o espaço euclidiano $\mathbb{R}^n$ (de vetores coluna) é um $\mathcal{R}$-módulo à esquerda, com a operação do módulo sendo a multiplicação matricial.*

Como vimos, a definição de módulo se inicia com um grupo. Qual o papel dos subgrupos na teoria de módulos? É o que veremos agora.

Definição 0.19 *Seja $\mathcal{M}$ um $\mathcal{R}$-módulo à esquerda e $\mathcal{N}$ um subgrupo de $\mathcal{M}$. Então $\mathcal{N}$ é um* ***$\mathcal{R}$-submódulo*** *se, para qualquer $n \in \mathcal{N}$ e $r \in \mathcal{R}$, o produto $rn \in \mathcal{N}$.*

Exemplo 0.20 *Se $\mathbb{G}$ é um grupo e $\mathbb{H}$ é um subgrupo de $\mathbb{G}$, então $\mathbb{H}$ é um $\mathbb{Z}$-submódulo do $\mathbb{Z}$-módulo $\mathbb{G}$.*

Definição 0.21 *Se $\mathcal{M}$ e $\mathcal{N}$ são $\mathcal{R}$-módulos à esquerda, então uma função $f : \mathcal{M} \to \mathcal{N}$ é um* ***$\mathcal{R}$-homomorfismo*** *de $\mathcal{R}$-módulos se, para quaisquer $m, n \in \mathcal{M}$ e $r, s \in \mathcal{R}$, vale a relação*

$$f(rm + sn) = rf(m) + sf(n).$$

Quando um $\mathcal{R}$-homomorfismo $f : \mathcal{M} \to \mathcal{N}$ é uma bijeção, dizemos que os $\mathcal{R}$-módulos $\mathcal{M}$ e $\mathcal{N}$ são **isomorfos**.

Definição 0.22 *Dizemos que um módulo $\mathcal{M}$ é* ***simples*** *quando $\mathcal{M}$ não possui submódulos não-triviais, isto é, diferentes de $\{0\}$ e $\mathcal{M}$. Alguns vezes também usamos a palavra* ***irredutível*** *ao invés de "simples".*

Se $\mathcal{M}$ é um $\mathcal{R}$-módulo, a ação de um elemento $r \in \mathcal{R}$ é uma função $\mathcal{M} \to \mathcal{M}$ que a cada $x \in \mathcal{M}$ associa rx. Pela definição de módulo, esta ação precisa ser um endomorfismo do grupo $\mathcal{M}$.

Proposição 0.23 *O conjunto de todos os endomorfismos de $\mathcal{M}$, denotado por $End(\mathcal{M})$, é um anel com as operações de soma e composição.*

À luz da proposição anterior temos que a aplicação $r \in \mathcal{R} \mapsto r \in End(\mathcal{M})$, que envia um elemento do anel na sua ação, define um homomorfismo de anéis de $\mathcal{R}$ em $End(\mathcal{M})$.

Este homomorfismo $\mathcal{R} \to End(\mathcal{M})$ é chamado de **representação** de $\mathcal{R}$ sobre $\mathcal{M}$. Desta forma, uma maneira alternativa, porém equivalente, de definirmos módulos é dizer que um $\mathcal{R}$-módulo à esquerda é um grupo abeliano $\mathcal{M}$ e uma representação de $\mathcal{R}$ sobre $\mathcal{M}$.

Uma representação é **fiel** quando a aplicação $\mathcal{R} \to End(\mathcal{M})$ é injetora. Em termos de módulos, se $r \in \mathcal{R}$ é tal que $rx = 0$ para todo $x \in \mathcal{M}$, então $r = 0$.

Exemplo 0.24 *Todo grupo abeliano é um módulo fiel sobre o anel dos inteiros.*

Álgebra Tensorial[1]

Os conhecimentos desta seção serão utilizados mais tarde, no Capítulo 3.

Sejam $\mathcal{V}$ e $\mathcal{W}$ espaços vetoriais de dimensão finita sobre um mesmo corpo $\mathbb{F}$. O **produto tensorial** entre $\mathcal{V}$ e $\mathcal{W}$, denotado por $\mathcal{V} \otimes \mathcal{W}$, é o espaço vetorial das aplicações bilineares

$$f : \mathcal{V}^* \times \mathcal{W}^* \to \mathbb{F},$$

onde $\mathcal{V}^*$ e $\mathcal{W}^*$ são os duais de $\mathcal{V}$ e $\mathcal{W}$.

Dados $v \in \mathcal{V}^*$ e $w \in \mathcal{W}^*$, o seu produto tensorial $v \otimes w$ é o funcional bilinear cujo valor quando aplicado em (α, β) é dado por

$$(v \otimes w)(\alpha, \beta) = \alpha(v)\beta(w).$$

O conjunto $\{v \otimes w \mid v \in \mathcal{V},\ w \in \mathcal{W}\}$ gera $\mathcal{V} \otimes \mathcal{W}$ e se $\{v_1, \ldots, v_n\}$ e $\{w_1, \ldots, w_m\}$ são bases de $\mathcal{V}$ e $\mathcal{W}$, respectivamente, então

$$\{v_i \otimes w_j \mid 1 \leq i \leq n,\ 1 \leq j \leq m\}$$

é uma base de $\mathcal{V} \otimes \mathcal{W}$. Portanto,

$$dim(\mathcal{V} \otimes \mathcal{W}) = dim(\mathcal{V})dim(\mathcal{W}).$$

De maneira mais geral, se $\mathcal{V}_1, \ldots, \mathcal{V}_n$ são espaços vetoriais sobre $\mathbb{F}$, o produto tensorial $\mathcal{V}_1 \otimes \cdots \otimes \mathcal{V}_n$ é o espaço das aplicações multilineares definidas em $\mathcal{V}_1^* \times \cdots \times \mathcal{V}_n^*$ que tomam valores em $\mathbb{F}$. Se $v_i \in \mathcal{V}_i$, para $1 \leq i \leq n$, então o produto tensorial $v_1 \otimes \cdots \otimes v_n$ é o funcional dado por

$$(v_1 \otimes \cdots \otimes v_n)(\alpha_1, \ldots, \alpha_n) = \alpha_1(v_1) \cdots \alpha_n(v_n),$$

onde $\alpha_i \in V_i^*$ para $1 \leq i \leq n$.

Como no caso bidimensional, a partir de bases $\{v_j^i\}_{j=1}^{n_i}$ de $\mathcal{V}_i$, com $1 \leq i \leq n$, podemos construir uma base de $\mathcal{V}_1 \otimes \cdots \otimes \mathcal{V}_n$ tomando todos os possíveis produtos da forma $v_{i_1}^1 \otimes \cdots \otimes v_{i_n}^n$. Assim, a dimensão do produto tensorial é o produto das dimensões dos fatores.

A propriedade principal do produto tensorial é que toda aplicação multilinear

$$f : \mathcal{V}_1 \times \cdots \times \mathcal{V}_n \to \mathcal{W},$$

onde $\mathcal{W}$ é um espaço vetorial qualquer, fatora-se numa aplicação linear do produto tensorial em $\mathcal{W}$, isto é, existe uma aplicação linear $\tilde{f} : \mathcal{V}_1 \otimes \cdots \otimes \mathcal{V}_n \to \mathcal{W}$ tal que o diagrama abaixo comuta, onde π é a imersão do produto cartesiano no produto tensorial, $\pi(v_1, \ldots, v_n) = v_1 \otimes \cdots \otimes v_n$:

$$\begin{array}{ccc} \mathcal{V}_1 \times \cdots \times \mathcal{V}_n & \longrightarrow^{f} & \mathcal{W} \\ \downarrow^{\pi} & \circlearrowleft & \nearrow_{\tilde{f}} \\ \mathcal{V}_1 \otimes \cdots \otimes \mathcal{V}_n & & \end{array}$$

[1]Esta seção segue o Apêndice A.5 de [SanMARTIN], págs. 437-441.

Quando os fatores de um produto tensorial são todos iguais a, por exemplo, $\mathcal{V}$, o produto de n de suas cópias é denotado por

$$\otimes^n \mathcal{V}.$$

A soma desses produtos define o espaço vetorial

$$\mathcal{T}(\mathcal{V}) = \sum_{k\geq 0} \otimes^k \mathcal{V},$$

que tem dimensão infinita caso $\mathcal{V} \neq \{\vec{0}\}$. Na expressão acima estamos convencionando $\otimes^0 \mathcal{V} = \mathbb{F}$ e $\otimes^1 \mathcal{V} = \mathcal{V}$. Note que $\mathcal{T}(\mathcal{V})$ é uma álgebra associativa com unidade quando munida da operação "produto tensorial".

Exemplo 0.25 *O espaço $\mathbb{L}(\mathcal{V}, \mathcal{W})$ das transformações lineares de $\mathcal{V}$ em $\mathcal{W}$ é isomorfo ao produto tensorial $\mathcal{V}^* \otimes \mathcal{W}$. O isomorfismo é obtido associando a $\alpha \otimes w \in \mathcal{V}^* \otimes \mathcal{W}$ a transformação linear $T_{\alpha,w} : \mathcal{V} \to \mathcal{W}$ definida por*

$$T_{\alpha,w}(u) = \alpha(u)w,$$

ou seja

$$\begin{aligned} \Phi : \mathcal{V}^* \otimes \mathcal{W} &\to \mathbb{L}(\mathcal{V}, \mathcal{W}) \\ \alpha \otimes w &\mapsto T_{\alpha,w} : \mathcal{V} \to \mathcal{W} \\ &\qquad\quad u \mapsto \alpha(u)w \end{aligned}$$

Note que Φ é mesmo um isomorfismo, pois se $\Phi(\alpha \otimes w) = 0$, então $T_{\alpha,w}(u) = 0$, para todo $u \in \mathcal{V}$. Logo, ou α é o funcional nulo ou w é o vetor nulo. Como $\alpha \otimes w$ é, na verdade a função de $\mathcal{V}^{**} \times \mathcal{W}^*$ em $\mathbb{F}$ definida por $(\alpha \otimes w)(\rho, \xi) = \rho(\alpha)\xi(w)$, para $(\rho, xi) \in \mathcal{V}^{**} \times \mathcal{W}^*$, e $\rho(0) = 0 = \xi(0)$, segue que se α ou w é nulo, $\alpha \otimes w = 0$. Portanto, Φ é injetora. O isomorfismo decorre da igualdade entre as dimensões e a linearidade "fica como exercício".

Se $\mathcal{V}$ e $\mathcal{W}$ tem dimensão finita, fixando bases $\beta = \{\beta_i\}_{i=1}^n$ e $\gamma = \{\gamma_j\}_{j=1}^m$ de $\mathcal{V}$ e $\mathcal{W}$, respectivamente, e escrevendo $w = \sum \lambda_i \gamma_i$ temos que

$$T_{\alpha,w}(\beta_k) = \alpha(\beta_k)w = \alpha(\beta_k) \sum \lambda_i \gamma_i = \alpha(\beta_k)\lambda_1\gamma_1 + \ldots + \alpha(\beta_k)\lambda_m\gamma_m.$$

Portanto,

$$[T_{\alpha,w}]_\gamma^\beta = \begin{pmatrix} \lambda_1\alpha(\beta_1) & \lambda_1\alpha(\beta_2) & \cdots & \lambda_1\alpha(\beta_n) \\ \lambda_2\alpha(\beta_1) & \lambda_2\alpha(\beta_2) & \cdots & \lambda_2\alpha(\beta_n) \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_m\alpha(\beta_1) & \lambda_m\alpha(\beta_2) & \cdots & \lambda_m\alpha(\beta_n) \end{pmatrix}$$

Definindo $[\alpha]_\beta = [\alpha(\beta_1) \;\; \alpha(\beta_2) \cdots \;\; \alpha(\beta_n)]$, temos que

$$[T_{\alpha,w}]_\gamma^\beta = [w]_\gamma [\alpha]_\beta$$

Portanto este isomorfismo não depende da base escolhida para $\mathcal{V}^*$, mas somente da escolha de bases para $\mathcal{V}$ e $\mathcal{W}$.

Exemplo 0.26 *Vamos a um caso numérico do exemplo anterior. Mantendo as notações, sejam $\mathcal{V} = \mathbb{R}^2$ e $\mathcal{W} = \mathbb{R}^3$, com bases canônicas $\mathcal{B}_1$ e $\mathcal{B}_2$, respectivamente. Como vimos antes, há um isomorfismo entre $\mathcal{V}^* \otimes \mathcal{W}$ e $\mathbb{L}(\mathcal{V}, \mathcal{W})$ tal que associa o elemento $\alpha \otimes w \in \mathcal{V}^* \otimes \mathcal{W}$ à matriz*

resultante do produto $[w]_{\mathcal{B}_2}[\alpha]_{\mathcal{B}_1}$. *Assim, se* $\alpha : \mathcal{V} \to \mathbb{R}$ *é o funcional dado por* $\alpha(x, y) = x$ *e* $w = (10, 4, 83)$, *podemos escrever*

$$\alpha \otimes w = [w]_{\mathcal{B}_2}[\alpha]_{\mathcal{B}_1} = \begin{bmatrix} 10 \\ 4 \\ 83 \end{bmatrix} [1\ 0] = \begin{bmatrix} 10 & 0 \\ 4 & 0 \\ 83 & 0 \end{bmatrix}$$

Observação 0.27 *Note que se* $\alpha \otimes w \in \mathcal{V}^* \otimes \mathcal{W}$, *então* $\alpha \otimes w$ *é uma aplicação de* $\mathcal{V}^{**} \times \mathcal{W}^*$ *em* $\mathbb{F}$. *Alguém ainda tem dúvidas quanto a importância de um isomorfismo entre* $\mathcal{V}^* \otimes \mathcal{W}$ *e um espaço de matrizes?*

Dois subespaços de $\otimes^n\mathcal{V}$ que devem ser destacados são o espaço de seus elementos simétricos e dos anti-simétricos. Dizemos que um elemento $v_1 \otimes \cdots \otimes v_n \in \otimes^n\mathcal{V}$ é **simétrico** se para toda permutação σ temos

$$v_1 \otimes \cdots \otimes v_n = v_{\sigma(1)} \otimes \cdots \otimes v_{\sigma(n)}$$

e **anti-simétrico** quando

$$v_1 \otimes \cdots \otimes v_n = (-1)^{|\sigma|} v_{\sigma(1)} \otimes \cdots \otimes v_{\sigma(n)},$$

onde $|\ \sigma\ |$ é a ordem da permutação (par ou ímpar).

O espaço dos elementos simétricos em $\otimes^n\mathcal{V}$ será denotado por $\odot^n\mathcal{V}$, enquanto que o dos anti-simétricos por $\wedge^n\mathcal{V}$. Esses espaços são obtidos pelas projeções[1]

$$\mathbf{S} : \otimes^n\mathcal{V} \to \odot^n\mathcal{V} \qquad \text{e} \qquad \mathbf{A} : \otimes^n\mathcal{V} \to \wedge^n\mathcal{V},$$

onde

$$\mathbf{S}(v_1 \otimes \cdots \otimes v_n) = \frac{1}{n!} \sum_\sigma v_{\sigma(1)} \otimes \cdots \otimes v_{\sigma(n)}$$

e

$$\mathbf{A}(v_1 \otimes \cdots \otimes v_n) = \frac{1}{n!} \sum_\sigma (-1)^{|\sigma|} v_{\sigma(1)} \otimes \cdots \otimes v_{\sigma(n)}.$$

Com estas projeções, podemos definir o **produto simétrico**

$$v_1 \odot \cdots \odot v_n = \mathbf{S}(v_1 \otimes \cdots \otimes v_n)$$

e o **produto exterior**

$$v_1 \wedge \cdots \wedge v_n = \mathbf{A}(v_1 \otimes \cdots \otimes v_n).$$

Os conjuntos $\odot\mathcal{V} = \sum_{k\geq 0} \odot^k\mathcal{V}$ e $\wedge\mathcal{V} = \sum_{k\geq 0} \wedge^k\mathcal{V}$ são álgebras com a simetrização e anti-simetrização do produto tensorial (funções $\mathbf{S}$ e $\mathbf{A}$ dadas acima), respectivamente. O primeiro destes conjuntos é chamado **Álgebra Simétrica** e o segundo **Álgebra Exterior** (ou **Álgebra de Grassmann**).

Note, para quaisquer v, w e v_i, $1 \leq i \leq k$, em $\wedge\mathcal{V}$, o produto exterior satisfaz:

i) $v \wedge v = 0$
ii) $v \wedge w = -w \wedge v$
iii) $v_1 \wedge v_2 \wedge \ldots \wedge v_k = 0$, sempre que $v_1, \ldots, v_k$ sejam linearmente independentes

[1] Operadores idempotentes que, quando restritos convenientemente a algum subespaço, agem identicamente.

Preliminares Computacionais

GAP - Groups, Algorithms and Programming

Introdução

GAP é uma abreviação para *Groups, Algorithms and Programming*. Este nome foi escolhido para descrever "explicitamente" o software, que inicialmente tratava somente de problemas da Teoria de Grupos e, em especial, problemas combinatórios.

Neste pequeno guia, que não tem a pretensão de ser um tutorial para o GAP iremos priorizar o uso de exemplos à sintaxes formais.

Aritmética

Após iniciado, o GAP apresenta a linha de comando

```
gap>
```

e fica aguardando um comando. Os comandos devem sempre ser terminados com ; (ponto-e-vírgula) ou : (dois-pontos). Porém, no segundo caso, o resultado não será mostrado na tela[1].

Para sair do GAP, o comando é

```
gap> quit;
```

Quando se desejar inserir um comentário numa linha de comando, isto é, uma frase para "marcar" uma linha, deve-se usar o símbolo #. Por exemplo:

```
gap> 2+2: #comentário que não terá efeito sobre o comando
```

A maneira como se manipulam expressões no GAP não é diferente da maneira que se faz nos outros softwares matemáticos (que, por sua vez, é a maneira natural). Por exemplo:

```
gap> (7-2)*(3+2);
25
```

Pode-se ainda usar múltiplas linhas para um mesmo comando (um comando só termina quando o último ponto-e-vírgula é digitado):

[1]Pode-se perguntar: "Ora, se eu não ver o resultado do cálculo, então de que me adianta o cálculo?". Na verdade, o uso do dois-pontos é mais comum em laços de programação, quando as contas intermediárias não interessam muito.

```
gap> (7-2)*(3+2)
> ;
25
```

Quando o comando está incompleto o GAP detecta onde possivelmente está o erro, e informa de uma maneira bem agradável, para que ele possa ser corrigido sem traumas:

```
gap> (7-2)*(3+2;
Syntax error: ) expected
(7-2)*(3+2;
          ^
```

O GAP não trabalha com aritmética de ponto flutuante. Assim, não espere que 1/2 seja 0.5 ou que 1/3 seja uma dízima periódica. Vamos a alguns comandos auto-explicativos.

```
gap> 12345/25;
2469/5

gap> 2^131
2722258935367507707706996859454145691648

gap> 15 mod 2
1

gap> 5<4
false

gap> true and false; true or false;
false true
```

Para atribuirmos valor a uma variável (que não precisa ser declarada) basta usar o sinal `:=` ("recebe"):

```
gap> a:=55;
55
gap> a;
55
```

Recomenda-se que as variáveis definidas pelo usuário se iniciem com letras minúsculas, pois os comandos do GAP são todos iniciados com maiúsculas.

Um comando muito útil no GAP é o `last` (na verdade, uma variável), que se refere ao último resultado[1]:

```
gap> 2*(3+1)+5*(9-3);
38
gap> b:=last:
gap> b;
```

[1]Outros comandos na mesma linha são last2 e last3, que guardam o penúltimo e ante-penúltimo comandos, respectivamente.

O GAP tem ainda inúmeras funções para lidar com números inteiros, como por exemplo `Factorial(n)`, `Gcd(n)`, `Lcm(n)` entre outros, que podem ser consultados em [GAP2].

Para definirmos uma função no GAP, a sintaxe é `função:=variavel->expressão`. Por exemplo

```
gap> poli:=x->x^2+3*x:
gap> poli(4)
28
```

Funções mais complexas podem ser criadas usando *procedimentos*. Detalhes em [GAP2].

Espaços Vetoriais e Transformações Lineares

O GAP lida muito bem com Espaços Vetoriais de dimensão finita sobre corpos de característica prima e/ou sobre extensões algébricas de $\mathbb{Q}$.

A sintaxe para criação de um Espaço Vetorial no GAP é `VectorSpace(F, [gens])`, onde `F` é o corpo base e `[gens]` é um conjunto de geradores para o espaço vetorial. Vejamos um exemplo:

```
gap> F:=Rationals; #define F como sendo o corpo dos números racionais
gap> V:=VectorSpace(F, [ [1,1,3],[0,0,1] ]);
<vector space over Rationals, with 2 generators>
```

Como todos os espaços vetoriais de uma mesma dimensão finita são isomorfos, o procedimento acima pode ser simplificado para:

```
gap> F:=Rationals; #define F como sendo o corpo dos números racionais
gap> V:=F^2;
<vector space over Rationals, with 2 generators}>
```

Note que usamos acima a notação usual para se definir um espaço vetorial de dimensão 2 sobre um corpo $\mathbb{F}$: $\mathbb{F}^2$ (porém, em linguagem própria, `F^2`). Para o espaço das matrizes $n \times m$, podemos usar algo semelhante: se $\mathbb{F}^{n\times m}$ é tal espaço, então para criá-lo no GAP basta um comando como:

```
gap> W:=Rationals^[2,2];
( Rationals}^[ 2, 2 ] )
gap> [ [1,3], [5,5] ] in W; # será que essa matriz pertence a W?
true
```

Para definirmos subespaços vetoriais, pode-se proceder como segue:

```
gap> V:= VectorSpace( Rationals, [ [ 1, 2, 3 ], [ 1, 1, 1 ] ] );;
gap> W:= Subspace( V, [ [ 0, 1, 2 ] ] );
<vector space over Rationals, with 1 generators>
```

Para verificar se um conjunto $\mathcal{V}$ é um espaço vetorial, o comando é `IsVectorSpace(V)`[1]. Para determinar a dimensão de $\mathcal{V}$, o comando é `Dimension(V)`. Já uma base de $\mathcal{V}$ pode ser extraída com `Basis(V)`, e os vetores desta base são elementos da lista `BasisVectors(Basis(V))`[1]. Por exemplo:

```
gap> A:=[1,2,3];;
gap> IsVectorSpace(A);
false
gap> V:=VectorSpace(Rationals, [[1,2,3],[1,1,1],[0,1,0] ]);;
gap> Dimension(V);
3
gap> BasisVectors( Basis(V) )[1];
[1, 2, 3]
gap> BasisVectors( Basis(V) )[2];
[1, 1, 1]
```

Podemos obter uma base canônica para um espaço vetorial com o comando:

```
gap> CanonicalBasis(V);
CanonicalBasis( <vector space of dimension 3 over Rationals> )
gap> BasisVectors(CanonicalBasis(V));
[ [ 1, 0, 0 ], [ 0, 1, 0 ], [ 0, 0, 1 ] ]
```

Para recuperar as coordenadas de um vetor $v \in \mathcal{V}$ em relação a uma base ordenada $\mathcal{B}$ o comando é `Coefficients( B, v)`. Já para determinar o vetor $w \in \mathcal{V}$ que tem como coordenadas $a_1, \ldots, a_n$ em relação à base $\mathcal{B}$ o comando é `LinearCombination(B, [a_1, ..., a_n])`.

```
gap> B:=Basis(V);
SemiEchelonBasis( <vector space of dimension 3 over Rationals> )
gap> Coefficients(B, [4, 5, 0]);
[ 4, -3, -6 ]
gap> LinearCombination(B, [1, -4, 3]);
[ 1, -2, -2 ]
```

Podemos definir transformações lineares (homomorfismos entre espaços vetoriais) com o comando `LeftModuleHomomorphismByImages( V, W, BV, img )`, onde `V`, `W` são espaços vetoriais, `BV=[v_1,...,v_n]` é uma base do domínio e `img=[w_1, ..., w_n]` é a imagem *ordenada* de `BV`. A imagem de um elemento $v \in \mathcal{V}$ por um homomorfismo f pode ser recuperada com o comando `Image(f, v)`. Por exemplo:

```
gap> V:=Rationals^2;;
gap> W:=VectorSpace( Rationals, [ [ 1, 0, 1 ], [ 1, 2, 3 ] ]);;
gap> f:=LeftModuleHomomorphismByImages( V, W,
> [ [ 1, 0 ], [ 0, 1 ]], [ [ 1, 0, 1 ], [ 1, 2, 3 ] ] );;
gap> Image( f, [1,1] );
[ 2, 2, 4 ]
```

[1]Repare que há uma diferença entre os dois "vês" desta frase. Primeiro ele aparece estilizado e, na sintaxe do comando, aparece com outra formatação. Para sermos fiéis aos tipos de fonte utilizados no texto, usaremos livremente a formatação. O leitor deve ficar atento: `V`=$\mathcal{V}$, cada qual no seu contexto.

[1]Para acessar o elemento na entrada `i` da lista `L`, usamos o comando `L[i]`.

Outra maneira de definirmos transformações lineares é por matrizes.

```
gap> V:=Rationals^2;;
gap> W:=VectorSpace(Rationals, [ [1,4,5,6],[0,0,2,2]  ]);;
gap> m:=[[1, 3],[0, 1]];;
gap> f:=LeftModuleHomomorphismByMatrix(Basis(V), m, Basis(W));
<linear mapping by matrix, ( Rationals^ 2 ) ->
<vector space over Rationals, with 2 generators>>
gap> Image(f, [4,5]);
[ 4, 16, 37, 41 ]
gap> Image(f, [4,6]);
[ 4, 16, 38, 42 ]
```

Para obtermos o espaço vetorial de todas as transformações lineares entre dois espaços vetoriais $\mathcal{V}$ e $\mathcal{W}$ sobre um corpo $\mathbb{F}$, o comando é `Hom(F, V, W)`. Já se quisermos a álgebra das transformações lineares entre $\mathcal{V}$ e $\mathcal{W}$, o comando é `End(F, V, W)`.

Mantendo $\mathcal{V}$ e $\mathcal{W}$ como no último exemplo, vamos obter uma base para $\mathcal{L}(\mathcal{V}, \mathcal{W})$ e analisar o comportamento desta base sobre o espaço.

```
gap> lvv:=Hom(Rationals, V, W);;
gap> Dimension(lvv);
4
gap> baseLVV:=BasisVectors(Basis(lvv));;
gap> baseB:=BasisVectors(Basis(V));;
gap> Image(baseLVV[1],baseB[1]);
[ 1, 4, 5, 6 ]
gap> Image(baseLVV[2], baseB[2]);
[ 0, 0, 0, 0 ]
gap> Image(baseLVV[3], baseB[2]);
[ 1, 4, 5, 6 ]
gap> Image(baseLVV[4], baseB[2]);
[ 0, 0, 1, 1 ]
gap> Dimension(Image(baseLVV[1])); Dimension(Image(baseLVV[2]));
1, 1
gap> Dimension(Image(baseLVV[3])); Dimension(Image(baseLVV[4]));
1, 1
```

Note que o GAP constrói a base de $\mathcal{L}(\mathcal{V}, \mathcal{W})$ segundo o procedimento padrão, que pode ser encontrado em [CL].

Álgebras

Ao definirmos uma multiplicação[1] em um espaço vetorial, criamos uma álgebra. Assim, um corpo $\mathbb{F}$ pode ser visto como uma álgebra sobre si mesmo.

Para álgebras de dimensão finita, vale o Teorema 0.11, o que torna as álgebras matriciais muito importantes. No GAP, não há diferenças entre a *álgebra* das matrizes e o *espaço vetorial*

[1] Uma operação num conjunto A que mereça ser chamada de multiplicação deve ser fechada para os elementos de A, além de ser associativa em relação à multiplicação por um elemento do corpo base.

das matrizes[2]. Assim, criamos uma álgebra de matrizes da mesma maneira que criamos um espaço vetorial de tal natureza:

```
gap> V:=GF(2)^[2,2] # O corpo finito de dois elementos
(GF(2)^[ 2, 2 ] )
```

Os comandos citados na seção anterior que tratavam de base e dimensão podem ser igualmente aplicados para álgebras, já que as definições destes entes para álgebras são herdadas da definição dada para espaços vetoriais.

Outra maneira de construírmos álgebras no GAP é definindo constantes de estrutura para a álgebra. Fixada uma base $\mathcal{B} = \{v_1, ..., v_n\}$ de uma álgebra $\mathcal{A}$, as constantes de estrutura são as coordenadas dos produtos $v_i v_j$ em relação à base $\mathcal{B}$. Maiores detalhes, consulte a seção **Constantes de Estrutura**, na página 6.

No GAP, uma tabela de constantes de estrutura é representada por uma "matriz 3D". Para criar uma matriz de constantes de estrutura, o comando é `EmptySCTable( d, z, opção )`, onde `d` é a dimensão da álgebra, `z` é o zero da álgebra e `opção` pode ser "*symmetric*", "*antisymmetric*" ou não tem valor nenhum, de acordo com a característica da multiplicação que se quer definir. Para editarmos a matriz das constantes de estrutura de uma álgebra, o comando é

```
SetEntrySCTable(T,i,j,[c_{ij}^{k_1}, k_1, ..., c_{ij}^{k_n}, k_n] );
```

onde `c_{ij}^{k_r}` é o coeficiente do vetor `k_r`-ésimo vetor da base no produto de v_i por v_j. Quando `c_{ij}^{k_r}=0`, esse coeficiente pode ser omitido. Por exemplo:

```
gap> T:=EmptySCTable( 2, 0, ``symmetric`` );;
gap> SetEntrySCTable( T, 1, 1, [1,1,] );;
gap> SetEntrySCTable( T, 2, 2, [1,2] );;
gap> A:=AlgebraByStructureConstants( Rationals, T );
<algebra of dimension 2 over Rationals>
```

Note que a álgebra $\mathcal{A}$ acima poderia, alternativamente, ser definida como a álgebra gerada pelas matrizes $\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ e $\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$.

Algumas álgebras "especiais" tem as constantes de estrutura definidas internamente no GAP. Para criá-las mais rapidamente, existem alguns comandos especiais, por exemplo:

```
gap> QuaternionAlgebra(Rationals);
<algebra-with-one of dimension 4 over Rationals>

gap> OctaveAlgebra(Rationals);
<algebra of dimension 8 over Rationals>
```

[2]Pode-se, usando os comandos `AsVectorSpace` e `AsAlgebra`, diferenciar estas duas estruturas. Porém, não trataremos com tal rigor do assunto.

```
gap> A:=FullMatrixAlgebra(Rationals, 20);
(Rationals^[ 20, 20 ])
gap> Dimension(A);
400

gap> A:=NullAlgebra(Rationals);
<algebra over Rationals>
gap> Dimension(A);
0
```

Criada uma álgebra, podemos definir subálgebras da seguinte forma:

```
gap> m:=[ [ 0, 1, 2 ], [ 0, 0, 3], [ 0, 0, 0 ] ];;
gap> A:=Algebra(Rationals, [m]);
<algebra over Rationals, with 1 generators>
gap> B:=Subalgebra(A, [m[2]] );
<algebra over Rationals, with 1 generators>
```

Podemos também criar ideais de uma álgebra:

```
gap> m:= [ [ 0, 2, 3 ], [ 0, 0, 4 ], [ 0, 0, 0] ];;
gap> A:=AlgebraWithOne(Rationals, [m]);;
gap> I:=Ideal(A, [m]); #cria o ideal de A gerado pela matriz m
<two-sided ideal in <algebra-with-one of dimension 3 over Rationals>,
(1 generators)>
gap> Dimension(I);
2
gap> GeneratorsOfIdeal(I); #quem são os geradores de I?
[ [ [ 0, 2, 3 ], [ 0, 0, 4 ], [ 0, 0,0 ] ] ]
gap> BasisVectors(Basis(I)); #agora a base de I, que tem dimensão 2
[ [ [ 0, 1, 3/2 ], [ 0, 0, 2 ], [0, 0, 0 ] ],
[ [ 0, 0, 1 ], [ 0, 0, 0 ], [ 0, 0, 0 ] ] ]

gap> A:=FullMatrixAlgebra(Rationals, 4);;
gap> m:=NullMat(4, 4);; m[1][4]:=1;;
gap> I:=LeftIdeal(A, [m]);
<left ideal in (Rationals^[4, 4]), (1 generators)>
gap> Dimension(I)
4
```

Podemos obter os elementos que geram uma álgebra da seguinte forma:

```
gap> m:=[ [ 0, 1, 2 ], [ 0, 0, 3 ], [ 0, 0, 0] ];;
gap> A:=AlgebraWithOne(Rationals, [m]);
<algebra-with-one over Rationals, with 1 generators>
gap> GeneratorsOfAlgebra(A);
[ [ [ 1, 0, 0 ], [ 0, 1, 0 ], [ 0, 0, 1 ] ],
[ [ 0, 1, 2 ], [ 0,0, 3 ], [ 0, 0, 0 ] ] ]
gap> # claro que A é gerada pela unidade e pela matriz m dada
```

Podemos testar se um dado conjunto é uma álgebra, uma álgebra com unidade, ou uma álgebra de Lie:

```
gap> A:=MatAlgebra(Rationals, 3);;
gap> IsAlgebra(A);
true
gap> IsAlgebraWithOne(A);
true
gap> IsLieAlgebra(A);
true
```

Para álgebras associativas e álgebras de Lie (sobre corpos de característica 0 ou maior que 5), podemos testar se a álgebra é simples, isto é, se só possui ideais triviais:

```
gap> A:=FullMatrixLieAlgebra(Rationals, 3);;
gap> IsSimpleAlgebra(A); #a álgebra de Lie das matrizes é simples?
false
gap> A:=MatAlgebra(Rationals, 3);;
gap> IsSimpleAlgebra(A); #e a álgebra associativa das matrizes?
true
```

Dado um elemento x numa álgebra $\mathcal{A}$, podemos considerar a matriz adjunta de x, denotada por $\mathbf{ad}x$ (veja pág. 41). Se temos uma base $\mathcal{B}$ de $\mathcal{A}$, podemos definir a base adjunta de $\mathcal{B}$ como sendo a base $\mathcal{C}$ de $\mathbf{ad}\mathcal{A}$, formada pelas transformações adjuntas dos elementos da base de $\mathcal{A}$. Para calculá-la:

```
gap> A:=QuaternionAlgebra(Rationals);;
gap> AdjointBasis(Basis(A));
Basis( <vector space over Rationals, with 4 generators>,
[ [ [ 1,0, 0, 0 ], [ 0, 1, 0, 0 ], [ 0, 0, 1, 0 ], [ 0, 0, 0, 1]],
[ [ 0, -1, 0, 0 ], [ 1, 0, 0, 0 ], [ 0, 0, 0, -1 ], [ 0, 0, 1,0 ] ],
[ [ 0, 0, -1, 0 ], [ 0, 0, 0, 1 ], [ 1, 0, 0, 0 ], [ 0, -1,0, 0 ] ],
[ [ 0, 0, 0, -1 ], [ 0, 0, -1, 0 ], [ 0, 1, 0, 0 ], [ 1,0, 0, 0 ] ] ]
```

Em uma álgebra de matrizes, podemos calcular o centralizador de um conjunto de matrizes na álgebra. O comando é:

```
gap> A:= QuaternionAlgebra( Rationals );;
gap> mats:=List(BasisVectors(Basis( A ) ), x ->AdjointMatrix(Basis(A), x ));;
gap> FullMatrixAlgebraCentralizer(Rationals, mats );
<algebra-with-one of dimension 4 over Rationals>
```

Para calcular o Radical de uma álgebra $\mathcal{A}$ (veja pág. 4), o comando é

```
gap> m:= [ [ 0, 1, 2 ], [ 0, 0, 3 ], [ 0, 0, 0 ] ];;
gap> A:= AlgebraWithOneByGenerators( Rationals, [ m ] );
<algebra-with-one over Rationals, with 1 generators>
gap> RadicalOfAlgebra(A);
<algebra of dimension 2 over Rationals>
```

Álgebras de Lie

Iremos definir o que é uma álgebra de Lie na página 39. Um modo corriqueiro de se criar uma álgebra de Lie é tomar uma álgebra associativa e definir o "colchete" como

$$[xy] = xy - yx.$$

No GAP, expressões como $[xy]$ e $[x, y]$ são reservadas para listas. Assim, o produto na álgebra de Lie é denotado, como um produto "normal", por `*`. O sistema interno do GAP consegue, na maior parte das vezes, distinguir entre produtos na álgebra associativa e na álgebra de Lie. Quando houver necessidade, pode-se usar o comando `LieObject` para evitar ambiguidades (consulte [GAP2] para detalhes).

O processo para "criar" uma álgebra de Lie é o mesmo usado para álgebras associativas, descrito na seção anterior. Uma das maneiras é por constantes de estrutura, com o comando `LieAlgebraByStructureConstants( Corpo, Tabela, Propriedade)`, onde `Corpo` é o corpo base, `Tabela` é o nome da variável que armazena as constantes de estrutura e `Propriedade` é uma propriedade que deve ser satisfeita pela álgebra (usualmente, "*antisymmetric*"). Por exemplo

```
gap> T:= EmptySCTable( 2, 0, ‘‘antisymmetric‘‘ );;
gap> SetEntrySCTable( T, 1, 2, [ 1/2, 1 ] );
gap> L:=LieAlgebraByStructureConstants( Rationals, T );
<Lie algebra of dimension 2 over Rationals>
```

Acima, criamos uma álgebra de Lie de dimensão 2 sobre o corpo dos racionais. Deve-se tomar cuidado, pois o comando `LieAlgebraByStructureConstants` não verifica a identidade de Jacobi da tábua, podendo ocorrer erros futuros caso se tente crie uma álgebra de Lie sem tal propriedade. Para verificar a identidade de Jacobi, o comando é `TestJacobi(T)`[1].

Outra maneira de criamos álgebras de Lie é com o comando `LieAlgebra( F, gens )`, onde `F` é o corpo e `gens` é uma lista com geradores para a álgebra.

```
gap> A:=FullMatrixAlgebra(GF(7), 4);;
gap> L:= LieAlgebra( A );
<Lie algebra of dimension 16 over GF(7)>

gap> mats:= [[[ 1, 0 ], [ 0, -1 ]], [[ 0, 1 ], [ 0, 0 ]],
> [[ 0,0 ], [ 1, 0]] ];;
gap> L:= LieAlgebra( Rationals, mats );
<Lie algebra over Rationals, with 3 generators>
```

Para as álgebras de Lie simples, existe um procedimento especial. Elas podem ser construídas com o comando `SimpleLieAlgebra( tipo, n, F )`, onde `tipo`, `n` e `f` são, respectivamente, o tipo da álgebra, o posto e o corpo base. Por exemplo:

```
gap> SimpleLieAlgebra( ‘‘E‘‘, 6, Rationals );
<Lie algebra of dimension 78 over Rationals>
```

[1] Este é um recurso no GAP para "encurtar" a criação de uma álgebra de Lie. Verificar a identidade de Jacobi é um processo lento, pois deve-se checar a relação para quaisquer três elementos da base da álgebra. Em caso de dúvida, deve-se usar o comando `TestJacobi()` para evitar surpresas desagradáveis no decorrer dos cálculos.

```
gap> SimpleLieAlgebra( ‘‘A‘‘, 6, GF(5) );
<Lie algebra of dimension 48 over GF(5)>

gap> SimpleLieAlgebra(‘‘W‘‘, [1,2], GF(5) );
<Lie algebra of dimension 250 over GF(5)>

gap> SimpleLieAlgebra(‘‘H‘‘, [1,2], GF(5) );
<Lie algebra of dimension 123 over GF(5)>
```

Os comandos para cálculo de subálgebras de ideais descritos na seção anterior valem também para álgebras de Lie. Vamos agora a alguns procedimentos para cálculo de subálgebras especiais de álgebras de Lie.

O centro de $\mathcal{L}$, que é o núcleo da representação adjunta:

```
gap> L:= FullMatrixLieAlgebra( GF(3), 3 );
<Lie algebra over GF(3), with 5 generators>
gap> LieCentre( L );
<two-sided ideal in <Lie algebra of dimension 9 over GF(3)>, (dimension 1)>
```

O centralizador de $\mathcal{S} \subset \mathcal{L}$, definido como o conjunto $\{a \in \mathcal{L} \mid [a, s] = 0, \forall s \in \mathcal{S}\}$:

```
gap> L:=SimpleLieAlgebra(‘‘G‘‘, 2, Rationals);
<Lie algebra of dimension 14 over Rationals>
gap> b:=BasisVectors( Basis( L ) );;
gap> LieCentralizer( L, Subalgebra( L, [ b[1], b[2] ] ) );
<Lie algebra of dimension 1 over Rationals>
```

O normalizador de $\mathcal{U}$ em $\mathcal{L}$, definido como sendo o conjunto $\{x \in \mathcal{L} \mid [x,\mathcal{U}] \subset \mathcal{U}\}$:

```
gap> L:= SimpleLieAlgebra( ‘‘G‘‘, 2, Rationals );
<Lie algebra of dimension 14 over Rationals>
gap> b:= BasisVectors( Basis( L ) );;
gap> LieNormalizer( L, Subalgebra( L, [ b[1], b[2] ] ) );
<Lie algebra of dimension 8 over Rationals>
```

A subálgebra derivada $\mathcal{L}' = [\mathcal{L}, \mathcal{L}]$ é assim calculada:

```
gap> L:=FullMatrixLieAlgebra( GF( 3 ), 3 );
<Lie algebra over GF(3), with 5 generators>
gap> LieDerivedSubalgebra( L );
<Lie algebra of dimension 8 over GF(3)>
```

Os radicais (solúvel e nilpotente) de $\mathcal{L}$:

```
gap> mats:= [ [[1,0],[0,0]], [[0,1],[0,0]],[[0,0],[0,1]]];;
gap> L:=LieAlgebra( Rationals, mats );;
gap> LieNilRadical( L );
<two-sided ideal in <Lie algebra of dimension 3 over Rationals>,
(dimension 2)>
```

```
gap> L:= FullMatrixLieAlgebra( Rationals, 3 );;
gap> LieSolvableRadical( L );
<two-sided ideal in <Lie algebra of dimension 9 over Rationals>,
(dimension 1)>
```

E, por fim, mas não menos importante, vamos calcular a subálgebra de Cartan de uma álgebra de Lie $\mathcal{L}$:

```
gap> L:= SimpleLieAlgebra( ``G``, 2, Rationals );;
gap> CartanSubalgebra( L );
<Lie algebra of dimension 2 over Rationals>


gap> V:=GF(3)^[4,4];;
gap> L:=LieAlgebra(V);;
gap> CartanSubalgebra(L);
<Lie algebra of dimension 4 over GF(3)>
```

Podemos calcular as séries derivada e central de uma álgebra de Lie. Os comandos são

```
gap> mats:=[ [[1,0],[0,0]], [[0,1],[0,0]],[[0,0],[0,1]]];;
gap> L:= LieAlgebra( Rationals, mats );;
gap> LieDerivedSeries( L);
[ <Lie algebra of dimension 3 over Rationals>,
<Liealgebra of dimension 1 over Rationals>,
<Lie algebra of dimension 0 over Rationals> ]
```

```
gap> mats:=[ [[ 1, 0 ], [ 0, 0 ]],[[0,1],[0,0]], [[0,0],[0,1]]];;
gap> L:=LieAlgebra(Rationals, mats);;
gap> LieLowerCentralSeries(L);
[ <Lie algebra of dimension 3 over Rationals>,
<Lie algebra of dimension 1 over Rationals> ]
```

Agora vamos explorar um pouco as propriedades de uma álgebra de Lie. Queremos testar comutatividade, nilpotência e solubilidade. Faremos um teste com a álgebra de Lie trivial.

```
gap> T:=EmptySCTable(5, 0, ‘‘antisymmetric‘‘);;
gap> L:=LieAlgebraByStructureConstants( Rationals, T );
<Lie algebra of dimension 5 over Rationals>
gap> IsLieNilpotent( L );
true
gap> IsLieAbelian( L);
true
gap> IsLieSolvable(L);
true
```

A decomposição em soma direta de ideais pode ser feita com o comando:

```
gap> L:=FullMatrixLieAlgebra( Rationals, 5 );;
gap> DirectSumDecomposition( L );
[ <two-sided ideal in
   <two-sided ideal in
    <Lie algebra of dimension 25 over Rationals>, (dimension 1)>,
   (dimension 1)>,
   <two-sided ideal in <two-sided ideal in
    <Lie algebra of dimension 25 over Rationals>, (dimension 24)>,
   (dimension 24)> ]
```

Agora vamos a algumas funções para trabalharmos com álgebras de Lie semi-simples.

Se $\mathcal{L}$ é uma álgebra de Lie semi-simples, para descobrirmos o "tipo" de $\mathcal{L}$, ou seja, o tipo das álgebras simples somandos diretos de $\mathcal{L}$, o comando é `SemiSimpleType(L)`.

Podemos calcular o sistema de raízes de $\mathcal{L}$ e também verificar se um sistema de raízes é de uma álgebra de Lie. Por exemplo:

```
gap> L:=SimpleLieAlgebra(‘‘G‘‘, 2, Rationals);;
gap> R:=RootSystem(L);
```

```
<root system of rank 2>
gap> IsRootSystem(R);
true
gap> IsRootSystemFromLieAlgebra(R);
true
```

Outro recurso disponível é o cálculo das raízes positivas/negativas, e da matriz de Cartan, além da álgebra de Lie que tem R como sistema de raízes.

```
gap> L:=SimpleLieAlgebra(``G``, 2, Rationals );;
gap> R:=RootSystem( L );;
gap> UnderlyingLieAlgebra( R );
<Lie algebra of dimension 14 over Rationals>
gap> PositiveRoots(R );
[[ 2, -1 ], [ -3, 2 ], [ -1, 1 ], [ 1, 0], [ 3, -1 ], [ 0, 1 ]]
gap> x:= PositiveRootVectors( R );
[ v.1, v.2, v.3, v.4, v.5, v.6]
```

Complexidade Computacional

Complexidade Computacional e Assintótica

Computacionalmente, um mesmo problema pode freqüentemente ser resolvido com algoritmos que diferem em eficiência. As diferenças entre os algoritmos podem ser irrelevantes para processar um pequeno número de ítens de dados, mas crescem proporcionalmente com a quantidade de dados. Para comparar a eficiência de algoritmos, uma medida do grau de dificuldade de um algoritmo, chamada de **complexidade computacional** foi desenvolvida por Juris Hartmanis e Richard E. Stearns.

A complexidade computacional indica quanto esforço é necessário para se aplicar um algoritmo, ou o quão custoso ele é. Este custo pode ser medido em termos do tempo de execução do algoritmo ou da quantidade de espaço (memória) que ele ocupa durante a sua execução. Para os algoritmos que iremos tratar o fator tempo é bem mais relevante e é nele que iremos nos concentrar.

A fim de serem comparados, os algoritmos precisam ter as mesmas características, além de estarem rodando em um mesmo computador. Algoritmos iguais podem ter desempenhos diferentes se escritos em diferentes linguagens, ou mesmo se compilados utilizando compiladores diferentes. Aliás, mesmo o fato dele ser compilado já gera uma diferença enorme: um programa escrito em C ou em Pascal (que precisam ser compilados) pode ser até 20 vezes mais rápido do que o mesmo programa codificado em BASIC ou LISP (que, geralmente, são interpretados).

A medida da eficiência de um algoritmo não é feita em unidades de tempo reais, como segundos ou nanosegundos. Geralmente, unidades lógicas que expressam uma relação entre o tamanho n de uma matriz de dados (ou de um arquivo) e a quantidade de tempo t exigida para processar os dados precisam ser usadas. Se houver uma relação linear entre o tamanho n e o tempo t, isto $t_1 = cn_1$, então um aumento de 5 vezes na quantidade de dados de entrada acarretará em um aumento de 5 vezes no tempo de execução. Similarmente, se $t_1 = log_2 n$, então, duplicando-se n, temos $t_2 = log_2(2n) = log_2(2) + log_2(n) = 1 + t_1$, ou seja, duplicando o tamanho da entrada, aumentamos o tempo de execução em somente uma unidade de tempo.

Uma função que expressa a relação entre n e t usualmente é muito mais complexa e o cálculo dessa função é importante somente em relação a grandes quantidades de dados. Assim, os termos que não modifiquem substancialmente a grandeza da função podem ser eliminados da função. Logo, a função resultante dá somente uma medida aproximada da função original. No entanto, essa aproximação é bem satisfatória, especialmente para funções que processam grandes quantidades de dados. Essa medida de eficiência é chamada de **complexidade assintótica** e é especialmente útil quando calcular uma função que mostre exatamente o comportamento de um dado algoritmo é difícil (ou mesmo impossível) e somente aproximações podem ser encontradas.

Para ilustrar, considere a função

$$f(n) = n^2 + 100n + log(n) + 106.$$

Na tabela abaixo, a função aplicada em alguns pontos, e a contribuição de cada um dos termos para o valor total da função.

n	$f(n)$	n^2	$100n$	$log(n)$	10^6
1	1000101	0,00%	0,01%	0,0000000%	99,99%
10	1001101	0,01%	0,10%	0,0000999%	99,89%
50	1007501,699	0,25%	0,50%	0,0001686%	99,26%
100	1020002	0,98%	0,98%	0,0001961%	98,04%
500	1300002,699	19,23%	3,85%	0,0002076%	76,92%
1000	2100003	47,62%	4,76%	0,0001429%	47,62%
2000	5200003,301	76,92%	3,85%	0,0000635%	19,23%
5000	26500003,7	94,34%	1,89%	0,0000140%	3,77%
1000	2100003	47,62%	4,76%	0,0001429%	47,62%
15000	227500004,2	98,90%	0,66%	0,0000018%	0,44%
20000	403000004,3	99,26%	0,50%	0,0000011%	0,25%
50000	2506000005	99,76%	0,20%	0,0000002%	0,04%

Para pequenos valores de n, o último termo, 10^6, é o maior, portanto, o que mais contribui para a função. Quando $n = 100$, a contribuição do termo quadrático se iguala à do termo linear. Porém, quando n passa de 100, a contribuição do termo n^2 se torna cada vez maior. Assim, para n grande, f depende principalmente do valor de n^2, e podemos desprezar os outros termos.

Notação O-grande

A notação mais usada para especificar a complexidade assintótica, isto é, para estimar a taxa de crescimento de uma função, é a notação **O-grande**.

Definição 1 *Dadas duas funções positivas, f e g, dizemos que $f(n)$ **é** $O(g(n))$ se existem números positivos c e N tais que $f(n) \leq cg(n)$ para todo $n \geq N$. Ou, mais rigorosamente,*

$$O(g(n)) = \{f : \mathbb{N} \to \mathbb{R}_+ \mid (\exists c \in \mathbb{R}_+^*)(\forall n > n_0) : f(n) < cg(n)\}.$$

Outra definição equivalente, usando limite, é a seguinte:

Definição 2 *Uma função $f(n)$ é dita **de classe** $O(g(n))$, e escreve-se $f(n) = O(g(n))$ quando $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c < \infty$, onde c é uma constante qualquer.*

Pela definição acima, f é **O**-grande de g se existe algum número positivo c tal que $f(n)$ é menor ou igual a $cg(n)$ para todos os valores de n maiores que um certo N. Assim, $g(n)$ é um limitante superior ao valor de $f(n)$ ou f cresce no máximo tão rápido quanto g.

É claro que se $f = O(n)$, então $f = O(n^2) = O(n^3)$... Para evitar este tipo de confusão, escolhe-se a *menor* função.

Pode-se ainda, para detalhar um pouco mais o crescimento da função, usar a notação **O**-grande somente em alguns termos da função, pode exemplo, se $g(n) = 2n^2 + 15n + 60$, podemos escrever $g(n) = 2n^2 + O(n)$ ao invés de simplesmente $g(n) = O(n^2)$.

Seguem algumas propriedades da notação **O**-grande.

Propriedade 1 (Transitividade) *Se f(n)=O(g(n)) e g(n)=O(h(n)), então f(n)=O(h(n)).*

Propriedade 2 *Se f(n)=O(h(n)) e g(n)=O(h(n)), então (f+g)(n)=O(h(n)).*

Propriedade 3 *Se f(n)=an^k, com $a, k \in \mathbb{R}$, então f(n)=O(n^k).*

Propriedade 4 *Se f(n)=an^k, com $a, k \in \mathbb{R}$, então f(n)=O(n^{k+j}), para todo j positivo.*

Propriedade 5 *Se f(n)=cg(n), com $c \in \mathbb{R}$, então f(n)=O(g(n)).*

Dos fatos acima, se $p(n) = a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0$ é uma função polinomial, então $p(n) = O(x^n)$.

Uma das funções mais importantes na avaliação da eficiência dos algoritmos é a função logarítmica. Se puder ser estabelecido que um algoritmo é da ordem de uma função logarítmica, então o algoritmo pode ser considerado muito bom. Claro que existem infinitas funções melhores que a logarítmica, porém, somente $O(1)$ ou $O(loglogn)$ (ou qualquer outra composição de logaritmos) tem utilidade na prática.

Propriedade 6 *A função $log_a(n)$ é $O(log_b(n))$ para quaisquer números positivos a e b diferentes de* 1.

A Propriedade 6 pode ser facilmente demonstrada usando a Propriedade 5 e a "regra" de mudança de base para logaritmos.

Propriedade 7 *A função $log_a(n)$ é $O(lg(n))$, para qualquer $a \neq 1$, onde $lg(n) = log_2(n)$.*

Agora, algumas regras para determinação da classe **O**-grande de duas funções, usando limites.

1. $\lim\limits_{n\to\infty} \dfrac{f(n)}{g(n)} = c < \infty, c \neq 0 \Rightarrow f(n) \in O(g(n))$ *e* $g(n) \in O(f(n))$

2. $\lim\limits_{n\to\infty} \dfrac{f(n)}{g(n)} = 0 \Rightarrow f(n) \in O(g(n))$ *e* $g(n) \notin O(f(n))$

3. $\lim\limits_{n\to\infty} \dfrac{f(n)}{g(n)} = +\infty \Rightarrow f(n) \notin O(g(n))$ *e* $g(n) \in O(f(n))$

Notações Ω e Θ

Assim como a notação **O**-grande se refere aos limites assintóticos superiores das funções, existem notações para os limites assintóticos inferiores e os "limites exatos".

Definição 3 *Uma função $f(n)$ é $\Omega(g(n))$ se existem números positivos c e N tais que $f(n) \geq cg(n)$ para todo $n \geq N$. Formalmente,*

$$\Omega(g(n)) = \{f : \mathbb{N} \to \mathbb{R}_+ \mid (\exists c \in \mathbb{R}_+^*)(\forall n > n_0) : f(n) > cg(n)\}.$$

Se $f(n) = \Omega(g(n))$, $g(n)$ é um limite inferior ao crescimento de $f(n)$, ou seja, $f(n)$ cresce no mínimo na mesma taxa que $g(n)$.

As notações Ω e **O**-grande são relacionadas pela seguinte regra:

Propriedade 8 *$f(n)$ é $\Omega(g(n))$ se, e só se, $g(n)$ é $O(f(n))$.*

A notação Ω sofre do mesmo problema que a notação **O**-grande: há um número infinito de funções menores que uma dada função. Assim, a função $f(n) = 5n^2$ é $\Omega(n^2)$, $\Omega(n)$, $\Omega(\sqrt{n})$, $\Omega(logn)$, $\Omega(loglogn)$, ... Para propósitos práticos, somente as Ω's mais próximas são interessantes, ou seja, geralmente estamos interessados nos maiores limites inferiores - assim como nos menores limites superiores, no caso da notação **O**-grande. Isto motiva uma outra definição:

Definição 4 *Uma função $f(n)$ é $\Theta(g(n))$ se existem números positivos c_1, c_2 e N tais que $c_1 g(n) \leq f(n) \leq c_2 g(n)$.*

Usando a notação Θ podemos limitar a complexidade superior e inferiormente, conseguindo uma maior garantia sobre a eficiência do algoritmo.

Na prática, aplicamos a notação Θ para descrever um limite inferior para o melhor caso e a notação $\mathbf{O} - grande$ para descrever um limite superior para o pior caso. Veremos mais tarde que que o algoritmo *Insertion Sort*, para ordenação de números, é $\Theta(n)$ e $O(n^2)$.

Existe uma adaptação da regra do limite dada antes para a notação Θ:

1. $\lim\limits_{n\to\infty} \dfrac{f(n)}{g(n)} \in \mathbb{R}_+^* \Rightarrow f(n) \in \Theta(g(n))$

2. $\lim\limits_{n\to\infty} \dfrac{f(n)}{g(n)} = 0 \Rightarrow f(n) \in O(g(n)),\ f(n) \notin \Theta(g(n))$

3. $\lim\limits_{n\to\infty} \dfrac{f(n)}{g(n)} = \infty \Rightarrow f(n) \in \Omega(g(n)),\ f(n) \notin \Theta(g(n))$

Classes de Comportamento Assintótico

A notação **O**-grande é reflexiva, simétrica e transitiva. Assim, ela determina uma partição no conjunto das funções $f : N \to \mathbb{R}_+$. Esta partição determina as **classes de comportamento assintótico** ou **classes de complexidade**.

O(1) Algoritmos na classe $O(1)$ são ditos de **complexidade constante**, e independentemente do tamanho da entrada, realizam o mesmo número de passos.

O(log(n)) Um algoritmo de complexidade $O(log(n))$ é dito ter **complexidade logarítmica**. Geralmente ocorrem em algoritmos que transformam um problema grande em outros problemas menores. Pode-se considerar, nestes casos, que o tempo de execução é menor que uma constante suficientemente grande.

O(n) Um algoritmo com complexidade $O(n)$ é dito ter **complexidade linear**. Assim, multiplicando n por uma constante k, o tempo de execução também ficará multiplicado por k.

O(nlog n) Complexidade típica de algoritmos que quebram um problema em outros menores, resolve cada um dos pequenos problemas independentemente e une as soluções. O tempo de execução é um pouco mais alto que os algoritmos de complexidade linear.

O(n^2) Algoritmos de **complexidade quadrática**. Ocorrem quando os ítens são processados ao pares, geralmente quando existem laços sendo compostos. Úteis para resolver problemas de tamanho pequeno a médio.

O(n^3) Algoritmos de **complexidade cúbica** são úteis somente para pequenos problemas.

O(2^n) Algoritmos de **complexidade exponencial**. Não são muito úteis, e para grandes problemas, nunca terminam.

O(n!) Complexidade também exponencial, apesar da complexidade fatorial $O(n!)$ ter um comportamento muito pior que a verdadeira exponencial $O(2^n)$.

A relação entre as classes acima é:

$$O(1) < O(log\ n) < O(n) < O(nlog\ n) < O(n^2) < O(n^3) < O(2^n) < O(n!)$$

A seguir, algumas tabelas comparando os tempos de execução para algoritmos em cada uma das classes acima para variados tamanhos de entrada, considerando sua execução em um computador com processador *Pentium IV 1,6 Ghz*, com capacidade de armazenamento infinita[1], funcionando no limite de sua capacidade.

Tamanho da Entrada		Classes de Complexidade		
	constante	logarítmico	linear	n log n
1000	83	2	0,0000008	0,000003
5000	83	3	0,0000042	0,000015
1000000	83	5	0,0008333	0,005000
50000000	83	6	0,0416667	0,320790
Escala de Tempo:	(nanosegundos)	(nanosegundos)	(segundos)	(segundos)

tamanho da entrada	Classe quadrática	escala de tempo	classe cúbica	escala de tempo
1000	0,0008	Segundos	0,83	Segundos
5000	0,02	Segundos	1,7	Minutos
50000	2	Segundos	29	Horas
1000000	13	Minutos	26	Anos
1000000000	26	Anos	5,7	ISS[2]
50000000000	66	Milênios	718065	ISS

tamanho da entrada	classe exponencial	escala de tempo
10	0,0008	Segundos
50	260,00	Dias
75	1000,0000	Milênios
100	7200,00	ISS

[1]Esta estranha expressão quer dizer simplesmente que a lentidão nos tempos de execução dos algoritmos não foi acarretada pela falta de memória ou pela lentidão desta, e sim pela limitação do número de operações por segundo que processador em questão consegue executar.

[2]ISS: idade do nosso Sistema Solar, estimada em 4,5 bilhões de anos. Assim, 5,7 ISS quer dizer um tempo de execução de 25 bilhões de anos. Haja paciência!

tamanho da entrada	classe fatorial	escala de tempo
5	0,0000001	Segundos
20	64	Anos
25	409881	Milênios
27	62	ISS

Alguns algoritmos mal projetados, ou cuja complexidade não pode ser melhorada não tem aplicações práticas nos computadores disponíveis. Para processar um milhão de ítens num computador capaz de efetuar um bilhão de operações por segundo (um *Pentium IV 1.6 Mhz* em seus melhores dias é capaz de realizar por volta disso), o algoritmo quadrático terminaria em $16,7$ segundos, enquanto o cúbico exigiria 31 anos. Assim, a análise de complexidade dos algoritmos é extremamente importante, e não pode ser abandonada por conta do argumento de que entramos em uma era em que, a um custo relativamente pequeno, um computador pessoal pode realizar alguns bilhões de operações por segundo.

Agora vamos falar rapidamente sobre a diferença entre problemas computacionais fáceis e difíceis, e entre algoritmos rápidos e lentos.

É sempre desejável trabalhar com algoritmos que tenham o tempo de execução como uma função polinomial do tamanho da entrada. Um algoritmo é dito **lento** se, seja lá qual for o polinômio $p(n)$, existe um tamanho B para a entrada tal que o algoritmo precisa de mais que $p(B)$ unidades de tempo para processar, ou seja, ele não tem complexidade polinomial.

Um problema computacional é **intratável** (ou **difícil**) se não existe um algoritmo polinomial para resolvê-lo, e é **tratável** (ou **fácil**) quando existe tal algoritmo. Provar que um algoritmo é intratável é uma tarefa das mais árduas dentro da Teoria da Complexidade de Algoritmos.

Encontrando a Complexidade Assintótica

Na maioria dos casos estamos interessados na complexidade de tempo, que usualmente mede o número de atribuições e comparações realizadas durante a execução de um programa. Iniciaremos lidando apenas com o número de atribuições.

Vejamos um primeiro exemplo, onde será usado um laço para se calcular a soma dos números em um vetor de tamanho n.

A linguagem de programação usada aqui é a mesma usada no GAP.

```
gap> soma:=0;
0
gap> for i in [1..10] do
>    soma:=soma+i;
>    od;
gap> soma;
55
```

No início, a variável `soma` é iniciada, e recebe o valor `0`. Logo no início do laço `for`, a variável `i` recebe o valor de `0`. A partir de então, em cada execução são realizadas duas atribuições: `i:=i+1` e `soma:=soma+i`. Como o laço é executado n vezes, temos n atribuições para `soma` e n

para `i`, ou seja, $2+2n$ atribuições para a rodada completa desse laço. Assim, sua complexidade assintótica é $O(n)$.

Geralmente a complexidade cresce quando compomos dois laços. Por exemplo, vejamos um algoritmo para o cálculo da soma dos elementos de uma matriz. Primeiro, vamos fazer um caso particular para só depois analisar a complexidade do caso geral, das matrizes $n \times n$ (o algoritmo começa no `for`).

```
gap> A:=[ [1,2,3], [3,4,5], [0,5,0] ];; gap> soma:=0;;
gap> for i in [1..3] do
>       for j in [1..3] do
>        soma:=soma+A[i][j];
>       od;
>   od;
gap> soma;
23
```

Agora um rastreio do algoritmo anterior:

	i	j	soma	
inicio	-	-	0	início
exec. 1	1	1	1	primeira
exec. 2	1	2	3	execução do
exec. 3	1	3	6	primeiro laço
exec. 4	2	1	9	segunda
exec. 5	2	2	13	execução do
exec. 6	2	3	18	primeiro laço
exec. 7	3	1	18	terceira
exec. 8	3	2	23	execução do
exec. 9	3	3	23	primeiro laço

Quantas atribuições foram feitas? É só contar o número de alterações nas diferentes células da tabela acima, o que dá um total de 22 atribuições.

E se a matriz fosse 4x4? Bem, o rastreio será omitido, mas (acredite!) são realizadas 37 operações - ou seja, apesar de termos incrementado o valor de n em apenas uma unidade, aumentamos em 68% o tempo de execução do algoritmo. Na tabela seguinte, mais alguns valores de n com o respectivo número de operações.

valor de n	número de atribuições			total
	i	j	$soma$	
3	3	9	10	22
4	4	16	17	37
5	5	25	26	56
6	6	36	37	79
10	10	100	101	211
50	50	2500	2501	5051
100	100	10000	10001	20101

A próxima tabela relaciona o total de atribuições feitas pelo algoritmo anterior, dados na tabela anterior, com os valores das funções $f(n) = 2.5n^2$ e $g(n) = 2n^2$:

n	$g(n) = 2n^2$	atribuições	$f(n) = 2.5n^2$
3	18	22	23
4	32	37	40
5	50	56	63
6	72	79	90
10	200	211	250
50	5000	5051	6250
100	20000	20101	25000

Como o número de atribuições realizadas para uma matriz de ordem n sempre está entre $g(n)$ e $f(n)$, se tivéssemos que escolher uma função a fim de que o algoritmo anterior ser **O**-grande dela, a função $f(n) = n^2$ seria uma forte candidata (já que $O(n^2) = O(2.5\ n^2)$). Na verdade, é isto que acontece. Vejamos o porquê, analisando o caso geral: uma matriz $n \times n$.

Uma atribuição sempre é feita: a variável `soma` recebe `0`. A partir daí, começa um processo repetitivo. Para cada valor de `i` (número da linha), `j` assumirá n valores, e, com isso, a variável `soma` também será atualizada n vezes. Assim, como `i` pode assumir n valores, `j` será incrementada n^2 vezes, e, com isso, `soma` será atualizada também n^2 vezes. Assim, $h(n) = 1 + n + n^2 + n^2 = 1 + n + 2n^2$ é a função que define a complexidade do algoritmo anterior e, é claro, $h(n) = O(n^2)$.

Melhor, Médio e Pior Caso

Nos dois exemplos anteriores, calcular a complexidade assintótica foi fácil, já que o número de vezes que cada laço seria executado era muito bem conhecido. O cálculo começa a ficar complicado quando o número de iterações que os laços realizam não é sempre o mesmo.

Como exemplo, considere um algoritmo clássico de ordenação de listas: o *Insertion Sort*, que é bem eficaz para ordenar pequenas listas de números. O funcionamento do *Insertion Sort* é bem simples, por ser basicamente o que fazemos sempre que vamos ordenar cartas de um baralho: faça uma pilha com todas as cartas, remova uma e separe. Retire outra carta do baralho, compare com a carta inicial e decida se ela fica atrás ou na frente. Retire outra (a terceira) carta da pilha inicial, compare-a com cada uma das cartas da nova pilha e coloque-a no lugar correto. O algoritmo segue abaixo:

```
gap> isort:=function(v);
>       atual:=0;
>       for j in [2..Size(v)] do;
>           atual:=v[j];
>           i:=j-1;
>           while i>0 and v[i]>atual do;
>               v[i+1]:=v[i];
>               i:=i-1;
>           od;
>           v[i+1]:=atual;
>       od;
>   end;
```

Agora um exemplo do uso:

```
gap> v:=[34,5,6,3,2,5,6,7,5,4,3,2,4,5,6,43,32,2]
[ 34, 5, 6, 3, 2, 5, 6, 7, 5, 4, 3, 2, 4, 5, 6, 43, 32, 2 ]
gap> isort(v);
gap> v;
[ 2, 2, 2, 3, 3, 4, 4, 5, 5, 5, 5, 6, 6, 6, 7,32,34,43]
```

Voltando ao baralho, se as cartas estão em ordem, então nada há para fazer - não é preciso nenhuma reordenação. Porém, se elas estão na ordem inversa, então todas precisarão ser movidas. E é exatamente isto que acontece com o *Insertion Sort*: se a lista está ordenada, seu tempo de execução é bem mais baixo que quando a lista está o mais desordenada possível - isto é, na ordenação reversa. Vamos ver se isto realmente acontece, considerando algumas listas especiais e o tempo necessário para ordená-las.

Primeiro uma lista já ordenada (o comando "`time`" mede o tempo de execução do último comando, em milisegundos) :

```
gap> lista:=[1..1000];
gap> for i in [1..Size(lista)] do;
>     lista[i]:=i;
>    od;
gap> isort(lista);time;
1
```

Agora uma lista totalmente desordenada:

```
gap> lista:=[1..1000];
gap> for i in [1..Size(lista)] do;
>     lista[i]:=Size(lista)-i;
>    od;
gap> isort(lista);time;
464
```

Por último, uma lista qualquer - de entradas aleatórias.

```
gap> lista:=[1..1000];
gap> for i in [1..Size(lista)] do;
>     lista[i]:=Random([1..Size(lista)]);
>    od;
gap> isort(lista);time;
247
```

Os casos acima são, respectivamente, o melhor, o pior e o caso médio para o *Insertion Sort*. Veremos agora como calcular a complexidade em cada um dos casos.

O que acontece no melhor caso, quando a lista já está ordenada? Considere um vetor $v = [v_1, v_2, v_3, ..., v_n]$, onde $v_i \leq v_{i+1}$. Seguindo o algoritmo, quando `j=2`, `atual=v[2]` e `i=1`. Ocorre de `i>0`, porém, `v[i]<atual=v[2]`. Logo, o comando `while` não será executado. Assim, em cada passo do algoritmo (para $j = 2, 3, \ldots, n$) são executadas 3 atribuições. Como `atual:=0` no início da função `isort`, são realizadas um total de $3(n-1)+1$ atribuições em todo o algoritmo, ou seja, o melhor caso do `isort` é $O(3n-2) = O(n)$ - ou, mais especificamente, $\Theta(n)$.

No pior caso, temos $v = [v_1, v_2, v_3, ..., v_n]$, com $v_i \geq v_{i+1}$. Bem, sempre temos uma atribuição: `atual:=0`. Para cada $2 \leq j \leq n$ temos outras 3 atribuições: `atual`, `i` e `v[i+1]`. Assim, já são $3(n-1) = 3n-2$ atribuições, como antes.

Quantas vezes o laço `while` será executado? Para cada valor de `j`, ele será executado sempre que `i>0` e `v[j-1]>v[j]`, sendo que esta última condição sempre acontece. Resta saber quando é que temos `i>0`.

Quando `j=2` temos `i=1`. Assim, já na primeira execução atingimos `i=0`. Assim, só duas atribuições são feitas. Quando `j=3`, `i=2`. Assim, o laço será executado uma vez, e `i` passa a valer `1`. Na próxima vez, `i=0`. Portanto, o `while` é executado 2 vezes. Para `j=4`, `i=3`, e o laço será executado 3 vezes. Continuando, veremos que o número de vezes que o `while` é executado é dado por $\sum_{j=2}^{n} = \frac{n(n+1)}{2} - 1$, perfazendo um total de n^2+n-1 atribuições (dentro do `while`).

Somando tudo, o número de atribuições realizadas pelo algoritmo *Insertion Sort*, no pior caso, é dado por $h(n) = n^2 + 4n - 3 = O(n^2)$ - mais especificamente $\Theta(n^2)$.

E no caso médio? Bom, para o cálculo do caso médio, alguma probabilidade deve ser considerada, como por exemplo, qual a probabilidade de ocorrer um ordenamento parcial de tamanho k no vetor?

Deixemos o caso médio aos cientistas da computação. Para nossos fins, é mais importante o pior caso ou, em algumas vezes, o caso ótimo. O motivo disso é simples:

i. O pior caso é um limitante superior no tempo de execução para qualquer entrada. Conhecendo-o, teremos a garantia de que o algoritmo nunca demorará mais que aquele limite, e podemos evitar as considerações excessivas sobre a origem dos dados.

ii. Para alguns algoritmos, o pior caso ocorre freqüentemente - senão, não seria necessário um algoritmo. Em algoritmos de busca, principalmente, o pior caso sempre acontece quando o elemento procurado não existe - ou não está onde deveria estar.

iii. O caso médio é quase sempre tão ruim quando o pior caso e, mesmo que seja um pouco melhor, seu cálculo é imensamente mais complicado.

Um algoritmo é considerado **mais eficiente** que outro se seu **tempo de execução no pior caso** tem uma **ordem de grandeza menor**.

P=NP?

O problema em sua forma *non-geek*-readable[1]

Até 1960, somente problemas pequenos podiam ser resolvidos. Sabia-se que problemas de natureza combinatória eram geralmente difíceis, mas não se tinha nenhuma teoria sobre complexidade de problemas.

Com o advento da computação eletrônica, tornou-se possível resolver problemas maiores, e observou-se o seguinte: alguns problemas são tratáveis, podendo ser resolvidos em um tempo de computação que cresce de maneira razoável com o tamanho do problema; outros são intratáveis, necessitando um número de operações que cresce rapidamente. Nesta época, Steve Cook observou um fato simples e ao mesmo tempo surpreendente: se um problema pudesse ser resolvido em tempo polinomial, poderíamos também verificar se uma dada possível solução é correta em tempo polinomial (dizemos que o algoritmo pode ser certificado em tempo polinomial). Provar a recíproca da afirmação de Cook é, grosso modo, provar que P=NP. Com uma definição simples de "tamanho de um problema", foram construídas várias classes de problemas de acordo com sua dificuldade. Três classes são importantes:

Classe P (*Polynomial-time*): um problema é da classe P se existe um algoritmo para resolvê-lo cujo tempo de computação varia de modo polinomial com o tamanho do problema.

Classe NP (*Non-deterministic Polynomial-time*): um problema é da classe NP se é fácil verificar se uma dada instância resolve o problema, mas não se tem necessariamente um método eficiente para encontrar a solução (obviamente, NP contém P).

Classe NP-completo: os problemas desta classe são os problemas mais difíceis da classe NP, no sentido de que eles são, dentre todos os problemas NP, os que têm mais chances de não estar na classe P.

O resultado surpreendente é o seguinte: todos os problemas NP-completos são equivalentes, no sentido de que se um deles for resolvido em tempo polinomial, todos serão. Isto é, se for possível encontrar um método polinomial para um desses problemas, esse método poderá ser adaptado para todos os outros. Nesse caso, P=NP. Se alguém mostrar que um desses problemas (por exemplo o do caixeiro viajante) necessita de um tempo exponencial para ser resolvido, mostra-se que P está propriamente contido em NP.

Um exemplo de problema NP-completo é o problema da soma de elementos de um dado subconjunto, ou seja, dado um conjunto finito de inteiros, determinar quando qualquer subconjunto destes tenha soma maior que zero. Este problema ilustra bem uma característica dos NP-completos: dada uma suposta solução, é imediato verificar se ela está correta ou não, mas ninguém conhece uma maneira rápida de se resolver o problema - atualmente o que se faz é tentar com todos os possíveis subconjuntos, o que é muito lento.

Outro exemplo de grande importância hoje em dia é o problema de descobrir se um dado número inteiro é composto, ou seja, se ele não é primo. Suponha que queiramos descobrir se 4294967297 é um número composto. Não existe uma maneira eficiente (rápida) de fazer isto. De fato tal tarefa pode ser realizada pela utilização do crivo de Eratóstenes, testando os possíveis

[1]Esta expressão é usada com frequência em contextos de computação/programação, e pode ser entendida como "material pouco técnico, voltado para o público em geral".

divisores do número, o que pode demandar um tempo excessivo de computação. Entretanto existe uma maneira sucinta de certificar que aquele número é composto: basta verificar que o produto de 6700417 por 641 é exatamente 4294967297. Mas como achar de maneira rápida estes fatores? Neste caso, levou 92 anos. A fatoração de 4294967297 foi encontrada por *Leonard Euler* em 1732, noventa e dois anos após *Pierre de Fermat* ter conjecturado erroneamente que tal número era primo.

É imediato verificar que P $\subseteq$ NP, pois se um algoritmo A está em P, para verificar se uma dada solução S está correta, basta encontrar a solução (o que é possível, pois $A \in$ P) e comparar com S.

Dados dois números x e y, vamos analisar o problema de determinar se y é múltiplo de algum inteiro entre 1 e x. Por exemplo, podemos verificar se 42355145 é múltiplo de algum inteiro entre 1 e 750. A resposta é sim, apesar de muito trabalho precisar ser feito para encontrar a solução. Entretanto, essa solução pode ser rapidamente testada multiplicando 58745 por 721 (que é menor que 750). Logo, este problema pertence à classe NP.

Não é tão fácil assim descobrir se o problema anterior está na classe P. Foi provado em 2002 que um caso especial, quando $x = y$ (o que reduz o problema ao de provar se um dado número é primo), está na classe P, sendo dado um algoritmo em tempo polinomial para isto, o famoso algoritmo AKS. Mais detalhes, veja [COUTINHO].

Um fato interessante é que a maioria dos problemas NP-completos é, de alguma maneira, ligado a um *puzzle*[1]. A recíproca também é verdadeira: a maioria dos *puzzles* para duas pessoas é um problema NP-completo, ou pior ainda: são problemas EXPTIME-Completos[2] ou PSPACE-Completos[3]. Para mais detalhes, consulte [EPPSTEIN], um belo artigo sobre a ligação entre problemas NP-Completos e Puzzles.

Além da relação $P \subseteq NP$, valem as seguintes inclusões: $P \subseteq NP \subseteq PSPACE$, $P \subseteq NP \subseteq PSPACE \subseteq EXPTIME \subseteq EXPSPACE$, $PSPACE - Completo \subseteq PSPACE$. Há suspeitas de que todas as inclusões nas três seqüências anteriores sejam próprias. Para a primeira e a última, é sabido que pelo menos uma delas é própria, porém, ainda não se sabe qual.

Não nos delongaremos mais sobre o problema "P=NP?", pois uma formalização de tal questão fugiria dos objetivos propostos neste trabalho. Quem se interessar por um estudo com tal ênfase pode consultar as referências online [WIKKIPEDIA], [MALAGUTTI], [DRUMMOND], [BOERES], [KAYE], [WILF] e [ARAÚJO]. Para desencargo de consciência, citaremos também os livros clássicos [CLR] e [DROZDEK], além de quase toda a literatura do grande mestre Donald Knuth, aqui citada em [KNUTH].

---

[1] Puzzle é um termo em inglês usado para designar jogos que envolvem algum raciocínio, como campo minado, Go, Hex, TIC-TAC-TOE (Jogo da Velha), entre outros.

[2] EXPTIME: Outra das classes de problemas: os que podem ser resolvidos por uma máquina de Turing determinística usando um algoritmo cuja complexidade é O($2^{p(n)}$), onde p(n) é um polinômio. A classe EXPTIME-Completo contém os problemas mais difíceis da classe EXPTIME, assim como a classe NP-Completo contém os da classe NP.

[3] PSPACE: Classe dos problemas de decisão que podem ser resolvidos por uma máquina de Turing usando uma quantidade polinomial de memória - ainda que em um tempo ilimitado. Para PSPACE-Completo, a definição é análoga às outras "*-Completo".

A referência [WIKKIPEDIA] acima citada pode ser consultada também sobre qualquer outro assunto que seja do interesse do leitor. Ela consiste na maior enciclopédia online existente. Em Matemática, as definições e teoremas mais famosos ali se encontram sem exceção e muito bem explicados.

Para os leitores com apurado faro capitalista, vale lembrar que ao resolver a questão "P=NP?", além do prestígio no mundo da matemática e o nome para sempre na história, a solucão pode trazer grandes vantagens materiais; explico: a solução do problema vale a bagatela de 1 milhão de dólares, pagos pelo Instituto Clay. Detalhes, veja [CLAY1]. A descrição oficial do problema, nos moldes do que o instituto entende como solução está em [COOK].

Capítulo 1

Álgebras de Lie

Começaremos agora o estudo da principal estrutura algébrica deste trabalho. Salvo menção contrária, $\mathcal{L}$ será uma álgebra de Lie de dimensão finita sobre um corpo $\mathbb{F}$ de característica 0, uma hipótese forte para o que será desenvolvido neste capítulo, já que grande parte dos resultados também vale para o caso de característica positiva.

Os resultados não demonstrados podem ser encontrados nas referências [HUMPHREYS], [SanMARTIN] ou em [JACOBSON].

1.1 Definições iniciais

As álgebras de Lie aparecem naturalmente como espaços vetoriais de transformações lineares munidos de uma nova operação, que não é comutativa nem associativa: $[F, G] = FG - GF$ (onde a justaposição significa a composição de transformações lineares).

Definição 1.1 *Uma* ***álgebra de Lie*** *consiste de um espaço vetorial $\mathcal{L}$ sobre um corpo $\mathbb{F}$ munido de um produto (chamado* ***colchete*** *ou comutador) bilinear*

$$[\ ,\ ] : \mathcal{L} \times \mathcal{L} \longrightarrow \mathcal{L}$$

que satisfaz a propriedade $[x, x] = 0$, $\forall x \in \mathcal{L}$ e também a ***Identidade de Jacobi***

$$[x, [y, z]] + [y, [z, x]] + [z, [x, y]] = 0,$$

para todos $x, y, z \in \mathcal{L}$.

Observação 1.2 *Iremos nos referir à bilinearidade como axioma L1 e à $[x, x] = 0, \forall x \in \mathcal{L}$ como axioma L2.*

Observação 1.3 *O axioma $[x, x] = 0$ é equivalente a $[x, y] = -[y, x]$ para $x, y \in \mathcal{L}$, desde que a característica do corpo $\mathbb{F}$ seja diferente de dois. Para ver isso, basta aplicar L1 e L2 ao colchete $[x + y, x + y]$.*

Observação 1.4 *O "colchete" de $x, y \in \mathcal{L}$, denotado acima por $[x, y]$, será escrito simplesmente como $[xy]$ quando a ausência da vírgula não prejudicar o entendimento.*

Observação 1.5 *Álgebras de Lie não são associativas.*

Alguns subconjuntos de $\mathcal{L}$ são muito importantes. Isto motiva a:

Definição 1.6 *Um subespaço $\mathcal{W}$ de $\mathcal{L}$ é dito uma* ***subálgebra*** *(de Lie) de $\mathcal{L}$ se $[xy] \in \mathcal{W}$, $\forall x, y, \in \mathcal{W}$. Usaremos a notação $\mathcal{W} \leq \mathcal{L}$ ou $\mathcal{W} < \mathcal{L}$ para significar que $\mathcal{W}$ é uma subálgebra de $\mathcal{L}$, sendo a última usada para subálgebras próprias.*

Note que se $\mathcal{L}$ tiver dimensão finita (como espaço vetorial) e $\mathcal{W}$ for uma subálgebra de $\mathcal{L}$ unidimensional, então $\mathcal{W}$ será uma álgebra de Lie trivial (i.e., $[xy] = 0$, $\forall x, y \in \mathcal{W}$).

Observação 1.7 *As noções de dimensão e geração finita usadas acima coincidem com aquelas usadas para espaços vetoriais.*

1.2 Álgebras de Lie Lineares

A estrutura de um espaço vetorial de dimensão finita é muito bem conhecida. Assim, seria muito útil se conseguíssemos uma estrutura correlata à ela para estudar uma álgebra de Lie.

Seja $\mathcal{V}$ um espaço vetorial de dimensão finita n sobre um corpo $\mathbb{F}$. É usual denotar por $\mathbb{L}(\mathcal{V}, \mathcal{V})$ o conjunto dos operadores lineares em $\mathcal{V}$. Como espaço vetorial sobre $\mathbb{F}$, $\mathbb{L}(\mathcal{V}, \mathcal{V})$ tem dimensão n^2 e é um anel com as operações usuais de soma e composição de operadores lineares, ou seja, $\mathbb{L}(\mathcal{V}, \mathcal{V})$ é uma álgebra (associativa) sobre $\mathbb{F}$.

Definindo em $\mathbb{L}(\mathcal{V}, \mathcal{V})$ a operação colchete por $[x, y] = xy - yx$, esta álgebra se torna uma álgebra de Lie sobre $\mathbb{F}$. Para distinguir esta estrutura da antiga (o anel), denotaremos a álgebra de Lie $\mathbb{L}(\mathcal{V}, \mathcal{V})$ por $\mathbf{gl}(\mathcal{V})$. As subálgebras de $\mathbf{gl}(\mathcal{V})$ são denominadas **álgebras de Lie lineares**. O nome é herdado do grupo linear geral $\mathbf{GL}(\mathcal{V})$, formado pelos operadores lineares invertíveis em $\mathbb{L}(\mathcal{V}, \mathcal{V})$.

Observação 1.8 *Fixada uma base para $\mathcal{V}$, podemos identificar $\mathbf{gl}(\mathcal{V})$ com o conjunto de todas as matrizes $n \times n$ sobre $\mathbb{F}$, denotado por $\mathbf{gl}(n, \mathbb{F})$ (com a multiplicação de Lie). Para fins de cálculo, esta abordagem é bem melhor que a anterior, já que cálculos com matrizes podem ser efetuados com relativa eficiência.*

1.3 Primeiros exemplos

Exemplo 1.9 *Para qualquer álgebra associativa $\mathcal{A}$, defina o colchete pelo comutador*

$$[xy] = xy - yx,$$

com $x, y \in \mathcal{A}$. Este colchete determina em $\mathcal{A}$ uma estrutura de álgebra de Lie.

Exemplo 1.10 *Particularizando o exemplo anterior, seja $\mathcal{L} = \mathbf{gl}(n, \mathbb{F})$ a álgebra de Lie das matrizes $n \times n$. Determinaremos as constantes de estrutura para $\mathbf{gl}(n, \mathbb{F})$, considerando sua base canônica formada pelas matrizes e_{ij}, com 1 na posição ij e 0 nas demais. Vimos no Exemplo 0.14 que $e_{ij}e_{kl} = \delta_{jk}e_{il}$. Assim,*

$$[e_{ij}, e_{kl}] = \delta_{jk}e_{il} - \delta_{li}e_{kj}.$$

Exemplo 1.11 (Álgebras abelianas) *Entende-se por álgebras abelianas as álgebras de Lie $\mathcal{L}$ onde $[x, y] = 0, \forall x, y \in \mathcal{L}$. Neste caso, a estrutura de álgebra de Lie não acrescenta nada à estrutura de espaço vetorial. Vejamos alguns destes casos:*
(a) Se dim$\mathcal{L} = 1$, $\mathcal{L}$ é abeliana, já que, como $\mathcal{L} = \langle v \rangle$ dados $u, w \in \mathcal{L}$, teremos que $u = \lambda v$ e $w = \beta v$. Assim, $[u, w] = [\lambda v, \beta v] = \lambda\beta[v, v] = 0$.
(b) Por (a), todo subespaço de dimensão 1 de uma álgebra de Lie é uma subálgebra abeliana.

Em Matemática, objetos aos quais são delegados nomes especiais são, não raramente, elementos chaves de uma teoria. É o caso de algumas subálgebras de $\mathbf{gl}(n, \mathbb{F})$.

Exemplo 1.12 *O conjunto $\boldsymbol{so}(n, \mathbb{F}) = \{x \in \mathbf{gl}(n, \mathbb{F}) \mid x + x^t = 0\}$, onde x^t significa a transposta da matriz x, é uma subálgebra de $\mathbf{gl}(n, \mathbb{F})$.*

Exemplo 1.13 *O conjunto $\mathbf{sl}(n, \mathbb{F}) = \{x \in \mathbf{gl}(n, \mathbb{F}) \mid \boldsymbol{tr}(x) = 0\}$, onde $\boldsymbol{tr}(x)$ significa o traço da matriz x é uma subálgebra de $\mathbf{gl}(n, \mathbb{F})$.*

Exemplo 1.14 *O subespaço das matrizes triangulares superiores e o das matrizes estritamente triangulares superiores, denotados por $\mathbf{t}(n, \mathbb{F})$ e $\mathbf{n}(n, \mathbb{F})$, respectivamente, são subálgebras de $\mathbf{gl}(n, \mathbb{F})$ (das mais importantes!).*

Exemplo 1.15 *O conjunto $\mathbf{d}(n, \mathbb{F})$, das matrizes diagonais é uma subálgebra de $\mathbf{gl}(n, \mathbb{F})$. Mais ainda, $\mathbf{d}(n, \mathbb{F})$ é uma álgebra de Lie abeliana, visto que matrizes diagonais comutam.*

1.4 Derivações

As álgebras de Lie nasceram originalmente num contexto de equações diferenciais. Isto explica alguns dos nomes e conceitos que aqui aparecem. E com as derivações não é diferente. Primeiro, o conceito mais geral:

Definição 1.16 *Seja $\mathcal{A}$ uma álgebra de dimensão n sobre um corpo $\mathbb{F}$. Uma **derivação** de $\mathcal{A}$ é uma função $\delta : \mathcal{A} \to \mathcal{A}$ que satisfaz uma regra semelhante à regra do produto para derivadas:*

$$\delta(ab) = a\delta(b) + \delta(a)b$$

Vamos denotar por $Der(\mathcal{A})$ o conjunto das derivações de $\mathcal{A}$.

Proposição 1.17 *$Der(\mathcal{A})$ é um subespaço vetorial de $\mathbb{L}(\mathcal{A}, \mathcal{A})$.*

Exemplo 1.18 *Um importante exemplo de derivação é a **adjunta**. Fixado $x \in \mathcal{L}$, defina $\boldsymbol{ad}x : \mathcal{L} \to \mathcal{L}$, $\boldsymbol{ad}x(y) = [xy]$. O fato de $\boldsymbol{ad}x$ ser uma derivação segue da identidade de Jacobi, pois $\boldsymbol{ad}x(yz) = [x, [yz]] = [[xy]z] + [y[xz]] = y\boldsymbol{ad}x(z) + z\boldsymbol{ad}x(y)$.*

Algumas vezes, será necessário olhar $x \in \mathcal{L}$ ou como elemento de $\mathcal{L}$ ou como elemento de uma subálgebra $\mathcal{W} < \mathcal{L}$. Nestes casos, para evitar ambigüidades, usaremos a notação $\mathbf{ad}_{\mathcal{L}}x$ ou $\mathbf{ad}_{\mathcal{W}}x$ para indicar se x está agindo em $\mathcal{L}$ ou em $\mathcal{W}$, respectivamente.

1.5 Ideais, Homomorfismos e Representações

Os ideais desempenham nas álgebras de Lie o mesmo papel que os subgrupos normais desempenham nos grupos e que os ideais bilaterais nos anéis. Mais ainda, todas estas estruturas têm uma origem comum: nos núcleos de homomorfismos. Usaremos a simbologia $\mathcal{I} \trianglelefteq \mathcal{L}$ para dizer que $\mathcal{I}$ é um ideal de $\mathcal{L}$. Formalmente, temos a:

Definição 1.19 *Um subespaço $\mathcal{I}$ de uma álgebra de Lie $\mathcal{L}$ é chamado de* ***ideal*** *de $\mathcal{L}$ se, para todos $x \in \mathcal{L}$ e $y \in \mathcal{I}$, temos $[xy] \in \mathcal{I}$.*

Vamos a alguns exemplos bem elucidativos.

Exemplo 1.20 (Ideais triviais) *Obviamente, 0 (o subespaço contendo somente o vetor nulo) e $\mathcal{L}$ são ideais de $\mathcal{L}$.*

Exemplo 1.21 (O centro) *O* ***centro*** *$Z(\mathcal{L}) = \{z \in \mathcal{L} \mid [xz] = 0, \forall x \in \mathcal{L}\}$ da álgebra de Lie $\mathcal{L}$ é um ideal de $\mathcal{L}$. Com efeito, se $x \in Z(\mathcal{L})$, então $[xy] = 0, \forall y \in \mathcal{L}$. Logo $[[zx]y] = [x[yz]] + [z[xy]] = 0$, para todo $z \in \mathcal{L}$. Note que $\mathcal{L}$ é abeliana se, e só se, $Z(\mathcal{L}) = \mathcal{L}$.*

Exemplo 1.22 (A álgebra derivada) *A* ***álgebra derivada*** *de $\mathcal{L}$, denotada por $[\mathcal{L}, \mathcal{L}]$, é o conjunto de todas as combinações lineares de $[xy]$, com $x, y \in \mathcal{L}$. Novamente, $\mathcal{L}$ é abeliana se, e só se, $[\mathcal{L}, \mathcal{L}] = 0$.*

Dados dois ideais $\mathcal{I}, \mathcal{J} \subseteq \mathcal{L}$, vamos definir os conjuntos $\mathcal{I} + \mathcal{J} = \{x + y \mid x \in \mathcal{I},\ y \in \mathcal{J}\}$ e $\mathcal{I}\mathcal{J} = \{\sum x_i y_j \mid x_i \in \mathcal{I},\ y_i \in \mathcal{J}\}$. Mostra-se da maneira usual que tanto $\mathcal{I} + \mathcal{J}$ quanto $\mathcal{I}\mathcal{J}$ são ideais de $\mathcal{L}$.

Observação 1.23 *Repare que $[\mathcal{L}\mathcal{L}]$ é um caso especial do produto de ideais $\mathcal{I}\mathcal{J}$, quando $\mathcal{I} = \mathcal{J} = \mathcal{L}$.*

É bem natural analisar a estrutura de uma álgebra de Lie olhando para seus ideais. Se $\mathcal{L}$ não tiver nenhum ideal além dos triviais (0 e $\mathcal{L}$), e se $[\mathcal{L}\mathcal{L}] \neq 0$, dizemos que $\mathcal{L}$ é **simples**. A condição $[\mathcal{L}\mathcal{L}] \neq 0$ ou, equivalentemente, $\mathcal{L}$ não-abeliana, é só para evitar o aparecimento de álgebras de Lie abelianas na classe das álgebras simples, já que se $\mathcal{L}$ é simples, então $Z(\mathcal{L}) = 0$ e $[\mathcal{L}\mathcal{L}] = \mathcal{L}$.

Exemplo 1.24 ($\mathbf{sl}(n, \mathbb{F})$ é simples) *Vamos fazer o caso $\mathbf{sl}(2, \mathbb{F})$, que pode ser generalizado naturalmente. Considere a base "canônica" $\{x, h, y\}$ de $\mathbf{sl}(2, \mathbb{F})$, cujas constantes de estrutura já foram calculadas no Exemplo 1.30.*

Se

$$z = ax + bh + cy,\ a, b, c \in \mathbb{F},$$

então

$$\boldsymbol{ad}x(z) = -2bx + ch,\ (\boldsymbol{ad}x)^2(z) = -2cx.$$

Assim, se $z \neq 0$, então z, $\boldsymbol{ad}x(z)$ ou $(\boldsymbol{ad}x)^2(z)$ é um múltiplo não-nulo de x. Desta forma, para qualquer ideal $\mathcal{I} \trianglelefteq \mathcal{L}$, teremos que $x \in \mathcal{I}$. Como

$$h = -[y, x],\ y = 1/2[y, h],$$

segue que $y, h \in \mathcal{I}$, ou seja,

$$\mathcal{I} = \mathbf{sl}(2, \mathbb{F}).$$

Este mesmo resultado vale para $\mathbf{sl}(n, \mathbb{F})$, desde que a característica de $\mathbb{F}$ seja diferente de dois.

No caso de $\mathcal{L}$ não ser simples é possível encontrar um ideal próprio $\mathcal{I}$ de $\mathcal{L}$, e "fatorar" $\mathcal{L}$, obtendo uma álgebra de dimensão inferior. Fatorar $\mathcal{L}$ por $\mathcal{I}$ significa construir o **quociente** $\mathcal{L}/\mathcal{I}$, construção esta análoga à feita para anéis ou espaços vetoriais. A operação no quociente é dada por $[x+\mathcal{I}, y+\mathcal{I}] = [xy]+\mathcal{I}$, que é bem definida, já que se $x+\mathcal{I} = x'+\mathcal{I}$ e $y+\mathcal{I} = y'+\mathcal{I}$, então $x' = x+u, u \in \mathcal{I}$ e $y' = y+v, v \in \mathcal{I}$. Conseqüentemente, $[x'y'] = [xy]+([uy]+[xv]+[uv])$. Como os termos entre parênteses estão em $\mathcal{I}$, segue que $[x'y'] + \mathcal{I} = [xy] + \mathcal{I}$.

Denominamos de **normalizador** de uma subálgebra $\mathcal{W} \leq \mathcal{L}$ o conjunto

$$N_{\mathcal{L}}(\mathcal{W}) = \{x \in \mathcal{L} \mid [x\mathcal{W}] \subset \mathcal{W}\},$$

que é uma subálgebra de $\mathcal{L}$. O normalizador de $\mathcal{W}$ pode ser descrito como a maior subálgebra de $\mathcal{L}$ da qual $\mathcal{W}$ é um ideal. Se $\mathcal{W} = N_{\mathcal{L}}(\mathcal{W})$, dizemos que $\mathcal{W}$ é **auto-normalizante**. Já o **centralizador** do conjunto $\mathcal{X} \subseteq \mathcal{L}$ é definido como

$$C_{\mathcal{L}}(\mathcal{X}) = \{y \in \mathcal{L} \mid [y\mathcal{X}] = 0\}.$$

É fácil ver que $C_{\mathcal{L}}(\mathcal{L}) = Z(\mathcal{L})$.

A definição de homomorfismo para uma álgebra de Lie não deve em nada às definições análogas para outras estruturas tais como Grupos, Anéis e Álgebras (principalmente esta última).

Definição 1.25 *Uma transformação linear $\phi : \mathcal{L} \to \mathcal{L}'$ é um* ***homomorfismo de álgebras de Lie*** *se $\phi([xy]) = [\phi(x), \phi(y)]$, para todo $x, y \in \mathcal{L}$. Caso $Ker\phi = \{x \in \mathcal{L} \mid \phi(x) = 0\} = 0$, dizemos que ϕ é um* ***monomorfismo****, e se $Im(\phi) = \mathcal{L}'$, então ϕ é dito* ***epimorfismo****. Caso ϕ seja* mono *e* epi*, dizemos que ϕ é um* ***isomorfismo****.*

Definição 1.26 *Dizemos que duas álgebras de Lie $\mathcal{L}$ e $\mathcal{L}'$ são* ***isomorfas*** *se existir um isomorfismo $\phi : \mathcal{L} \to \mathcal{L}'$.*

Vamos explicitar agora a relação entre ideais, núcleos e homomorfismos, timidamente citada no começo da seção anterior. Na verdade, existe uma correspondência 1-1 entre homomorfismos e ideais, que induz a outra correspondência: a cada homomorfismo ϕ, associamos o ideal $Ker(\phi)$, e a cada ideal $\mathcal{I}$, associamos a **projeção canônica** $\mathcal{L} \to \mathcal{L}/\mathcal{I}$, $x \mapsto x + \mathcal{I}$. Esta é uma das consequências do **Primeiro Teorema do Homomorfismo**:

Teorema 1.27 *Sejam $\mathcal{L}$ e $\mathcal{L}'$ duas álgebras de Lie, $\phi : \mathcal{L} \to \mathcal{L}'$ um homomorfismo, $\mathcal{I}$ e $\mathcal{J}$ ideais de $\mathcal{L}$. Então:*
(a) $\mathcal{L}/Ker\phi \cong Im\phi$ e se $\mathcal{I} \trianglelefteq Ker\phi$, existe um único homomorfismo $\psi : \mathcal{L}/\mathcal{I} \to \mathcal{L}'$ tal que, se π é a projeção canônica $x \mapsto x + \mathcal{I}$, então $\psi \circ \pi = \phi$, ou seja, o diagrama abaixo comuta:

$$\begin{array}{ccc} \mathcal{L} & \longrightarrow^{\phi} & \mathcal{L}' \\ & \searrow^{\pi} \quad \nearrow_{\psi} & \\ & \dfrac{\mathcal{L}}{\mathcal{I}} & \end{array}$$

(b) Se $\mathcal{I} \subset \mathcal{J}$, então $\mathcal{J}/\mathcal{I}$ é um ideal de $\mathcal{L}/\mathcal{I}$ e $(\mathcal{L}/\mathcal{I})/(\mathcal{J}/\mathcal{I}) \cong \mathcal{L}/\mathcal{J}$
(c) $(\mathcal{I} + \mathcal{J})/\mathcal{J} \cong \mathcal{I}/(\mathcal{I} \cap \mathcal{J})$

Dada uma álgebra de Lie abstrata qualquer, procuramos sempre representar esta álgebra em uma estrutura bem conhecida, geralmente um conjunto de matrizes.

Definição 1.28 *Uma* ***representação*** *de $\mathcal{L}$ em $\mathcal{V}$ é um homomorfismo $\phi : \mathcal{L} \to \mathbf{gl}(\mathcal{V})$, onde $\mathcal{V}$ é um espaço vetorial sobre $\mathbb{F}$.*

Uma representação é dita **fiel** quando $Ker\ \phi = 0$. A justificativa do nome, que nada tem a ver com a índole da álgebra de Lie (ou ainda de Sophus Lie), é que, desta maneira, teremos $\mathcal{L}$ isomorfa a um subconjunto de $\mathbf{gl}(\mathcal{V})$.

Note que uma representação ρ de $\mathcal{L}$ em $\mathcal{V}$ induz um módulo, com a ação definida por $x \cdot a = \rho(x)(a)$.

Uma representação é **irredutível** quando o módulo $\mathcal{L} \times \mathcal{V}$ induzido pela representação é simples, isto é, não possui submódulos não triviais, e **completamente redutível** quando tal módulo é soma direta de módulos simples.

Exemplo 1.29 *Se $\mathcal{L}$ é uma subálgebra de $\mathbf{gl}(\mathcal{V})$, então a inclusão define uma representação de $\mathcal{L}$ em $\mathcal{V}$, conhecida como* ***representação canônica****.*

Um importante exemplo é a **representação adjunta**:

$$\begin{aligned} \mathbf{ad} : \mathcal{L} &\to Der(\mathcal{L}) \\ x &\mapsto \mathbf{ad}\ x : \ \mathcal{L} \to \mathcal{L} \\ & \qquad\quad y \mapsto [xy] \end{aligned}$$

Exemplifique-mo-la:

Exemplo 1.30 *A aplicação $\rho : \boldsymbol{sl}(2, \mathbb{F}) \to \boldsymbol{sl}(3, \mathbb{F})$ dada por*

$$\begin{pmatrix} a & b \\ c & -a \end{pmatrix} \in \mathbf{sl}(2, \mathbb{F}) \mapsto \begin{pmatrix} 2a & -2b & 0 \\ -c & 0 & b \\ 0 & 2c & -2a \end{pmatrix} \in \mathbf{sl}(3, \mathbb{F})$$

é uma representação de $\mathbf{sl}(2, \mathbb{F})$. De fato, seja a base $\{x, h, y\}$ de $\mathbf{sl}(2, \mathbb{F})$, onde

$$x = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix},\ h = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},\ y = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}.$$

Os colchetes dos elementos da base são dados por

$$[h, x] = 2x,\ [h, y] = 2y,\ [y, x] = h.$$

Temos que $\{\rho(x), \rho(y), \rho(h)\}$ é uma base da álgebra $Im(\rho)$ que tem as mesmas constantes de estrutura.

Exemplo 1.31 *Repare que se calcularmos $\boldsymbol{ad}(\mathbf{sl}(2, \mathbb{F}))$ teremos:*

$$\begin{aligned} \boldsymbol{ad}x(x) = 0, \boldsymbol{ad}x(y) = h, \boldsymbol{ad}x(h) = -2x \\ \boldsymbol{ad}y(x) = -h, \boldsymbol{ad}y(y) = 0, \boldsymbol{ad}y(h) = 2y \end{aligned}$$

$$\boldsymbol{ad}h(x) = 2x, \boldsymbol{ad}h(y) = -2y, \boldsymbol{ad}h(h) = 0$$

Em forma matricial:

$$\boldsymbol{ad}x = \begin{pmatrix} 0 & 0 & -2 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \boldsymbol{ad}y = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 2 \\ -1 & 0 & 0 \end{pmatrix}, \boldsymbol{ad}h = \begin{pmatrix} 2 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Assim, o exemplo anterior é a matriz de $\boldsymbol{ad}$ *em relação à base* $\{x, h, y\}$*!*

Qual será o núcleo da representação adjunta? Bom, se $x \in \mathcal{L}$ é tal que $\mathbf{ad}x = 0$, então $\mathbf{ad}x(y) = 0, \forall y \in \mathcal{L}$, ou seja, $[xy] = 0$, $\forall y \in \mathcal{L}$. Assim, $Ker(\mathbf{ad}) = Z(\mathcal{L})$, pois é claro que se $x \in Z(\mathcal{L})$ então $\mathbf{ad}x = 0$.

Sabemos que se $\mathcal{L}$ for simples (e não-abeliana), então $Z(\mathcal{L})$=0. Logo $Ker(\mathbf{ad}) = 0$, e $\mathbf{ad}$ é um monomorfismo. Portanto,

$$\mathcal{L} \cong W \subseteq \mathbf{gl}(\mathcal{L}),$$

ou seja, toda álgebra de Lie simples é isomorfa a uma álgebra de Lie linear. Para deixar registrado:

Lema 1.32 *Se* $\mathcal{L}$ *é uma álgebra de Lie simples, então* $\mathcal{L}$ *é isomorfa a uma álgebra de Lie linear.*

1.6 Álgebras solúveis

Seja $\mathcal{L}$ é uma álgebra de Lie. Dados dois conjuntos $A, B \subseteq \mathcal{L}$, usaremos o símbolo $[A, B]$ para denotar o subespaço gerado por $\{[x, y] \mid x \in A, y \in B\}$.

Definamos os subespaços

$$\mathcal{L}^{(0)} = \mathcal{L},\ \mathcal{L}^{(1)} = [\mathcal{L}, \mathcal{L}],\ \mathcal{L}^{(2)} = [\mathcal{L}^{(1)}, \mathcal{L}^{(1)}],\ \ldots\ \mathcal{L}^{(i)} = [\mathcal{L}^{(i-1)}, \mathcal{L}^{(i-1)}].$$

Proposição 1.33 *Os subespaços* $\mathcal{L}^{(i)}$ *definidos acima são ideais de* $\mathcal{L}$ *e* $\mathcal{L}^{(i+1)} \subseteq \mathcal{L}^{(i)}$*.*

A sequência de ideais definida acima é chamada **série derivada de** $\mathcal{L}$, e suas componentes são as **subálgebras derivadas de** $\mathcal{L}$.

Definição 1.34 *Uma álgebra de Lie* $\mathcal{L}$ *é* ***solúvel*** *se alguma de suas álgebras derivadas se anula, isto é, se* $\mathcal{L}^{(k)} = 0$*, para algum* $k \geq 1$*.*

Exemplo 1.35 *Se* $\mathcal{L}$ *é abeliana, então* $\mathcal{L}$ *é solúvel, pois* $[\mathcal{L}, \mathcal{L}] = 0$*, ou seja, a primeira álgebra derivada,* $\mathcal{L}'$*, se anula.*

Exemplo 1.36 *Álgebras de Lie simples não são solúveis, pois se* $\mathcal{L}$ *é simples, então* $\mathcal{L} = [\mathcal{L}, \mathcal{L}]$ *e as álgebras derivadas são sempre iguais à própria álgebra* $\mathcal{L}$*.*

Exemplo 1.37 (A solubilidade de $\mathrm{t}(n,\mathbb{F})$) *Seja $\mathcal{L} = \mathrm{t}(n,\mathbb{F})$ a álgebra das matrizes triangulares superiores. Que $\mathcal{L}$ é uma álgebra de Lie segue do fechamento para o colchete*

$$[xy] = xy - yx,\ x, y \in \mathcal{L}.$$

Uma base para $\mathcal{L}$ é a formada pelas matrizes elementares e_{ij} com $i \leq j$; logo, sua dimensão é $1 + 2 + \ldots + n = n(n+1)/2$. Vamos mostrar que $\mathcal{L}$ é solúvel.
Repare que podemos decompor $\mathcal{L}$ como

$$\mathcal{L} = \mathrm{u}(n,\mathbb{F}) \oplus \mathrm{d}(n,\mathbb{F}),$$

onde $\mathrm{u}(n,\mathbb{F})$ é a álgebra das matrizes triangulares estritamente superiores e $\mathrm{d}(n,\mathbb{F})$ a álgebra das matrizes diagonais.
Como $e_{ii}e_{il} = e_{il}$ para $i < l$ e $\mathrm{u}(n,\mathbb{F})$ é fechado para a multiplicação, segue que $\mathrm{u}(n,\mathbb{F}) \subseteq [\mathcal{L},\mathcal{L}]$.
Como $e_{ii}e_{jj} = 0$ para $i \neq j$, $[\mathrm{d}(n,\mathbb{F}), \mathrm{d}(n,\mathbb{F})] = 0$. Consequentemente, $[\mathcal{L},\mathcal{L}] = \mathrm{n}(u,\mathbb{F})$.
Definamos o nível de um elemento e_{ij} como $j - i$. Assim, assumindo $i < j$ e $k < l$,

$$[e_{ij}, e_{kl}] = \delta_{jk} e_{il}.$$

Assim, $\mathcal{L}^{(2)}$ será gerado pelos e_{ij} de nível maior ou igual a 2. Intuitivamente, em $\mathcal{L}$, os elementos com menor nível eram os e_{ii}. Já em $[\mathcal{L},\mathcal{L}]$, estes passaram a ser os $e_{i,i+1}$. Agora, em $\mathcal{L}^{(2)}$, são os $e_{i,i+2}$. Quando estivermos em $\mathcal{L}^{(n-1)}$, os de menor nível serão os elementos $e_{i,i+n-1}$ e restará para i assumir o valor n, ou seja, teremos a álgebra gerada pela matriz e_{nn}. Mas

$$[e_{nn}, e_{nn}] = 0$$

e, portanto, existirá um k tal que $\mathcal{L}^{(k)} = 0$. Logo, $\mathcal{L}$ é solúvel.

Vamos agora a algumas observações sobre solubilidade.

Proposição 1.38 *Seja $\mathcal{L}$ uma álgebra de Lie.*
(a) Se $\mathcal{L}$ é solúvel então todas as subálgebras e imagens de $\mathcal{L}$ por algum homomorfismo são solúveis.
(b) Se $\mathcal{I} \trianglelefteq \mathcal{L}$ é um ideal e $\mathcal{L}$ é solúvel, então $\mathcal{L}/\mathcal{I}$ é solúvel.
(c) Se $\mathcal{I} \trianglelefteq \mathcal{L}$ é um ideal solúvel tal que $\mathcal{L}/\mathcal{I}$ é solúvel, então $\mathcal{L}$ é solúvel.
(d) Se $\mathcal{I}, \mathcal{J} \trianglelefteq \mathcal{L}$ são ideais solúveis, então $\mathcal{I} + \mathcal{J}$ é um ideal solúvel.
Prova. *(a) Para a primeira parte, basta observar que se W é uma subálgebra de $\mathcal{L}$, $W^{(i)} \subset \mathcal{L}^{(i)}$. Para a segunda, use o Teorema 1.27.*
(b) Novamente usando o Teorema 1.27, como $(\mathcal{L}/\mathcal{I})^{(k)} = \pi(\mathcal{L}^{(k)})$, se $\mathcal{L}^{(k)} = 0$ teremos que $(\mathcal{L}/k)^{(k)} = 0$.
(c) Seja k tal que $(\mathcal{L}/\mathcal{I})^{(k)} = 0$. Como $(\mathcal{L}/\mathcal{I})^{(k)} = \pi(\mathcal{L}^{(k)})$, temos $\pi(\mathcal{L}^{(k)}) = 0$. Logo $(\mathcal{L}/\mathcal{I})^{(k)} \subset \mathcal{I}$. Como $\mathcal{I}$ é solúvel, existe m tal que $\mathcal{I}^{(m)} = 0$. Assim, $\mathcal{L}$ é solúvel.
(d) Se $\mathcal{I}^{(n)} = 0$ e $\mathcal{J}^{(m)} = 0$, basta tomar a $(n+m)$-ésima derivada de $\mathcal{I} + \mathcal{J}$. ∎

1.7 Álgebras nilpotentes

Assim como na seção anterior, vamos definir uma sequência de subálgebras.

Definamos a série central descendente de $\mathcal{L}$ por

$$\mathcal{L}^0 = \mathcal{L},\ \mathcal{L}^1 = [\mathcal{L},\mathcal{L}](= \mathcal{L}^{(1)}),\ \mathcal{L}^2 = [\mathcal{L},\mathcal{L}^1],\ \ldots, \mathcal{L}^i = [\mathcal{L},\mathcal{L}^{i-1}].$$

Proposição 1.39 *1.* $[\mathcal{L}^i, \mathcal{L}^j] \subset \mathcal{L}^{i+j}$
2. $\mathcal{L}^k = \langle [x_1, \ldots, [x_{k-1}, x_k], \ldots] \mid x_i \in \mathcal{L} \rangle$*, ou seja,* $\mathcal{L}^k$ *é gerado por todos os possíveis colchetes envolvendo* k *elementos de* $\mathcal{L}$*.*

Observação 1.40 *Pela caracterização de* $\mathcal{L}^k$ *dada na proposição anterior, segue que:*
1. $\mathcal{L}^{k+1} \subset \mathcal{L}^k$*, pois um produto de* $k+1$ *elementos é também produto de* k *elementos.*
2. $\mathcal{L}^k$ *é um ideal, pois* $[\mathcal{L}, \mathcal{L}^k] = \mathcal{L}^{k+1} \subset \mathcal{L}^k$*.*
3. O nome série descendente se justifica, pois $\mathcal{L} = \mathcal{L}^1 \supseteq \mathcal{L}^2 \supseteq \ldots \supseteq \mathcal{L}^k \supseteq \ldots$

Definição 1.41 *Uma álgebra de Lie* $\mathcal{L}$ *é* ***nilpotente*** *se algum termo de sua série central descendente se anular em algum momento, isto é, se* $\mathcal{L}^k = \{0\}$*, para algum* $k > 0$*.*

Exemplo 1.42 *Se* $\mathcal{L}$ *for abeliana, então* $\mathcal{L}$ *será nilpotente, pois* $\mathcal{L}^2 = 0$*.*

Claramente, $\mathcal{L}^{(i)} \subseteq \mathcal{L}^i$ para algum i. Assim, álgebras nilpotentes são solúveis. A recíproca, no entanto, é falsa. Vejamos.

Exemplo 1.43 *Considere novamente* $\mathcal{L} = \mathrm{u}(t, \mathbb{F})$*. Como já foi visto antes,* $\mathcal{L}^1 = \mathcal{L}^{(1)} = \mathrm{u}(n, \mathbb{F})$ *e* $\mathcal{L}^2 = [\mathcal{L}, \mathcal{L}^1] = \mathcal{L}^1$*. É fácil perceber que* $\mathcal{L}^i = \mathcal{L}^1$*, para todo* $i > 1$*, ou seja,* $\mathcal{L}$ *não é solúvel. Por outro lado,* $\mathcal{L}$ *é nilpotente, visto que* $\mathcal{L}^i$ *é gerada pelos* e_{ij} *de nível maior ou igual a* $i + 1$*.*

Outros exemplo naturais podem ser construídos com base na seguinte proposição:

Proposição 1.44 *O centro de uma álgebra de Lie nilpotente* $\mathcal{L}$ *é não-trivial.*
Prova. *Seja* k *tal que* $\mathcal{L}^k \neq 0$ *e* $\mathcal{L}^{k+1} = 0$*. Então* $\mathcal{L}^k \subset Z(\mathcal{L})$*, pois* $[\mathcal{L}, \mathcal{L}^k] = \mathcal{L}^{k+1} = 0$*.* ■

Por outro lado, o centro de uma álgebra solúvel pode se anular. Veremos exemplos mais tarde. Vamos a uma proposição nos moldes da Proposição 1.38.

Proposição 1.45 *Seja* $\mathcal{L}$ *uma álgebra de Lie.*
(a) Se $\mathcal{L}$ *é nilpotente, então todas as imagens de* $\mathcal{L}$ *via homomorfismos e todas as subálgebras de* $\mathcal{L}$ *são nilpotentes.*
(b) Se $\mathcal{L}/Z(\mathcal{L})$ *é nilpotente, então* $\mathcal{L}$ *é nilpotente.*
(c) Se $\mathcal{L}$ *é nilpotente e* $\mathcal{L} \neq 0$*, então* $Z(\mathcal{L}) \neq 0$*.*

Se $\mathcal{L}$ é uma álgebra nilpotente, existe um inteiro k tal que todos os colchetes envolvendo k elementos de $\mathcal{L}$ se anulam. Em particular, $[x, \ldots, [x, y] \ldots] = 0$, com x aparecendo $k-1$ vezes. Melhorando a expressão, teremos

$$(\mathbf{ad}x)^{k-1} = 0, \ \forall x \in \mathcal{L}.$$

Assim, nas álgebras nilpotentes, as adjuntas de seus elementos são transformações lineares nilpotentes. A recíproca também é verdade: se $\mathcal{L}$ é uma álgebra de dimensão finita tal que $\mathbf{ad}x$ é nilpotente para todo $x \in \mathcal{L}$, então $\mathcal{L}$ é nilpotente. Este é o conteúdo do Teorema de Engel, e desenvolveremos algumas ferramentas antes de dar sua demonstração.

1.8 Radicais

Iremos definir aqui o que são os radicais solúvel e nilpotente em uma álgebra de Lie. O radical solúvel é importante para determinar a semi-simplicidade de uma álgebra de Lie e o nilpotente é peça fundamental para os algoritmos que iremos apresentar nos próximos capítulos.

Sejam $\mathcal{L}$ uma álgebra de Lie, $\mathcal{S}$ um ideal solúvel maximal e $\mathcal{N}$ um ideal nilpotente maximal.

Suponha que $\mathcal{I}$ seja um ideal solúvel maximal diferente de $\mathcal{S}$. Assim, $\mathcal{S}+\mathcal{I} = \mathcal{S}$ e $\mathcal{S}+\mathcal{I} = \mathcal{I}$, pelas maximalidades de $\mathcal{S}$ e $\mathcal{I}$. Mas como $\mathcal{I} \subsetneq \mathcal{S}+\mathcal{I}$ e $\mathcal{S} \subsetneq \mathcal{S}+\mathcal{I}$, temos um absurdo. Portanto só existe um ideal solúvel maximal em uma álgebra de Lie $\mathcal{L}$, denotado por $Rad_S(\mathcal{L})$ e denominado **Radical Solúvel** de $\mathcal{L}$

Com um raciocínio análogo ao anterior, encontramos que também é único o ideal nilpotente maximal, denotado por $Rad(\mathcal{L})$ e denominado **Radical Nilpotente** de $\mathcal{L}$. Apresentamos um algoritmo para o cálculo desta estrutura na pág. 83.

Observação 1.46 *Como álgebras nilpotente são solúveis (veja pág. 47), segue que* $Rad(\mathcal{L}) \subset Rad_S(\mathcal{L})$.

Observação 1.47 *A definição de radical nilpotente aqui apresentada para álgebras de Lie pode parecer um pouco diferente daquela apresentada na Seção 1 para álgebras associativas, considerando elementos fortemente nilpotentes. Na verdade, as definições são equivalentes, com o refinamento de se definir o conceito de nilpotentes fortes em álgebras de Lie, já que* $[x, x] = 0$, *para todo* $x \in \mathcal{L}$. *Veja que, buscando o anulamento da série derivada, queremos na verdade que* $[x_1, [x_2, [x_3, [x_4 \dots [x_n, x_{n+1}] \dots]$ *seja zero, para quaisquer* $x_i \in \mathcal{L}$, *com* $1 \leq i \leq n+1$, *o que é* quase *a definição de elementos nilpotentes fortes, pois* $x \in \mathcal{L}$ *é sempre nilpotente (*$[x, x] = 0$*).*

Seguem algumas propriedades e exemplos.

Proposição 1.48 *Seja* $\mathcal{L}$ *uma álgebra de Lie. Então* $\mathcal{L}$ *é solúvel se, e só se,* $\mathcal{L} = Rad_S(\mathcal{L})$.

Alguns exemplos:

Exemplo 1.49 $Rad_S(\mathbf{gl}(n, \mathbb{F})) = Z(\mathbf{gl}(n, \mathbb{F}))$
Com efeito, note que

$$Z(\mathbf{gl}(n, \mathbb{F})) = \langle 1_{\mathcal{A}} \rangle$$

e que, como $Z(\mathbf{gl}(n, \mathbb{F}))$ *é abeliano, também é solúvel (a notação* $\langle x \rangle$ *significa "gerado por x"). Além do mais,*

$$\mathbf{gl}(n, \mathbb{F}) = \mathbf{sl}(n, \mathbb{F}) \oplus Z(\mathbf{gl}(n, \mathbb{F}))$$

e, portanto, $\mathbf{gl}(n, \mathbb{F})/Z(\mathbf{gl}(n, \mathbb{F})) \cong \mathbf{sl}(n, \mathbb{F})$. *Assim,* $Z(\mathbf{gl}(n, \mathbb{F}))$ *é o único ideal solúvel de* $\mathbf{gl}(n, \mathbb{F})$, *pois* $\mathbf{sl}(n, \mathbb{F})$ *é simples.*

Agora vamos a um dos resultados mais importantes sobre o radical nilpotente de uma álgebra de Lie, que nos dá um certo "procedimento" para calculá-lo. Este teorema foi enunciado e provado por Jacobson por volta de 1945.

Teorema 1.50 (Jacobson) *Seja $\mathcal{L}$ uma álgebra de Lie e $\mathcal{A}$ a álgebra associativa gerada pelas transformações lineares $\boldsymbol{ad}x$, com $x \in \mathcal{L}$, ou seja, $\mathcal{A} = \boldsymbol{ad}(\mathcal{L})$. Então um elemento $x \in \mathcal{L}$ pertence ao $Rad(\mathcal{L})$ se, e só se, $\boldsymbol{ad}x \in Rad(\mathcal{A})$.*[1]

A demonstração foge aos objetivos desta monografia e pode ser encontrada em [JACOBSON].

1.9 Representações de Álgebras Nilpotentes

Na verdade, o que iremos fazer aqui é mostrar que para uma álgebra de Lie nilpotente de transformações lineares é possível encontrar uma base em que as matrizes desses elementos são todas triangulares superiores com zeros na diagonal principal, e desse resultado sairá o Teorema de Engel e, deste, que numa representação qualquer de uma álgebra nilpotente, os elementos da álgebra se decompõem, em alguma base, em blocos triangulares superiores semelhantes aos blocos de Jordan em que se decompõe uma transformação linear qualquer.

Relembrando alguns conceitos, a forma canônica de Jordan para um endomorfismo T sobre um corpo algebricamente fechado é uma expressão de T na forma matricial como soma de blocos

$$\begin{pmatrix} a & 0 & 0 & 0 & 0 \\ 1 & a & 0 & 0 & 0 \\ 0 & 1 & a & 0 & 0 \\ & & \ddots & \ddots & \\ 0 & 0 & 0 & 1 & a \end{pmatrix}.$$

Como a matriz diagonal $(a, \ldots, a)$ comuta com a matriz nilpotente com $1's$ abaixo da diagonal principal e 0 nas outras posições, T pode ser escrita como a soma de duas matrizes, uma diagonal e uma nilpotente, que comutam.

Dizemos que uma matriz x é **semi-simples** se as raízes de seu polinômio minimal forem todas distintas. Se $\mathbb{F}$ é um corpo algebricamente fechado, então x é semi-simples se, e só se, x é **diagonalizável**. Como duas transformações diagonalizáveis que comutam podem ser simultâneamente diagonalizáveis, a soma de duas transformações semi-simples é também semi-simples.

Outro fato da Álgebra Linear que vale ser mencionado aqui é que, se $x \in End\ \mathcal{V}$, então existem x_s e x_n, semi-simples e nilpotente, respectivamente, tais que $x = x_n + x_s$, e, mais que isso, tanto x_n como x_s são polinômios em x.

Definição 1.51 *Dizemos que uma representação ρ de $\mathcal{L}$ no espaço vetorial $\mathcal{V}$ é uma **representação nilpotente** (nil-representação) se $\rho(x)$ é nilpotente para todo $x \in \mathcal{L}$.*

Exemplo 1.52 *A representação adjunta é uma nil-representação de uma álgebra nilpotente.*

[1] Note que $Rad(\mathcal{A})$ e $Rad(\mathcal{L})$ são conjuntos definidos de maneira diferente, apesar da notação. O primeiro deles está definido na página 4, enquanto o segundo só aparece na página 48.

Proposição 1.53 *Seja $\mathcal{V}$ um espaço vetorial de dimensão finita e $y \in \mathbf{gl}(\mathcal{V})$. Suponha que y seja nilpotente. Então $\boldsymbol{ad}y$ também é nilpotente e, portanto, se*

$$\rho : \mathcal{L} \to \mathbf{gl}(\mathcal{V})$$

é uma nil-representação, então

$$x \mapsto \boldsymbol{ad}\rho(x)$$

também é uma nil-representação.
Prova. *Como y é nilpotente, em alguma base de $\mathcal{V}$, y se escreve como uma matriz triangular superior com zeros na diagonal. Com isso, é fácil ver que*

$$Im(\boldsymbol{ad}y)^k \subset \{c = (c_{ij})_{n\times n} \mid i - j \geq n - k \Rightarrow c_{ij} = 0\}.$$

Logo, $(\boldsymbol{ad}y)^{2n}$ se anula. ■

Vamos agora a um lema que ajudará a demonstrar muitas das proposições a seguir. As demonstrações omitidas podem ser encontradas em [SanMARTIN, p. 59-61].

Lema 1.54 *Seja $\mathcal{V} \neq 0$ um espaço vetorial de dimensão finita e $\mathcal{L}$ uma álgebra de Lie linear. Suponha que todo $x \in \mathcal{L}$ é nilpotente. Então existe $v \in \mathcal{V}$, $v \neq 0$, tal que $xv = 0$, $\forall x \in \mathcal{L}$.*

Os resultados a seguir nos garantem a existência de uma base na qual todos os elementos de uma nil-representação são triangulares superiores - além de provar o Teorema de Engel.

Teorema 1.55 *Seja $\mathcal{V}$ um espaço vetorial de dimensão finita e $\mathcal{L}$ uma álgebra de Lie linear tal que $\forall x \in \mathcal{L}$, x é nilpotente. Então existem subespaços*

$$0 = \mathcal{V}_0 \subset \mathcal{V}_1 \subset \ldots \subset \mathcal{V}_{n-1} \subset \mathcal{V}_n = \mathcal{V}$$

tais que $x\mathcal{V}_i \subset \mathcal{V}_{i-1}$ para $i = 1, \ldots, n$. Esses subespaços podem ser definidos indutivamente por

$$\mathcal{V}_0 = 0$$

$$\mathcal{V}_i = \{v \in \mathcal{V} \mid xv \in \mathcal{V}_{i-1}, \forall x \in \mathcal{L}\}.$$

Em particular, estendendo sucessivamente as bases dos subespaços $\mathcal{V}_i$, chega-se a uma base β de $\mathcal{V}$ tal que a matrix de x em relação a β é triangular superior com zeros na diagonal para todo $x \in \mathcal{L}$

No Exemplo 1.43 verificamos que a álgebra das matrizes triangulares superiores é uma álgebra nilpotente. Pelo teorema anterior, qualquer álgebra de matrizes cuja representação canônica é uma nil-representação está contida na álgebra das matrizes triangulares superiores e, como tal, é nilpotente.

Agora um corolário importantíssimo desse teorema.

Corolário 1.56 *Seja $\mathcal{L}$ uma álgebra de Lie de dimensão finita e suponha que $\boldsymbol{ad}$ seja uma nil-representação. Então a série central ascendente satisfaz*

$$0 = \mathcal{L}_0 \subset \mathcal{L}_1 \subset \ldots \subset \mathcal{L}_n = \mathcal{L}$$

para algum $n \in \mathbb{N}$.

Teorema 1.57 (Teorema de Engel) *Seja $\mathcal{L}$ uma álgebra de Lie de dimensão finita e suponha que, para todo $x \in \mathcal{L}$, $\mathbf{ad}x$ seja nilpotente. Então $\mathcal{L}$ é nilpotente.*
Prova. *Pelo corolário anterior, a série central ascendente termina em $\mathcal{L}_n = \mathcal{L}$. Dessa forma, procedendo por indução e usando o fato de que $[\mathcal{L}, \mathcal{L}_i] \subset \mathcal{L}_{i-1}$, mostra-se que a série central descendente está contida na ascendente com a relação $\mathcal{L}^i \subset \mathcal{L}_{n-i+1}$, para $i = 1, \ldots, n$. Assim, $\mathcal{L}^{n+1} = 0$ e $\mathcal{L}$ é nilpotente.* ■

Este último teorema é realmente muito forte. Uma álgebra é nilpotente se todos os produtos que envolvem uma certa quantidade de elementos se anulam. No entanto, para que a representação adjunta de uma álgebra seja nilpotente, pede-se bem menos que isso: basta que certos produtos, que envolvem somente dois elementos, um deles aparecendo uma única vez, se anulem. E apenas com isso já teremos a nilpotência da álgebra de Lie.

O Teorema de Engel nos dá outro método para mostrar que uma álgebra de Lie $\mathcal{L}$ é não-nilpotente. Basta encontrar $x \in \mathcal{L}$ tal que $\mathbf{ad}x$ não é nilpotente. Vamos a um exemplo:

Exemplo 1.58 *Seja $\mathcal{L}$ a álgebra de Lie de dimensão 5 com base $\{x_1, \ldots, x_5\}$ satisfazendo $[x_1, x_4] = 2x_1$, $[x_2, x_3] = x_1$, $[x_2, x_4] = x_2$, $[x_2, x_5] = -x_3$, $[x_3, x_4] = x_3$ e $[x_3, x_5] = x_2$. Assim, como $\mathbf{ad}x_4(x_1) = -2x_1$, a transformação $\mathbf{ad}x_4$ é não-nilpotente. Logo, $\mathcal{L}$ não é nilpotente.*

1.10 Decomposições de Jordan e pesos das representações

Já temos condições de obter algumas informações sobre as representações das álgebras nilpotentes. Em geral não há motivos para que uma representação de uma álgebra nilpotente seja nilpotente.

Exemplo 1.59 *Se $\mathcal{L} = \mathbf{d}(n, \mathbb{F})$ é a álgebra das matrizes diagonais $n \times n$, então $\mathcal{L}$ é abeliana e nilpotente. A representação canônica de $\mathcal{L}$, dada pela inclusão, não é uma nil-representação, pois uma matriz diagonal não é nilpotente, a menos que ela seja a matriz nula.*

Exemplo 1.60 *Seja $\mathcal{L} \subseteq \mathbf{t}(n, \mathbb{F})$ a álgebra das matrizes triangulares superiores com os elementos da diagonal todos iguais, isto é, $\mathcal{L} = \mathbf{n}(n, \mathbb{F}) \oplus \mathbb{F}$. A representação canônica de $\mathcal{L}$ não é nilpotente, pois matrizes diagonais não são nilpotentes.*

A diferença entre uma representação arbitrária e uma nil-representação de uma álgebra nilpotente é que, em geral, podem aparecer autovalores não-nulos, desde que com um certo padrão de repetição, como no caso do segundo exemplo acima. Esse padrão de repetição é dado pelas decomposições de Jordan dos elementos da álgebra que, como vai ser visto a seguir, são compatíveis entre si, isto é, se realizam de maneira simultânea.

Vamos relembrar alguns conceitos de Álgebra Linear. Seja $\mathcal{V}$ um espaço vetorial de dimensão finita e $T : \mathcal{V} \to \mathcal{V}$ uma transformação linear. O Teorema da Decomposição Primária decompõe $\mathcal{V}$ em subespaços T-invariantes

$$\mathcal{V} = \mathcal{V}_1 \oplus \ldots \oplus \mathcal{V}_s$$

que são os auto-espaços generalizados

$$\mathcal{V}_i = \{v \in \mathcal{V} \mid p_i(T)^k(v) = 0, k \geq 1\}$$

onde os polinômios irredutíveis p_i, $i = 1, \ldots,$ s são as componentes primárias do polinômio minimal de T, $p = p_1^{m_1} \cdots p_s^{m_s}$.

No caso em que o corpo de escalares é algebricamente fechado, $p_i(T) = T - \lambda_i I$ com λ_i autovalor de T e os subespaços da decomposição primária se escrevem como

$$\mathcal{V}_i = \{v \in \mathcal{V} \mid (T - \lambda_i I)^k(v) = 0, k \geq 1\}.$$

Denotaremos esses subespaços por $\mathcal{V}_{\lambda_j}$.

Proposição 1.61 *Suponha que o corpo de escalares é algebricamente fechado. Sejam T, U transformações lineares de $\mathcal{V}$, e $\mathcal{V}_{\lambda_i}$ os auto-espaços generalizados de T. Então $U\mathcal{V}_{\lambda_i} \subset \mathcal{V}_{\lambda_i}$ para todo i se, e só se, $\boldsymbol{ad}(T)^q(U) = 0$ para algum $q \geq 1$.*

Usando linguagem de representações, a proposição anterior garante que podemos decompor o espaço da representação em auto-espaços generalizados que são invariantes pela ação de qualquer elemento da álgebra. Em outras palavras, podemos obter uma decomposição da representação de uma álgebra nilpotente.

Teorema 1.62 *Suponha que $\mathbb{F}$ seja algebricamente fechado e tome uma representação de $\mathcal{L}$ em $\mathcal{V}$, espaço vetorial sobre $\mathbb{F}$, com dim $\mathcal{V} < \infty$ e $\mathcal{L}$ nilpotente. Então existem funcionais lineares $\lambda_1, \ldots, \lambda_s$ tais que se*

$$\mathcal{V}_{\lambda_i} = \{v \in \mathcal{V} \mid \forall x \in \mathcal{L}, \exists n \geq 1, (\rho(x) - \lambda_i(x))^n v = 0\}$$

então $\mathcal{V}_{\lambda_i}$ é $\mathcal{L}$-invariante e

$$\mathcal{V} = \mathcal{V}_{\lambda_1} \oplus \ldots \oplus \mathcal{V}_{\lambda_s}$$

Prova. *Daremos um esboço da prova. Considerando a decomposição dada por cada $\rho(x)$, existirão $\lambda_i : \mathcal{L} \to \mathbb{F}$ e subespaços $\mathcal{W}_1, \ldots \mathcal{W}_s$ tais que $\mathcal{V} = \mathcal{W}_1 \oplus \ldots \oplus \mathcal{W}_s$ e $\mathcal{W}_i \subset \mathcal{V}_{\lambda_i}$. Mostra-se que λ_i é linear, pois*

$$\lambda_i(x) = \frac{tr\ \rho_i(x)}{dim\ \mathcal{V}_{\lambda_i}}.$$

Como os funcionais $\lambda_i - \lambda_j$ são não-nulos e são em quantidade finita, pode-se considerar o auto-espaço generalizado associado a cada $\lambda_i(x)$, ou seja, $\mathcal{V}_{\lambda_i(x)}$. Como os $\lambda_i(x)$ são os autovalores de $\rho(x)$ e estes são distintos, a soma $\mathcal{V}_{\lambda_1(x)} + \ldots + \mathcal{V}_{\lambda_s(x)}$ é direta. Como $\mathcal{W}_i \subset \mathcal{V}_{\lambda_i}$, segue que $\mathcal{W}_i = \mathcal{V}_{\lambda_i}$ e a soma anterior coincide com $\mathcal{V}$. ■

Definição 1.63 *Se $\mathcal{L}$ uma álgebra de Lie e ρ uma representação de $\mathcal{L}$ em $\mathcal{V}$. Um **peso** de ρ é um funcional linear $\lambda : \mathcal{L} \to \mathbb{F}$ tal que o subespaço $\mathcal{V}_\lambda \subset \mathcal{V}$ definido por*

$$\mathcal{V}_\lambda = \{v \in \mathcal{V} \mid \forall x \in \mathcal{L}, \exists n \geq 1, (\rho(x) - \lambda(x)I)^n v = 0\}$$

*satisfaz $\mathcal{V}_\lambda \neq 0$. O subespaço $\mathcal{V}_\lambda$ é chamado de **subespaço de pesos associado a** λ. A dimensão de $\mathcal{V}_\lambda$ é chamada de multiplicidade de λ.*

Os pesos de uma representação são, portanto, os autovalores dos elementos da álgebra. Pelo teorema anterior, existe a garantia de que, se o corpo de escalares for algebricamente fechado, as representações de dimensão finita das álgebras nilpotentes admitem pesos. Vale ressaltar que pedir o fechamento algébrico do corpo de escalares não é, de forma alguma, uma hipótese forte, já que, em geral, isto é exigido. Já a nilpotência da álgebra não pode ser descartada.

Exemplo 1.64 *Se $\mathcal{L}$ é a álgebra das matrizes diagonais em relação à base $\{e_1, \ldots, e_n\}$, os funcionais λ_i definidos por*

$$\lambda_i : \mathcal{L} \to \mathbb{F}, \quad \begin{pmatrix} a_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & a_n \end{pmatrix} \mapsto a_i$$

são pesos da representação canônica de $\mathcal{L}$. Neste caso, $\mathcal{V}_{\lambda_i}$ é o subespaço gerado por e_i. Estes são os únicos pesos desta representação.

Exemplo 1.65 *Seja agora*

$$\mathcal{L} = \{ \begin{pmatrix} \lambda & & * \\ & \ddots & \\ 0 & & \lambda \end{pmatrix} \}$$

a álgebra das matrizes triangulares superiores com todos os elementos da diagonal iguais. O único peso da representação canônica é dado pelo funcional

$$\begin{pmatrix} \lambda & & * \\ & \ddots & \\ 0 & & \lambda \end{pmatrix} \mapsto \lambda.$$

Assim, o subespaço de pesos associado é toda a representação.

Exemplo 1.66 *Considere* $\mathbf{sl}(2, \mathbb{C})$. *Pela representação canônica, os auto-espaços de*

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

são os subespaços gerados pelos elementos da base canônica

$$\{(1,0), (0,1)\},$$

enquanto os auto-espaços de

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

são os subespaços gerados por

$$\{(1,1), (1,-1)\}.$$

Como esses subespaços tem interseção nula, essa representação não admite pesos.

1.11 Critérios de Cartan

Após a introdução às álgebras de Lie dada nas seções anteriores, onde definimos a maioria das estruturas gerais relacionadas a tais álgebras e provamos o importante Teorema de Engel, seguiremos na direção da determinação de critérios eficientes para determinar se uma álgebra é solúvel ou nilpotente.

Nesta seção, a hipótese de que $\mathbb{F}$ é um corpo de característica 0 é fortificada para que $\mathbb{F}$ seja também algebricamente fechado.

Definição 1.67 *Seja $\mathcal{L}$ uma álgebra de Lie. Diremos que $\mathcal{L}$ é uma álgebra de Lie semi-simples quando $Rad_S(\mathcal{L}) = \{0\}$.*

Qual a justificativa do nome "semi-simples"? Como as álgebras de Lie semi-simples se relacionam com as simples? É o que veremos na próxima proposição.

Proposição 1.68 *Se $\mathcal{L}$ é uma álgebra de Lie simples, então $\mathcal{L}$ é semi-simples.*
Prova. *De fato, se $\mathcal{L}$ não tem ideais próprios, logo, não terá ideais solúveis não-triviais. Assim, $Rad_S(\mathcal{L}) = \{0\}$ e $\mathcal{L}$ é semi-simples.* ■

Exemplo 1.69 *Além dos muitos exemplos que podem ser conseguidos à luz da proposição anterior, também as álgebras de Lie abelianas são semi-simples, pelo mesmo motivo que as simples.*

Note que para qualquer álgebra de Lie $\mathcal{L}$, o quociente $\mathcal{L}/Rad_S(\mathcal{L})$ é sempre semi-simples. As álgebras de Lie semi-simples serão um dos elementos principais deste trabalho.

Antes de continuarmos, vamos relembrar algumas definições e propriedades que serão importantes para o que segue. Estas tangem sobre *formas bilineares*. Para maiores detalhes, consulte [CL, cap. 8].

Definição 1.70 *Seja $\mathcal{V}$ um espaço vetorial sobre um corpo $\mathbb{F}$. Uma forma bilinear sobre $\mathcal{V}$ é uma função $f : \mathcal{V} \times \mathcal{V} \to \mathbb{F}$ que satisfaz:*
(a) $f(\lambda u_1 + u_2, v) = \lambda f(u_1, v) + f(u_2, v),\ \forall \lambda \in \mathbb{F},\ \forall u_1, u_2, v \in \mathcal{V}$.
(b) $f(u, \lambda v_1 + v_2) = \lambda f(u, v_1) + f(u, v_2),\ \forall \lambda \in \mathbb{F},\ \forall u, v_1, v_2 \in \mathcal{V}$

Dizemos que uma forma bilinear f sobre $\mathcal{V}$ é **simétrica** se

$$f(u, v) = f(v, u)$$

para quaisquer $u, v \in \mathcal{V}$.

Exemplo 1.71 *O produto interno usual em $\mathcal{V} = \mathbb{R}^n$ dado por*

$$\langle v, u \rangle = v_1 u_1 + \ldots + v_n u_n$$

é uma forma bilinear simétrica.

Definição 1.72 *Sejam $\mathcal{V}$ um espaço vetorial sobre $\mathbb{F}$ e f uma forma bilinear sobre $\mathcal{V}$. A função $q : \mathcal{V} \to \mathbb{F}$ dada por $q(v) = f(v, v)$ é denominada* ***forma quadrática associada a*** *f.*

Exemplo 1.73 *Seja $\mathcal{V} = \mathbb{R}^n$ e f a forma bilinear definida pelo produto interno usual. A forma quadrática q associada a f é*

$$q(v) = f(v, v) = v_1^2 + \ldots + v_n^2,$$

ou seja, $q(v)$ é o quadrado da norma de v.

Proposição 1.74 (Fórmulas de Polarização) *Seja $\mathcal{V}$ um espaço vetorial sobre $\mathbb{F}$, f uma forma bilinear simétrica e q a forma quadrática associada a f. Então para cada u, v em $\mathcal{V}$ temos* $q(u+v) - q(u) - q(v) = f(u+v, u+v) - f(u,u) - f(v,v)$
$= f(u,v) + f(u,u) + f(v,u) + f(v,v) - f(u,u) - f(v,v)$
$= 2f(u,v)$

e, assim,

$$\frac{1}{2}\left(q(u+v) - q(u) - q(v)\right) = f(u,v).$$

Esta última igualdade é conhecida como ***Fórmula de Polarização****.*

Se $\mathcal{V}$ é um espaço vetorial e f uma forma bilinear simétrica, dizemos que f é *não-degenerada* quando para todo $\vec{0} \neq \alpha \in \mathcal{V}$, existe $\beta \in \mathcal{V}$ tal que $f(\alpha, \beta) \neq 0$, ou seja, nenhum $\alpha \in \mathcal{V}$ não-nulo tem a propriedade de que $f(\alpha, \beta) = 0$ para todo $\beta \in \mathcal{V}$.

Dada uma forma bilinear simétrica f, definimos uma transformação linear $L_f : \mathcal{V} \to \mathcal{V}^*$ por $L_f(\alpha) = f_\alpha = f(a, \cdot)$, ou seja, $L_f(\alpha)(\beta) = f(\alpha, \beta)$. Note que, para cada $\alpha \in \mathcal{V}$ realmente temos $f_\alpha \in \mathcal{V}^*$ (f_α é um funcional linear).

Definimos o **posto** de uma forma bilinear simétrica f como o posto da transformação L_f definida acima.

Proposição 1.75 *Uma forma bilinear simétrica f sobre um espaço vetorial n-dimensional é não-degenerada se, e só se, o posto da transformação linear $L_f : \mathcal{V} \to \mathcal{V}^*$ é n (L_f isomorfismo).*

A demonstração do resultado anterior leva em conta o **Teorema da Representação de Riez** para formas bilineares, que, sob certas condições, associa a cada forma bilinear f uma transformação linear T_f tal que $f(\alpha, \beta) = \langle \alpha, T_f(\beta) \rangle$, onde $\langle \cdot\ ,\ \cdot \rangle$ é um produto interno, e pode ser encontrada em [HK].

Corolário 1.76 *Seja $\mathcal{B} = \{v_1, \ldots, v_n\}$ uma base ortonormal de $\mathcal{V}$ e $f : \mathcal{V} \times \mathcal{V} \to \mathbb{F}$ uma forma bilinear simétrica. Então posto(f) = posto$([f]_\mathcal{B})$, onde $[f]_\mathcal{B}$ é a matriz de f dada por $[f]_\mathcal{B} = (a_{ij})$ com $a_{ij} = f(v_i, v_j)$.*

Como a forma f é não-degenerada se, e só se, *posto*$(f) = n$, segue que uma maneira prática de testar a não-degeneração é calcular o determinante da matriz $[f]_\mathcal{B}$. A forma será não-degenerada se, e só se, esse determinante for não-nulo.

Voltando para nossas álgebras de Lie, dada uma representação ρ de dimensão finita de $\mathcal{L}$, define-se em $\mathcal{L}$ a **forma traço** β_ρ que é a forma bilinear simétrica dada por

$$\beta_\rho(x, y) = \mathbf{tr}(\rho(x)\rho(y)).$$

Exemplo 1.77 *Se ρ é uma representação nilpotente, então $\beta_\rho(x^2) = 0$ para todo $x \in \mathcal{L}$, pois o traço de uma transformação linear nilpotente é zero.*

Para nossos interesses trataremos aqui somente da representação ρ como sendo a adjunta. Assim, denotaremos $\beta_{\mathbf{ad}}(x, y)$ por

$$\kappa(x, y) = \mathbf{tr}(\mathbf{ad}x \cdot \mathbf{ad}y)^1.$$

À essa forma, dá-se o nome de **forma de Cartan-Killing**.

[1] As expressões $\mathbf{tr}(\mathbf{ad}x \cdot \mathbf{ad}y)$ e $\mathbf{tr}(\mathbf{ad}x\ \mathbf{ad}y)$ têm o mesmo significado. O "pontinho" para denotar a multiplicação será usado somente para evitar ambigüidades.

Exemplo 1.78 *Vamos considerar ρ a inclusão de $\mathbf{sl}(2,\mathbb{F})$ no espaço das matrizes 2×2. Se $x\in\mathbf{sl}(2,\mathbb{F})$, então*

$$x=\begin{pmatrix} a & b\\ c & -a\end{pmatrix}.$$

Assim,

$$\boldsymbol{tr}(x\cdot x)=2(a^2+bc)$$

e essa é a expressão de $\beta_\rho(x,x)$. Considerando a base canônica de $\mathbf{sl}(2,\mathbb{F})$ dada por $\{x,h,y\}$, temos que

$$[\beta_\rho]=\begin{pmatrix} 0 & 0 & 1\\ 0 & 2 & 0\\ 1 & 0 & 0\end{pmatrix}.$$

Exemplo 1.79 *Novamente, vamos à forma de Cartan-Killing de $\mathbf{sl}(2,\mathbb{F})$, agora, considerando a representação adjunta, com a mesma base do exemplo anterior. Como já calculamos as matrizes adjuntas dos elementos da base canônica $\{x,h,y\}$ no Exemplo 1.30, vamos omitir os cálculos.*
Como $\kappa(x,y)=\boldsymbol{tr}(\boldsymbol{ad}x\cdot\boldsymbol{ad}y)$, temos a igualdade

$$\begin{pmatrix} \kappa(x,x) & \kappa(x,h) & \kappa(x,y)\\ \kappa(h,x) & \kappa(h,h) & \kappa(h,y)\\ \kappa(y,x) & \kappa(y,h) & \kappa(y,y)\end{pmatrix}=\begin{pmatrix} 0 & 0 & 4\\ 0 & 8 & 0\\ 4 & 0 & 0\end{pmatrix}$$

Note que

$$det\begin{pmatrix} 0 & 0 & 4\\ 0 & 8 & 0\\ 4 & 0 & 0\end{pmatrix}=-128\neq 0$$

e, portanto, κ é não-degenerada em $\mathbf{sl}(2,\mathbb{F})$.

Para o primeiro dos critérios de Cartan, usaremos os seguintes resultados, cujas demonstrações podem ser encontrada em [SanMARTIN, p. 85].

Lema 1.80 *Seja $\mathcal{L}$ uma álgebra de Lie de dimensão finita e suponha que sua forma de Cartan-Killing seja identicamente nula. Então $\mathcal{L}$ é solúvel.*

Teorema 1.81 *Uma álgebra de Lie $\mathcal{L}$ é solúvel se, e só se, $\kappa(x,y)=\boldsymbol{tr}(\boldsymbol{ad}x\cdot\boldsymbol{ad}y)=0$, para todo $x\in\mathcal{L}'$ e $y\in\mathcal{L}$.*

O teorema anterior bem poderia ser enunciado como um lema para o:

Teorema 1.82 *A forma de Cartan-Killing de $\mathcal{L}$ é não-degenerada se, e só se, $\mathcal{L}$ é semi-simples.*

Os critérios acima mostram o contraste existente entre as álgebra solúveis e as semi-simples. Por um lado, a forma de Cartan-Killing de uma álgebra solúvel é quase toda nula, revelando pouca informação acerca da álgebra. Já com as álgebras semi-simples, acontece o contrário. Como a forma de Cartan-Killing de uma álgebra semi-simples é não-degenerada, ela induz na álgebra um "produto interno", que oferece bem mais recursos que aqueles dados pela estrutura bilinear. Mais que isso, a existência de uma forma bilinear não-degenerada nas álgebras semi-simples limita a quantidade de classes de equivalência dessas álgebras, permitindo que elas possam ser bem classificadas e distinguidas, o que não acontece com as álgebras solúveis.

1.11.1 Uma breve visita às álgebras semi-simples

À luz dos comentários feitos no parágrafo anterior, vamos estabelecer alguns resultados sobre álgebras semi-simples usando as facilidades introduzidas pela forma de Cartan-Killing.

Primeiramente, sejam $\mathcal{L}$ uma álgebra semi-simples e $\mathcal{I} \trianglelefteq \mathcal{L}$ um ideal não-trivial de $\mathcal{L}$. Se $\langle .,. \rangle_{\mathcal{I}}$ é a forma de Cartan-Killing restrita à $\mathcal{I}$ e se $\mathcal{I}^{\perp}$ denota o complemento ortogonal de $\mathcal{I}$ em relação à $\langle .,. \rangle$, então $\mathcal{I}^{\perp}$ é um ideal complementar à $\mathcal{I}$.

Vamos a um análogo do Teorema de Wedderburn (0.12) para álgebras de Lie:

Teorema 1.83 *Seja $\mathcal{L}$ uma álgebra semi-simples. Então $\mathcal{L}$ se decompõe em soma direta $\mathcal{L} = \mathcal{L}_1 \oplus \ldots \oplus \mathcal{L}_s$, com $\mathcal{L}_i$ ideais simples. Nessa decomposição, $[\mathcal{L}_i, \mathcal{L}_j] = 0$, se $i \neq j$. Além do mais:*
1. o ortogonal $\mathcal{L}_i^{\perp}$ de uma componente simples, em relação à forma de Cartan-Killing é a soma das demais componentes;
2. os ideais de $\mathcal{L}$ são somas de algumas dessas componentes;
3. a decomposição é única.

Como após todo grande teorema, agora as grandes consequências:

Corolário 1.84 *Se $\mathcal{L}$ é simples, $\mathcal{L}' = \mathcal{L}$.*
Prova. *Como $\mathcal{L}'$ é um ideal de $\mathcal{L}$, o teorema anterior garante que existe um ideal $\mathcal{I}$ que complementa $\mathcal{L}'$. Dados $x, y \in \mathcal{I}$, temos que $[x, y] \in \mathcal{L}' \cap \mathcal{I}$, ou seja, $\mathcal{I}$ é abeliano. Logo $\mathcal{I} = 0$, ou seja, $\mathcal{L} = \mathcal{L}'$.* ■

Corolário 1.85 *Se $\mathcal{L}$ é semi-simples e $\mathcal{H}$ é uma álgebra abeliana, então o único homomorfismo de $\mathcal{L}$ em $\mathcal{H}$ é o nulo. Em particular, a única representação unidimensional de $\mathcal{L}$ é a nula e, para uma representação ρ qualquer, $\boldsymbol{tr}\rho(x) = 0$ para todo $x \in \mathcal{L}$*
Prova. *Se $\phi : \mathcal{L} \to \mathcal{H}$ é um homomorfismo, então $\phi[x, y] = 0$ para todo $x, y \in \mathcal{L}$ e, como $\mathcal{L} = \mathcal{L}'$, temos que $\phi = 0$.* ■

Corolário 1.86 *Se $\mathcal{L}$ é uma álgebra semi-simples e $\mathcal{I} \trianglelefteq \mathcal{L}$, então $\mathcal{L}/\mathcal{I}$ é semi-simples.*

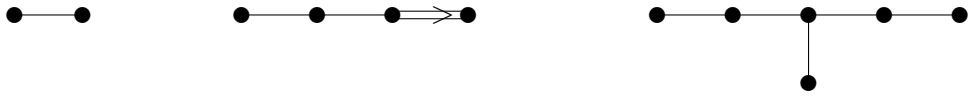
Capítulo 2

Espaços de Raízes e Diagramas de Dynkin

Neste capítulo, nossas álgebras são sobre corpos de característica 0.

Para o que iremos desenvolver aqui, serão necessárias algumas das definições dadas na Subseção 1.10, principalmente no que diz respeito a pesos e espaços de peso. As demonstrações omitidas podem ser encontradas em [HUMPHREYS] ou [SanMARTIN].

Iremos caracterizar as álgebras semi-simples por entes denominados "raízes", e conseguiremos descrever completamente uma álgebra de Lie por desenhos simpáticos como os abaixo:



2.1 Representações de $\mathbf{sl}(2,\mathbb{F})$

A álgebra de Lie das matrizes 2×2 sobre $\mathbb{F}$ de traço zero, $\mathbf{sl}(2,\mathbb{F})$, é das mais importantes[1].

Vamos deixar registrado aqui um teorema que será fundamental para o que segue:

Teorema 2.1 (Weyl) *Se ρ é uma representação de uma álgebra de Lie semi-simples, então ρ é completamente redutível.*

Como já vimos antes, a base canônica para $\mathbf{sl}(2,\mathbb{F})$ é

$$x = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \quad y = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \quad h = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$

que verificam as relações

$$[h,x] = 2x,\ [h,y] = -2y,\ [x,y] = h.$$

A álgebra $\mathbf{sl}(2,\mathbb{F})$ é simples (veja Proposição 1.30). Assim, pelo Teorema de Weyl (pág. 58), as representações de $\mathbf{sl}(2,\mathbb{F})$ se decompõem em representações irredutíveis. Portanto, para estudar tais representações, basta estudar as irredutíveis.

[1] Esta importância será melhor percebida após a Proposição 2.13, pág. 62.

Em uma representação de $\mathbf{sl}(2,\mathbb{F})$, a ação do elemento h é diagonal, ou seja, se $\rho : \mathbf{sl}(2,\mathbb{F}) \to End(\mathcal{V})$ é uma representação, então $\rho(h)v_i = \lambda_i v_i$.

Seja ρ uma representação irredutível de $\mathbf{sl}(2,\mathbb{F})$, e v um autovetor de $\rho(h)$ associado ao autovalor λ, ou seja, $\rho(h)(v) = \lambda v$. Como $\rho[h,x] = \rho(h)\rho(x) - \rho(x)\rho(h)$, segue $\rho(h)\rho(x) = \rho[h,x] + \rho(x) + \rho(h)$. Portanto

$$\rho(h)\rho(x)(v) = \rho[h,x](v) + \rho(x)\rho(h)(v) = (\lambda+2)\rho(x)(v)$$

Analogamente,

$$\rho(h)\rho(y)(v) = (\lambda-2)\rho(y)(v).$$

Indutivamente, segue

$$\rho(h)\rho(x)^k(v) = (\lambda+2k)\rho(x)^k(v)$$
$$\rho(h)\rho(y)^k(v) = (\lambda-2k)\rho(y)^k(v)$$

Vamos denotar a transformação $\rho(h)$ simplesmente por h, assim como x para $\rho(x)$ e y para $\rho(y)$. Como h age diagonalmente, podemos decompor $\mathcal{V}$ como soma direta de espaços

$$\mathcal{V}_\lambda = \{v \in \mathcal{V} \mid hv = \lambda v\}.$$

Quando $\mathcal{V}_\lambda \neq \{\vec{(}0)\}$, diremos que λ é um **peso** de h em $\mathcal{V}$ e que $\mathcal{V}_\lambda$ é um **espaço de peso**.

Com esta nova notação, equações anteriores podem ser reescritas como

Lema 2.2 *Seja $v \in \mathcal{V}_\lambda$. Então $x^n v \in \mathcal{V}_{\lambda+2n}$ e $y^n v \in \mathcal{V}_{\lambda-2n}$.*

Se $\mathcal{V}$ é um espaço vetorial de dimensão finita, existem no máximo n pesos, já que $\mathcal{V} = \coprod \mathcal{V}_\lambda$. Assim, existem $\tilde{\lambda},\ \lambda_0$ tais que $\mathcal{V}_{\tilde{\lambda}} \neq \{\vec{0}\},\ \mathcal{V}_{\lambda_0} \neq \{\vec{0}\}$ e $\mathcal{V}_{\tilde{\lambda}+k} = \{\vec{0}\} = \mathcal{V}_{\lambda_0}$, para todo $k > 0$.

Seja $v \in \mathcal{V}_\lambda$. Como $x : \mathcal{V}_\lambda \to \mathcal{V}_{\lambda+2}$ e $y : \mathcal{V}_\lambda \to \mathcal{V}_{\lambda-2}$, para índices $n_0, n_1 \in \mathbb{Z}$, teremos $x^{n_0}(v) = 0$ e $y^{n_1}(v) = 0$, pois acontecerá $x^{n_0}(v) \in \mathcal{V}_{\tilde{\lambda}+i}$ e $y^{n_1}(v) \in \mathcal{V}_{\lambda_0-i}$, $i > 0$. Logo, x e y são representados por endomorfismos nilpotentes. Se $v \in \mathcal{V}_{\tilde{\lambda}}$, v é chamado **vetor maximal**.

Teorema 2.3 *Seja ρ uma representação irredutível de $\mathbf{sl}(2,\mathbb{F})$ em $\mathcal{V}$ tal que dim $\mathcal{V} = n+1$. Então existe uma base $\{v_0,\ldots,v_n\}$ de $\mathcal{V}$ tal que, definindo $v_{-1} = 0 = v_{n+1}$, temos:*

$$\begin{cases} xv_i = i(n-i+1)v_{i-1} \\ hv_i = (n-2i)v_i \\ yv_i = v_{i+1} \end{cases}$$

Pelo teorema anterior, podemos perceber que as transformações lineares $x,\ y,\ h$, em relação à base $\mathcal{B} = \{v_0,\ldots,v_n\}$ dada pelo teorema anterior, têm matrizes:

$$[x] = \begin{pmatrix} 0 & n & 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 2(n-1) & 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 3(n-2) & & \ldots & 0 \\ 0 & 0 & 0 & 0 & 4(n-3) & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & \ldots & n \\ 0 & 0 & 0 & 0 & 0 & \ldots & 0 \end{pmatrix}$$

$$[y] = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 2 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 3 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & m-1 & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & m & 0 \end{pmatrix}$$

$$[h] = \begin{pmatrix} n & 0 & 0 & \dots & 0 \\ 0 & n-2 & 0 & \dots & 0 \\ 0 & 0 & n-4 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & -n \end{pmatrix}$$

O peso $\lambda = n$ é denominado **peso maximal**. Note que, se $\mathcal{V}_\mu \neq \{\vec{0}\}$, então $dim\ \mathcal{V}_\mu = 1$. Em particular, o vetor maximal é único (a menos de múltiplos).

O teorema seguinte nos garante a unicidade das representações de $\mathbf{sl}(2, \mathbb{F})$:

Teorema 2.4 *Para cada $n \geq 1$, existe uma única representação irredutível de dimensão $n+1$ de $\mathbf{sl}(2, \mathbb{F})$.*

2.2 Espaços de Raízes

Se $\mathcal{L}$ é uma álgebra de Lie que só tem elementos nilpotentes, então $\mathcal{L}$ será nilpotente. Caso isso não ocorra, podemos encontrar $x \in \mathcal{L}$ tal que a componente semi-simples de x, denotada por x_s, seja não-nula. Isso mostra que $\mathcal{L}$ possui uma subálgebra que só tem elementos semi-simples. Denominaremos tal subálgebra **toral**.

Lema 2.5 *Toda subálgebra toral de $\mathcal{L}$ é abeliana.*

Existem teoremas (veja [HUMPHREYS, cap. 4]) que garantem a existência de subálgebras torais em uma álgebra de Lie $\mathcal{L}$. Se existem subálgebras torais, deve existir uma subálgebra toral maximal. Denotaremos tal subálgebra por $\mathbb{H}$, por razões históricas e a nomearemos **subálgebra de Cartan** da álgebra de Lie $\mathcal{L}$.

Exemplo 2.6 *Se $\mathcal{L} = \mathbf{sl}(2, \mathbb{F})$, então $\mathbb{H}$ é a subálgebra das matrizes diagonais (que têm traço zero).*

Como $\mathbb{H}$ é abeliana, o conjunto $ad(\mathbb{H})$ é uma família de endomorfismos semi-simples que comutam entre si. Logo podem ser simultâneamente diagonalizáveis. Em outras palavras, $\mathcal{L}$ é soma direta de subespaços do tipo

$$\mathcal{L}_\alpha = \{x \in \mathcal{L} \mid [hx] = \alpha(h)x, \forall h \in \mathbb{H}\},$$

com α em $\mathbb{H}^*$, o dual de $\mathbb{H}$.

Note que

$$\mathcal{L}_0 = \{x \in \mathcal{L} \mid [hx] = 0, \forall h \in \mathbb{H}\} = C_{\mathcal{L}}(\mathbb{H}),$$

o centralizador de $\mathbb{H}$. Como $\mathbb{H}$ é comutativo, $\mathbb{H} \subset C_{\mathcal{L}}(\mathbb{H})$. O conjunto dos $\alpha \in \mathbb{H}^*$ tais que $\mathcal{L}_\alpha \neq 0$ é denotado por Φ. Os elementos de Φ são chamados de **raízes** de $\mathcal{L}$ relativas a $\mathbb{H}$.

Com as notações acima, temos uma decomposição em espaços de raízes (ou decomposição de Cartan) da álgebra, da seguinte forma

$$\mathcal{L} = C_{\mathcal{L}}(\mathbb{H}) \oplus \coprod_{\alpha \in \Phi} \mathcal{L}_\alpha$$

Vamos agora a um resultado que, como se não bastasse sua importância para a teoria, ainda tem um caráter "compactador", já que reduzirá a expressão $C_{\mathcal{L}}(\mathbb{H})$ a somente uma de suas "letras".

Proposição 2.7 *$C_{\mathcal{L}}(\mathbb{H}) = \mathbb{H}$.*

Exemplo 2.8 *Novamente, se $\mathcal{L} = \mathbf{sl}(2, \mathbb{F})$, temos que a decomposição acima corresponde à decomposição de $\mathcal{L}$ dada pela base canônica, a saber,*

$$\mathcal{L} = \langle x \rangle \oplus \langle h \rangle \oplus \langle y \rangle.$$

Proposição 2.9 *1. Se $\alpha, \beta \in \mathbb{H}^*$ então $[\mathcal{L}_\alpha, \mathcal{L}_\beta] \subset \mathcal{L}_{\alpha+\beta}$.*
2. Se $x \in \mathcal{L}_\alpha$ com $\alpha \neq 0$, então $\mathbf{ad}x$ é nilpotente.
3. Se $\alpha, \beta \in \mathbb{H}^*$ e $\alpha + \beta \neq 0$, então $\kappa(\mathcal{L}_\alpha, \mathcal{L}_\beta) = 0$.

Agora mais alguns resultados sobre a decomposição em espaços de raízes, usando a forma de Cartan-Killing.

Proposição 2.10 *1. Se α e β são duas raízes de $\mathbb{H}$, então dados $x \in \mathcal{L}_\alpha$ e $y \in \mathcal{L}_\beta$ temos que $\kappa(x, y) = 0$, a menos que $\alpha + \beta = 0$.*
2. Se α é uma raiz, então $-\alpha$ também é raiz.
3. Para todo $h \in \mathbb{H}$ e toda raiz α, temos que $\mathbf{ad}h \mid_{\mathcal{L}_\alpha} = \alpha(h)id$ e as transformações lineares $\mathbf{ad}h$ são simultaneamente diagonalizáveis.

Agora um resultado muito útil, que nos dará conseqüências importantes.

Proposição 2.11 *O conjunto Φ das raízes gera o dual $\mathbb{H}^*$ de $\mathbb{H}$, isto é, $h = 0$ se $\alpha(h) = 0$ para toda raiz α.*
Prova. *Pela parte (3) da proposição anterior, $\mathbf{ad}h = 0$ se $\alpha(h) = 0$ para toda raiz α. Como o núcleo de $\mathbf{ad}$ se anula, temos que $h = 0$ se $\alpha(h) = 0$ para toda raiz α.* ■

O que iremos fazer agora é obter mais relações entre as raízes e as conseqüências destas relações no conjunto $\mathbb{H}^*$.

Primeiramente vamos explicitar o isomorfismo $\mathbb{H}$ e $\mathbb{H}^*$: dado $\phi \in \mathbb{H}^*$, tome $t_\phi \in \mathbb{H}$ tal que

$$\phi(h) = \kappa(t_\phi, h), \ \forall h \in \mathbb{H}.$$

Em particular,

$$\Phi \cong \{t_\alpha \mid \alpha \in \Phi\} \subset \mathbb{H}.$$

Usaremos a notação $\kappa(\cdot, \cdot)$ livremente, nos momentos em que isso não causar nenhuma confusão, tanto para a forma de Cartan-Killing em $\mathbb{H}$ quanto para a forma induzida em $\mathbb{H}^*$.

De posse disso, vamos a mais uma proposição.

Proposição 2.12 *1. Sejam $\alpha \in \Phi$, $x \in \mathcal{L}_\alpha$ e $y \in \mathcal{L}_{-\alpha}$. Então $[xy] = \kappa(x,y)t_\alpha$, com t_α como definido acima.*
2. $\alpha(t_\alpha) = \kappa(t_\alpha, t_\alpha) \neq 0$, para todo $\alpha \in \Phi$.
3. O subespaço $[\mathcal{L}_\alpha, \mathcal{L}_{-\alpha}]$ é unidimensional e tem como gerador t_α, para toda raiz $\alpha \in \Phi$.

Com as proposições acima, já sabemos que:
1. Φ gera o dual de $\mathbb{H}$;
2. Se α é uma raiz, então $-\alpha$ também é raiz;
3. $[xy] = \kappa(x,y)t_\alpha$, onde t_α é dado pela relação $\kappa(t_\alpha, h) = \alpha(h)$, com $h \in \mathbb{H}$.
4. $\kappa(\mathcal{L}_\alpha, \mathcal{L}_\beta) = 0$, para α, β em $\mathbb{H}^*$, com $\alpha + \beta \neq 0$.

Mostraremos mais. Primeiro, mostraremos que a cada raiz podemos associar uma subálgebra de $\mathcal{L}$ isomorfa a $\mathbf{sl}(2,\mathbb{F})$. Depois disso, mostraremos que se α é raiz, as únicas raízes múltiplas de α são $\pm\alpha$.

Proposição 2.13 *Se $\alpha \in \Phi$ e x_α é qualquer elemento não-nulo de $\mathcal{L}_\alpha$, então existe y_α em $\mathcal{L}_{-\alpha}$ tal que x_α, y_α e $h_\alpha = [x_\alpha, y_\alpha]$ geram uma subálgebra simples de $\mathcal{L}$ de dimensão 3, isomorfa a $\mathbf{sl}(2,\mathbb{F})$ pelo isomorfismo*

$$x_\alpha \mapsto \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix},\ y_\alpha \mapsto \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix},\ h_\alpha \mapsto \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Prova. *Dado $0 \neq x_\alpha \in \mathcal{L}_\alpha$, encontre $y_\alpha \in \mathcal{L}_{-\alpha}$ tal que*

$$\kappa(x_\alpha, y_\alpha) = \frac{2}{\kappa(t_\alpha, t_\alpha)}.$$

Repare que tal y_α pode ser encontrado, já que $\alpha(t_\alpha) = \kappa(t_\alpha, t_\alpha) \neq 0$. Seja

$$h_\alpha = \frac{2t_\alpha}{\kappa(t_\alpha, t_\alpha)}.$$

Assim,

$$[x_\alpha, y_\alpha] = t_\alpha,$$

já que $[x,y] = \kappa(x,y)t_\alpha$, para $x \in \mathcal{L}_\alpha$ e $y \in \mathcal{L}_{-\alpha}$. Analogamente,

$$[h_\alpha, x_\alpha] = \frac{2}{\alpha(t_\alpha)}[t_\alpha, x_\alpha] = \frac{2}{\alpha(t_\alpha)}\alpha(t_\alpha)x_\alpha = 2x_\alpha$$

e

$$[h_\alpha, y_\alpha] = -2y_\alpha.$$

Desta forma, a tabela de multiplicação de $\langle x_\alpha, h_\alpha, y_\alpha\rangle$ é a mesma de $\mathbf{sl}(2,\mathbb{F})$, ou seja, estas álgebras são isomorfas. ■

Observação 2.14 *Da demonstração da Proposição anterior, decorre que*

$$h_\alpha = \frac{2t_\alpha}{\kappa(t_\alpha, t_\alpha)}.$$

Dado $x_\alpha \in \mathcal{L}_\alpha$, existe e é único o elemento $y_\alpha \in \mathcal{L}_{-\alpha}$ tal que $[x_\alpha, y_\alpha] = h_\alpha$. Assim, para cada raiz α, seja $\mathcal{S}_\alpha \cong \mathbf{sl}(2, \mathbb{F})$ a subálgebra de uma álgebra de Lie $\mathcal{L}$ associada à raiz α. Então $\mathcal{S}_\alpha = \mathcal{L}_\alpha + \mathcal{L}_{-\alpha} + H_\alpha$, onde $H_\alpha = [\mathcal{L}_\alpha, \mathcal{L}_{-\alpha}]$. Repare que $\mathcal{L}_\alpha$ é unidimensional, para cada $\alpha \in \Phi$.

O próximo teorema nos dirá informações importantes sobre as raízes. Sua demonstração pode ser encontrada em [HUMPHREYS, p. 38].

Teorema 2.15 *1. Se $\alpha \in \Phi$, os únicos múltiplos de α que são raízes são $\pm\alpha$.*
2. Se $\alpha, \beta \in \Phi$ são tais que $\alpha + \beta \in \Phi$, então $[\mathcal{L}_\alpha, \mathcal{L}_\beta] = \mathcal{L}_{\alpha+\beta}$.
3. $\mathcal{L} = \langle \mathcal{L}_\alpha \mid \alpha \in \Phi \rangle$.
4. Se $\alpha, \beta \in \Phi$, então $\beta(h_\alpha) \in \mathbb{Z}$ e $\beta - \beta(h_\alpha)\alpha \in \Phi$.

Sejam $\alpha, \beta \in \Phi$ tais que $\beta \neq \pm\alpha$, e sejam r e q os maiores inteiros tais que $\beta - r\alpha$ e $\beta + q\alpha$ são raízes. Então $\beta + i\alpha \in \Phi$ é raiz, para $-r \leq i \leq q$ e $\beta(h_\alpha) = r - q$, e a sequência

$$\beta - r\alpha, \beta - (r-1)\alpha, \ldots, \beta, \ldots, \beta + (q-1)\alpha, \beta + q\alpha$$

é denominada $\alpha-$**seqüência iniciada em** β.

Exemplo 2.16 *Como visto no Exemplo 2.6, a subálgebra de Cartan $\mathbb{H} < \mathbf{sl}(n, \mathbb{F})$ é a subálgebra das matrizes com traço zero. Para cada $1 < i, j < n$, $i \neq j$, o conjunto das matrizes e_{ij} e $e_{ii} - e_{jj}$ forma uma base de $\mathbf{sl}(n, \mathbb{F})$.*
Dado $h \in \mathbb{H}$,

$$h = \begin{pmatrix} a_1 & 0 & \ldots & 0 \\ 0 & a_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \\ 0 & 0 & 0 & a_n \end{pmatrix}, \text{ com } a_1 + \ldots + a_n = 0.$$

Portanto,

$$\boldsymbol{ad}h e_{ij} = (a_i - a_j)e_{ij},$$

ou seja, as raízes de $\mathbb{H}$ são os funcionais lineares $\alpha_{ij} = \lambda_i - \lambda_j$, com $i \neq j$, onde λ_i é dado por

$$\lambda_i : \begin{pmatrix} a_1 & 0 & \ldots & 0 \\ 0 & a_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \\ 0 & 0 & 0 & a_n \end{pmatrix} \mapsto a_i.$$

Os espaços de raízes são os subespaços unidimensionais gerados por e_{ij}, com $i \neq j$. Tomando $h \in \mathbb{H}$, temos que

$$\kappa(h, h) = 2n \sum_{i=1}^{n} a_i^2.$$

Pelas fórmulas de polarização na Proposição 1.74, se $h, h' \in \mathbb{H}$, temos que $\kappa(h, h') = 2n(a_1 b_1, \ldots, a_n b_n)$, onde $h' = (b_{ij})_{n \times n}$. Assim,

$$h_{\alpha_{ij}} = \frac{e_{ii} - e_{jj}}{2n}.$$

Portanto, os valores da forma de Cartan-Killing nas raízes são os racionais

$$\kappa(\alpha_{ij}, \alpha_{rs}) = \frac{\delta_{ir} - \delta_{is} - \delta_{jr} + \delta_{js}}{2n},$$

onde $\delta_{ij} = 1$ *se* $i = j$ *e* $\delta_{ij} = 0$ *se* $i \neq j$. *Em particular,*

$$\kappa(\alpha_{ij}, \alpha_{ij}) = \frac{1}{n}.$$

Como $[e_{ij}, e_{ji}] = e_{ii} - e_{jj} = 2nh_{\alpha_{ij}}$, *segue que*

$$\kappa(e_{ij}, e_{ji}) = 2n.$$

2.3 Sistemas Simples de Raízes

Como já foi visto, o conjunto Φ das raízes associado a uma subálgebra de Cartan $\mathbb{H} < \mathcal{L}$ gera o dual $\mathbb{H}^*$. Da mesma forma, os elementos h_α, com $\alpha \in \Phi$, duais das raízes em relação à forma de Cartan-Killing, geram $\mathbb{H}$. Iremos agora escolher bases especiais para $\mathbb{H}$ e $\mathbb{H}^*$.

Tais bases serão escolhidas de maneira que, em relação a elas, os elementos de Φ possam ser escritos com coordenadas inteiras.

Definição 2.17 *Uma raiz* $\alpha \in \Phi$ *é* ***simples*** *se* $\alpha > 0$ *e se não existem* $\beta, \gamma \in \Phi$ *tais que* $\beta > 0$, $\gamma > 0$ *e* $\alpha = \beta + \gamma$. *O conjunto das raízes simples será denotado por* Σ.

Observação 2.18 *A notação* $\alpha > 0$ *nos diz implicitamente que existe uma ordem em* Φ. *Uma dessas ordens pode ser a seguinte: se as coordenadas de* α *em relação à base de* $\mathbb{H}^*$ *forem todas positivas, então* $\alpha > 0$. *Para maiores detalhes, consulte [SanMARTIN, p.159].*

Lema 2.19 $\Sigma \neq \emptyset$.
Prova. *Se* α *é uma raiz,* $-\alpha$ *também é. Assim, pode-se escolher uma raiz* α *tal que para nenhuma outra raiz* $\beta \in \Sigma$ *ocorre* $\beta > 0$ *e* $\beta < \alpha$. *A existência de tal raiz decorre do fato de que* Σ *é finito.* ■

Na proposição a seguir, considere $\Sigma = \{\alpha_1, \ldots, \alpha_l\}$.

Proposição 2.20 *1. Se* $\alpha, \beta \in \Sigma$ *e* $\alpha \neq \beta$, *então* $\kappa(\alpha, \beta) \leq 0$.
2. Σ *é um conjunto linearmente independente.*
3. Se $\beta \in \Phi$ *e* $\beta > 0$, *então* $\beta = x_1\alpha_1 + \ldots + x_l\alpha_l$, *com* $x_1, \ldots, x_l$ *inteiros não-negativos. Em particular,* Σ *gera* $\mathbb{H}^*$.

Definição 2.21 *Um subconjunto* $\Sigma \subseteq \Phi$ *tal que* Σ *é base de* $\mathbb{H}^*$ *e toda raiz* β *pode ser escrita como* $\beta = x_1\alpha_1 + \ldots + x_n\alpha_n$ *é denominado* ***sistema simples de raízes***.

Exemplo 2.22 *Continuando o Exemplo 2.16, um sistema simples de raízes para* $\mathbf{sl}(n, \mathbb{F})$ *pode ser o conjunto* $\Sigma = \{\alpha_{12}, \ldots, \alpha_{n-1,n}\}$, *pois* $\alpha_{ij} = \alpha_{i,i+1} + \ldots + \alpha_{j-1,j}$ $(i < j)$, *já que* $\alpha_{ij} = \lambda_i - \lambda_j$.

2.4 Matrizes de Cartan

Fixemos uma base ordenada $\{\alpha_1, \ldots, \alpha_l\}$ de Σ. A matriz

$$C = \left(\frac{2\kappa(\alpha_i, \alpha_j)}{\kappa(\alpha_j, \alpha_j)} \right)_{l \times l}$$

é denominada **matriz de Cartan** de Σ e os números

$$\frac{2\kappa(\alpha_i, \alpha_j)}{\kappa(\alpha_j, \alpha_j)}$$

são chamados **números de Killing**.

Proposição 2.23 *Se θ denota o ângulo entre α e β, então*

$$cos\ \theta \in \left\{ 0, \pm 1, \frac{\pm\sqrt{(}3)}{2}, \frac{\pm\sqrt{(}2)}{2}, \frac{\pm 1}{2} \right\}.$$

Prova. *Como os números*

$$\frac{2\kappa(\alpha, \beta)}{\kappa(\alpha, \alpha)}, \ \frac{2\kappa(\alpha, \beta)}{\kappa(\beta, \beta)}$$

são inteiros, segue que

$$\frac{4\kappa(\alpha, \beta)^2}{\kappa(\alpha, \alpha)\kappa(\beta, \beta)} \in \mathbb{Z}$$

é também inteiro. Mas

$$cos\theta = \frac{\kappa(\alpha, \beta)}{\kappa(\alpha, \alpha)},$$

ou seja,

$$4cos^2\theta = \frac{4\kappa(\alpha, \beta)^2}{\kappa(\alpha, \alpha)\kappa(\beta, \beta)}$$

e isso conclui a prova. ■

Lema 2.24 *Os possíveis valores para os números de Killing são*

$$\frac{2\kappa(\alpha, \beta)}{\kappa(\alpha, \alpha)} = 0, \pm 1, \pm 2, \pm 3.$$

Prova. *Como o produto*

$$\frac{2\kappa(\alpha, \beta)}{\kappa(\beta, \beta)} \frac{2\kappa(\alpha, \beta)}{\kappa(\alpha, \alpha)}$$

é igual a

$$0, \pm 1, \ \pm 2, \ \pm 3 \ ou \ \pm 4$$

e cada fator é um inteiro, estes fatores podem assumir os valores

$$0, \ \pm 1, \ \pm 2, \ \pm 3 \ ou \ \pm 4.$$

Suponha que

$$\frac{2\kappa(\alpha, \beta)}{\kappa(\beta, \beta)} = 4.$$

Então

$$4cos^2\theta = \frac{4\kappa(\alpha,\beta)^2}{\kappa(\alpha,\alpha)\kappa(\beta,\beta)} = 4$$

e, com isso,

$$cos\theta = \pm 1,$$

ou seja, $\beta = \pm\alpha$ *e*

$$\frac{2\kappa(\alpha,\beta)}{\kappa(\beta,\beta)} = 2,$$

o que contradiz a suposição inicial. ■

Pelos dois resultados anteriores, os elementos fora da diagonal da matriz de Cartan assumem apenas os valores

$$0,\ -1,\ -2,\ \text{ou}\ -3.$$

O valor para o cosseno do ângulo entre duas raízes limita as possibilidades para o ângulo entre elas, que ficam sendo

$$0,\ \pi/2,\ 3\pi/2,\ 3\pi/4 \text{ e } 5\pi/6.$$

Além do mais, como

$$cos^2\theta < 1,$$

se

$$2\frac{\kappa(\alpha_i,\alpha_j)}{\kappa(\alpha_j,\alpha_j)} = -2 \text{ ou } 3$$

então

$$2\frac{\kappa(\alpha_j,\alpha_i)}{\kappa(\alpha_i,\alpha_i)} = -1,$$

para $i \neq j$.

Resumindo:

Proposição 2.25 *Seja* $C = (c_{ij})$ *a matriz de Cartan de um sistema simples de raízes. Então:*
1. $c_{ii} = 2$*, para todo* i*;*
2. $c_{ij} = 0, -1, -2$ *ou* -3*;*
3. $c_{ji} = -1$ *se* $c_{ij} = -2$ *ou* -3*;*
4. $c_{ij} = 0$ *se, e só se,* $c_{ji} = 0$*.*

Exemplo 2.26 *Se* $\mathcal{L} = \mathbf{sl}(3,\mathbb{F})$*, a matriz de Cartan é*

$$\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix},$$

dada pelas raízes simples α_{12} *e* α_{23} *(como no Exemplo 2.16). A única raiz positiva é* $\alpha_{12}+\alpha_{23}$*.*

Exemplo 2.27 *A matriz de Cartan de* $\mathbf{sl}(4,\mathbb{F})$ *é*

$$\begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}.$$

As raízes de altura 1 são as raízes simples, α_{12}, α_{23} *e* α_{34}*. As de altura dois são* $\alpha_{12}+\alpha_{23}$ *e* $\alpha_{23}+\alpha_{34}$*. A soma* $\alpha_{12}+\alpha_{34}$ *não é raiz, pois* $\dfrac{2\kappa(\alpha_{12},\alpha_{34})}{\kappa(\alpha_{34},\alpha_{34})} = 0$ *(não existe* α_{34}*-sequência iniciada em* α_{12}*).*

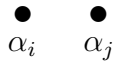
Exemplo 2.28 *Em geral, a matriz de Cartan de* $\mathbf{sl}(n, \mathbb{F})$ *é*

$$\begin{pmatrix} 2 & -1 & 0 & & & & \\ -1 & 2 & -1 & & & 0 & \\ 0 & -1 & 2 & & & & \\ & & & \ddots & & & \\ & & & & 2 & -1 & 0 \\ & 0 & & & -1 & 2 & -1 \\ & & & & 0 & -1 & 2 \end{pmatrix}.$$

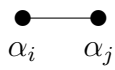
2.5 Diagramas de Dynkin

O **diagrama de Dynkin** é um grafo que contém as mesmas informações que a matriz de Cartan. Ele é definido a partir de um sistema simples de raízes fixado $\Sigma = \{\alpha_1, \ldots \alpha_l\}$. O diagrama contém um nó para cada raiz (l nós). Os vértices são ligados (ou não) por uma, duas ou três arestas de acordo com as seguintes regras:

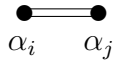
1. Se
$$\frac{2\kappa(\alpha_j, \alpha_i)}{\alpha_i, \alpha_i} = \frac{2\kappa(\alpha_i, \alpha_j)}{\alpha_j, \alpha_j} = 0$$
não existe ligação:



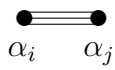
2. Se
$$\frac{2\kappa(\alpha_j, \alpha_i)}{\alpha_i, \alpha_i} = \frac{2\kappa(\alpha_i, \alpha_j)}{\alpha_j, \alpha_j} = -1$$
existe uma ligação simples entre α_i e α_j:



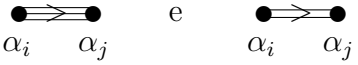
3. Se
$$\frac{2\kappa(\alpha_j, \alpha_i)}{\alpha_i, \alpha_i} \text{ ou } \frac{2\kappa(\alpha_i, \alpha_j)}{\alpha_j, \alpha_j} = -2$$
existe uma ligação dupla entre α_i e α_j:



4. Se
$$\frac{2\kappa(\alpha_j, \alpha_i)}{\alpha_i, \alpha_i} ou \frac{2\kappa(\alpha_i, \alpha_j)}{\alpha_j, \alpha_j} = -3$$
existe uma ligação tripla entre α_i e α_j:



É fácil recuperar a matriz de Cartan a partir do diagrama de Dynkin. Se houver uma ligação entre as raízes α_i e α_j, então $c_{ij} = 1$; se elas não estiverem ligadas, $c_{ij} = 0$. Caso haja duas ou três ligações, pode não ficar claro qual das entradas c_{ij} ou c_{ji} é -2 ou -3. Para distinguir isso, orienta-se a ligação na direção da raiz α_j se $c_{ij} = -2$ ou -3. Obtemos desta maneira, as ligações orientadas



Observação 2.29 *Na página 94 apresentamos um algoritmo que gera o diagrama de Dynkin de uma álgebra de Lie simples.*

Exemplo 2.30 *A matriz de Cartan*

$$\begin{pmatrix} 2 & -3 \\ -1 & 2 \end{pmatrix}$$

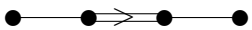
define o diagrama

Exemplo 2.31 *A matriz de Cartan de* $\mathbf{sl}(n, \mathbb{F})$ *define o diagrama*

Exemplo 2.32 *A matriz de Cartan*

$$\begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -2 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix}$$

define o diagrama



2.6 Misteriosos Diagramas

Não foi a intenção aqui detalhar toda a teoria por detrás dos diagramas de Dynkin. Como pôde se ver acima, dada uma álgebra semi-simples, podemos associar a ela um (e só um!) diagrama de Dynkin. Esta unicidade nos permite classificar as álgebras semi-simples e, por conseguinte, todas as álgebras de Lie, usando somente diagramas de Dynkin.

Pode-se mostrar que existem somente alguns poucos tipos possíveis de diagramas e a cada um corresponde uma álgebra de Lie clássica (ou alguma de um tipo excepcional). Os diagramas destas álgebras se realizam muitas vezes em espaços de dimensão maior que 3.

Para álgebras de Lie *modulares* (sobre corpos de característica positiva), a classificação ainda não está terminada. Se a característica for muito maior que 0, uma abordagem semelhante, usando diagramas de Dynkin pode ser utilizada. Porém, para o caso de características pequenas como 2 e 3, a história se complica e a associação da álgebra a um ente geométrico mais amigável não é tão natural - nem os entes tão amigáveis, já que geralmente emergem da geometria diferencial e topologia.

Para maiores detalhes sobre diagramas de Dynkin veja o Capítulo 7 de [SanMARTIN] ou o Capítulo 3 de [HUMPHREYS].

A justificativa para o título desta subseção vem agora. Diagramas de Dynkin têm aparecido já a algum tempo em áreas distintas da matemática, sempre relacionados à classificação de estruturas e objetos. Inicialmente, na caracterização das álgebras de Lie simples complexas, trabalho começado por Killing e Cartan em fins do século passado e completado por Van der Waerden em 1933; posteriormente, na caracterização dos grupos finitos de movimentos euclidianos gerados por reflexão, obtida por Coxeter em 1934 e em outros contextos diversos.

Como ou porquê os diagramas de Dynkin são tão especiais comparados com muitos outros diagramas combinatóriamente tanto ou mais especiais e interessantes que eles? Esta é uma questão intrigante. Tanto que ela foi incluída na lista dos grandes problemas da Matemática publicada em [BROWDER]. Não há ainda uma resposta satisfatória, mas as contribuições para uma solução continuam a aparecer, como por exemplo em [HPR].

Capítulo 3

Algoritmos para Álgebras Associativas

Neste capítulo, apresentaremos alguns algoritmos para álgebras associativas. Nos guiaremos pelo exposto em [GRAAF1], [GIR], [GRAAF3], [GAP2], [GR], [CCISR] e [CIW].

Dedicaremos algum tempo na discussão do algoritmo para cálculo do radical (definição na pág. 4) de uma álgebra associativa, que será fundamental no próximo capítulo.

Os exemplos serão descritos da mesma maneira que são executados no GAP para incentivar o leitor a refazê-los ou criar outros, variando alguns parâmetros.

3.1 Algoritmos Básicos

3.1.1 Matriz da Adjunta

Se $x \in \mathcal{A}$ é um elemento da álgebra associativa $\mathcal{A}$, a adjunta de x é a transformação linear

$$\begin{aligned} \mathbf{ad}x : \mathcal{A} &\to \mathcal{A} \\ a &\mapsto xa \end{aligned}$$

Seja $\mathcal{B} = \{a_1, \ldots, a_n\}$ uma base para $\mathcal{A}$. Então a matriz de adx em relação à base $\mathcal{B}$ é dada por

$$[\mathbf{ad}x]_{\mathcal{B}} = \begin{pmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} & \cdots & \gamma_{1n} \\ \gamma_{21} & \gamma_{22} & \gamma_{23} & \cdots & \gamma_{2n} \\ \gamma_{31} & \gamma_{32} & \gamma_{33} & \cdots & \gamma_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \gamma_{n1} & \gamma_{n2} & \gamma_{n3} & \cdots & \gamma_{nn} \end{pmatrix}$$

onde estamos supondo

$$\begin{cases} xa_1 = \gamma_{11}a_1 + \gamma_{21}a_2 + \gamma_{31}a_3 + \ldots + \gamma_{n1}a_n \\ xa_2 = \gamma_{12}a_1 + \gamma_{22}a_2 + \gamma_{32}a_3 + \ldots + \gamma_{n2}a_n \\ \vdots \\ xa_n = \gamma_{1n}a_1 + \gamma_{2n}a_2 + \gamma_{3n}a_3 + \ldots + \gamma_{nn}a_n \end{cases}$$

Observe que se tomarmos outra base para $\mathcal{A}$, digamos $\mathcal{B}'$, então $[\mathbf{ad}x]_{\mathcal{B}'}$ será semelhante a $[\mathbf{ad}x]_{\mathcal{B}'}$.

Assim, podemos calcular facilmente a transformação $\mathbf{ad}x$ resolvendo um sistema linear (basta encontrar os coeficientes γ_{ij}).

Vamos a alguns exemplos.

Exemplo 3.1 *Vamos calcular a matriz da adjunta de um elemento* 3×3 *na álgebra associativa*

```
gap> A:=Rationals^[3,3];;
gap> x:=[[4, 5, 6], [2, 0, 0], [1, 1, 3]];;
gap> AdjointMatrix(Basis(A), x);
[ [ 4, 0, 0, 5, 0, 0, 6, 0, 0 ],
[ 0, 4, 0, 0, 5, 0, 0, 6, 0],
[ 0, 0, 4, 0, 0, 5, 0, 0, 6 ],
[ 2, 0, 0, 0, 0, 0, 0, 0, 0  ],
[ 0, 2, 0, 0, 0, 0, 0, 0, 0 ],
[ 0, 0, 2, 0, 0, 0, 0, 0, 0  ],
[ 1, 0, 0, 1, 0, 0, 3, 0, 0 ],
[ 0, 1, 0, 0, 1, 0, 0, 3, 0  ],
[ 0, 0, 1, 0, 0, 1, 0, 0, 3 ] ]
```

Exemplo 3.2 *Vamos calcular as matrizes adjuntas dos elementos da base de* $\mathbf{sl}(2,\mathbb{F})$.

```
gap> L:=SimpleLieAlgebra(``A``, 1, Rationals);
<Lie algebra of dimension 3 over Rationals> # Criamos sl(2,Q)
gap> B:=BasisVectors(Basis(L) );;
gap> AdjointMatrix( Basis(L), B[1]);
[ [ 0, 0, -2 ],
[ 0, 0, 0 ],
[ 0, 1, 0 ] ]
gap> AdjointMatrix( Basis(L), B[2]);
[ [ 0, 0, 0 ],
[ 0, 0, 2 ],
[ -1, 0, 0 ] ]
gap> AdjointMatrix( Basis(L), B[3]);
[ [ 2, 0, 0 ],
[ 0, -2, 0 ],
[ 0, 0, 0 ] ]
gap> AdjointMatrix(Basis(L), B[1]+B[2]+B[3]);
[ [ 2, 0, -2 ],
[ 0, -2, 2 ],
[ -1, 1, 0 ] ]
```

Note pelo exemplo anterior que que a matriz da adjunta de uma soma de elementos é igual à soma das matrizes das adjuntas de cada elemento. Traduzindo este complicado jogo de palavras:

$$[\sum x_i]_{\mathcal{B}} = \sum [x_i]_{\mathcal{B}}.$$

3.1.2 Teste de nilpotência

Para calcular o radical de uma álgebra associativa, precisamos encontrar elementos nilpotentes. Mas como encontrar tais elementos?

Um elemento x é dito **nilpotente** quando sua matriz adjunta $[\mathbf{ad}x]_{\mathcal{B}}$ é nilpotente[1] (compare com a definição na pág. 4). Assim, para testarmos se x é nilpotente, basta verificar se $[\mathbf{ad}x]^{dim\mathcal{A}} = 0$. Formalmente, temos:

ALGORITMO TesteDeNilpotencia
Entrada: um elemento $x \in \mathcal{A}$, com $\mathcal{A}$ dada por constantes de estrutura
Saída: "sim" caso x seja nilpotente e "não" caso contrário

n:=$dim\ \mathcal{A}$
M:=$[\mathbf{ad}x]_{\mathcal{B}}$
P:=M^n
se P=0 **então retorne** "sim"; **senão retorne** "não"

Podemos refinar o algoritmo acima verificando se $[\mathbf{ad}x]^i = 0$ para algum i entre 1 e n, o que diminuirá os cálculos. Porém, nos basta que o algoritmo seja polinomial.

3.2 O Radical de uma Álgebra Associativa

Vamos considerar o problema de calcular uma base para o radical $Rad(\mathcal{A})$ de uma álgebra $\mathcal{A}$, onde $\mathcal{A}$ é uma álgebra associativa n-dimensional sobre um corpo $\mathbb{F}$, apresentada por uma coleção de constantes de estrutura. Este problema foi resolvido em [CIW].

Lembremos que o radical de $\mathcal{A}$ é o conjunto dos elementos fortemente nilpotentes de $\mathcal{A}$, ou seja,

$$Rad(\mathcal{A}) = \{x \in \mathcal{A} \mid x^n = 0, (xy)^n = 0, \forall y \in \mathcal{A}\}.$$

3.2.1 Um caso simples: char(F)=0

Quando $char\mathbb{F} = 0$, o problema é equivalente ao de resolver alguns sistemas lineares sobre o corpo $\mathbb{F}$, como foi caracterizado por Dickson.

Teorema 3.3 (Teorema de Dickson) *Seja $\mathcal{A}$ uma álgebra de matrizes n-dimensional sobre um corpo $\mathbb{F}$, $char\mathbb{F} = 0$. Então $Rad(\mathcal{A}) = \{x \in \mathcal{A} \mid Tr(yx) = 0,\ \forall y \in \mathcal{A}\}$.*

Logo, se $\{a_1, ..., a_n\}$ constitui uma base de $\mathcal{A}$ sobre $\mathbb{F}$, para encontrar $Rad(\mathcal{A})$ é suficiente resolver o sistema linear $Tr(a_i x) = 0$, $i = 1, ..., n$, onde $x = (x_1, \ldots,\ x_n)$.

Vamos a alguns exemplos.

Exemplo 3.4 *Seja $\mathcal{A} = \mathbb{R}^{3\times3}$ e $\mathcal{B}$ a base canônica de $\mathcal{A}$. Então temos que*

$$Rad(\mathcal{A}) = \{x \in \mathbb{R}^{3\times3} \mid Tr(e_{ij}x) = 0,\ i = 1..3,\ j = 1..3\}.$$

Pondo $x = (x_{ij})_{3\times3}$, teremos:

[1] Aqui $\mathcal{B}$ é uma base qualquer de $\mathcal{A}$ e algumas vezes será omitida da notação.

$$e_{11}x = 0 \Leftrightarrow \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$e_{12}x = 0 \Leftrightarrow \begin{pmatrix} x_{21} & x_{22} & x_{23} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$e_{13}x = 0 \Leftrightarrow \begin{pmatrix} x_{31} & x_{32} & x_{33} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\vdots$$

$$e_{32}x = 0 \Leftrightarrow \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ x_{21} & x_{22} & x_{23} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$e_{33}x = 0 \Leftrightarrow \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ x_{31} & x_{32} & x_{33} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Resolvendo os 9 sistemas acima, encontramos que $x \equiv 0$. Assim, o único elemento no radical é a matriz nula, ou seja, $\mathcal{A} = \mathbb{R}^{3\times 3}$ é semi-simples.

Exemplo 3.5 *Vamos a um exemplo "parecido" com o anterior. Tomaremos $\mathcal{T}$ como sendo a álgebra das matrizes triangulares superiores de ordem 3 sobre o corpo $\mathbb{R}$. Claro que $\mathcal{T} \leq \mathbb{R}^{3\times 3}$. É fácil obter que a dimensão de $\mathcal{T}$ é 6 e que $\mathcal{T}$ tem como base as matrizes e_{ij} com $j \geq i$, $j = 1, \ldots, 3,\ i = 1, \ldots, 3$.*
Com cálculos semelhantes aos do exemplo anterior, vemos que $Rad(\mathcal{T}) \neq 0$. Mais precisamente,

$$Rad(\mathcal{T}) = \langle e_{12}, e_{13}, e_{23} \rangle,$$

já que

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Assim, $\mathcal{T}$ não é uma álgebra semi-simples. Note ainda que, ao contrário de $\mathcal{A}$ (exemplo anterior), $\mathcal{T}$ tem pelo menos um ideal, $Rad(\mathcal{T})$.

Os dois exemplos anteriores ilustram uma situação curiosa. Note que, ao considerarmos $\mathcal{T}$ como álgebra, diminuímos as exigências para que um conjunto $\mathcal{I} \subseteq \mathcal{T} \subseteq \mathcal{A}$ seja um ideal. Em $\mathcal{A}$, $\mathcal{I}$ precisaria ser fechado para multiplicação por qualquer matriz. Já em $\mathcal{T}$, basta que $\mathcal{I}$ seja fechado para a multiplicação por matrizes triangulares superiores. **"Diminuímos"a álgebra, "aumentamos"o Radical**.

Vamos usar o GAP e refazer os dois exemplos anteriores.

Exemplo 3.6
```
gap> A:=Rationals^[3,3];
( Rationals^[ 3, 3 ] )
gap> R:=RadicalOfAlgebra(A);time;
<algebra over Rationals>
154 # tempo de cálculo, em milisegundos
gap> Dimension(R);
0
212
```

Mantendo a mesma álgebra $\mathcal{A}$:

Exemplo 3.7
```
gap> B:=BasisVectors(Basis(A));;
gap> T:=Subalgebra(A, [B[1], B[2], B[3], B[5], B[6], B[9]]);
<algebra over Rationals, with 6 generators>
gap> RadicalOfAlgebra(T);time;
<algebra of dimension 3 over Rationals>
2260 # tempo de cálculo, em milisegundos
```

Note que, no segundo exemplo, o GAP levou mais tempo para calcular o Radical, já que este é não-nulo.

3.2.2 Caso interessante: char(F)=p

Bem entendido o caso $char\mathbb{F} = 0$ (e sua simplicidade), vamos ampliar um pouco nosso cenário. Suponha $char\ \mathbb{F} = p$, onde p é um número primo qualquer. Assim, $\mathbb{F} \cong \mathbb{F}_q$, com $q = p^t$ para algum $t \in \mathbb{N}$. Assumiremos que $\mathcal{A}$ é uma subálgebra de $\mathbb{F}^{n\times n}$. Usando a representação regular (consulte a prova do 0.11), podemos atingir esta situação.

Infelizmente, ao contrário da seção anterior, não contaremos com a ajuda preciosa de Dickson, que tem como hipótese a característica do corpo ser 0. Para motivar o restante desta seção e mostrar que essa hipótese é realmente imprescindível, iremos usar o GAP para criar alguns contra-exemplos para o Dickson quando estamos em características primas.

O plano é o seguinte: vamos criar um procedimento no GAP que calcule o conjunto que, quando temos uma álgebra $\mathcal{A}$ sobre um corpo de característica zero, nos dá o radical de $\mathcal{A}$. Denotemos tal conjunto por $Dickson(\mathcal{A})$. Assim

$$Dickson(\mathcal{A}) = \{x \in \mathcal{A} \mid Tr(yx) = 0,\ \forall y \in \mathcal{A}\}.$$

Um possível algoritmo para calcular $Dickson(\mathcal{A})$ é:

ALGORITMO Dickson
Entrada: uma álgebra $\mathcal{A}$
Saída: o conjunto $Dickson(\mathcal{A})$

```
Dickson:= function( A )
n:= Dimension( A );
bas:= BasisVectors( Basis( A ) );
eqs:= List( [ 1 .. n ], x -> [] );

for i in [1..n] do
    for j in [i..n] do
        eqs[i][j]:= TraceMat( bas[i] * bas[j] );
        eqs[j][i]:= eqs[i][j];
    od;
od;
return SubalgebraNC( A, List( NullspaceMat( eqs ),x ->
                                        LinearCombination( bas, x ) ), "basis");
end;
```

Agora estamos a procura de uma álgebra $\mathcal{A}$ sobre um corpo $\mathbb{F}$ de característica prima tal que $Dickson(\mathcal{A}) \neq Rad(\mathcal{A})$. O primeiro deles vem a seguir, numa Álgebra de Grupo.

Exemplo 3.8
```
gap> G:=SymmetricGroup(4); # o grupo S4
Sym( [ 1 .. 4 ] )
gap> A:=GroupRing(GF(2), G);
<algebra-with-one over GF(2), with 2 generators>
gap> # vamos definir um isomorfismo entre A e uma algebra de matrizes
gap> f:=IsomorphismMatrixAlgebra(A);
<op. hom. AlgebraWithOne(GF(2), ... ) -> matrices of dim. 24>
gap> M:=Image(f, A); # M será nossa álgebra de matrizes isomorfa a A por f
<algebra over GF(2), with 2 generators>
gap> Dimension(M); # claro que dimensao(M)=dimensao(A)
24
gap> RadicalOfAlgebra(M); #primeiro calculamos o radical de M...
<algebra of dimension 19 over GF(2)>
gap> Dickson(M);\# .. e agora o conjunto de Dickson de M
<algebra of dimension 24 over GF(2)>
```

No exemplo anterior, pela diferença entre as dimensões de $Dickson(\mathcal{M})$ e $Rad(\mathcal{M})$, percebe-se que se trata de subálgebras diferentes.

Se alterarmos a característica do corpo de 2 para 3, ainda temos $Dickson(\mathcal{M}) \neq Rad(\mathcal{M})$. A diferença é que agora $Rad(\mathcal{M})$ tem dimensão 5, e não mais 19:

Exemplo 3.9
```
gap> A3:=GroupRing(GF(3), G);;
gap> f3:=IsomorphismMatrixAlgebra(A3);;
gap> M3:=Image(f3, A3);;
gap> Dickson(M3);
<algebra of dimension 24 over GF(3)>
```

```
gap> RadicalOfAlgebra(M3);
<algebra of dimension 4 over GF(3)>
```

Se nos exemplos anteriores tivéssemos considerado a álgebra $\mathcal{A}$ sobre $\mathbb{Q}$, é claro que teríamos obtido $Rad(\mathcal{A}) = Dickson(\mathcal{A})$, pois neste caso as duas definições coincidem. E sobre corpos de característica prima para primos maiores que 3? Vamos tentar responder a essa pergunta com um algoritmo.

Queremos um procedimento que compare os conjuntos $Rad(\mathcal{A})$ e $Dickson(\mathcal{A})$ para a álgebra $\mathcal{A} = (\mathbf{GF}(p), \mathcal{S}_4)$, variando os primos p envolvidos. Abaixo está proposto um algoritmo, que cria uma lista com os resultados das comparações, para todos os números primos até 101.

```
p:=1;; exec:=0;; result:=[];;
G:=SymmetricGroup(4);;
while p<100 do
    p:=NextPrimeInt(p);;
    A:=GroupRing(GF(p), G);;
    f:=IsomorphismMatrixAlgebra(A);;
    M:=Image(f, A);;
    exec:=exec+1;;
    if RadicalOfAlgebra(M)=Dickson(M) then
        result[exec]:=[p,"iguais"];
    else
        result[exec]:=[p,"diferentes"];
    fi;
od;
```

Após executado, este algoritmo deixa gravado na lista result os resultados das comparações. E o resultado é:

- Primos 2 e 3: os conjuntos são diferentes.
- Outros primos menores que 102: os conjuntos são iguais.

Não fomos totalmente leais nos exemplos anteriores. Quando, para mostrar a ineficácia do Teorema de Dickson para álgebras sobre corpos de característica prima, comparamos a execução do algoritmo que calcula o radical com nosso algoritmo que calcula o conjunto de Dickson, *usamos* o algoritmo existente para provar que ele é necessário. Os leitores com aguçado pensamento lógico poderiam nos acusar de "usar a tese" que queremos provar. Para desencargo de consciência, vamos justificar melhor o porquê do radical da álgebra de grupo $\mathcal{A} = (\mathbf{GF}(2), \mathcal{S}_4)$, do exemplo 3.8, não poder ter a mesma dimensão que o conjunto de Dickson. Para os outros exemplos, vale uma abordagem análoga.

Ao invés de $\mathcal{A} = (\mathbf{GF}(2), \mathcal{S}_4)$, vamos trabalhar com $\tilde{\mathcal{A}}$, a álgebra de matrizes isomorfa a $\mathcal{A}$. Seja $\mathcal{V}$ um espaço vetorial de dimensão 24 sobre $\mathbf{GF}(2)$. O isomorfismo em questão é dado pela restrição do contra-domínio à imagem no homomorfismo $\Upsilon : \mathcal{A} \to End(V)$, com $\Upsilon(\sigma)(v_i) = v_{\sigma(i)}$. Desta maneira, estamos definindo a álgebra $\tilde{\mathcal{A}}$ por $\tilde{\mathcal{A}} = Im(\Upsilon)$.

Precisamos encontrar uma matrix $a \in \tilde{\mathcal{A}}$ que não seja nilpotente, ou para a qual exista uma matriz $b \in \tilde{\mathcal{A}}$ tal que $(ab)^n \neq 0$, para todo n natural. Desta forma, $a \notin Rad(\tilde{\mathcal{A}})$, ou seja, $Rad(\tilde{\mathcal{A}}) \neq \tilde{\mathcal{A}}$, e com isso $dim\ Rad(\tilde{\mathcal{A}}) < 24$.

Considerando o elemento $(2\ 3) \in \mathcal{A}$, após tediosos cálculos, encontramos que o polinômio minimal de $\Upsilon((2\ 3))$ é $x^2 + 1$. Logo $\Upsilon((2\ 3))$, que é uma matriz 24×24, não é nilpotente. Portanto, não pode estar em $Rad(\tilde{\mathcal{A}})$. Daí, a dimensão deste radical não pode ser 24.

Em paz com nossa consciência, vamos continuar.

Exemplo 3.10 *Vamos a outro contra-exemplo para o Teorema de Dickson sobre corpos de característica positiva, agora usando a álgebra dos Quatérnios.*

```
gap> A:=QuaternionAlgebra( GF(2) );
<algebra of dimension 4 over GF(2)>
gap> f:=IsomorphismMatrixAlgebra(A);;
gap> M:=Image(f, A);;
gap> RadicalOfAlgebra(M);
<algebra of dimension 3 over GF(2)>
gap> Dickson(M);
<algebra of dimension 4 over GF(2)>
```

O leitor curioso pode fazer uma discussão semelhante àquela feita quando trabalhávamos com a Álgebra de Grupo para encontrar, ainda que heuristicamente, quais outros primos além do 2 deixam o exemplo acima na *categoria* dos contra-exemplos para o Teorema de Dickson sobre corpos finitos.

Convencidos da (*pseudo*-)inutilidade do Teorema de Dickson para esta seção, em especial para álgebras sobre corpos de característica 2 e 3, vamos rumo a um resultado que melhor nos auxilie.

O "pseudo" adjetivando "inutilidade" no parágrafo acima se justifica: se o Teorema de Dickson não nos é útil para calcular $Rad(\mathcal{A})$, ele pelo menos dá uma dica de onde começar a procurar. A seguir, provaremos que, para uma álgebra associativa modular[1] $\mathcal{A}$, o radical está contido no que já definimos como "conjunto de Dickson"

$$Dickson(\mathcal{A}) = \{x \in \mathcal{A} \mid Tr(yx) = 0, \forall y \in \mathcal{A}\}$$

que, para álgebras sobre corpos infinitos, dá o Radical.

Iremos mostrar como encontrar $Rad(\mathcal{A})$ resolvendo alguns sistemas lineares. Vamos começar definindo um conjunto

$$\mathcal{A}_1 = \{a \in \mathcal{A} \mid Tr(a) = 0,\ Tr(ab) = 0,\ \forall b \in A\},$$

que é uma subálgebra de $\mathcal{A}$.

Se $x \in Rad(\mathcal{A})$, então $x^n = 0$ para algum n. Logo, o polinômio característico $\chi(x, t)$ de x na indeterminada t é

$$\chi(x, t) = t^n.$$

[1] A palavra "modular" é das mais pertinentes, já que o corpo $\mathbb{F}_p$ é definido por um quociente $\mathbb{Z}/\langle p \rangle$.

Vamos entender melhor como obter essa situação ($\chi(x,t) = t^n$). Para cada $i \in \{0, \ldots, n\}$, considere a i-ésima álgebra exterior $\wedge^i(\mathcal{W})$ e a ação de $x \in \mathcal{A}$ em $\wedge^i(\mathcal{W})$. Vamos denotar o traço de x nesta ação por $Tr(i,x)$. Com esta notação, temos $Tr(1,x) = Tr(x)$. Assim

$$\chi(x,t) = \sum_{i=0}^{n} (-1)^i Tr(i,x) t^{n-1}.$$

Definindo

$$\tilde{\chi}(x,t) = \sum_{i=0}^{n} Tr(i,x) t^i$$

segue que

$$x^n = 0 \Longleftrightarrow Tr(i,x) = 0, i \in \{1, \ldots, n\},$$

pois, neste caso, somente $Tr(1,x)$ é não nulo. Disso segue que $Rad(\mathcal{A}) \subseteq \mathcal{A}_1$.

Lema 3.11 *$\mathcal{A}_1$ é um ideal de $\mathcal{A}$ que contém $Rad(\mathcal{A})$.*
Prova. *Como Tr é linear, $\mathcal{A}_1$ é um subespaço. Sejam $a \in \mathcal{A}_1$ e $b \in \mathcal{A}$. Precisamos mostrar que $ab \in \mathcal{A}_1$, ou seja, que $Tr((ba)c) = 0$, para todo $c \in \mathcal{A}$. Mas:*

$$Tr((ba)c) = Tr(b(ac)) = Tr((ac)b) = Tr(a(cb)) = 0,$$

pois $a \in \mathcal{A}_1$. Assim, $ab \in \mathcal{A}_1$ e $\mathcal{A}_1$ é um ideal. ■

Pelas propriedades do Radical, temos que $Rad(\mathcal{A}) = Rad(\mathcal{A}_1)$ e o problema de calcular $Rad(\mathcal{A})$ fica reduzido ao de calcular $Rad(\mathcal{A}_1)$.

Lema 3.12 *Todos os elementos de $\mathcal{A}_1$ têm a seguinte propriedade: se $Tr(a^j) = 0$, para todo $1 < j < n$, os autovalores não-nulos de a ocorrem com multiplicidade p para característica p e não ocorrem em característica 0.*

Assim, em característica 0 ou no caso de $p > n$ (característica maior que a dimensão), o ideal $\mathcal{A}_1$ coincide com $Rad(\mathcal{A})$, já que ele contém $Rad(\mathcal{A})$ e só tem elementos nilpotentes. Assim, encontramos o radical: $\mathcal{A}_1$.

Vamos então supor que não estamos em característica 0 e que $p < n$. Definamos os ideais $\mathcal{A}_2 = \{a \in \mathcal{A}_1 \mid Tr(p,a) = 0,\ Tr(p,ab) = 0,\ \forall b \in \mathcal{A}_1\}$ e, indutivamente,

$$\mathcal{A}_j = \{a \in \mathcal{A}_{j-1} \mid Tr(p^{j-1},a) = 0,\ Tr(p^{j-1},ab) = 0,\ \forall b \in \mathcal{A}_{j-1}\}.$$

Teorema 3.13 *Para todo $j \geq 1$, vale o seguinte:*
(a) Cada $\mathcal{A}_j$ é um ideal em $\mathcal{A}_{j-1}$ que contém $Rad(\mathcal{A})$.
(b) As multiplicidades dos autovalores não-nulos dos elementos de $\mathcal{A}_j$ são múltiplos de p^j.
(c) Para todos $a, b \in \mathcal{A}_j$ temos que $Tr(p^j, a+b) = Tr(p^j,a) + Tr(p^j,b)$ e $Tr(k,a) = 0$, para todo $k < p^j$.

O teorema anterior nos garante que os autovalores não-nulos dos elementos em $\mathcal{A}_j$ têm multiplicidade múltiplas de p^j. Logo, quando $p^j > n$, as multiplicidades serão nulas e teremos $\mathcal{A}_j = Rad(\mathcal{A})$.

3.2.3 Algoritmo e complexidade

Seja $\mathcal{A}$ uma álgebra associativa sobre $\mathbb{F}$, um corpo de característica $p > 0$. O método descrito na seção anterior sugere um algoritmo para calcular $Rad(\mathcal{A})$ baseado na resolução de equações semilineares.

O algoritmo que apresentaremos só peca por pedir que se encontre certas raízes p-ésimas de elementos de $\mathbb{F}$, o que não é fácil. Consulte, por exemplo, [RÓNYAI1], [IRS] e [EBERLY].

Com a notação da seção anterior, temos o seguinte algoritmo:

ALGORITMO: RadicalAssociativa()
Entrada: uma base $\mathcal{B}$ de $\mathcal{A}$
Saída: uma base para $Rad(\mathcal{A})$

início
s:=1
enquanto $s \leq n$ do
  $\mathcal{B}' := \mathcal{B} \bigcup \{I_{n\times n}\}$
  $r :=\mid \mathcal{B} \mid$
  $r' :=\mid \mathcal{B}' \mid$
  $G = (g_{ij}) := (Tr(s, bb'))_{b\in\mathcal{B},b'\in\mathcal{B}'}$
  $\Gamma :=$ uma base do espaço solução de $\sum g_{ij}x_i = 0$
  se $1 < s <\mid \mathbb{F} \mid$ **então**
    $s' := 1$
    enquanto $s' < s$ **faça**
      $\Gamma :=$ uma base de $\{(\beta_1, \ldots, \beta_r) \in \mathbb{F}^r \mid (\beta_1^p, \ldots, \beta_r^p)\} \in \mathbb{F}\Gamma\}$ $\quad (*)$
      $s' := s'p$
    fim enquanto
  fim se
  $\mathcal{B} := \{\gamma_1 b_1 + \ldots \gamma_r b_r \mid \gamma_i \in \Gamma\}$
  $s := ps$
fim enquanto
retorne $\mathcal{B}$

A saída do algoritmo, o conjunto $\mathcal{B}$, é a base procurada para $Rad(\mathcal{A})$. Vamos agora analisar a complexidade desse algoritmo.

Com exceção de $(*)$, todos os outros passos do algoritmo podem ser executados com $n^{O(1)}$ operações (são polinomiais), pois recaem em encontrar soluções para sistemas lineares. Porém, para realizar $(*)$, precisamos resolver um sistema de equações p-semilineares sobre $\mathbb{F}$, o que torna o algoritmo lento.

Para calcular os polinômios característicos das ações nas álgebras exteriores, existem algoritmos com complexidade $\mathbf{O}(m(n)log^2\ n)$, onde $m(n)$ é o número de operações necessárias para multiplicar duas matrizes $n \times n$. A melhor estimativa para $m(n)$ é da ordem de $\mathbf{O}(n^{2,376})$.

3.2.4 Implementação

O algoritmo descrito nesta seção anterior está implementado no software GAP. Nesta implementação, é levado em conta a caracterísa do corpo, no seguinte sentido: caso $char\mathbb{F} = 0$ é aplicado o Teorema de Dickson; se $char\mathbb{F} = p > 0$, então o algoritmo anterior é aplicado. Segue o código do algoritmo:

```
# ALGORITMO: RadicalAlgAssociativa()
# Entrada: Uma álgebra associativa A
# Saída: Uma base para o radical de A

local F, # the field of A
    p, # the characteristic of F
    q, # the size of F
    n, # the dimension of A
    ident, # the identity matrix
    bas, # a list of basis vectors of A
    minusOne, # -1 in F
    eqs, # equation set
    i,j, # loop variables
    G, # Gram matrix
    I, # a list of basis vectors of an ideal of A
    I_prime, # I ident
    changed, # flag denoted if I_i¡¿I_{i-1} in ideal sequence
    pexp, # a power of the prime p
    dim, # the dimension of the vector space where A acts on
    charPoly, # characteristic polynomials of elements in A
    invFrob, # inverse of the Frobenius map of F
    invFrobexp, # a power of the invFrob
    r, r_prime; # the length of I and I_prime

# Check associativity.
if not IsAssociative(A) then
    TryNextMethod();
fi;

if Dimension(A) = 0 then return A; fi;

F:=LeftActingDomain(A);
p:=Characteristic(F);
n:=Dimension(A);
bas:=BasisVectors( Basis(A) );

if p = 0 then

    # First we treat the characteristic 0 case.
    # According to Dickson's theorem we have that in this case
    # the radical of A can be computed by solving a system of linear equations.
```

```
    eqs:= List( [ 1 .. n ], x -> [] );

    for i in [1..n] do
        for j in [i..n] do
            eqs[i][j]:=TraceMat(bas[i]*bas[j]);
            eqs[j][i]:=eqs[i][j];
        od;
    od;

    return SubalgebraNC(A, List(NullspaceMat(eqs),
        x -> LinearCombination(bas, x)), "basis");

else

    # If 'p' is greater than 0, then the situation is more difficult.
    # We implement the algorithm presented in
    # "Cohen, Arjeh M, Gábor Ivanyos, and David B. Wales,
    # 'Finding the radical of an algebra of linear tranformations,'
    # Journal of Pure and Applied Algebra 117 & 118 (1997), 177-193".

    q:=Size(F);
    dim:=Length( bas[1] );
    pexp:=1;
    invFrob:=InverseGeneralMapping(FrobeniusAutomorphism(F));
    invFrobexp:=invFrob;
    minusOne:=-One(F);
    ident:=IdentityMat( dim, F );
    changed:=true;
    I:=ShallowCopy(bas);

    # Compute the sequence of ideals I_i (see the paper by Cohen, et.al.)
    while pexp <= dim do
        # These values need recomputation only when I_{i} <> I_{i-1}
        if changed then
            I_prime:=ShallowCopy(I);
                if not ident in I_prime then
                    Add(I_prime, ident);
                fi;
                r:=Length(I);
                r_prime:=Length(I_prime);
                eqs:=NullMat(r, r_prime, F);
                charPoly:=List( [1..r], x -> [] );
                for i in [1..r] do
                    for j in [1..r_prime] do
                        charPoly[i][j] :=
                        CoefficientsOfUnivariatePolynomial(CharPolynomial(F, I[i]*I_prime[j]));
                    od;
                od;
```

```
            changed := false;
        fi;

        for i in [1..r] do
            for j in [1..r_prime] do
                eqs[i][j]:=minusOne^pexp * charPoly[i][j][dim-pexp+1];
            od;
        od;

        G:=NullspaceMat(eqs);

        if Length(G)=0 then
            return TrivialSubalgebra(A);
        elif Length(G) <> r then
            # I_{ i } <> I_{ i-1 }, so compute the basis for I_{i}
            changed := true;
            if 1 < pexp and pexp < q then
                G:=List(G, x -> List(x, y -> y^invFrobexp));
            fi;
            I := List( G, x -> LinearCombination( I, x ) );
        fi;
        # prepare for next step

        invFrobexp := invFrobexp * invFrob;
        pexp := pexp*p;
    od;

    return SubalgebraNC( A, I, "basis");
fi;
```

Capítulo 4

Algoritmos para Álgebras de Lie

Neste capítulo discutiremos alguns algoritmos para álgebras de Lie. No que segue, $\mathcal{L}$ será uma álgebra de Lie de dimensão n sobre um corpo $\mathbb{F}$.

4.1 Radical de uma Álgebra de Lie

O radical de que estamos falando aqui é o radical nilpotente de uma álgebra de Lie $\mathcal{L}$, ou seja, o único ideal nilpotente maximal de $\mathcal{L}$. A notação é a mesma usada para radical de uma álgebra associativa, $Rad(\mathcal{L})$, mas as definições são bem diferentes. Veja uma observação pertinente na página 48.

O radical de uma álgebra de Lie tem inúmeras importâncias práticas. Embora para matemáticos puros a simples existência do radical muitas vezes seja suficiente, para as aplicações é necessário "enxergá-lo". Seguiremos o que foi apresentado em [CCISR]

O algoritmo que será apresentado para calcular $Rad(\mathcal{L})$ é baseado no algoritmo para o cálculo do radical de uma álgebra associativa, apresentado no capítulo anterior. Vamos começar a "matematizar".

Vamos enunciar novamente o Teorema de Jacobson, já mencionado na página 49.

Teorema 4.1 (Jacobson) *Seja $\mathcal{L}$ uma álgebra de Lie e $\mathcal{A}$ a álgebra associativa gerada pelas transformações lineares $\boldsymbol{ad}x$, com $x \in \mathcal{L}$, ou seja, $\mathcal{A} = \boldsymbol{ad}(\mathcal{L})$. Então um elemento $x \in \mathcal{L}$ pertence ao $Rad(\mathcal{L})$ se, e só se, $\boldsymbol{ad}x \in Rad(\mathcal{A})$.*

Por este teorema, para calcular o radical de uma álgebra de Lie $\mathcal{L}$ é suficiente obtermos o radical da álgebra associativa $\mathcal{A} = \mathbf{ad}(\mathcal{L})$, o que já sabemos como fazer, e então calcularmos $\mathbf{ad}^{-1}(Rad(\mathcal{A}))$.

ALGORITMO:RadicalLie()
Entrada: uma álgebra de Lie $\mathcal{L}$
Saída: uma base para $Rad(\mathcal{L})$
início
A:=algebra gerada por $\mathbf{ad}(L)$;
retorne ad^{-1}(Rad(A))
fim

Como mostramos anteriormente, é possível calcular $Rad(\mathcal{A})$ em tempo polinomial. Assim, o algoritmo RadicalLie() também tem complexidade polinomial, pois ele simplesmente calcula o radical de uma álgebra associativa e a imagem inversa de um conjunto por uma aplicação linear, o que também tem complexidade polinomial, já que recai em resolver um sistema linear. Para formalizar, vamos enunciar isso em um:

Corolário 4.2 *Seja $\mathcal{L}$ uma álgebra de Lie de dimensão finita sobre $\mathbb{F}$. Suponha que $\mathcal{L}$ é apresentada como um conjunto de constantes de estrutura. Assim, o radical de $\mathcal{L}$, $Rad(\mathcal{L})$, pode ser calculado em tempo polinomial no tamanho da entrada (dimensão da álgebra).*

4.1.1 Implementação

O algoritmo anterior está implementado no GAP e o código é o seguinte:

```
# ALGORITMO: RadicalAlgLie()
# Entrada: Uma álgebra de Lie L
# Saída: Uma base para o radical de L

n:=Dimension(L);
bv:=BasisVectors(Basis(L));
adL:=List(bv, x -> AdjointMatrix(Basis(L),x));
A:=AdjointAssociativeAlgebra(L, L);
R:=RadicalOfAlgebra(A);

if Dimension(R)=0 then
    # Neste caso, a interseção de 'ad L' e 'R' é o centro de L.
    return LieCentre(L);
fi;

B:=BasisVectors(Basis(R));
t:=Dimension(R);

# Agora vamos calcular a interseção de 'R' '<ad L>'.
eqs:= NullMat(n+t,n*n,F);
for i in [1..n] do
    for j in [1..n] do
        for k in [1..n] do
            eqs[k][j+(i-1)*n]:=adL[k][i][j];
        od;
        for k in [1..t] do
            eqs[n+k][j+(i-1)*n]:= -B[k][i][j];
        od;
    od;
od;

# Agora a imagem inversa da interseção calculada antes
sol:=NullspaceMat(eqs);
I:= List(sol, x-> LinearCombination(bv, x[1..n]));
```

return SubalgebraNC(L, I, "basis");

fi;

4.2 Cálculos de estrutura

Nesta seção iremos seguir o exposto no artigo [GRAAF2]. Dada $\mathcal{L}$ uma álgebra de Lie, temos como meta:

⋆ Calcular uma subálgebra de Cartan de $\mathcal{L}$
⋆ Calcular a decomposição de Cartan (em espaços de raízes) de $\mathcal{L}$
⋆ Calcular uma subálgebra de Cartan de $\mathcal{L}$
⋆ Calcular a decomposição de $\mathcal{L}$ em componentes simples (soma direta de ideais)
⋆ Identificar o "tipo" de $\mathcal{L}$

Vê-se que temos muito trabalho. Vamos começar logo!

4.2.1 Calculando uma subálgebra de Cartan

Estaremos trabalhando agora com uma álgebra de Lie $\mathcal{L}$ sobre um corpo de característica 0, dada por constantes de estrutura.

Para cada $x \in \mathcal{L}$, vamos definir a subálgebra de Engel de x por:

$$\mathcal{L}_0(x) = \{y \in \mathcal{L} \mid (\mathbf{ad}x)^m(y) = 0, m > 0\}$$

Lema 4.3 *Sejam $x \in \mathcal{L}$ e $K = \mathcal{L}_0(x)$. Então $K \leq \mathcal{L}$ e $\mathcal{N}_{\mathcal{L}}(K) = \mathcal{L}$*

Pelo lema acima, as subálgebras de Engel são auto-normalizantes. Assim, se encontrarmos uma subálgebra de Engel nilpotente, teremos uma subálgebra de Cartan. Começando com $\mathcal{L}_0(x)$, vamos diminuí-la (se preciso) até termos uma subálgebra nilpotente.

Para tal, precisamos encontrar $x \in \mathcal{L}$ tal que $\mathbf{ad}x$ não seja nilpotente, pois se $\mathbf{ad}x$ for nilpotente, $\mathcal{L}_0(x) = \mathcal{L}$.

Proposição 4.4 *Seja $\mathcal{L}$ uma álgebra de Lie não-nilpotente sobre um corpo de característica zero e $\{x_1, \ldots, x_n\}$ base de $\mathcal{L}$. Então o conjunto $\{x_1, \ldots, x_n\} \bigcup \{x_i + x_j \mid 1 \leq i, j \leq n\}$ possui um elemento não-nilpotente.*

Com o x não-nilpotente da proposição anterior, vamos diminuir $\mathcal{L}_0(x)$ até fazer desta uma subálgebra de Cartan.

Proposição 4.5 *Sejam $\mathcal{L}$ uma álgebra de Lie não-nilpotente sobre $\mathbb{F}$, $\Omega \subseteq \mathbb{F}$ um conjunto de tamanho $dim(\mathcal{L}) + 1$ e $x \in \mathcal{L}$ um elemento não-nilpotente. Suponha que $\mathcal{L}_0(x)$ não é nilpotente e que $y \in \mathcal{L}_0(x)$ é um elemento não-nilpotente. Então existe $c_0 \in \Omega$ tal que $\mathcal{L}_0(x + c_0(y - x)) \subsetneq \mathcal{L}_0(x)$.*

Essa proposição nos dá o próximo algoritmo:

ALGORITMO: CartanSubalgebra()

Entrada: uma álgebra de Lie $\mathcal{L}$
Saída: uma subálgebra de Cartan de $\mathcal{L}$

1. Se $\mathcal{L}$ for nilpotente, retorne $\mathcal{L}$. Senão, vá para **2**.
2. Tome $x \in \mathcal{L}$ não-nilpotente. Se $\mathcal{L}_0(x)$ for nilpotente, retorne $\mathcal{L}_0(x)$. Senão, vá para **3**.
3. Seja $y \in \mathcal{L}_0(x)$, y não-nilpotente e c um escalar tal que $\mathcal{L}_0(x+c(y-x)) \subsetneq \mathcal{L}_0(x)$. Retorne para **2** com $x + c(y - x)$ no lugar de x.

Pela proposição anterior, o algoritmo acima termina após no máximo $dim\mathcal{L} + 1$ escolhas de escalares, encontrando uma subálgebra de Cartan. Assim, temos uma complexidade polinomial na dimensão da álgebra $\mathcal{L}$.

Exemplo 4.6 *Seja $\mathcal{L}$ a Álgebra de Lie sobre $\mathbb{Q}$ de dimensão 8 dada pela tabela de multiplicação:*

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
x_1	0	$2x_2$	$-2x_3$	0	$-x_5$	0	0	x_8
x_2	$-2x_2$	0	x_1	0	x_8	0	0	0
x_3	$2x_3$	$-x_1$	0	0	0	0	0	x_5
x_4	0	0	0	0	$-x_5$	$-2x_6$	$-x_6 - 2x_7$	$-x_8$
x_5	x_5	$-x_8$	0	x_5	0	0	0	$-x_6$
x_6	0	0	0	$2x_6$	0	0	0	0
x_7	0	0	0	$x_6 + 2x_7$	0	0	0	0
x_8	$-x_8$	0	$-x_5$	x_8	x_6	0	0	0

O elemento x_1 não é nilpotente, então se $x := x_1$ temos que

$$\mathcal{L}_0(x) = [x_1, x_4, x_6, x_7].$$

*Nesta subálgebra, o elemento $y := x_4$ não é nilpotente. Portanto, de acordo com o algoritmo CartanSubalgebra() precisamos encontrar $c_0 \in \mathbb{Q}$ tal que $\mathcal{L}_0(ad(x + c_0(y - x)))$ tem dimensão **menor** que $\mathcal{L}_0(x)$. É fácil ver que se $c_0 = 1$, não temos isso. Porém, para $c_0 = 2$ obtemos*

$$\mathcal{L}_0(ad(2y - x)) = \mathcal{L}_0(ad(2x_4 - x_1)),$$

e esta é uma subálgebra nilpotente, pois é abeliana (gerada por x_1 e x_4). Assim, encontramos uma subálgebra de Cartan de $\mathcal{L}$:

$$\mathbb{H} = [x_1, x_4].$$

Exemplo 4.7 *Vamos refazer o exemplo anterior, agora no GAP. Os comandos que definem as constantes de estrutura foram salvos no arquivo "algdimensao8.txt". Assim, vamos ler o arquivo, criar a álgebra de Lie, calcular sua subálgebra de Cartan e verificar se é gerada pelos mesmos elementos encontrados no último exemplo.*

```
gap> Read(‘‘algdimensao8.txt‘‘);
gap> L:=LieAlgebraByStructureConstants(Rationals, T);
```

```
<Lie algebra of dimension 8 over Rationals>
gap> H:=CartanSubalgebra(L);
<Lie algebra of dimension 2 over Rationals>
gap> BasisVectors( Basis(H) ); # E a base da subálgebra de Cartan é..
[ v.1, v.4 ]
```

A mesma de antes!

4.2.2 Calculando a decomposição de Cartan

Agora que sabemos como calcular uma subálgebra de Cartan $\mathbb{H}$ de uma álgebra $\mathcal{L}$, queremos calcular a decomposição de Cartan de $\mathcal{L}$, ou seja, uma decomposição de $\mathcal{L}$ em subespaços da forma

$$\mathcal{L}_\alpha = \{x \in \mathcal{L} \mid [hx] = \alpha(h)x, \forall h \in \mathbb{H}\}.$$

Seja h um elemento em $\mathbb{H}$ tal que seu polinômio minimal se decompõe completamente em $\mathbb{Q}$, $m(X) = f_1(X) \cdots f_s(X)$, $f_1(X) = X$. Seja

$$\mathcal{L} = \mathcal{L}_0 + \ldots + \mathcal{L}_s$$

a decomposição primária de $\mathcal{L}$ com respeito à transformação linear $\mathbf{ad}h$, onde

$$\mathcal{L}_i = \{x \in \mathcal{L} \mid f_i(\mathbf{ad}h)(x) = 0\}.$$

Assim, podemos garantir que a mesma decomposição acima é a decomposição de $\mathcal{L}$ com respeito a $\mathbb{H}$, tal que $\mathbb{H} \subseteq \mathcal{L}_0$.

Problema: Como achar um elemento cujo polinômio minimal se fatora completamente?

Teorema 4.8 *Um elemento $h \in \mathbb{H}$ tem seu polinômio minimal completamente decomposto se, e só se, este tem grau $dim\mathcal{L} - dim\mathbb{H} + 1$.*

Observação 4.9 *Existe um algoritmo* Las Vegas[1] *para se calcular elementos cujos minimais são decomponíveis em no máximo $N(N+1)$ passos, onde $N = dim\mathcal{L} - dim\mathbb{H}$. E isso é muito bom.*

Temos então um algoritmo para a decomposição de Cartan:

ALGORITMO: DecomposicaoCartan()

Entrada: uma álgebra de Lie $\mathcal{L}$ sobre $\mathbb{F}$, uma subálgebra de Cartan $\mathbb{H}$
Saída: a decomposição de $\mathcal{L}$ em relação à $\mathbb{H}$

1. Seja $\Omega \subset \mathbb{F}$ um conjunto de tamanho N(N+1). Calcule um elemento $h \in \mathbb{H}$ cujo minimal se decompõe completamente.

2. Calcule os fatores $f_0, \ldots, f_s$ do polinômio minimal de $\mathbf{ad}h$.

[1] Um algoritmo Las Vegas é, na verdade, uma heurística.

3. Para $i \in \{1, \ldots, s\}$, calcule o subespaço $\mathcal{L}_i = \{x \in \mathcal{L} \mid f_i(\mathbf{ad}h)(x) = 0\}$. Retorne $\{\mathcal{L}_0, \ldots, \mathcal{L}_s\}$.

Exemplo 4.10 *Seja $\mathcal{L}$ a Álgebra de Lie sobre $\mathbb{Q}$ de dimensão 8 dada no exemplo anterior, com subálgebra de Cartan $\mathbb{H} = [x_1, x_4]$. Vamos calcular a decomposição primária dada pela transformação linear $\boldsymbol{ad}x_1$. Com o auxílio do GAP, vamos calcular a matriz de $\boldsymbol{ad}x_1$, os polinômios característico e minimal desta matriz, e a fatoração do polinômio minimal:*

```
gap> M:=AdjointMatrix(Basis(L), BasisVectors(Basis(L))[1]);
[ [ 0, 0, 0, 0, 0, 0, 0, 0 ],
[ 0, 2, 0, 0, 0, 0, 0, 0 ],
[ 0, 0, -2, 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, 0, 0, 0, 0 ],
[ 0, 0, 0, 0, 0, 0, 0, 0 ],
[ 0, 0, 0, 0, 0, 0, 0, 1 ] ]
gap> CharacteristicPolynomial(M);
x_1^8-5*x_1^6+4*x_1^4
gap> m:=MinimalPolynomial(Rationals,M);
x_1^5-5*x_1^3+4*x_1
gap> Factors( PolynomialRing( Rationals ), m );
[ x_1-2, x_1-1, x_1, x_1+1, x_1+2 ]
```

Agora que temos a fatoração do polinômio minimal, vamos calcular a decomposição primária de $\mathcal{L}$ dada por $\boldsymbol{ad}x_1$. Para isso, vamos calcular $Ker(\boldsymbol{ad}x_1)$, $Ker(\boldsymbol{ad}x_1 - I)$, $Ker(\boldsymbol{ad}x_1 - 2I)$, $Ker(\boldsymbol{ad}x_1 + I)$ e $Ker(\boldsymbol{ad}x_1 + 3I)$. Como $\mathbb{H} = [x_1, x_4]$ e $\mathbb{H} \subseteq Ker(\boldsymbol{ad}x_1)$, a decomposição:

$$\mathcal{L} = \mathbb{H} \oplus [x_2] \oplus [x_3] \oplus [x_5] \oplus [x_6] \oplus [x_7] \oplus [x_8]$$

4.2.3 Decomposição em componentes simples

Agora trabalharemos com álgebras de Lie semi-simples ($Rad(\mathcal{L}) = 0$) de dimensão finita. Tentaremos decompor $\mathcal{L}$ como soma direta de seus ideais simples. Seja $\mathbb{H}$ a subálgebra de Cartan de $\mathcal{L}$ e $\{h_1, \ldots, h_l\}$ uma base de $\mathbb{H}$.

O teorema seguinte nos diz que uma decomposição de Cartan é compatível com uma decomposição em ideais simples.

Teorema 4.11 *Seja $\mathcal{L}$ uma álgebra de Lie semi-simples sobre um corpo de característica zero, $\mathbb{H}$ uma subálgebra de Cartan de $\mathcal{L}$ e $\mathcal{L} = \mathbb{H} \oplus \coprod_{i=1}^{s} \mathcal{L}_i$ a decomposição de Cartan de $\mathcal{L}$. Suponha que $\mathcal{L} = \mathcal{I}_1 \oplus \mathcal{I}_2$, $\mathcal{I}_j$ ideais simples. Então ou $\mathcal{L}_i \subset \mathcal{I}_1$ ou $\mathcal{L}_i \subset \mathcal{I}_2$.*

Pelo teorema acima, temos o seguinte algoritmo:

ALGORITMO: DecomposicaoEmIdeais()
Entrada: uma álgebra de Lie semi-simples $\mathcal{L}$
Saída: uma lista com bases para as somas diretas.

1. Calcule a decomposição $\mathcal{L} = \mathbb{H} \oplus \mathcal{L}_1 \oplus \ldots \oplus \mathcal{L}_s$.
2. Para $1 \leq i \leq s$, determine a base do ideal de $\mathcal{L}$ gerado por $\mathcal{L}_i$ (por exemplo, adicionando à $[\mathcal{L}_i]$ os resultados dos produtos $[x_{ik}, v_k]$, onde $\{v_i\}$ é base de $\mathcal{L}$ e $\{x_{ik}\}$ é base de $\mathcal{L}_i$.
3. Atualize a lista e recomece.

Este é um algoritmo em tempo polinomial, exceto possivelmente o passo **1**, onde é necessário um algoritmo para fatorar polinômios, nem sempre disponível.

Outra observação importante sobre o algoritmo anterior é que ele funciona perfeitamente para álgebras de Lie definidas sobre corpos de característica prima. Para isso, é suficiente que a forma de Killing seja não-degenerada e, nesse caso, a álgebra se comporta como uma álgebra de Lie de característica zero. No entanto, como não podia deixar de ser, esse primo precisa ser maior que 3, já que os números -2 e -3 aparecem na matriz da forma de Killing (pelo menos nos casos interessantes).

Exemplo 4.12 *Seja $\mathcal{L}$ a álgebra de Lie sobre $\mathbb{Q}$ de dimensão 6 com base $\{h_1, x_1, y_1, h_2, x_2, y_2\}$ e tábua de multiplicação dada por:*

$$\begin{array}{llll} [h_1,x_1] = & 2x_1 & [h_2,x_1] = & 2x_1 \\ [h_1,y_1] = & -2y_1 & [h_2,y_1] = & -2y_1 \\ [h_1,x_2] = & 2x_2 & [h_2,x_2] = & -2x_2 \\ [h_1,y_2] = & -2y_2 & [h_2,y_2] = & 2y_2 \\ [x_1,y_1] = & \frac{1}{2}h_1 + \frac{1}{2}h_2 & [x_2,y_2] = & \frac{1}{2}h_1 - \frac{1}{2}h_2 \end{array}$$

O determinante da matriz da forma de Killing (a matriz $(tr(\boldsymbol{ad}v_i \cdot \boldsymbol{ad}v_j))_{6\times 6}$, v_i e v_j vetores da base de $\mathcal{L}$) é 2^{16}. Assim, a forma de Killing é não degenerada se a característica do corpo base não for 2.

É fácil percebermos que $\mathbb{H} = [h_1, h_2]$. Vamos considerar $\mathcal{L}$ sobre $\mathbb{Q}$.

Como o polinômio minimal de $ad(h_1 + 2h_2)$ é

$$X(X+6)(X-6)(X+2)(X-2),$$

$h_1 + 2h_2$ nos ajudará a determinar a decomposição de Cartan de $\mathcal{L}$. E esta é:

$$\mathcal{L} = [h_1, h_2] \oplus [x_1] \oplus [x_2] \oplus [y_1] \oplus [y_2].$$

Note que o ideal $\mathcal{I}_1$ que contém $[x_1]$ é gerado por $\{x_1, y_1, (h_1+h_2)/2\}$ e o ideal $\mathcal{I}_2$ que contém $[x_2]$ é gerado por $\{x_2, y_2, (h_1 - h_2)/2)\}$. Assim, temos a decomposição de $\mathcal{L}$ em ideais simples,

$$\mathcal{L} = \mathcal{I}_1 + \mathcal{I}_2.$$

O mesmo exemplo funciona se o corpo base for, por exemplo, $\mathbb{F}_5$ ou qualquer outro corpo onde a tábua de multiplicação seja "a mesma"e a forma de Killing não seja degenerada.

Exemplo 4.13 *Vejamos o exemplo acima no GAP, onde poderemos verificar a validade de sua observação final, sobre a invariância da decomposição. Os comandos que definem as constantes de estrutura serão salvos no arquivo "algdimensao6.txt".*

```
gap> Read(‘‘algdimensao6.txt‘‘);
gap> L:=LieAlgebraByStructureConstants(Rationals, T);;
gap> H:=CartanSubalgebra(L);;
gap> BasisVectors( Basis(H) );
[ v.1, v.4 ]
gap> # vamos calcular os ideais gerados por x_1 e x_2;
gap> I1:=IdealByGenerators( L, [BasisVectors(Basis(L))[2]]);;
gap> BasisVectors(Basis(I1));
```

```
[ v.2, v.1+v.4, v.3 ]
gap> I2:=IdealByGenerators( L, [BasisVectors(Basis(L))[5]]);;
gap> BasisVectors(Basis(I2));
[ v.5, v.1+(-1)*v.4, v.6 ]
gap> DirectSumDecomposition(L);
[ <two-sided ideal in <Lie algebra of dimension 6 over Rationals>,
 (dimension 3)>,
  <two-sided ideal in <Lie algebra of dimension 6 over Rationals>,
 (dimension 3)> ]
gap> BasisVectors( Basis(DirectSumDecomposition(L)[1]));
[ v.2, v.3, v.1+v.4 ]
gap> BasisVectors( Basis(DirectSumDecomposition(L)[2]));
[ v.5, v.6, v.1+(-1)*v.4 ]
gap> DirectSumDecomposition(L)[1]=I1;
true
gap> DirectSumDecomposition(L)[2]=I2;
true
```

Como pode ser visto acima, encontramos os mesmos ideais $\mathcal{I}_1$ e $\mathcal{I}_2$ do exemplo anterior. Agora vamos variar o corpo, de olho na decomposição.

```
gap> # característica 3;
gap> L:=LieAlgebraByStructureConstants(GF(3), T);
<Lie algebra of dimension 6 over GF(3)>
gap> BasisVectors(Basis(DirectSumDecomposition(L)[1]));
[ v.5, v.6, v.1+(Z(3))*v.4 ]
gap> BasisVectors(Basis(DirectSumDecomposition(L)[2]));
[ v.3, v.2, v.1+v.4 ]

gap> # característica 7;
gap> L:=LieAlgebraByStructureConstants(GF(7), T);
<Lie algebra of dimension 6 over GF(7)>
gap> BasisVectors(Basis(DirectSumDecomposition(L)[1]));
[ v.5, v.6, v.1+(Z(7)^3)*v.4 ]
gap> BasisVectors(Basis(DirectSumDecomposition(L)[2]));
[ v.2, v.3, v.1+v.4 ]
```

Como havíamos observado, a decomposição não se altera!

4.2.4 Identificando o tipo de uma álgebra semi-simples

As álgebras de Lie semi-simples sobre corpos de característica zero foram classificadas. Uma álgebra de Lie semi-simples é isomorfa a somas de álgebras clássicas com álgebras excepcionais.

Dada uma álgebra de Lie semi-simples sobre $\mathcal{Q}$, queremos obter qual o seu tipo (i.e., obter o tipo das álgebras na soma de álgebras simples à qual ela é isomorfa). Para isso, precisamos determinar um isomorfismo entre a base dada e a base "genérica"de alguma álgebra de Lie na dimensão dada. E isso pode não ser tão simples.

Problema principal: Não raramente, precisaremos encontrar polinômios minimais. Assim, precisaremos fatorar polinômios e isso envolve encontrar extensões de $\mathbb{Q}$ de grau até $n!$ (onde n é a dimensão da álgebra).

Para melhorar a situação: Reduziremos a álgebra $\mathcal{L}$ módulo p, pois as extensões algébricas de $\mathbb{F}_p$ são mais simples de serem calculadas (é um problema combinatório).

Para um primo p que não divida o determinante da matriz da forma de Killing, a classificação de álgebras de Lie modulares é bem parecida com o caso racional e, pela observação acima, fica simplificada. Mas para isso precisamos provar que as duas álgebras, $\mathcal{L}$ e $\mathcal{L}_p = \mathcal{L}\ mod\ p$ possuem as mesmas matrizes de Cartan.

Vamos assumir agora que $\mathcal{L}$ é uma álgebra de Lie semi-simples com constantes de estrutura em $\mathbb{Z}$ (se elas estiverem em $\mathbb{Q}$, basta trocar a base, $\tilde{v_i} = \lambda v_i$, onde λ é produto dos denominadores das constantes de estrutura) e subálgebra de Cartan $\mathbb{H}$, com base $\{h_1, ..., h_l\}$. A forma de Killing em $\mathcal{L}$ será denotada por κ e $f_i = x^{m_i} g_i$ ($g_i(0) \neq 0$) denotará o polinômio característico de $\mathbf{ad}h_i$.

Fixemos um primo $p \geq 7$ que não divida o determinante da matriz da forma de Killing e que não divida $g_i(0)$. Vamos definir $\mathcal{L}_p = \mathcal{L} \otimes \mathbb{F}_{p^m}$, o que certamente é um abuso de notação.

Se estivermos na situação acima a forma de Killing será não-degenerada na álgebra de Lie $\mathcal{L}_p$ e a classificação segue o procedimento usado para as álgebras sobre corpos infinitos. Assim, usando os algoritmos anteriores, poderemos decompor $\mathcal{L}_p$ em ideais simples. Considerando cada um desses ideais como uma álgebra de Lie, calculamos suas subálgebras de Cartan e a matriz de Cartan de cada um deles. Pela matriz de Cartan, determinamos o tipo de cada um dos componentes da soma direta, determinando assim, o tipo da álgebra $\mathcal{L}_p$.

Isso nos dá o seguinte algoritmo:

ALGORITMO: IdentificaOTipo()

Entrada: uma álgebra de Lie semi-simples $\mathcal{L}$ sobre $\mathbb{Q}$
Saída: o tipo de $\mathcal{L}$

1. Calcule uma subálgebra de Cartan $\mathbb{H}$ de $\mathcal{L}$.
2. Estenda uma base de $\mathbb{H}$ até uma base de $\mathcal{L}$ e faça com que as constantes de estrutura relativas a essa base sejam inteiras.
3. Determine um primo-amigo p.
4. Considere a álgebra de Lie $\mathcal{L}_p = \mathcal{L} \oplus \mathbb{F}_{p^m}$, com m grande o bastante para que os polinômios característicos de $\mathbf{ad}h_i$ se quebrem em fatores lineares.
5. Decomponha $\mathcal{L}_p$ em soma de direta de ideais simples, $\mathcal{L}_p = \mathcal{I}_1 \oplus \ldots \oplus \mathcal{I}_s$.
6. Para $1 \leq i \leq s$, determine uma subálgebra de Cartan de $\mathcal{I}_i$ e a matriz de Cartan de $\mathcal{I}_i$. Com isso, obteremos o tipo de $\mathcal{I}_i$ e, conseqüentemente, o tipo de $\mathcal{L}$.

Complexidade

O inteiro m acima será o mínimo múltiplo comum dos graus dos fatores irredutíveis dos polinômios minimais de $\mathbf{ad}h_i$, $\{h_i\}_{i=1}^s$ base de $\mathbb{H}$. Não é possível garantir que este número seja polinomial no tamanho da álgebra, mas é bem inferior à $(dim\mathcal{L})!$. As outras etapas do algoritmo têm complexidade polinomial, já que recaem em resolver sistemas lineares.

Capítulo 5

Algoritmos produzidos

Apesar de não ter sido um dos objetivos iniciais do projeto, após compreendermos o código de alguns algoritmos para álgebras de Lie, nos aventuramos na tarefa de desenvolver algum algoritmo.

O resultado da aventura está neste capítulo: conseguimos construir dois algoritmos para álgebras de Lie, um que trabalha com diagramas de Dynkin e outro com representações de $\mathbf{sl}(2, \mathbb{F})$.

5.1 Diagramas de Dynkin

Na Seção 2.5 tratamos teoricamente dos diagramas de Dynkin, apresentando alguns exemplos e, ainda que superficialmente, explicitando sua ligação com a classificação das álgebras de Lie semi-simples, que pode ser feita via classificação de diagramas.

Um diagrama de Dynkin é um grafo construído a partir da matriz de Cartan de uma álgebra (veja pág. 65). A ordem da matriz (o posto da álgebra de Lie) nos dá o número de vértices do grafo, e o "tipo" de raiz nos dá a quantidade e a orientação das arestas que ligam estes vértices.

Nosso algoritmo consegue construir diagramas de Dynkin para álgebras simples. Para tal, usamos as funções do GAP para obter a matriz de Cartan da álgebra de Lie e o tipo da álgebra. De posse disso, usamos o procedimento descrito na página 67 para gerar o diagrama.

O problema é que, para as álgebras de Lie excepcionais, o GAP considera uma ordenação diferente do sistema de raízes. Assim, a matriz de Cartan não assume a forma padrão. Para contornar este problema, optamos por gerar em separado os diagramas destas álgebras. A seguir, a implementação do algoritmo no GAP.

```
# ALGORITMO: dynkin()
# Entrada: Uma álgebra de Lie simples L
# Saída: O diagrama de Dynkin de L

dynkin:= function(L)

tipo:=SemiSimpleType(L);

R:=RootSystem(L);

C:=CartanMatrix(R);

if tipo="F4"then;
    C:=[ [2, -1, 0, 0], [-1, 2, -2, 0], [0, -1, 2, -1], [0, 0, -1, 2]];
fi;

n:=Size(C); posto:=n;

D:=[];
for i in [1..n] do;
    D[i]:=[];
od;

for i in [1..n] do;
    for j in [1..n] do;
        D[i][j]:=Maximum(AbsInt(C[i][j]), AbsInt(C[j][i]));
    od;
od;

dyn:=[]; dyn[1]:="o"; dyn[2]:=[]; dyn[3]:=[]; dyn[4]:=[];
dyn[2]:=[ "–", "->-", "-<-"]; dyn[3]:=[ "==", "=>=", "=<="];
dyn[4]:=[ "-==-", "-=>=-", "-=<=-"];

diagrama:=[];
for i in [1..n] do;
    diagrama[2*i]:=dyn[1];
od;

diagrama[1]:=SemiSimpleType(L);

# Até agora definimos os símbolos que serão usados no
# diagrama e fizemos as atribuicoes basicas.
# Agora vamos passar a preencher uma lista com o diagrama
# propriamente dito. Para isso, vamos comparar os valores
# da matriz de Cartan, para determinar o numero de ligacoes
# e qual é a raiz maior.
```

```
for i in [1..(n-1)] do;
    if D[i][i+1]=1 then;
        if AbsInt(C[i][i+1])>AbsInt(C[i+1][i]) then;
            diagrama[2*i+1]:=dyn[2][2];
        elif AbsInt(C[i][i+1])<AbsInt(C[i+1][i]) then;
            diagrama[2*i+1]:=dyn[2][3];
        else diagrama[2*i+1]:=dyn[2][1];
        fi;
    elif D[i][i+1]=2 then;
        if AbsInt(C[i][i+1])>AbsInt(C[i+1][i]) then;
            diagrama[2*i+1]:=dyn[3][2];
        elif AbsInt(C[i][i+1])<AbsInt(C[i+1][i]) then;
            diagrama[2*i+1]:=dyn[3][3];
        else diagrama[2*i+1]:=dyn[3][1];
        fi;
    elif D[i][i+1]=3 then;
        if AbsInt(C[i][i+1])>AbsInt(C[i+1][i]) then;
            diagrama[2*i+1]:=dyn[4][2];
        elif AbsInt(C[i][i+1])<AbsInt(C[i+1][i]) then;
            diagrama[2*i+1]:=dyn[4][3];
        else diagrama[2*i+1]:=dyn[4][1];
        fi;
    fi;
od;

if tipo="E6" or tipo="E7" or tipo="E8" then;
    if posto=6 then;
        Print("              o \n");
        Print("E6            | \n");
        Print("o – o – o – o – o \n");
    elif posto=7 then;
        Print("              o \n");
        Print("E7            | \n");
        Print("o – o – o – o – o – o \n");
    else
        Print("              o \n");
        Print("E8            | \n");
        Print("o – o – o – o – o – o – o \n");
    fi;
else return JoinStringsWithSeparator(diagrama, "  ");
fi;

end;;
```

5.1.1 Exemplos

Vamos usar nosso algoritmo para obter alguns diagramas de Dynkin.

```
gap> L:=SimpleLieAlgebra(‘‘A‘‘, 3, Rationals);;
gap> dynkin(L);
A3 o -- o -- o

gap> L:=SimpleLieAlgebra(‘‘B‘‘, 7, Rationals);;
gap> dynkin(L);
B7 o -- o -- o -- o -- o -- o =>= o

gap> L:=SimpleLieAlgebra(‘‘E‘‘, 6, Rationals);;
gap> dynkin(L);
              o
E6            |
   o -- o --  o -- o -- o

gap> L:=SimpleLieAlgebra(‘‘G‘‘, 2, Rationals);;
gap> dynkin(L);
G2 o -=<=- o
```

5.2 Representações de $\mathbf{sl}(2,\mathbb{F})$

Se ρ é uma representação irredutível de $\mathbf{sl}(2,\mathbb{F})$ em $\mathcal{V}$, o Teorema 2.3 nos garante a existência de uma base $\{v_0,\ldots,v_n\}$ de $\mathcal{V}$ que satisfaz as relações

$$\begin{cases} \rho(x)(v_i) = i(n-i+1)v_{i-1} \\ \rho(h)(v_i) = (n-2i)v_i \\ \rho(y)(v_i) = v_{i+1} \end{cases}$$

Entretanto, como podemos obter uma representação e uma base que satisfaça o teorema? O algoritmo seguinte nos dá um procedimento.

ALGORITMO: BaseRepresentacao()
Entrada: Uma base $\mathcal{B}$ de $\mathcal{V}$ com $n+1$ vetores
Saída: as transformações que representam x, y e h

B:=base dada
C:=base canônica para V

matrizX:=
$$\begin{bmatrix} 0 & n & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 2(n-1) & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 3(n-2) & & \cdots & 0 \\ 0 & 0 & 0 & 0 & 4(n-3) & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & n \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

matrizY:=
$$\begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 2 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 3 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & m-1 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & m & 0 \end{bmatrix}$$

matrizH:=
$$\begin{bmatrix} n & 0 & 0 & \cdots & 0 \\ 0 & n-2 & 0 & \cdots & 0 \\ 0 & 0 & n-4 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & -n \end{bmatrix}$$

Pronto! As matrizes acima são as matrizes das representações
de x, y e h, respectivamente, em relação à base B.
Vamos obter uma expressão explicita para as transformacoes
M:=matriz mudança de base de B para C
repX:=M*matrizX*(M^(-1))
repY:=M*matrizY*(M^(-1))
repH:=M*matrizH*(M^(-1))
Agora sim, a ação de:
X em um vetor v é dada por repX*v
Y em um vetor v é dada por repY*v

H em um vetor v é dada por repH*v

Conclusão

O presente trabalho foi baseado nos artigos de W. A. de Graaf, G. Ivanyos, L. Ronyai e demais membros do *CIRCA - Centre for Interdisciplinary Research in Computational Algebra*, grupo de pesquisa em Álgebra Computacional na Universidade de St. Andrews, Escócia, que desenvolveram algoritmos para se calcular quase tudo em termos de álgebras associativas e de Lie, incluindo o software GAP, e continuam a trabalhar no ambicioso projeto *ACELA - Architecture of a Computer Environment for Lie Algebras*, que pretende ser, numa modesta descrição, um livro interativo sobre álgebras de Lie.

Em "Preliminares Algébricas", os pré-requisitos básicos para estudar Álgebras de Lie e Complexidade de Algoritmos foram apresentados, como uma introdução à teoria de Álgebras Associativas e um pequeno resumo sobre Módulos e Representações, além de uma seção sobre Álgebra Tensorial, este último tópico sendo necessário somente uma vez, no Capítulo 3, quando utilizamos o Produto Exterior.

Já em "Preliminares Computacionais", cobrimos os pré-requisitos computacionais que precisaríamos no restante do texto. Apresentamos também um tutorial do GAP, descrevendo os principais comandos e funções para álgebras, abordando de espaços vetoriais até álgebras de Lie, sempre com mais exemplos de uso do que sintaxes formais. No mesmo capítulo foi feita uma discussão sobre Complexidade Computacional, classes de complexidade e alguns exemplos de como encontrar a complexidade de um algoritmo, além de mencionarmos o famoso problema P=NP, ainda aberto na Teoria de Complexidade.

Nos Capítulos 1 e 2, falamos de álgebras de Lie, seguindo a abordagem *standard* para estruturas álgebricas: ideais, homomorfismos, solubilidade, nilpotência, simplicidade. Apresentamos também os critérios de Cartan, para determinação da simplicidade e semi-simplicidade de uma álgebra de Lie. Iniciamos o Capítulo 2 descrevendo as representações de $\mathbf{sl}(2, \mathbb{F})$ e, a seguir, estudamos os espaços de raízes de uma álgebra de Lie, culminando na classificação da álgebra por intermédio dos diagramas de Dynkin.

Concluída a parte teórica inicial, passamos para o estudo dos algoritmos para álgebras. No Capítulo 3 foram descritos alguns algoritmos para Álgebras Associativas, em especial um algoritmo para calcular o radical em uma álgebra associativa, tanto sobre corpos de característica zero como sobre corpos finitos. No último caso, foram dados alguns contra-exemplos não encontrados na literatura que justificam o abandono do Teorema de Dickson, quando consideramos álgebras sobre corpos finitos. Também foi apresentada uma análise da complexidade deste algoritmo, com alguns exemplos de execução e sua implementação no software GAP.

No Capítulo 4, apresentamos algoritmos para Álgebras de Lie. O primeiro deles calcula o radical de uma álgebra de Lie, baseando-se no cálculo do radical para uma álgebra associativa. A seguir, apresentamos algoritmos para o que denominamos de "Cálculos de Estrutura":

subálgebra e decomposição de Cartan, decomposição em componentes simples e identificação do tipo de uma álgebra de Lie. Estes algoritmos também tiveram sua complexidade analisada e foram apresentados alguns exemplos.

Finalmente, no Capítulo 5 foram apresentados dois algoritmos que produzimos durante este projeto: um para o cálculo do diagrama de Dynkin de uma álgebra de Lie e outro para o cálculo de uma representação irredutível de $\mathbf{sl}(2, \mathbb{F})$ em uma dada dimensão.

Durante o projeto, foram apresentados dois seminários sobre o assunto no Departamento de Matemática da UFV, o primeiro deles em 18 de novembro de 2004, com o título *Radicalizando nas Álgebras Associativas*, onde foi apresentado o conteúdo do Capítulo 3, e o segundo em 6 de abril de 2005, com título *Uma abordagem algorítmica às Álgebras de Lie*, cobrindo o conteúdo do Capítulo 4 deste trabalho. Os resultados deste projeto também foram apresentados no 25º Colóquio Brasileiro de Matemática, evento ocorrido no Instituto Nacional de Matemática Pura e Aplicada - IMPA, no período 23-29 de julho de 2005, na forma de pôster.

Com este trabalho, chegamos à conclusão que os algoritmos existentes para resolver problemas em álgebras associativas e de Lie cobrem grande parte da teoria básica e são geralmente algoritmos rápidos, pelo menos quando se trabalha com álgebras de dimensão pequena. No entanto, muito ainda há para se fazer na área de algoritmos para lidar com estruturas mais complexas que são temas de pesquisas atuais. Em especial para Álgebras de Lie, novos algoritmos serão de grande valia, dada a grande variedade de aplicações desta classe de álgebras em problemas físicos.

Índice Remissivo

Referências Bibliográficas

[ARAÚJO] ARAÚJO, F. J., *Complexidade Computacional: Notas de aula da disciplina Projeto e Análise de Algoritmos*, UFPI, 2004.

[BOERES] BOERES, M. C. S., *Algoritmos e Estruturas de Dados: Programa do Curso*, UFRGS, 2004.

[BROWDER] BROWDER, F. E., *Mathematical Developments Arising from Hilbert Problems*, Proc. Symposia Pure Math. 28, 1976.

[CIC] CIC-UnB, *http://www.cic.unb.br/tutores/turing/introduc.html*.

[CLAY1] Clay Math Institute, www.claymath.org, University of Toronto, Canadá.

[CL] COELHO, F. U., LOURENÇO, M. L., *Um Curso de Álgebra Linear*, EdUSP, 2001.

[CIW] COHEN, A. M., IVANYOS, G., WALES, D. B., *Finding the radical of an algebra of linear transformations*, Journal of Pure and Applied Algebra 117&118, 1997.

[CG] COHEN, A., GRAAF, W. A., *Lie Algebraic Computation*, Computer Physics Communications, 97(1996), 53-62.

[CCISR] COHEN, P. A. M., CUYPERS, H., IVANYOS, G., STERK, H., RÓNYAI, L., *Some Tapas of Computer Algebra, Chapter 5: Computations in Associative and Lie Algebras*, Algorithms and Computation in Mathematics 4, Springer Verlag, 1998.

[COOK] COOK, S., *The P versus NP Problem*, University of Toronto, 2000.

[CLR] CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., *Introduction to Algorithms*, The MIT Press, 1990.

[COUTINHO] COUTINHO, S. C., *Primalidade em tempo polinomial: uma introdução ao algoritmo AKS*, SBM, 2004.

[CR] CURTIS, C. W., REINER I., *Representation Theory of Finite Groups and Associative Algebras*, Interscience Publisher, John Wiley & Sons, Nova Iorque, 1962.

[DROZDEK] DROZDEK, A., *Estruturas de Dados e Algoritmos em C++*, Ed. Thomson, São Paulo, 2002.

[DRUMMOND] DRUMMOND, A. C., *Complexidade de Algoritmos: Resumo das aulas sobre problemas NP-Completo*, UFSCar, 2002.

[EBERLY] EBERLY, W. M., *Computations for algebras and group representations*, Ph.D. Thesis, Dept. of Computer Science, University of Toronto, 1989.

[EPPSTEIN] EPPSTEIN, D., *Computational Complexity of Games and Puzzles*, http://www.ics.uci.edu/ eppstein/cgt/hard.html.

[FR] FRIEDL, K., RÓNYAI, L., *Polynomial time solutions to some problems in computational algebra*, Proc. of the 17th ACMSTOC, Providence (1985), 153-162.

[FRITZSCHE] FRITZSCHE, B., *Sophus Lie: a sketch of his life and work*, Journal of Lie Theory 9, 1999, p. 1-38.

[FH] FULTON, W., HARRIS, J., *Representation Theory: A first course*, Graduate Texts in Mathematics (Reading in Maths) 129, Springer-Verlag, 1991

[GR] GALLIAN, J. A., RAINBOLT, J. G., *Abstract Algebra with GAP*, 2003.

[GAP1] GAP Group, The*GAP - Groups, Algorithms, and Programming, Version 4.4.2*; 2004 (http://www.gap-system.org)

[GAP2] GAP Group, The*GAP Reference Manual, release 4.4.2*; 2004 (http://www.gap-system.org)

[GRAAF1] GRAAF, W. A., *An Algorithm for the decomposition of semisimple Lie Algebras* (preprint)

[GRAAF2] GRAAF, W. A., *Calculating the structure of a semisimple Lie algebra*, Journal of Pure and Applied Algebra 117 & 118, 1997.

[GRAAF3] GRAAF, W. A., *Lie Algebras: Theory and Algorithms*, North-Holland Mathematical Library, vol. 56, Elsevier 2000.

[GIR] GRAAF, W. A., IVANYOS, G., RÓNYAI, L., *Computing Cartan subalgebras of Lie Algebras*, AAECC (to appear)

[HPR] HAPPEL, D., PREISER, U., RINGEL, C. M., *Binary Polyhedral Groups and Euclidean Diagrams*, Manuscripta Math. 31 (1980) p. 317-329.

[HK] HOFFMAN, K., KUNZE, R., *Álgebra Linear*, Livros Técnicos e Científicos, 1970.

[HUMPHREYS] HUMPHREYS, J. E., *Introduction to Lie Algebras and Representation Theory*, Graduate Texts in Mathematics, Springer-Verlag, New York, 1972.

[IRS] IVANYOS, G., RÓNYAI, L., SZANTO, A., *Decomposition of algebras over (X1,. .X,,,)*, Applicable Algebra Eng. Comm. Comput. 5 (1994) 71-90.

[JACOBSON] JACOBSON, N., *Lie Algebras*, Interscience Tracts in Pure and Applied Math. 10, New York, Interscience Publ., 1962.

[JM] JONES, A., MERKLEN, H. , *Representações de Álgebras - Métodos Diagramáticos*, Pub. do IME/USP - São Paulo.

[KAYE] KAYE, R., *How Complicated is Minesweeper?*, Birmingham University, http://www.mat.bham.ac.uk/R.W.Kaye/.

[KNUTH] KNUTH, D., *The Art of Computer Programming, Volume 1-3*, Addison-Wesley.

[MALAGUTTI] MALAGUTTI, P. L. A., *Hipertexto Pitágoras: P versus NP*, UFSCar, 2002.

[PIERCE] PIERCE, R. S., *Associative Algebras*, Graduate Texts in Mathematics 88, Springer-Verlag, 1982.

[POINCARÉ] POINCARÉ, H., *O valor da ciência*, Garnier, Rio de Janeiro, 1924.

[POLCINO] POLCINO MILIES, F. C., *Anéis e Módulos*, Publicações de Instituto de Matemática e Estatística da USP.

[RÓNYAI1] RÓNYAI, L., *Computing the structure of finite algebras*, J. Symbolic Comput. 9 (1990) 355-373.

[RÓNYAI2] RÓNYAI, L., *Factoring polynomials over finite fields*, Proc. 1987 IEEE Symp. on Foundations of Computer Science (1987), p. 132-137.

[SanMARTIN] SAN MARTIN, L. A. B., *Álgebras de Lie*, Editora da Unicamp, 1999.

[SILVA] SILVA, J. C., *Álgebras de Lie de Derivações Livres de Constantes*, Dissertação de Mestrado, UnB, 2004.

[WIKKIPEDIA] Wikkipedia, *Complexity Classes, NP-Complete, NP-Hard, Turing Machines*, http://www.wikipedia.org, 2004.

[WILF] WILF, H. S., *Algorithms and Complexity*, 1994.

Anexos

1. Certificado de Apresentação de Seminário: *Radicalizando nas Álgebras Associativas*. Apresentado no evento "Quintas de Álgebra", realizado no DMA/UFV.

2. Certificado de Apresentação de Seminário: *Uma abordagem Algorítmica às Álgebras de Lie*. Apresentado no evento "Quartas de Álgebra", realizado no DMA/UFV.

3. Resumo publicado no *25º Colóquio Brasileiro de Matemática*, realizado no Instituto Nacional de Matemática Pura e Aplicada - IMPA, referente à apresentação de poster com o título *Algoritmos para Álgebras Associativas e de Lie*, no dia 25 de julho de 2005.

CERTIFICADO

Certifico, para os devidos fins, que **RICARDO MIRANDA MARTINS** apresentou o seminário **Radicalizando nas Álgebras Associativas** no evento **Quintas de Álgebra**, realizado no Departamento de Matemática da Universidade Federal de Viçosa em 18 de novembro de 2004.

Viçosa, 15 de agosto de 2005.

Rogério Carvalho Picanço
Coordenador do Evento

CERTIFICADO

Certifico, para os devidos fins, que **RICARDO MIRANDA MARTINS** apresentou o seminário **Uma abordagem algorítmica às Álgebras de Lie** no evento **Quartas de Álgebra**, realizado no Departamento de Matemática da Universidade Federal de Viçosa em 6 de abril de 2005.

Viçosa, 15 de agosto de 2005.

_____

Rogério Carvalho Picanço
Coordenador do Evento

RESUMO
25o Colóquio Brasileiro de Matemática

Título: Algoritmos para Álgebras Associativas e de Lie
Autores: GUERREIRO, Marinês, PICANÇO, Rogério C., MARTINS, Ricardo Miranda
Instituições: Departamento de Matemática/Universidade Federal de Viçosa
Sessão Temática: Álgebra, Geometria Algébrica

Poucas áreas na Álgebra têm tido tantas aplicações na matemática e na física quando as Álgebras de Lie, e estas aplicações invariavelmente exigem muitos cálculos. Com o sucesso obtido na utilização de algoritmos e ferramentas computacionais para resolver problemas em matemática, especialmente nas últimas décadas, é razoável considerar que estes podem também ser úteis para álgebras associativas e de Lie.

O estudo da complexidade computacional e o desenvolvimento de bons algoritmos para cálculos em álgebras de Lie iniciou-se por volta de 1975. O sucesso maior foi obtido para álgebras de Lie de dimensão finita, onde se produzem novos e mais eficientes algoritmos constantemente, e as implementações computacionais já são uma realidade, principalmente nos sistemas de computação algébrica, como o GAP e o MAGMA.

O objetivo deste trabalho foi analisar os algoritmos e heurísticas existentes para realizar cálculos em estruturas algébricas, tanto do ponto de vista algébrico, justificando cada passo dos algoritmos através dos teoremas clássicos de estrutura, quanto do ponto de vista computacional, criticando sua complexidade.

Inicialmente foi estudada a teoria das álgebras associativas e também das álgebras de Lie de um ponto de visto algorítmico, ou seja, a cada nova definição e cada teorema, utilizávamos o software GAP para efetuar alguns cálculos elucidativos a cerca do objeto de estudo. Quando do aparecimento dos "famosos"teoremas de estrutura (Wedderburn, Teoremas de Lie e de Engel), procuramos testar os algoritmos implementados no GAP relacionados a tais teoremas com vários exemplos. Isso nos permitia analisar, ainda que experimentalmente, a relação entre a dimensão da álgebra e a velocidade da resposta.

Com alguns dados experimentais, os códigos-fonte dos algoritmos foram estudados, procurando estabelecer a complexidade de cada um deles e, para isso, foi necessário estudar um pouco de teoria de complexidade computacional e análise de algoritmos.

Após já estarmos familiarizados com o software GAP, alguns algoritmos foram desenvolvidos, como por exemplo um algoritmo para gerar o diagrama de Dynkin de uma álgebra de Lie simples a partir de suas constantes de estrutura.

Apresentaremos neste trabalho alguns algoritmos para Álgebras de Lie, bem como análises de complexidade e vários exemplos, onde procuramos mostrar o quanto o uso de ferramentas computacionais facilita os cálculos ao trabalharmos com Álgebras (Associativas e de Lie), ainda que de baixa dimensão.