



UNIVERSIDADE ESTADUAL DE SANTA CRUZ
PRO-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM COMPUTACIONAL
EM CIÊNCIA E TECNOLOGIA

RUDHERO MONTEIRO DOS SANTOS

MODELOS COMPUTACIONAIS PARA VERIFICAÇÃO DE IDENTIDADES POLINOMIAIS
EM ÁLGBRAS DOS TIPOS $M_n(E)$ E $M_{k,l}(E)$

ILHÉUS-BA
2017

RUDHERO MONTEIRO DOS SANTOS

**MODELOS COMPUTACIONAIS PARA VERIFICAÇÃO DE
IDENTIDADES POLINOMIAIS EM ÁLGEBRAS DOS TIPOS
 $M_n(E)$ E $M_{k,l}(E)$**

Dissertação apresentada ao Programa de Pós-Graduação em Modelagem Computacional em Ciência e Tecnologia da Universidade Estadual de Santa Cruz, como parte das exigências para obtenção do título de Mestre em Modelagem Computacional em Ciência e Tecnologia.

Orientador: Prof. Dr. Sérgio Mota Alves

Coorientador: Prof. Dr. Francisco Bruno Souza Oliveira

ILHÉUS-BA
2017

S237

Santos, Rudhero Monteiro dos.

Modelos computacionais para verificação de identidades polinomiais em álgebras dos tipos $M_n(E)$ e $M_k;l(E)$ / Rudhero Monteiro dos Santos. – Ilhéus, BA: UESC, 2017.

60 f.; Anexos.

Orientador: Sérgio Mota Alves.

Dissertação (Mestrado) – Universidade Estadual de Santa Cruz. Programa de Pós-Graduação em Modelagem Computacional em Ciência e Tecnologia.

Inclui referências.

1. Computação - Matemática. 2. Maple (Programa de computador). 3. Álgebra. 4. Polinômios.
I. Título.

CDD 004.0151

RUDHERO MONTEIRO DOS SANTOS

**MODELOS COMPUTACIONAIS PARA VERIFICAÇÃO DE
IDENTIDADES POLINOMIAIS EM ÁLGEBRAS DOS TIPOS**


$M_n(E)$ E $M_{k,l}(E)$

Ilhéus-BA, 23/02/2017

Comissão Examinadora



Prof. Dr. Sérgio Mota Alves
UESC
(Orientador)



Prof. Dr. Francisco Bruno Souza Oliveira
UESC
(Coorientador)



Prof. Dr. Jorge Henrique de Oliveira Sales
UESC



Prof. Dr. Fabíolo Moraes Amaral
IFBA

Dedico este trabalho aos meus pais Avenice e Ailton, aos meus 11 irmãos Jonatas, Airam, Urânio, Abda, Afrânio, Claudia, Clestia, Uerliton, Clestiane, Cledinea e Eugenio e à minha companheira de longa data Luana.

Agradecimentos

- À Deus pela vida.
- À Fundação de Amparo à Pesquisa do Estado da Bahia (FAPESB) pelo apoio financeiro.
- Aos professores Dr. Sérgio Mota e Dr. Francisco Bruno pelo aprendizado e orientações prestadas no presente trabalho.
- Aos professores Dr. Dany Sanchez, Dr. Gildson Queiroz e professora Dra. Susana Marrero pelos ensinamentos.
- Aos colegas do PPGMC pelas conversas e momentos de lazer.
- Aos amigos de hoje e de sempre, em especial Welber Gobis, pela motivação.
- Aos funcionários do PPGMC pela prestatividade e paciência por eu sempre ser o último a sair do laboratório.

Resumo

Nesse trabalho apresentamos uma abordagem computacional para tratar das álgebras que satisfazem identidades polinomiais. Mais precisamente, utilizamos o software Maple para verificar e identificar identidades polinomiais das álgebras de matrizes com entradas na álgebra de Grassmann E , em especial a álgebra $M_{k;l}(E)$, a qual Di Vincenzo e La Scala apresentam resultados interessantes quando $k = l = 1$, usando a noção de identidades polinomiais fracas. Foram criados alguns procedimentos em Maple para adequar o produto das matrizes segundo as propriedades de E , sendo esta uma álgebra não comutativa. Este software apresenta algumas funções previamente implementadas que permitem trabalhar com tais propriedades, porém, o tempo de processamento é consideravelmente maior em comparação com algumas das funções que implementamos. Finalizamos com estudo da conjectura dada por Kemer a cerca do grau mínimo do polinômio standard para a álgebra $M_n(E)$.

Palavras-chave: PI-álgebras; Maple; Computação Algébrica; Identidades Polinomiais Fracas.

Computational Models for Verify Polynomial Identities in algebras of the types $M_n(E)$ and $M_{k,l}(E)$

Abstract

In this work is presented a computational approach to deal with algebras that satisfy polynomial identities. More precisely, we used the Maple software to verify and identify polynomial identities of matrix algebras with entries in the Grassmann algebra E , especially the algebra $M_{k,l}(E)$, which Di Vincenzo and La Scala show interesting results when $k = l = 1$, using the notion of weak polynomial identities. Some procedures were created in Maple to suit the product of the matrices according to the properties of E , this being a non-commutative algebra. This software has some previously implemented functions that allow to work with such properties, however, the processing time is considerably higher in comparison with some of the functions that we implement. We conclude with a study of the conjecture given by Kemer at about the minimum degree of the standard polynomial for the algebra $M_n(E)$.

Keywords: PI-algebras; Maple; Algebraic computing; Weak Polynomial Identities.

Lista de figuras

Figura 1 – Descrição geral do Pacote	16
Figura 2 – Lista de funções disponíveis	17

Lista de tabelas

Tabela 1 – Tempo (em segundos) para calcular o comutador de comprimento n	21
Tabela 2 – Tempo para calcular o polinômio standard de grau n	38
Tabela 3 – Coeficientes das entradas do polinômio standard de grau n	39

Lista de quadros

Quadro 1 – Principais funções utilizadas do pacote <i>Physics</i>	17
Quadro 2 – Procedimentos com as propriedades da Álgebra de Grassmann	31

Lista de símbolos

$\alpha, \beta, \sigma, \phi$	Letras do alfabeto grego: alpha, beta, gamma, phi
$\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{C}$	Conjuntos numéricos: naturais, inteiros, racionais, reais, complexos
Σ	Letra grega Sigma usada para somatórios.
\cup, \cap	Operadores de união e interseção
\emptyset	Conjunto vazio
\circ	Operador de composição de funções
$*$	Operadores de multiplicação
\oplus	Soma direta
\forall	Para todo
S_n	Grupo das permutações dos elementos $\{1, \dots, n\}$.
$\text{sgn}(\sigma), (-1)^\sigma$	Sinal da permutação σ
$E, E(V)$	Álgebra de Grassmann sobre o espaço vetorial V
E_0, E_1	Subespaços da Álgebra de Grassmann
$\text{mindeg}(A)$	Grau mínimo para polinômios da álgebra A
$M_n(K)$	Álgebra das matrizes quadradas de ordem n com entradas no corpo K qualquer
$\text{char}K$	Característica do corpo K
$M_{k,l}(E)$	Subálgebra de $M_{k+l}(E)$ onde $k + l = n$
$M_n(E)$	Álgebra das matrizes quadradas de ordem n com entradas em E
1_A	Unidade da álgebra A
$Z(A)$	Centro da álgebra A
$T(A)$	Ideal das identidades polinomiais da álgebra A
$K\langle X \rangle$	Álgebra dos polinômios não comutativos
$ Y $	cardinalidade do conjunto Y

Sumário

1 – Introdução	1
1.1 Organização da Dissertação	2
2 – Noções Preliminares	4
2.1 O grupo S_n	4
2.2 Álgebras	6
2.3 Álgebras com Identidades Polinomiais	9
2.3.1 Álgebra dos Polinômios não-comutativos	9
2.3.2 Identidades Polinomiais	11
3 – Sobre o Maple	13
4 – Procedimentos em Maple para Verificar Identidades Polinomiais	19
4.1 A Identidade $[[x_1, x_2], x_3]$ da Álgebra de Grassmann	19
4.2 A identidade de Hall $[[x_1, x_2]^2, x_3]$	24
4.3 O Teorema de Amtsur e Levitzki	25
5 – Identidades Polinomiais para $M_{1,1}(E)$	31
6 – Grau mínimo do polinômio standard para $M_{k,l}(E)$	36
6.1 O Polinômio Standard para $M_{k,l}(E)$	36
6.2 Outros resultados sobre o Polinômio standard	39
7 – Considerações Finais	41
Referências	43
Anexos	45
ANEXO A – Procedimentos em Maple	46

1 Introdução

A teoria das álgebras com identidades polinomiais, ou simplesmente PI-álgebras, é uma teoria relativamente nova, tendo seu maior desenvolvimento ocorrido a partir de 1945, sobretudo com os trabalhos dos matemáticos N. Jacobson, J. Levitzki e I. Kaplansky, que tratavam da estrutura de anéis (ou álgebras) com identidades polinomiais. Antes desse período podemos destacar trabalhos como os de (DEHN, 1922) e (WAGNER, 1936), considerados os primeiros a exibirem polinômios em variáveis não comutativas que se anulam quando avaliados por elementos de uma álgebra, ambos motivados pela geometria.

Do ponto de vista combinatório e computacional, como um dos primeiros trabalhos relevantes podemos citar (AMITSUR; LEVITZKI, 1950) onde é provado que o polinômio standard (veja definição 2.3.5) de grau $2n$ é a identidade de menor grau da álgebra $M_n(K)$, onde K é um corpo de característica zero. Tal resultado é conhecido como o teorema de Amitsur-Levitzki, um dos mais importantes para essa teoria tendo sido demonstrado de formas diferentes por vários pesquisadores.

Em (KAPLANSKY, 1957) foi apresentado uma lista de problemas que motivaram significativamente a pesquisa da década. Um destes problemas diz respeito a existência de polinômios centrais para a álgebra das matrizes, $M_n(K)$ com $n > 2$. Uma versão revisada é apresentada em (KAPLANSKY, 1970). A positivação do problema de Kaplansky foi dada independentemente em 1972-1973 por (FORMANEK, 1991) e (RAZMYSLOV, 1973b), e este fato foi de grande importância para a teoria dos anéis.

Alguns dos teoremas importantes que foram estabelecidos ou simplificados usando a noção de polinômios centrais podem ser consultados em (ZIMMERMANN, 1975; ROWEN, 1980; FORMANEK, 1991). Combinando as ideias de Formanek e Razmyslov, vários polinômios centrais foram construídos (HALPIN, 1983; DRENSKY, 1995; GIAMBRUNO; VALENTI, 1996).

A noção de identidade polinomial fraca (ver definição 5) foi introduzida por (RAZMYSLOV, 1994) visando resolver vários problemas, um deles foi construir polinômios centrais para $M_n(K)$. Outro motivo foi estudar as identidades polinomiais ordinárias para $M_n(K)$.

Por exemplo, uma das observações de (RAZMYSLOV, 1973a) resulta que as identidades polinomiais de $M_2(K)$, quando o corpo K é de característica 0 (denotamos $\text{char } K = 0$), seguem das identidades polinomiais de grau menor que 7 e que as identidades fracas de $M_2(K)$ são consequências da identidade:

$$[x_1^2, x_2] = 0. \tag{1}$$

Um outro problema refere-se ao grau mínimo dos polinômios centrais para a álgebra de matrizes $M_n(K)$ quando a característica de K é zero. Os polinômios centrais de (FORMANEK, 1972) são de grau n^2 . Os polinômios obtidos por Razmyslov são de grau $3n^2 - 1$. Usando a noção de identidade fraca, Halpin reduziu os polinômios de Razmyslov para polinômios de grau n^2 . Em posse dessas informações, Formanek conjecturou que o grau mínimo dos polinômios centrais para $M_n(K)$ sobre um corpo K de característica zero é dada por:

$$\text{mindeg}(M_n(K)) = \frac{1}{2}(n^2 + 3n - 2). \quad (2)$$

Um fato importante é que a conjectura de Formanek está diretamente relacionada com vários problemas em aberto na teoria das álgebras com identidades polinomiais.

Para $n = 4$, (DRENSKY; CATTANEO, 1995) encontraram um polinômio central de grau $13 = \frac{1}{2}(4^2 + 3 \cdot 4 - 2)$. Mas não sabemos se existem polinômios centrais de grau 12 para $M_4(K)$. Eles usaram identidades fracas de grau 8 e 9, e os métodos de Razmyslov para obter o resultado, o qual (DRENSKY, 1995) generalizou para construir polinômios de grau $(n - 1)^2 + 4$ para $M_n(K)$ com $n > 2$.

Conseqüentemente, surge o problema a cerca do grau mínimo para uma identidade polinomial fraca da álgebra de matrizes $M_n(K)$ quando $n > 2$. Este foi positivado para $n = 2$ por (RAZMYSLOV, 1973b), o qual mostrou que todas as identidades polinomiais fracas de $M_2(K)$ são conseqüências da identidade fraca da equação 2.

Para $n = 3$, foram (DRENSKY; RASHKOVA, 1993) que descreveram todas as identidades polinomiais fracas de grau 6, essas identidades fracas deram lugar a polinômios centrais de grau 8 e 9.

Nesse trabalho, buscamos resultados parecidos para as álgebras $M_n(E)$ e $M_{k,l}(E)$ (ver definição 2.2.5), sendo E a álgebra de Grassmann sobre um corpo de característica positiva diferente de 2.

1.1 Organização da Dissertação

Nesse trabalho buscamos implementar modelos computacionais para identificar se um dado polinômio é, ou não, uma identidade para uma álgebra dada.

Para isso, apresentamos alguns conceitos inerentes ao estudo de álgebras com identidades polinomiais (ou PI-álgebras). Isso é evidenciado no capítulo 2, o qual dividimos em 4 seções. Na seção 2.1 são dadas algumas definições e resultados a cerca do grupo das permutações, visto que este aparece em diversos problemas ligados ao tema. Na seção 3 apresentamos o software Maple, sendo este uma boa ferramenta para trabalhar com computação simbólica. Nas duas seções subsequentes apresentamos os objetos principais do nosso texto: as álgebras e as identidades

polinomiais. São dados alguns exemplos clássicos presentes em basicamente todos os textos introdutórios que versam sobre o tema.

Alguns dos exemplos citados são verificados no Maple a partir de procedimentos próprios (usando programação em Maple), bem como funções e pacotes previamente disponibilizados pelo software. Isso compõe o capítulo 4, o qual é subdividido em 3 seções, sendo apresentado um exemplo de PI-álgebra em cada uma delas. Vale destacar a seção 4.3 onde apresentamos uma demonstração para o teorema de Amitsur e Levitzki e o verificamos usando $n = 1, 2, 3, 4, 5$ para o polinômio standard.

Para satisfazer as propriedades da álgebra de Grassmann foi preciso criar alguns procedimentos específicos apresentados no capítulo 5. Nele, introduzimos o conceito de identidades fracas e apresentamos as identidades encontradas por Di Vincenzo e La Scala para a álgebra $M_{1,1}(E)$. Além dessas, mostramos as identidades obtidas por Popov para essa mesma álgebra, em sua forma geral.

Tendo em vista o polinômio standard de grau $2n$ ser a identidade de menor grau para a álgebra $M_n(K)$, buscamos resultados análogos para a álgebra $M_{k,l}(E)$, em particular identificamos o menor grau para o polinômio standard de $M_{1,1}(E)$ quando a característica do corpo é 3. Para características $p > 3$ encontramos dificuldades computacionais para encontrar identidades devido as propriedades do grupo S_n , presente na definição do polinômio standard. Há trabalhos nessa direção para a álgebra $M_n(E)$, onde Kemer deixa uma conjectura com um "chute" inicial para o grau mínimo.

Finalizamos o trabalho com comparativos sobre a eficiência dos procedimentos próprios e das funções previamente disponibilizadas no Maple e fazemos algumas considerações a cerca da programação paralela em Maple.

2 Noções Preliminares

Nesse capítulo apresentamos algumas definições e resultados inerentes ao estudo de Álgebras com Identidades Polinomiais, ou simplesmente PI-álgebras. Apresentamos também uma introdução ao software Maple, com foco nos pacotes algébricos e programação que serão utilizados neste trabalho.

2.1 O grupo S_n

Iniciamos esta seção com a definição geral de grupo e apresentamos o grupo das permutações como exemplo, trazendo as principais propriedades e resultados interessantes a cerca do mesmo. Para mais detalhes, sugerimos a consulta do capítulo 6 do livro (ARMSTRONG, 1988).

Definição 2.1.1. *Um grupo é um par $(G, *)$, constituído por um conjunto G e uma operação binária $* : G \times G \rightarrow G$, que satisfaz os seguintes axiomas:*

- (1) *associatividade: $x * (y * z) = (x * y) * z, \forall x, y, z \in G$*
- (2) *existência de elemento neutro: $\exists e \in G$ tal que $x * e = e * x = x$*
- (3) *existência de elemento simétrico: $\forall x \in G, \exists x' \in G$ tal que $x * x' = x' * x = e$.*

Se, além disso, $x * y = y * x$ para todos os elementos $x, y \in G$, o grupo $(G, *)$ diz-se *abeliano* ou *comutativo*.

A fim de simplificar a notação, falaremos no grupo G , em vez de $(G, *)$, sempre que não exista ambiguidade quanto à operação considerada. Usaremos, em geral, notação multiplicativa, substituindo $x * y$ por xy . Neste caso, o inverso de um elemento x será denotado por x^{-1} . Denotaremos o elemento neutro de G simplesmente por e . A *ordem* de um grupo finito G é o número de elementos do conjunto subjacente que se denota por $|G|$. Um grupo com um número infinito de elementos diz-se que tem ordem infinita.

A seguir apresentamos um exemplo de grupo, conhecido como *grupo simétrico* ou *grupo das permutações*.

Por permutação de um conjunto X entende-se uma função bijetiva $\alpha : X \rightarrow X$. O conjunto S_X das permutações de X é um grupo com a operação \circ , que é a composição de funções. Se X é infinito, S_X é um grupo infinito. Se X tem n elementos, por exemplo $X = \{1, 2, \dots, n\}$, o grupo simétrico correspondente denota-se por S_n e tem ordem $n!$.

Permutações $\alpha \in S_n$ podem ser representadas na forma matricial como segue

$$\begin{pmatrix} 1 & 2 & \dots & n \\ \alpha(1) & \alpha(2) & \dots & \alpha(n) \end{pmatrix}$$

Para efetuar a composição de dois elementos

$$\alpha = \begin{pmatrix} 1 & 2 & \dots & n \\ \alpha(1) & \alpha(2) & \dots & \alpha(n) \end{pmatrix} \text{ e } \beta = \begin{pmatrix} 1 & 2 & \dots & n \\ \beta(1) & \beta(2) & \dots & \beta(n) \end{pmatrix}$$

Procedemos da seguinte forma

$$\begin{aligned} \alpha \circ \beta &= \begin{pmatrix} 1 & 2 & \dots & n \\ \alpha(1) & \alpha(2) & \dots & \alpha(n) \end{pmatrix} \circ \begin{pmatrix} 1 & 2 & \dots & n \\ \beta(1) & \beta(2) & \dots & \beta(n) \end{pmatrix} \\ &= \begin{pmatrix} 1 & \dots & i & \dots & n \\ \alpha(\beta(1)) & \dots & \alpha(\beta(i)) & \dots & \alpha(\beta(n)) \end{pmatrix}, \quad i = 1, 2, \dots, n \end{aligned}$$

Exemplo 2.1.1. Os elementos do grupo S_3 são representados matricialmente por:

$$\begin{aligned} e &= \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}, \quad \alpha_1 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}, \quad \alpha_2 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}, \\ \alpha_3 &= \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix}, \quad \alpha_4 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}, \quad \alpha_5 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}. \end{aligned}$$

Observemos como se obtém $\alpha_1 \circ \alpha_4$, por exemplo:

$$\alpha_1 \circ \alpha_4 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix} \circ \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix} = \alpha_5$$

Se a_1, a_2, \dots, a_n são elementos distintos de X , por $(a_1 a_2 \dots a_k)$ denota-se a permutação que aplica a_1 em a_2 , a_2 em a_3 , \dots , a_{k-1} em a_k , a_k em a_1 e fixa os restantes elementos de X . Uma tal permutação chama-se permutação *cíclica* ou *ciclo* de comprimento k . Ciclos de comprimento $k = 2$ chamam-se *transposições*. Os ciclos $(a_1 a_2 \dots a_k)$ e $(b_1 b_2 \dots b_s)$ dizem-se *disjuntos* se $\{a_1, a_2, \dots, a_k\} \cap \{b_1, b_2, \dots, b_s\} = \emptyset$.

Ciclos de comprimento um não se escrevem, em geral. Usando esta notação podemos reescrever os elementos de S_3 como sendo

$$S_3 = \{e, \alpha_1 = (123), \alpha_2 = (132), \alpha_3 = (23), \alpha_4 = (12), \alpha_5 = (13)\}.$$

Para $n \geq 3$ têm-se que S_n é um grupo não comutativo. Os teoremas a seguir são importantes no estudo de S_n e suas respectivas demonstrações são simples, sendo encontradas na referência (ARMSTRONG, 1988).

Teorema 2.1.1. Todo elemento $\alpha \neq e$ de S_n pode ser escrito de forma única, a menos da ordem dos fatores, como um produto de ciclos disjuntos.

Teorema 2.1.2. *O conjunto das transposições de S_n gera S_n .*

Sejam A_n e B_n os subconjuntos de S_n constituídos pelas permutações que podem escrever-se como produto de um número par de transposições e de um número ímpar de transposições, respectivamente.

Prova-se que $S_n = A_n \cup B_n$ e $A_n \cap B_n = \emptyset$, o que nos permite classificar as permutações em *permutações pares*, as que se podem escrever como produto de um número par de transposições, e *permutações ímpares*, no caso contrário. O *signal* de uma permutação α denotado por $\text{sgn}(\alpha)$ é definido como $+1$ se α é uma permutação par e -1 se α é uma permutação ímpar.

2.2 Álgebras

No texto que segue, K denotará um corpo e, a menos que se diga o contrário, todas as álgebras e espaços vetoriais serão definidos sobre K .

Definição 2.2.1. *Seja A um espaço vetorial sobre K munido de uma operação binária $*$: $A \times A \rightarrow A$. Dizemos que A é uma K -álgebra (álgebra sobre K ou simplesmente álgebra) se para qualquer $\alpha \in K$ e quaisquer $a, b, c \in A$, valer as seguintes propriedades:*

- (1) $(a + b) * c = a * c + b * c$
- (2) $a * (b + c) = a * b + a * c$
- (3) $\alpha(a * b) = (\alpha a) * b = a * (\alpha b)$

Em outras palavras, para que A seja uma álgebra, basta que tenha estrutura de espaço vetorial e seja munida de uma operação bilinear a qual chamamos de multiplicação. Na definição anterior, usamos o símbolo $*$ para a multiplicação de dois elementos de A a fim de distingui-la da multiplicação por escalar, mas a partir de agora, para simplificar a notação, denotaremos ambas as operações do mesmo modo, isto é, $a * b$ será escrita como ab .

Definição 2.2.2. *Uma álgebra A é dita ser:*

- (1) *comutativa, se $ab = ba$ para quaisquer $a, b \in A$;*
- (2) *associativa, se $(ab)c = a(bc)$ para quaisquer $a, b, c \in A$;*
- (3) *unitária, se existir $1_A \in A$ chamado de unidade de A tal que $1_A a = a 1_A = a$ para qualquer $a \in A$ (escreveremos 1 ao invés de 1_A).*
- (4) *álgebra de Lie, se para quaisquer $a, b, c \in A$ valem $a^2 = aa = 0$ e $(ab)c + (bc)a + (ca)b = 0$ (identidade de Jacobi).*

(5) *nil*, se para cada $a \in A$, existe $n \in \mathbb{N}$ tal que $a^n = 0$. O elemento a é chamado de *nilpotente* e o menor natural n com tal propriedade é denominado *índice de nilpotência* de a . Quando existe $n \in \mathbb{N}$ tal que $a^n = 0$ para todo $a \in A$, dizemos que A é *nil de índice limitado*.

(6) *nilpotente*, se existe $n \in \mathbb{N}$ tal que o produto de quaisquer $n + 1$ elementos de A com qualquer disposição de parênteses é igual a zero (se A é de Lie ou associativa, isto equivale a dizer que $a_1 a_2 \dots a_n a_{n+1} = 0$ para quaisquer $a_1, a_2, \dots, a_n, a_{n+1} \in A$). Neste caso, definimos o *índice* (ou *classe*) de *nilpotência* de A como sendo o menor n que satisfaz esta condição.

Definição 2.2.3. Uma transformação linear $\varphi : A \rightarrow B$ de álgebras é um **homomorfismo de álgebras**, se $\varphi(ab) = \varphi(a)\varphi(b)$ para quaisquer $a, b \in A$ e além disso $\varphi(1_A) = 1_B$. Analogamente às demais estruturas algébricas, chamaremos φ de **isomorfismo** quando φ for um homomorfismo bijetor, **mergulho** quando φ for injetor, **endomorfismo** quando φ for um homomorfismo de A em A e **automorfismo** quando φ for um endomorfismo bijetor.

Observe que se A é uma álgebra nilpotente, então é nil de índice limitado. Claramente, uma álgebra nil não pode ter unidade.

Apresentaremos a seguir alguns exemplos importantes de álgebras de interesse a presente pesquisa.

Exemplo 2.2.1. O K -espaço vetorial das matrizes $n \times n$ com entradas no corpo K , denotado por $M_n(K)$, munido com a multiplicação usual de matrizes, tem estrutura de álgebra.

Exemplo 2.2.2. Seja V um K -espaço vetorial com base enumerável $\{e_i \mid i \in \mathbb{N}\}$. A álgebra de Grassmann (ou exterior) $E = E(V)$ é a álgebra tendo como base $\beta = \{1, e_{i_1} e_{i_2} \dots e_{i_k} \mid i_1 < i_2 < \dots < i_k, k \geq 1\}$ satisfazendo as relações $e_i e_j = -e_j e_i$ e $e_i^2 = 0$ para todos $i, j \in \mathbb{N}$. Se $\text{char} K \neq 2$ essa última condição segue da primeira. Além disso, destacamos em E os seguintes subespaços vetoriais:

(1) E_0 , gerado pelo conjunto $\beta_0 = \{1, e_{i_1} e_{i_2} \dots e_{i_m} \mid i_1 < i_2 < \dots < i_m, m = 2, 4, 6, \dots\}$

(2) E_1 , gerado pelo conjunto $\beta_1 = \{e_{i_1} e_{i_2} \dots e_{i_k} \mid i_1 < i_2 < \dots < i_k, k = 1, 3, 5, \dots\}$

Claramente, $E = E_0 \oplus E_1$ como espaço vetorial. Desde que $e_i e_j = -e_j e_i$ temos $(e_{i_1} e_{i_2} \dots e_{i_m})(e_{j_1} e_{j_2} \dots e_{j_k}) = (-1)^{mk} (e_{j_1} e_{j_2} \dots e_{j_k})(e_{i_1} e_{i_2} \dots e_{i_m})$ para quaisquer $m, k \in \mathbb{N}$, e assim podemos concluir que $ax = xa$ para quaisquer $a \in E_0$ e $x \in E$, e $bc = -cb$ para quaisquer $b, c \in E_1$.

Observamos facilmente que se $\text{char} K = 2$, então E é uma álgebra comutativa. De fato, teríamos

$$2e_i e_j = 0 \longrightarrow e_i e_j = -e_i e_j \longrightarrow e_i e_j = -(-e_j e_i) = e_j e_i, \forall i, j.$$

Considerando E' a álgebra com base $\{e_{i_1} e_{i_2} \cdots e_{i_k} \mid i_1 < i_2 < \dots < i_k, k \geq 1\}$, temos que E' não tem unidade e é chamada de álgebra Grassmann sem unidade.

Exemplo 2.2.3. O K -espaço vetorial das matrizes $n \times n$ com entradas na álgebra de Grassmann E , denotado por $M_n(E)$, munido com a multiplicação usual de matrizes, tem estrutura de álgebra.

Definição 2.2.4. Seja A uma álgebra e B um K -subespaço vetorial de A . Dizemos que B é uma K -subálgebra de A , se tiver estrutura de álgebra, isto é, se B for fechado com respeito a multiplicação definida em A . O subespaço B será denominado um ideal à esquerda de A , se $AB \subset B$. De modo similar, definiremos ideal à direita de A . Um ideal bilateral (ideal à direita e à esquerda simultaneamente) será simplesmente denominado de ideal.

Definição 2.2.5. Um ideal I de uma álgebra A é dito ser um T -ideal, se I for invariante sob todos os endomorfismos φ de A , isto é, se $\varphi(I) \subseteq I$ para todo endomorfismo $\varphi : A \longrightarrow A$.

Da mesma forma que se define o quociente de espaços vetoriais, podemos definir o quociente de uma álgebra A por um ideal I , estendendo as operações de A às classes de equivalência de A/I . As operações são bem definidas uma vez que I é subespaço vetorial, desde que a álgebra seja unitária.

Apresentaremos a seguir alguns exemplos de subálgebras de interesse na presente pesquisa.

Exemplo 2.2.4. Seja A uma álgebra. O conjunto $Z(A) = \{a \in A \mid ab = ba, \forall b \in A\}$ é uma subálgebra de A , o qual denominamos de **centro** de A e seus elementos são ditos **centrais**. Note que se $A = E$ temos que $Z(E) = E_0$, isto é, E_0 é o centro da álgebra de Grassmann.

Exemplo 2.2.5. Sejam $a, b \in \mathbb{N}$ com $a + b = n$, é fácil verificar através de multiplicação de matrizes, que o K -subespaço de $M_{a+b}(E)$ dado por

$$M_{a,b}(E) = \left\{ \begin{pmatrix} A & B \\ C & D \end{pmatrix} \mid A \in M_a(E_0), B \in M_{a \times b}(E_1), C \in M_{b \times a}(E_1), D \in M_b(E_0) \right\},$$

é uma subálgebra de $M_{a+b}(E)$.

Vale ressaltar a importância das regras das álgebras apresentadas nos exemplos 2.2.2 e 2.2.5 para a implementação dos procedimentos no Maple na obtenção de identidades polinomiais, sobre as quais discorreremos a seguir.

2.3 Álgebras com Identidades Polinomiais

Nessa seção, iniciaremos com a definição de álgebras livres, seguido da construção do conjunto dos polinômios com variáveis não-comutativas. Além disso são definidos os polinômios homogêneos, multilineares e próprios. Por fim, definimos as álgebras com identidades polinomiais, seguida de exemplos importantes.

2.3.1 Álgebra dos Polinômios não-comutativos

Definição 2.3.1. *Seja \mathcal{V} uma classe de álgebras. Dizemos que uma álgebra $F \in \mathcal{V}$ é uma álgebra livre de \mathcal{V} se existe um subconjunto Y (enumerável) gerador de F tal que para toda álgebra $A \in \mathcal{V}$ e toda aplicação $h : Y \rightarrow A$ existe um homomorfismo de álgebras $\varphi : F \rightarrow A$ estendendo h . F é então dita ser livremente gerada por Y e a cardinalidade $|Y|$ do conjunto Y é chamada de posto de F .*

Seja $X = \{x_1, x_2, \dots\}$ um conjunto não vazio e enumerável de variáveis não comutativas. Uma palavra em X é uma sequência $x_1 x_2 \dots x_n$ onde $n \in \mathbb{N}$ e $x_{i_j} \in X$ (para $n = 0$ temos a palavra vazia e denotaremos por 1). Considerando $S(X)$ o conjunto de todas as palavras em X e definindo que duas palavras $x_{i_1} x_{i_2} \dots x_{i_n}$ e $x_{j_1} x_{j_2} \dots x_{j_m}$ são iguais se $n = m$ e $i_1 = j_1, i_2 = j_2, \dots, i_n = j_n$, denotaremos por $K\langle X \rangle$ o espaço vetorial que tem como base o conjunto $S(X)$. Os elementos de $K\langle X \rangle$ serão, portanto, somas (formais) de termos que são produtos (formais) de um escalar por uma palavra em X .

Definindo em $S(X)$ a seguinte multiplicação

$$(x_{i_1} \dots x_{i_n})(x_{j_1} \dots x_{j_m}) = x_{i_1} \dots x_{i_n} x_{j_1} \dots x_{j_m}, \quad \forall x_{i_k} x_{j_l} \in X, \quad (3)$$

é fácil mostrar que $K\langle X \rangle$, com essa multiplicação, é uma álgebra associativa livre com unidade (a palavra vazia). De fato, seja A uma álgebra associativa unitária e $f : X \rightarrow A$ uma aplicação qualquer. Para cada inteiro positivo i (menor ou igual a $|X|$ no caso em que X é finito), denotemos por a_i a imagem de x_i pela f . Seja $\varphi : K\langle X \rangle \rightarrow A$ definida para cada monômio de $K\langle X \rangle$ por $\varphi(x_{i_1} x_{i_2} \dots x_{i_k}) = a_{i_1} a_{i_2} \dots a_{i_k}$ e estendida por linearidade.

Note que, pela definição de φ segue que $\varphi(0) = 0$ e $\varphi(1) = 1$. Sejam $c, d \in K\langle X \rangle$, então $c = \lambda_1 x_{c_{11}} \dots x_{c_{1k}} + \dots + \lambda_n x_{c_{n1}} \dots x_{c_{nl}} + m \cdot 1$ e $d = \mu_1 x_{d_{11}} \dots x_{d_{1p}} + \dots + \mu_r x_{d_{r1}} \dots x_{d_{rq}} + s \cdot 1$. Logo, temos que

$$\varphi(c+d) = \varphi((\lambda_1 x_{c_{11}} \dots x_{c_{1k}} + \dots + \lambda_n x_{c_{n1}} \dots x_{c_{nl}}) + (\mu_1 x_{d_{11}} \dots x_{d_{1p}} + \dots + \mu_r x_{d_{r1}} \dots x_{d_{rq}})). \quad (4)$$

Como na definição de φ a estendemos por linearidade, segue de (4) que

$$\begin{aligned} \varphi(c+d) &= (\lambda_1 a_{c_{11}} \dots a_{c_{1k}} + \dots + \lambda_n a_{c_{n1}} \dots a_{c_{nl}}) + (\mu_1 a_{d_{11}} \dots a_{d_{1p}} + \dots + \mu_r a_{d_{r1}} \dots a_{d_{rq}}) \\ &= \varphi(c) + \varphi(d). \end{aligned} \quad (5)$$

Analogamente, com a multiplicação definida em (3), demonstra-se que $\varphi(cd) = \varphi(c)\varphi(d)$ e $(\lambda b) = \lambda\varphi(b)$, para quaisquer $c, d \in K\langle X \rangle$ e qualquer $\lambda \in K\langle X \rangle$. Logo, φ é um homomorfismo de álgebras e, portanto, $K\langle X \rangle$ é livre na classe em questão.

Os produtos de um escalar por uma palavra são chamados de **monômios** e os elementos de $K\langle X \rangle$ são chamados de **polinômios**. Notemos que a álgebra $K\langle X \rangle$ é, em outras palavras, a álgebra dos polinômios não-comutativos.

Definição 2.3.2. Um monômio m tem grau k em x_i se a variável x_i ocorre em m exatamente k vezes. Um polinômio é homogêneo de grau k em x_i se todos os seus monômios têm grau k em x_i . Denotamos este fato por $\deg_{x_i} f = k$. Um polinômio linear em x_i é um polinômio de grau 1 em x_i .

Definição 2.3.3. Um polinômio é **multihomogêneo** se para cada variável x_i todos os seus monômios têm o mesmo grau em x_i . Um polinômio é **multilinear** se é linear em cada variável. O grau de um polinômio é o maior grau entre todos os seus monômios.

Sejam f um polinômio de $K\langle X \rangle$ de grau n e x_k uma variável de f . Podemos escrever f como uma soma $f = \sum_{i=0}^n f_i$, onde cada polinômio f_i é homogêneo de grau i na variável x_k . Cada polinômio f_i é a componente homogênea de grau i em x_k do polinômio f .

Os polinômios multilineares e multihomogêneos desempenham um papel importante na busca de bases para as identidades polinomiais sobre determinados tipos de corpos. Quando a álgebra é unitária podemos restringir a nossa busca de identidades polinomiais a um determinado tipo de polinômios definido a seguir.

Definição 2.3.4. O comutador de comprimento n é definido indutivamente a partir de $[x_1, x_2] = x_1x_2 - x_2x_1$ tomando $[x_1, x_2, \dots, x_n] = [[x_1, x_2, \dots, x_{n-1}], x_n]$ para $n > 2$. Um polinômio $f \in K\langle X \rangle$ é chamado **polinômio próprio** (ou comutador), se ele é uma combinação linear de produtos de comutadores, isto é,

$$f(x_1, x_2, \dots, x_n) = \alpha_{i_1, \dots, j_1} [x_{i_1}, \dots, x_{j_1}] \dots [x_{j_1}, \dots, x_{j_q}] \text{ com } \alpha_{i_1, \dots, j_1} \in K$$

(Assumindo que 1 é um produto de um conjunto vazio de comutadores). Denotamos por $B(X)$ o conjunto de todos os polinômios próprios de $K\langle X \rangle$.

A seguir apresentamos alguns exemplos importantes de polinômios que aparecerão nos próximos capítulos.

Definição 2.3.5. O polinômio

$$st_n(x_1, \dots, x_n) = \sum_{\sigma \in S_n} (-1)^\sigma x_{\sigma(1)} \dots x_{\sigma(n)}$$

é chamado **polinômio standard** de grau n . Aqui $(-1)^\sigma$ é o sinal da permutação σ do grupo simétrico S_n .

Definição 2.3.6. O polinômio

$$e_k(x_1, \dots, x_n) = \sum_{1 \leq i_1 < \dots < i_k \leq n} x_{i_1} \dots x_{i_k}$$

onde $k \leq n$ e as variáveis são não comutativas, é chamado de **polinômio simétrico** de grau k em n variáveis.

Definição 2.3.7. O polinômio

$$p_k(x_1, \dots, x_n) = x_1^k + \dots + x_n^k$$

é chamado **soma de potências** de grau k em n variáveis.

2.3.2 Identidades Polinomiais

Agora, podemos definir uma identidade polinomial.

Definição 2.3.8. Um polinômio $f(x_1, \dots, x_n) \in K\langle X \rangle$ é denominado uma **identidade polinomial** da álgebra A , se $f(a_1, \dots, a_n) = 0$ para quaisquer $a_1, \dots, a_n \in A$. Uma **álgebra com identidade polinomial (PI-álgebra)** é uma álgebra que satisfaz uma identidade polinomial não trivial (polinômio não nulo).

Observemos que $f = f(x_1, \dots, x_n)$ é uma identidade de A se, e somente se, f pertence aos núcleos de todos os homomorfismos de $K\langle X \rangle$ em A . Denotamos por $T(A)$ o conjunto de todas as identidades polinomiais de A , e assim, dizemos que A é uma PI-álgebra se $T(A) \neq \{0\}$.

A seguir, apresentamos alguns exemplos de PI-álgebras as quais verificaremos com procedimentos no Maple no capítulo seguinte.

Exemplo 2.3.1. Toda álgebra comutativa A é uma PI-álgebra, pois o polinômio comutador $f(x_1, x_2) = [x_1, x_2] = x_1x_2 - x_2x_1$ é uma identidade polinomial para A .

Exemplo 2.3.2. A álgebra de Grassmann E é uma PI-álgebra, pois um cálculo direto usando os elementos da base de E mostra que o polinômio comutador $f(x_1, x_2, x_3) = [[x_1, x_2], x_3]$ é uma identidade polinomial para E .

Exemplo 2.3.3. A álgebra de Grassmann sem unidade E' sobre um corpo de característica $p \neq 0$ é uma PI-álgebra, pois $f(x) = x^p$ é uma identidade polinomial para E' .

Exemplo 2.3.4. A álgebra $M_2(K)$ satisfaz a identidade $f(x_1, x_2, x_3) = [[x_1, x_2]^2, x_3]$ conhecida como a identidade de Hall.

Exemplo 2.3.5. (Teorema de Amitsur-Levitzki) A álgebra $M_n(K)$ satisfaz o polinômio standard de grau $2n$

$$st_{2n}(x_1, \dots, x_{2n}) = \sum_{\sigma \in S_{2n}} (-1)^\sigma x_{\sigma(1)} \dots x_{\sigma(2n)}$$

onde S_{2n} é o grupo das permutações de $\{1, 2, \dots, 2n\}$ e $(-1)^\sigma$ é o sinal da permutação σ .

Note que, devido as propriedades do grupo S_n , temos que o polinômio standard é identicamente nulo, sempre que duas de suas variáveis são iguais. Isso nos permite enunciar o seguinte teorema.

Teorema 2.3.1. *Seja A uma álgebra de dimensão finita, digamos n . Então, ela satisfaz o polinômio standard de grau $n + 1$.*

Demonstração. Pelas definições 2.3.3 e 2.3.5 temos que o polinômio standard é multilinear e anti-simétrico, logo ele se anula quando calculado sobre elementos linearmente dependentes. Particularmente, isso ocorre quando dois de seus argumentos são iguais. Assim, sejam $B = \{e_1, \dots, e_n\}$ uma base de A e a_1, \dots, a_{n+1} elementos de A . Podemos escrever cada a_i como combinação linear de elementos de B . Temos então que $st_{n+1}(a_1, \dots, a_{n+1})$ é uma combinação linear com coeficientes em K de elementos da forma $st_{n+1}(e_{i_1}, \dots, e_{i_{n+1}})$ onde cada $e_{i_k} \in B$. Sendo assim algum e_{i_k} se repetirá, e portanto, $st_{n+1}(e_{i_1}, \dots, e_{i_{n+1}}) = 0$ implicando que $st_{n+1}(a_1, \dots, a_{n+1}) = 0$. ■

Um exemplo de uma álgebra que não satisfaz nenhuma identidade polinomial não nula é a álgebra $K\langle X \rangle$. Isto pode ser compreendido por um argumento simples. Suponhamos, por absurdo, que $f(x_1, \dots, x_n)$ seja uma identidade polinomial não nula de $K\langle X \rangle$. Assim, $f(f_1(x_1), \dots, f_n(x_n)) = 0$ onde $f_i(x_i) = x_i$ para $i = 1, \dots, n$, o que é um absurdo, pois $f(x_1, \dots, x_n) \neq 0$.

O exemplo 2.3.5 foi muito importante para auxiliar na implementação computacional do polinômio standard de grau n para a álgebra $M_{k,l}(E)$, objeto central do nosso trabalho. Alguns exemplos dos procedimentos criados, bem como aplicação dos mesmos, é mostrado no capítulo a seguir.

3 Sobre o Maple

Para este capítulo, seguimos as notas de (PORTUGAL, 2002), bem como o guia de ajuda disponibilizado no próprio software ou online no site <http://www.maplesoft.com/support/help>. Indicamos, também, o livro (ANDRADE, 2004) que apresenta uma boa introdução à computação algébrica com Maple tratando de diversos temas da matemática. Aqui, utilizamos a versão 18 do software.

O Maple é um ambiente de computação com diversos aspectos integrados: algébrico, numérico, gráfico e de programação. Ao realizar um procedimento algébrico, definindo uma função, por exemplo, o mesmo pode ser convertido em dados numéricos e gráficos de forma rápida. O Maple é, sobretudo, uma linguagem de programação simbólica. O mesmo teve uma pequena parte escrita em C (o núcleo com cerca de 5%) e todo o restante foi desenvolvido na linguagem Maple. É óbvio que o Maple não elimina completamente a necessidade do uso de outras linguagens de programação, inclusive, o mesmo oferece recursos que facilitam na conversão de programas escritos em Maple para linguagens bem conhecidas como C, Fortran e Matlab.

Nesse trabalho, utilizamos diversos procedimentos simbólicos de álgebra abstrata, o que justifica o uso deste software. A parte do programa que não faz parte do núcleo, consiste de duas: a biblioteca principal e um conjunto de vários pacotes (*packages*) separados. Assim como os comandos que fazem parte do núcleo, os comandos da biblioteca principal são carregados automaticamente na hora da inicialização e estão, de imediato, prontos para serem usados. Segue um exemplo de produto entre matrizes com entradas simbólicas.

Exemplo 3.0.1. *Sejam A e B duas matrizes quadradas quaisquer de ordem 2. Mostramos como podem ser definidas tais matrizes no Maple de forma simples e como utilizar algumas das operações usuais. As linhas de comando que foram inseridas aparecem na cor preta e a resposta do Maple aparece na cor azul, em geral centralizada.*

$A := \text{Matrix}(2, 2, (i, j) \longrightarrow a[i, j]);$

$$\begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix}$$

$B := \text{Matrix}(2, 2, (i, j) \longrightarrow b[i, j]);$

$$\begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix}$$

$A.B - B.A;$

$$\begin{bmatrix} a_{1,2}b_{2,1} - a_{2,1}b_{1,2} & a_{1,1}b_{1,2} - b_{1,1}a_{1,2} + a_{1,2}b_{2,2} - b_{1,2}a_{2,2} \\ -b_{2,1}a_{1,1} + a_{2,1}b_{1,1} - b_{2,2}a_{2,1} + a_{2,2}b_{2,1} & -a_{1,2}b_{2,1} + a_{2,1}b_{1,2} \end{bmatrix}$$

Note que, no exemplo acima, utilizamos o símbolo $.$ (*dot*) entre A e B visto esse ser um operador não comutativo para o Maple, próprio para tratar de matrizes, vetores e outros objetos em que não vale a propriedade comutativa. Vale ressaltar que, da forma como está escrito os comandos acima, o Maple pressupõe que as entradas das matrizes, mesmo que simbólicas, são elementos comutativos. Caso não queiramos que o Maple exiba o resultado, basta finalizar com $:$ cada linha de comando. Vale ressaltar que nas versões atuais, não é obrigatório a utilização dos símbolos $:$ ou $;$ ao término de um único comando na folha de trabalho.

Se substituirmos o símbolo $.$ por $*$ o maple retornará a seguinte mensagem de erro: *Error, (in rtable/Product) invalid arguments*. Isso se deve ao fato de $*$ ser definido, no Maple, como um operador binário entre dois elementos que comutam entre si.

O fato de o Maple possuir um operador não comutativo o torna ainda mais interessante para o nosso trabalho visto que estudamos álgebras com tais propriedades. Vale ressaltar que esse operador é associativo e possui o 1 como elemento neutro, porém não vale a distributividade em relação a soma.

Para acrescentar a propriedade distributiva em relação a soma do operador *dot* recorreremos a programação no Maple.

Um procedimento (ou programa) no Maple é essencialmente um conjunto de comandos (do próprio Maple ou estruturas usuais de programação *if, else, for, while*, entre outros) organizados de forma sequenciada e lógica, sendo executados manualmente ou de forma automática na ordem em que aparecem da esquerda para a direita e de cima para baixo.

Procedimento 3.0.1. Função distributiva à direita com o operador $.$ (*dot*)

Entrada: Duas expressões $x_1 + x_2 + \dots + x_n$ e $y_1 + y_2 + \dots + y_n$.

Saída: Uma expressão $x_1.y_1 + \dots + x_i.y_j + \dots + x_n.y_n$

```

1  distribui:=proc(exp1,exp2)
2      local i,j,expaux;
3      if (type(exp2,'+')) then
4          # "caso exp2 e exp1 recebam uma soma ab+bcd+..."
5          if (type(exp1,'+')) then
6              expaux:= 0;
7              for j to nops(exp2) do
8                  for i to nops(exp1) do
9                      expaux:= expaux + op(i,exp1) . op(j,exp2);
10                 end do;

```

```

11         end do;
12         # "caso exp2 recebe uma soma e exp1 um unico termo."
13     else
14         expaux:= 0;
15         for j to nops(exp2) do
16             expaux:= expaux + exp1 . op(j,exp2);
17         end do;
18     end if;
19 else
20     # "caso exp1 recebe uma soma e exp2 um termo."
21     if (type(exp1, '+')) then
22         expaux:= 0;
23         for i to nops(exp1) do
24             expaux:= expaux + op(i,exp1) . exp2;
25         end do;
26     else
27         # "caso exp1 e exp2 recebam um unico termo."
28         expaux:= exp1 . exp2;
29     end if;
30 end if;
31 return expaux;
32 end proc:

```

┘

Nesse procedimento, as variáveis i, j são utilizadas como controladores de estruturas de repetição e $expaux$ armazena a expressão de saída. Cada caso está devidamente comentado após o símbolo `#`. Após este símbolo, tudo que for escrito na mesma linha é desconsiderado pelo compilador do Maple. Além disso, caso se queira um bloco de comentários pode-se usar a estrutura (`* comentários *`).

Note que não precisamos declarar o tipo das variáveis dentro de um procedimento no Maple. A função `type` identifica automaticamente diversos tipos de variáveis que podem ser numéricas, algébricas, simbólicas ou até mesmo outros procedimentos. Porém, deve-se ter cuidado quanto a omissão do tipo da variável visto que os procedimentos podem gerar resultados errados (ou não esperados) a partir de alguns dados de entrada.

Exemplo 3.0.2. Executando, numa mesma folha de trabalho no Maple, o procedimento 3.0.1 e os comandos abaixo, obtemos a expressão de saída a seguir

$$x := a.b.c + e.f.g - a.b :$$

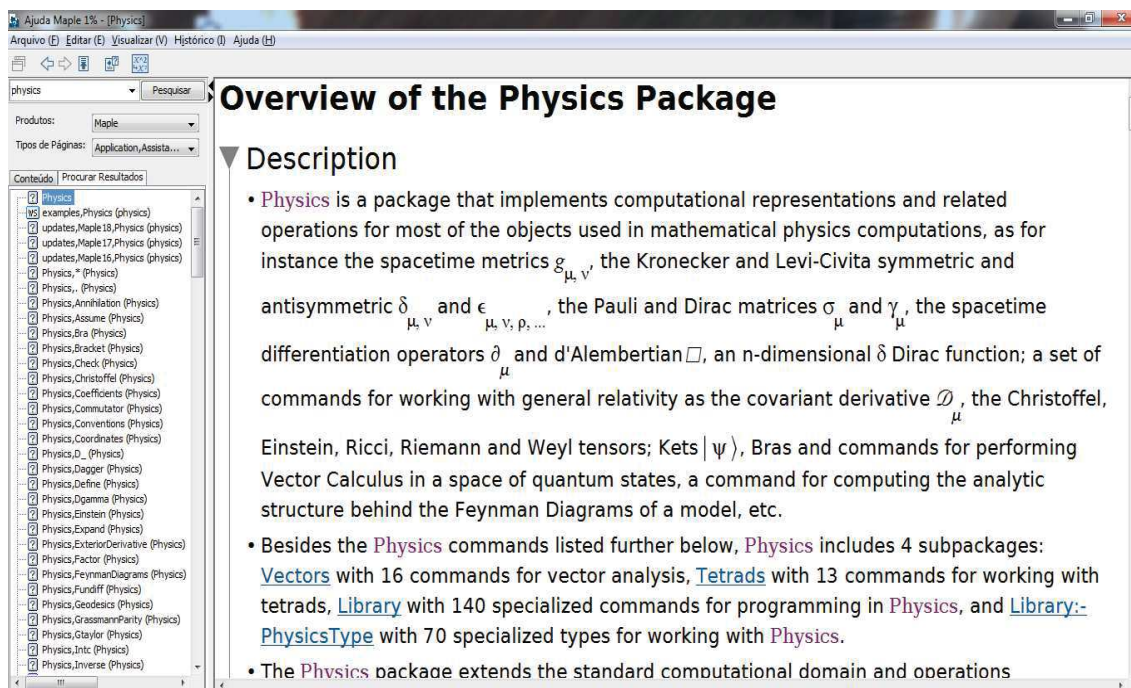
$$y := b + g :$$

$distribui(x, y);$

$$a.b.c.b + e.f.g.b - a.b.b + a.b.c.g + e.f.(g^2) - a.b.g$$

Como dito no início desta seção, o Maple possui vários pacotes separados que podem ser carregados na folha de trabalho com o comando $with(nome\ do\ pacote)$ para que se torne disponíveis algumas funções pré-implementadas. Alguns dos pacotes de nosso interesse foram *LinearAlgebra*, *group*, *combinat*, *Physics*, que permitiram trabalhar com manipulação de matrizes, permutações, análise combinatória entre outros. O pacote *Physics* merece uma atenção especial, visto que o mesmo permite trabalhar de forma simples com álgebra não-comutativa. Por isso, tratamos algumas de suas principais funções que usamos no presente estudo. Ao digitar $?Physics$ na folha de trabalho, abrirá uma página de ajuda com informações sobre o pacote, conforme figura 3.

Figura 1 – Descrição geral do Pacote

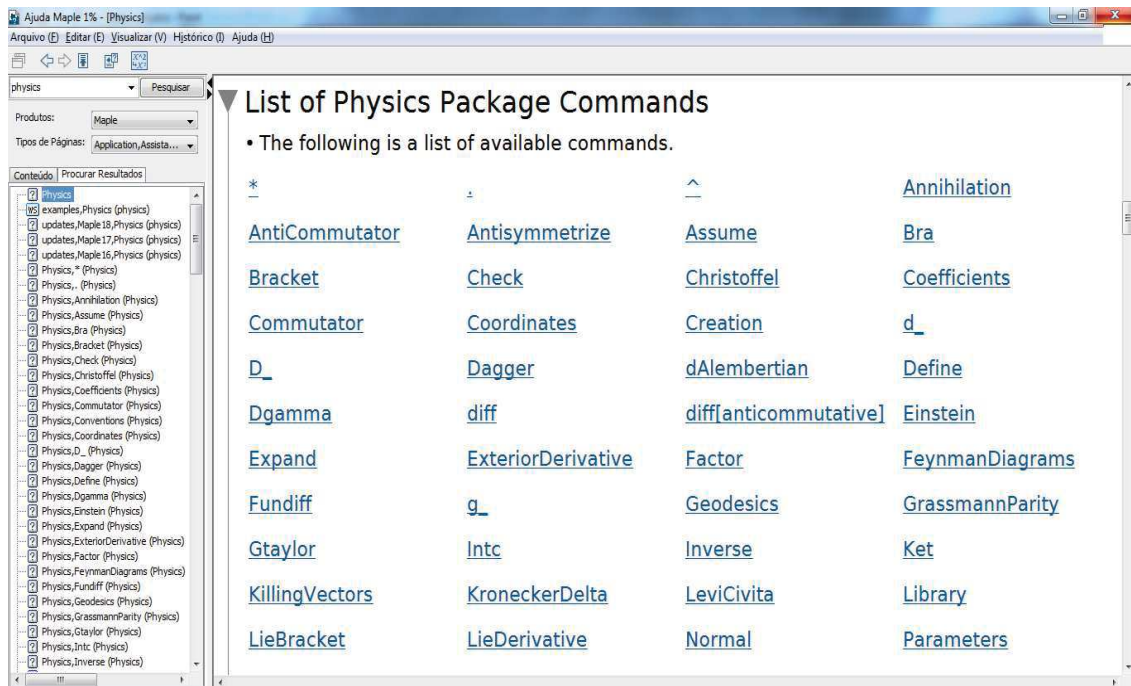


Fonte: Guia de ajuda do Maple 18 - [Physics]

Note que pela descrição apresentada na Figura 3, o pacote Physics pode ser usado para resolver diversos problemas ligados a física-matemática, temas não abordados no presente texto.

Existe uma boa quantidade de funções disponíveis (cf. Figura 3), dentre elas destaca-se a função *Setup* que é usado para definir, limpar e consultar o ambiente computacional usado pelos comandos do pacote *Physics*.

Figura 2 – Lista de funções disponíveis



Fonte: Guia de ajuda do Maple 18 - [Physics]

Destacamos as atribuições de algumas das funções que aparecem na Figura 3 condensadas no quadro 1.

Quadro 1 – Principais funções utilizadas do pacote Physics

Funções do pacote Physics	Atribuições
*	Produto generalizado que opera como um produto comutativo, anticomutativo ou não comutativo dependendo das propriedades de seus operandos.
$Commutator(A, B)$	Retorna o comutador $AB - BA$, onde A e B são duas expressões algébricas quaisquer.
$AntiCommutator(A, B)$	Retorna o anti-comutador $AB + BA$, onde A e B são duas expressões algébricas quaisquer.
$Expand(expr)$	Expande produtos não comutativos sobre somas, Comutadores, etc.
$Parameters(a, b, ...)$	Conjunto de símbolos para usar como parâmetros constantes.
$Setup(parameter = value)$	Define e consulta o ambiente computacional para os comandos do pacote Physics.

Fonte: Guia de ajuda do Maple 18. Tradução livre.

Com a função *Setup*, podemos definir as variáveis como não-comutativas ou anti-comutativas com auxílio dos parâmetros *noncommutativeprefix*, *anticommutativeprefix*, res-

pectivamente. Além disso, para utilizar a notação habitual de matemática, usamos o parâmetro *mathematicalnotation* atribuindo valores lógicos (*true* ou *false*) dentro da função *Setup*. Digamos, por exemplo, que se queira definir x, y como variáveis não-comutativas e w, z como anti-comutativas. Segue uma configuração simples com alguns exemplos e resultados retornados pelo Maple:

with(Physics):

Setup(mathematicalnotation = true,

anticommutativeprefix = {w, z},

noncommutativeprefix = {x, y}):

$f := x \cdot y - y \cdot x$

$$xy - yx$$

$g := w \cdot z - z \cdot w$

$$2wz$$

$h := x \cdot z \cdot w \cdot y + 2 \cdot x \cdot z - z \cdot y \cdot w \cdot z$

$$-wzxy + 2xz$$

Note que em h o último termo é eliminado, já que vale $z^2 = 0$. Vale ressaltar que da forma como está escrito acima, as variáveis x_i, y_j são não-comutativas para quaisquer i, j . Analogamente, vale para w e z . Além das funções supracitadas, vale destacar também as funções *Commutator* e *AntiCommutator*, as quais usaremos frequentemente nos próximos capítulos.

Uma descrição detalhada de cada função ou pacote, com exemplos das possíveis aplicações, podem ser consultadas no guia de ajuda do próprio Maple.

4 Procedimentos em Maple para Verificar Identidades Polinomiais

No capítulo anterior, apresentamos alguns exemplos de álgebras satisfazendo identidades polinomiais. Neste, apresentamos alguns procedimentos computacionais, elaborados em Maple, para verificar a validade de tais identidades, bem como encontrar identidades novas.

4.1 A Identidade $[[x_1, x_2], x_3]$ da Álgebra de Grassmann

Na seção 3, vimos que existe uma função do Maple no pacote *Physics*, denominada *Commutator*, que calcula o comutador entre duas variáveis. Em notação matemática do Maple, o comutador de duas variáveis x, y é dado por $[x, y]_- = xy - yx$ visto que é usado uma notação semelhante para o anti-comutador (função *AntiCommutator*) dado por $[x, y]_+ = xy + yx$. Vale lembrar que são usados apenas dois parâmetros em cada função.

Logo, assumindo que $x_1, x_2, x_3 \in E_1$ (Caso alguma variável seja de E_0 é óbvia a identidade, já que o comutador entre duas delas será nulo), podemos verificar a identidade da seguinte forma:

with(Physics):

Setup(mathematicalnotation = true, anticommutativeprefix = x):

f := Commutator(x[1], x[2])

$[x_1, x_2]_-$

f := Commutator(f, x[3])

0

Se y é uma variável não-comutativa, o comutador $[y_1, y_2, y_3]$ pode ser calculado acrescentando o comando:

Setup(noncommutativeprefix = y):

g := Commutator(y[1], y[2]):

g := Commutator(g, y[3])

$y_1 y_2 y_3 - y_2 y_1 y_3 - y_3 y_1 y_2 + y_3 y_2 y_1$

Porém, um teste de eficiência (ver Tabela 1) da função *Commutator* mostra que a mesma não é uma boa opção para o simples cálculo de comutadores de comprimento maior que 6 quando as variáveis são não-comutativas. Por conta disso, implementamos funções independentes do pacote *Physics* (ver Quadro 2), para resolver esse e outros problemas.

Vale lembrar que o comutador de comprimento $n = 2$ é dado por $[x_1, x_2] = x_1x_2 - x_2x_1$, ou seja, para $n > 2$ basta tomar a diferença entre a distributiva do elemento x_i à esquerda e à direita da expressão anterior, para cada $i = 3, 4, \dots, n$. Logo, podemos utilizar o procedimento 3.0.1 apresentado na seção 3.

Procedimento 4.1.1. *Função comutador (colchete de Lie).*

Entrada: Uma lista $L := [x_1, x_2, \dots, x_n]$ com $n \geq 2$.

Saída: Uma expressão dada pelo colchete de Lie.

```

1  comutador:= proc(L::list)
2      local i, aux;
3      aux:= distribui(op(1,L), op(2,L)) - distribui(op(2,L), op(1,
4          L));
5      for i from 3 to nops(L) do
6          aux:= distribui(aux, op(i,L)) - distribui(op(i,L), aux);
7      end do;
8      return aux;
9  end proc;
```

┘

Neste procedimento, declaramos o tipo de entrada de dados (*list*) usando o símbolo `::` e usamos a variável i como controlador de *loop*, enquanto aux armazena a expressão de saída. Note que utilizamos o procedimento 3.0.1 dentro do procedimento 4.1.1 o que permitiu uma redução de linhas de comandos no mesmo.

Usando o procedimento 4.1.1 (que deve ser executado na mesma folha de trabalho do procedimento 3.0.1) podemos, por exemplo, reescrever a identidade $f(x_1, x_2, x_3) = [[x_1, x_2], x_3]$ de E entrando com os seguintes comandos no Maple

```
f:=[x[1], x[2], x[3]]:
```

```
comutador(f);
```

$$x_1.x_2.x_3 - x_2.x_1.x_3 - x_3.x_1.x_2 + x_3.x_2.x_1$$

Note que obtemos um resultado semelhante ao anterior, exceto por questões de notação (o operador $.$ fica visível), sem usar o *Setup*. Calculamos o tempo real gasto para obter o comutador de comprimento $n = 2, \dots, 10$ usando as funções *comutador* e *Commutator*, separadamente. Para isso, utilizou-se a função *time[real]()* que retorna o tempo real decorrido desde a inicialização do núcleo do Maple, em segundos, truncado em apenas 3 casas decimais. Os resultados aparecem na tabela 1.

Tabela 1 – Tempo (em segundos) para calcular o comutador de comprimento n

n	Usando <i>comutador</i>	Usando <i>Commutator</i>
2	1E-3	1E-3
3	1E-3	1E-3
4	2E-3	6E-3
5	7E-3	1E-2
6	1.4E-2	3E-2
7	4E-2	1.2941E1
8	6.7E-2	4.737E1
9	1.32E-1	2.55582E2
10	2.75E-1	4.31738E2

Note que, a partir de $n = 6$ a função *Commutator* passa a ser muito lenta. Porém, apesar de calcular de forma mais rápida, a função *comutador* não leva em consideração nenhuma propriedade da variável (lembrando que o operador $.$ não comuta as variáveis). Ou seja, caso queiramos que as variáveis anti-comute, precisamos criar um novo procedimento, o qual descrevemos a seguir.

Para atender a regra do produto da álgebra de Grassmann, isto é, $x_1x_2 = -x_2x_1$, criamos o procedimento a seguir

Procedimento 4.1.2. *Função para ordena os $x_i \in E_1$ usando a regra anti-comutativa: $x_ix_j = -x_jx_i$.*

Entrada: Um produto com elementos indexados x_i .

Saída: Um produto de x_i com índices organizados em ordem crescente e o sinal.

```

1  ordenaE1 := proc (expr)
2      local sinal, aux, vet, expaux, expaux2, i, j, cont;
3      if op(1, expr) = -1 then
4          expaux := -expr;
5          sinal := -1;
6      else
7          expaux := expr;
8          sinal := 1;

```



```

49             expaux:= subsop(j = vet[nops(vet)]+1, expaux
50                 );
51             break;
52         end if;
53     elif (evalb(op([j, 1], expaux)=vet[i])) then
54         expaux2:= expaux2 .~ op(j,expaux);
55         expaux:= subsop(j = vet[nops(vet)]+1, expaux);
56         break;
57     end if;
58 end do;
59 return sinal .~ expaux2;
60 end proc;

```

┘

Neste procedimento é realizado uma cópia do parâmetro de entrada em *expaux* e a leitura do sinal e dos índices de cada parcela do produto *expr*, os quais são armazenados nas variáveis *sinal* e em uma lista *vet*, respectivamente. Esta última é organizada em ordem crescente. Note que esse procedimento considera potências de elementos indexados. Na ordenação de *vet*, para cada mudança de posição de um elemento dessa lista é somado 1 a variável *cont* a qual tem o importante papel referente ao *sinal* da expressão final (1 se *cont* for par e -1 se *cont* for ímpar). A variável *expaux2* armazena o produto de saída, isto é, com índices já ordenados. No caso de polinômios não-lineares (alguma variável aparece mais de uma vez) é necessário algumas mudanças, as quais fizemos em um outro procedimento (A.0.2).

Como exemplo de aplicação da função, utilizamos o procedimento A.0.1 em cada termo da expressão obtida pelo *comutador*(*f*) (Nesse caso, os dois procedimentos devem ser executados na mesma folha de trabalho) da seguinte forma

$$f := [x[1], x[2], x[3]]:$$

$$f := \text{comutador}(f) :$$

$$f1 := \text{ordenaE1}(x_1.x_2.x_3)$$

$$x_1.x_2.x_3$$

$$f2 := \text{ordenaE1}(x_2.x_1.x_3)$$

$$-x_1.x_2.x_3$$

$$f3 := \text{ordenaE1}(x_3.x_1.x_2)$$

$$x_1.x_2.x_3$$

$$f4 := ordenaE1(x_3.x_2.x_1)$$

$$-x_1.x_2.x_3$$

$$f1 - f2 - f3 + f4$$

$$0$$

Ou simplesmente

aux := 0 :

for *i* to nops(*f*) do *aux*:= *aux*+ordenaE1(op(*i*,*f*)); end do:

aux;

$$0$$

Isso é suficiente para mostrar que $f(x_1, x_2, x_3) = [[x_1, x_2], x_3] = 0, \forall x_i \in E$, logo E é uma PI-álgebra. Note que nesse procedimento não acrescentamos a regra $x_i^2 = 0, \forall x_i \in E_1$. Isso se faz necessário quando tomamos polinômios que não são multilineares, por exemplo, se a identidade de Hall fosse para a álgebra das matrizes de ordem 2×2 com entradas E .

4.2 A identidade de Hall $[[x_1, x_2]^2, x_3]$

Agora, vamos verificar que o polinômio do exemplo 2.3.4 é uma identidade polinomial para as matrizes quadradas de ordem 2 sobre K .

Vimos que a identidade de Hall é dada por $f(x_1, x_2, x_3) = [[x_1, x_2]^2, x_3]$. Para calcular $[x_1, x_2]^2$ basta combinarmos os procedimentos 4.1.1 e 3.0.1 da seguinte forma

for *i* to 3 do *x*[*i*] := Matrix([[*a*[*i*], *b*[*i*]], [*c*[*i*], *d*[*i*]]]); end do;

$$\begin{bmatrix} a_1 & b_1 \\ c_1 & d_1 \end{bmatrix}, \begin{bmatrix} a_2 & b_2 \\ c_2 & d_2 \end{bmatrix}, \begin{bmatrix} a_3 & b_3 \\ c_3 & d_3 \end{bmatrix}$$

f:= comutador([*x*[1],*x*[2]]):

f:= distribui(*f*,*f*): # ou apenas *f*:= *f*.*f*:

f:= comutador([*f*,*x*[3]]):

`simplify(%);`

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Como as entradas das matrizes são elementos de um corpo (logo comutam entre si) o Maple nos dá um resultado direto apenas com os procedimentos construídos até aqui. Nesse caso, o comando `simplify` faz as simplificações necessárias de acordo as propriedades do corpo (associatividade, distributividade, comutatividade, etc.) e mostra que f é, de fato, uma identidade polinomial de $M_2(K)$. O tempo médio foi de 1.48 segundos. De forma análoga, usando a função `Commutator` obtemos o resultado após 1.65 segundos. Mas, esta função seria interessante caso as entradas das matrizes fossem não-comutantes, como veremos no próximo capítulo.

4.3 O Teorema de Amitsur e Levitzki

Nessa seção, dentre as várias demonstrações existentes do teorema de Amitsur e Levitzki, apresentamos a de Rosset de 1976 por sua simplicidade e utilidade devido ao uso de conceitos, técnicas e resultados interessantes para a teoria das PI-álgebras. Ao final da seção, verificamos o teorema no maple para os casos $n = 1, 2, 3, 4, 5$. Quando $n > 5$, tornou-se inviável visto que o tempo de processamento é muito grande devido ao polinômio standard ser formado por uma soma de $(2n)!$ produtos de $2n$ matrizes simbólicas de ordem n .

Para tanto, necessitamos de alguns resultados importantes para prova de tal teorema, os quais enunciamos a seguir e ocultamos algumas das demonstrações visto apresentarem conceitos além dos disponíveis no presente texto. Sugerimos a leitura de (GALVÃO, 2003).

Lema 4.3.1. *Seja K um corpo qualquer. Se st_{2n} é uma identidade polinomial de $M_n(\mathbb{Q})$ então também é de $M_n(K)$.*

Demonstração: Como st_{2n} é multilinear, basta mostrar que st_{2n} se anula quando calculado sobre matrizes pertencentes a uma base de $M_n(K)$. Escolhemos a base usual de $M_n(K)$ formada pelas matrizes e_{ij} . Então podemos considerar somente as matrizes de $M_n(K_0)$ onde K_0 é o subcorpo primo de K . Se a característica de K for zero o corpo primo de K é isomorfo a \mathbb{Q} e se a característica de K for um número primo p , tal corpo primo será isomorfo a \mathbb{Z}_p . Como \mathbb{Z}_p é um quociente de \mathbb{Z} e $\mathbb{Z} \subset \mathbb{Q}$, basta considerar K de característica zero, e $K = \mathbb{Q}$. Sejam $r_p = \sum_{i,j=1}^n \alpha_{ij}^{(p)} e_{ij}$ onde $\alpha_{ij}^{(p)} \in K$ e $p = 1, \dots, 2n$, matrizes em $M_n(K)$. Temos

$$st_{2n}(r_1, \dots, r_{2n}) = \sum_{\sigma \in S_{2n}} (-1)^\sigma r_{\sigma(1)} \cdots r_{\sigma(2n)} = \sum_{\sigma \in S_{2n}} (-1)^\sigma \left(\sum_{i,j} \alpha_{ij}^{(\sigma(1))} e_{ij} \right) \cdots \left(\sum_{i,j} \alpha_{ij}^{(\sigma(2n))} e_{ij} \right)$$

Mas veja que

$$e_{ij}e_{nl} = \begin{cases} e_{il}, & \text{se } j = n \\ 0, & \text{se } j \neq n \end{cases}$$

Logo $st_{2n}(r_1, \dots, r_{2n})$ é uma combinação linear de $st_{2n}(e_{i_1j_1}, \dots, e_{i_{2n}j_{2n}})$ que é zero pois, por hipótese, st_{2n} é uma identidade de $M_n(\mathbb{Z}) \subset M_n(\mathbb{Q})$. ■

Lema 4.3.2. *Sejam K um corpo, n e l dois números naturais e r_1, \dots, r_l matrizes de $M_n(K)$. Se l é par então $tr(st_l(r_1, \dots, r_l)) = 0$.*

Demonstração: Se r e s são matrizes de $M_n(K)$, é fácil ver que $tr(rs - sr) = 0$. Seja $\sigma \in S_l$, onde S_l é o grupo das permutações de l elementos, considere $\tau = \sigma \circ (123 \dots l) \in S_l$. As paridades de σ e τ são distintas pois como l é par, $(123 \dots l)$ é ímpar. Fazemos então $r = r_{\sigma(1)}$ e $s = r_{\sigma(2)} \dots r_{\sigma(l)}$. Logo, $sr = r_{\sigma(2)} \dots r_{\sigma(l)}r_{\sigma(1)}$ e como $\tau(1) = [\sigma \circ (123 \dots l)](1) = \sigma(2), \tau(2) = [\sigma \circ (123 \dots l)](2) = \sigma(3), \dots, \tau(l-1) = \sigma(l)$, temos $sr = r_{\sigma(2)} \dots r_{\sigma(l)}r_{\sigma(1)} = r_{\tau(1)} \dots r_{\tau(l-1)}r_{\tau(l)}$. Assim $tr(sgn(\sigma)r_{\sigma(1)} \dots r_{\sigma(l)} + sgn(\tau)r_{\tau(1)} \dots r_{\tau(l)}) = tr(sign(\tau)rs + sign(\sigma)sr) = 0$ pois $sgn(\sigma) = -sgn(\tau)$ (Lembrando que $sgn(\sigma)$ é o sinal da permutação $\sigma \in S_l$). Desta forma $tr(st_l(r_1, \dots, r_l)) = 0$. ■

Lema 4.3.3. *(Fórmulas de Newton) Sejam K um corpo e, n e k números naturais. Então se $k \leq n$*

$$(p_k) - (p_{k-1})(e_1) + (p_{k-2})(e_2) + \dots + (-1)^{k-1}(p_1)(e_{k-1}) + (-1)^k k(e_k) = 0$$

e se $k \geq n$ então

$$(p_k) - (p_{k-1})(e_1) + (p_{k-2})(e_2) + \dots + (-1)^{n-1}(p_{k-n+1})(e_{n-1}) + (-1)^n(p_{k-n})(e_n) = 0$$

Demonstração: Ver (GALVÃO, 2003) pag. 42.

Lema 4.3.4. *Sejam K um corpo de característica zero e B uma álgebra associativa e comutativa sobre K . Se $a \in M_n(B)$ e $tr(a^k) = 0$ para todo $k = 1, 2, \dots, n$, então $a^n = 0$*

Demonstração: Seja $a \in M_n(B)$ uma matriz com entradas $a_{ij} \in B$. Considere a álgebra comutativa $K[a_{ij} | i, j = 1, \dots, n]$, ela é imagem homomorfa da álgebra dos polinômios comutativos $K[x_1, \dots, x_{n^2}]$. Mas esta última álgebra é um subanel do corpo das funções racionais $K(x_1, \dots, x_{n^2}) = L$. Se o lema for verdadeiro para matrizes com entradas no corpo L então ele será válido para matrizes com entradas na álgebra B . Assim basta demonstrar o lema somente no caso onde $B = L$ é um corpo de característica 0. Sejam então $\lambda_1, \dots, \lambda_n$ as raízes características de a no fecho algébrico de L . Logo, o polinômio característico de a é

$$p(x) = \det(xI - a) = x^n - e_1(\lambda_1, \dots, \lambda_n)x^{n-1} + e_2(\lambda_1, \dots, \lambda_n)x^{n-2} + \dots \\ \dots + (-1)^{n-1}e_{n-1}(\lambda_1, \dots, \lambda_n)x + (-1)^n e_n(\lambda_1, \dots, \lambda_n).$$

Como as raízes de a^k são $\lambda_1^k, \dots, \lambda_n^k$ e $\text{tr}(a^k) = 0$ para todo $k = 1, 2, \dots, n$, temos a igualdade $p_k(x_1, \dots, x_n) = 0$. Pelas fórmulas de Newton (Lema 4.3.3) e pelo fato da característica de K ser zero, obtemos que $e_i(\lambda_1, \dots, \lambda_n) = 0$ para todo $i \in 1, 2, \dots, n$. Assim $p(x) = x^n$ e pelo teorema de Cayley-Hamilton, temos $a^n = 0$. ■

De posse desses resultados, podemos provar o teorema a seguir, o qual aparece em diversas aplicações da PI-Teoria em áreas correlatas à Matemática, em especial na área de física quântica (ver (JAMIOŁKOWSKI, 2013; MUNDIN; MUNDIN, 1997; BENTIN; ALVES, 2012)).

Teorema 4.3.5. *A álgebra $M_n(K)$ satisfaz o polinômio standard de grau $2n$*

$$st_{2n}(x_1, \dots, x_{2n}) = \sum_{\sigma \in S_{2n}} (-1)^\sigma x_{\sigma(1)}, \dots, x_{\sigma(2n)}$$

onde S_{2n} é o grupo das permutações de $\{1, 2, \dots, 2n\}$ e $(-1)^\sigma$ é o sinal da permutação σ .

Demonstração: Pelo lema 4.3.1 podemos considerar $K = \mathbb{Q}$. Seja E a álgebra de Grassmann de um espaço vetorial sobre K gerado por e_1, e_2, \dots, e_{2n} e seja $E = E_0 \oplus E_1$ a \mathbb{Z}_2 -gradação de E . Temos que E_0 é uma subálgebra comutativa de E . Sejam então r_1, r_2, \dots, r_{2n} matrizes em $M_n(\mathbb{Q})$ e teremos que

$$b = r_1 e_1 + \dots + r_{2n} e_{2n} \tag{6}$$

é uma matriz $n \times n$ com entradas na álgebra de Grassmann que é não comutativa. Mas, visto que $e_i e_j = -e_j e_i$ temos

$$a = b^2 = \sum_{i=1}^{2n} \sum_{j=1}^{2n} r_i r_j e_i e_j = \sum_{i < j} (r_i r_j - r_j r_i) e_i e_j. \tag{7}$$

Logo, a é uma matriz com entradas da álgebra E_0 que é comutativa. Veja então que

$$a^q = (b^2)^q = \sum_{1 \leq i_1 < \dots < i_{2q} \leq 2n} st_q(r_{i_1}, \dots, r_{i_{2q}}) e_{i_1} \dots e_{i_{2q}}. \tag{8}$$

para todo q natural. Assim, usando propriedades do traço de matrizes, obtemos

$$\text{tr}(a^q) = \sum_{1 \leq i_1 < \dots < i_{2q} \leq 2n} \text{tr}(st_{2q}(r_{i_1}, \dots, r_{i_{2q}})) e_{i_1} \dots e_{i_{2q}}. \tag{9}$$

Pelo lema 4.3.2, $\text{tr}(st_{2q}(r_{i_1}, \dots, r_{i_{2q}})) = 0$ para todos r_1, \dots, r_{2q} em $M_n(\mathbb{Q})$ e pelo lema 4.3.4

$$a^n = st_{2n}(r_1, \dots, r_{2n}) e_1 \dots e_{2n} = 0. \tag{10}$$

Mas $e_1 \cdots e_{2n} \neq 0$ pois é um elemento da base de E . Portanto, $st_{2n}(r_1, \dots, r_{2n}) = 0$.



Para verificar o teorema 4.3.5 no maple, usaremos o seguinte procedimento

Procedimento 4.3.1. *Função para calcular o polinômio standard de $M_n(K)$.*

Entrada: Uma lista de matrizes $L := [x_1, x_2, \dots, x_{2n}]$ com $n > 1$.

Saída: Uma matriz (polinômio standard de grau n).

```

1 Stn := proc (L::list)
2     local i, j, Perm, pol, standn, sinal;
3     uses combinat, group;          # // Libera as funcoes numbperm
4     Perm := permute([seq(i, i = 1 .. nops(L))], nops(L));
5     standn := 0;
6     for i to nops(Perm) do
7         pol := L[Perm[i][1]].L[Perm[i][2]];
8         for j from 3 to nops(Perm[i]) do
9             pol := pol.L[Perm[i][j]];
10        end do;
11        sinal := parity(convert(Perm[i], 'disjyc')); # //
12        pol := sinal*pol; # // parity retorna 1 se a
13        standn := standn + pol;
14    end do;
15    return standn;
16 end proc;
```

┘

Por padrão, o Maple considera as entradas das matrizes sobre um corpo de característica zero (\mathbb{R} ou \mathbb{C}). Note que no teorema supracitado aparece o grupo simétrico sobre $\{1, 2, \dots, 2n\}$, bem como o sinal de uma dada permutação. O maple disponibiliza pacotes com algumas funções pré-implementadas para trabalhar com esse tipo de problema, a saber o pacote *group* e *combinat*, onde podemos, dentre outras coisas, calcular e exibir todas as permutações de um dado conjunto finito (no formato de lista) com a função *permute*, e obter o sinal de cada permutação com a função *parity*. Para obter o sinal de uma permutação, é necessário reescrevê-la em ciclos disjuntos, o que é feito usando a função *disjyc* como parâmetro da função *convert*.

Como o polinômio standard, nesse caso, é formado por uma soma de produtos de matrizes com entradas em um corpo (logo as entradas podem comutar) podemos utilizar simplesmente o

operador *dot* para calcular tais produtos de forma recursiva. A partir daí, é utilizado a função *simplify* para simplificar o máximo possível as expressões obtidas seguindo as propriedades de um corpo.

Segue alguns exemplos com os comandos necessários de entrada dos dados no Maple e as matrizes de saída.

```
n:=[1,2,3,4,5]:
for i to nops(n) do
  for k to 2*n[i] do
    x[k]:= M[k]:=Matrix(n[i], n[i], (i, j) -> a[k][i,j]):
  end do:
  L := [seq(M[k], k = 1 .. 2*n[i])]:
  A[i]:= Stn(L);
end do;
```

$$\begin{bmatrix} 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

O que, de fato, mostra valer o Teorema 4.3.5 para valores particulares $n = 1, 2, 3, 4, 5$. Como supracitado, quando $n > 5$ nos deparamos com problemas computacionais relativo ao tempo de processamento para obtenção do polinômio standard. Por conta disso fizemos algumas tentativas de paralelização do procedimento 4.3.1.

O próprio Maple disponibiliza algumas ferramentas para isso, com as funções do pacote *Threads*, podendo ser feita a programação paralela de duas formas diferentes: o modelo de programação de tarefas (*Task Programming Model*) e o modelo de programação em rede *Grid Programming*. O modelo de programação de tarefas é um modelo de programação de alto nível, concebido para tornar a programação *multithreaded* mais fácil. Neste modelo, os usuários criam tarefas em vez de *threads*. Uma tarefa é uma chamada de função (um procedimento com argumentos) que pode ser executado em um segmento. O Maple programa automaticamente as tarefas para threads. Isso permite que o código escrito usando o modelo de programação de tarefas seja redimensionado para computadores com um grande número de processadores.

Aparentemente esse problema parecia fácil, uma vez que o polinômio standard é obtido a partir de uma soma de parcelas independentes, ou seja, pode ser paralelizado de forma simples. Porém, tivemos dificuldades para garantir o acesso seguro à memória pelas multitarefas criadas e, por isso, foram necessários alguns cuidados usando as funções *Mutex:-Lock(m)* e *Mutex:-Unlock(m)* que permitem o bloquear e desbloquear o acesso a memória, isto é, duas tarefas quaisquer não pode obter o mesmo dado alocado provisoriamente na memória. Mas, ao fazer esse controle, nosso procedimento perdeu eficiência, tendo um tempo de processamento um pouco maior do que o normal, mesmo contando com um computador de 8 CPU's.

Porém, ao utilizar os procedimentos de forma paralelizado para verificar o teorema de Amitsur-Levitzki, conseguimos os mesmos resultados supracitados com um tempo de processamento maior. Por isso, por falta de domínio na área de programação paralela, nos atentamos a resolver problemas específicos (casos particulares) de PI-álgebras usando apenas os procedimentos até então apresentados no presente texto e em seu anexo. Um desses casos particulares é apresentado no capítulo a seguir.

5 Identidades Polinomiais para $M_{1,1}(E)$

Neste capítulo, apresentamos uma verificação no maple das identidades polinomiais fracas encontradas em (VINCENZO; SCALA, 2005), bem como as identidades polinomiais de $M_{1,1}(E)$ encontradas em (POPOV, 1982).

Aqui K denotará um corpo infinito de característica diferente de 2 e $K\langle X \rangle$ será uma álgebra associativa livre gerada pelo conjunto enumerável $X = \{x_1, x_2, x_3, \dots\}$. Denotamos por B a subálgebra de $K\langle X \rangle$ gerada por todos os comutadores $[x_{i_1}, x_{i_2}, \dots, x_{i_l}]$ de comprimento $l \geq 2$.

Definição 5.0.1. *Seja R uma álgebra associativa e $W \subset R$ um espaço vetorial sobre K . Dizemos que o polinômio $f(x_1, x_2, \dots, x_n) \in K\langle X \rangle$ é uma identidade polinomial fraca para o par (R, W) se $f(w_1, w_2, \dots, w_n) = 0$, para todos os elementos $w_1, w_2, \dots, w_n \in W$.*

O conjunto de todas identidades polinomiais fracas $I_w = I(R, W)$ é um ideal de $K\langle X \rangle$.

Em particular, se $E = E_0 \oplus E_1$ é a álgebra de Grassmann como no exemplo 2.2.2 definamos R da seguinte forma:

$$R = M_{1,1}(E) = \left\{ \begin{pmatrix} a & b \\ c & d \end{pmatrix} \mid a, d \in E_0, b, c \in E_1 \right\}.$$

Além disso, definamos o *supertraço* de A como $str(A) = a - d$, para todos elementos $A \in R$ e tomamos W como o espaço vetorial dado por todas as matrizes de R com supertraço igual a zero, isto é, $W = \{A \in R \mid str(A) = 0\}$.

Para avaliar passo a passo a verificação de que um dado polinômio seja uma identidade, utilizamos cada propriedade da álgebra de Grassmann separadamente divididas nos procedimentos do quadro 2

Quadro 2 – Procedimentos com as propriedades da Álgebra de Grassmann

Nomes das funções	Atribuições de Propriedades
<i>ordenaE0</i>	Atribui a propriedade $ax = xa$, $\forall a \in E_0$ e $x \in E$, ou seja, os elementos de E_0 comutam com todos elementos de E .
<i>ordenaE1</i>	Além da propriedade satisfeita por <i>ordenaE0</i> , atribui-se a propriedade de que $b_i c_j = -c_j b_i$, para quaisquer $b_i, c_j \in E_1$, ou seja, os elementos de E_1 anticomutam entre si.
<i>ordenaEIEI</i>	Além das propriedades satisfeitas por <i>ordenaE1</i> , é atribuída a propriedade $b_i^2 = c_j^2 = 0$, para quaisquer $b_i, c_j \in E_1$

Fonte: Autor.

Em (VINCENZO; SCALA, 2005), os autores mostram que $c := [x_1, x_2, x_3]$ e o polinômio próprio $p := [x_2, x_1][x_3, x_1][x_4, x_1]$ são identidades polinomiais fracas para a álgebra R . Mais ainda, é provado, nessas condições, que c e p geram o ideal I_w . O fato de c ser identidade segue de que $str(A) = 0$ (então $a = d$), logo o comutador $[x_1, x_2]$ pertence ao centro de R . Apesar de ser um cálculo simples, podendo ser feito no ambiente papel e lápis, verificamos essa identidade no maple como segue:

```

h := comutador([x[1], x[2], x[3]]): i:=nops(h):

for k to i do

    if op(1, op(k, h)) = -1 then p[k]:= -op(k, h); s[k]:=-1; else p[k]:= op(k, h); s[k]:=1;
end if;

    p[k] := convert(p[k], list);

end do:

for k to 3 do x[k]:= Matrix([[a[k], b[k]], [c[k], a[k]]]); end do:

c:=add(s[k]*ordenamE1(prodn(p[k])),k=1..i);

```

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Como o procedimento 4.1.1 retorna um polinômio com sinais alternados, armazenamos cada monômio e seu sinal nas variáveis $p[k]$ e $s[k]$, respectivamente, onde $k = 1, \dots, i$ e i corresponde a quantidade de termos do polinômio. Isso foi feito para utilizarmos a função *convert* que transforma cada monômio em uma lista a fim de utilizarmos os procedimentos A.0.7 e A.0.4. O resultado, atribuído a variável c , foi obtido com uso da função *add* que calcula cumulativamente uma soma de termos. Portanto, verificamos que $c \in I_w$.

De forma análoga, verificamos o polinômio próprio $p = [x_2, x_1][x_3, x_1][x_4, x_1]$, como segue

```

h[1]:= comutador([x[2], x[1]]): h[2]:= comutador([x[3], x[1]]): h[3]:= comutador([x[4],
x[1]]):

p=distribui(distribui(h[1],h[2]),h[3]); contp:= nops(p):

```

$$x_2 \cdot x_1 \cdot x_3 \cdot x_1 \cdot x_4 \cdot x_1 - x_2 \cdot x_1 \cdot x_3 \cdot (x_1^2) \cdot x_4 - x_2 \cdot (x_1^2) \cdot x_3 \cdot x_4 \cdot x_1 + x_2 \cdot (x_1^2) \cdot x_3 \cdot x_1 \cdot x_4 - x_1 \cdot x_2 \cdot x_3 \cdot x_1 \cdot x_4 \cdot x_1 + x_1 \cdot x_2 \cdot x_3 \cdot (x_1^2) \cdot x_4 + x_1 \cdot x_2 \cdot x_1 \cdot x_3 \cdot x_4 \cdot x_1 - x_1 \cdot x_2 \cdot x_1 \cdot x_3 \cdot x_1 \cdot x_4$$

```

for k to contp do

```

```

v[k] := [];

if op(1, op(k, p)) = -1 then h:= -op(k, p); s[k]:=-1; else h:= op(k, p); s[k]:=1; end if;

for j while j <= nops(h) do

    if type(op(j, h), `^`) then v[k] := [op(v[k]), op([j, 1], h), op([j, 1], h)]; else v[k] :=
[op(v[k]), op(j, h)]; end if;

end do;

end do;

for k to 4 do x[k]:= Matrix([[a[k], b[k]], [c[k], a[k]]]); end do;

p:=add(s[k]*ordenamEI(prodn(v[k])),k=1..contp);

```

$$\begin{bmatrix} -b_1.(c_1^2).b_2.c_3.b_4 - b_1^2.c_1.c_2.b_3.c_4 + b_1^3.c_2.c_3.c_4 + \\ +b_1.(c_1^2).b_2.b_3.c_4 + b_1^2.c_1.b_2.c_3.c_4 + b_1^2.c_1.c_2.c_3.b_4 + \\ +b_1.(c_1^2).c_2.b_3.b_4 + c_1^3.b_2.b_3.b_4 & 0 \\ 0 & -b_1.(c_1^2).b_2.c_3.b_4 - b_1^2.c_1.c_2.b_3.c_4 + b_1^3.c_2.c_3.c_4 + \\ +b_1.(c_1^2).b_2.b_3.c_4 + b_1^2.c_1.b_2.c_3.c_4 + b_1^2.c_1.c_2.c_3.b_4 + \\ +b_1.(c_1^2).c_2.b_3.b_4 + c_1^3.b_2.b_3.b_4 \end{bmatrix}$$

```

p:=add(s[k]*ordenamEIEI(prodn(v[k])),k=1..contp);

```

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Note que usamos alguns passos a mais do que a verificação anterior visto que o polinômio p apresenta termos com fatores x_1^2 , sendo preciso duplicar este fator ao converter em listas. Vale lembrar que, ao convertermos as variáveis em matrizes, o Maple calcula x_1^2 na forma usual (comutando o produto entre as entradas de x_1), o que não nos é interessante. Além disso, os procedimentos utilizados tomaram apenas o fato de que $a_i \in E_0$ e que vale $b_i c_j = -c_j b_i, \forall i, j$, obtendo o resultado da matriz acima. Mas, sabemos que se $b_i, c_j \in E_1$ então $b_i^2 = c_j^2 = 0, \forall i, j$, o que nos permite concluir que $p \in I_w$.

Com os mesmos procedimentos construídos até aqui, podemos verificar que os polinômios $f := [[x_1, x_2]^2, x_2]$ e $g := [x_1, x_2, [x_3, x_4], x_5]$ são identidades para R . De forma geral, é provado em (POPOV, 1982) que f e g geram o ideal $I(R)$ quando $\text{char}K = 0$. Um pequeno detalhe, é que o procedimento A.0.1 faz apenas uma organização nos índices, colocando-os em ordem crescente. Porém, não é levado em consideração a ordem lexicográfica para os b_i e c_i , $i \in \mathbb{N}$. Por exemplo, usando a função *ordena* no monômio $c_2.c_1.b_3.b_1.c_1$, obtemos o monômio $-c_1.b_1.c_1.c_2.b_3$. Mas, tal produto pode ser reescrito, de forma mais simplificada, como sendo $b_1.c_1^2.c_2.b_3 = 0$. Isso acontece sempre que tratamos de polinômios não-lineares. Dessa forma,

foi necessário, além de ordenar os índices como no procedimento A.0.1, criar um novo procedimento, denotado por *ordenaE1E1*, capaz de resolver esse impasse. Para isso, bastou tomar listas v_i formada por termos $b_i = 1$ e $c_i = 2$ para cada índice i que aparece em cada monômio (Nesse caso, o monômio deve conter apenas termos b_i e c_i , $i \in \mathbb{N}$). Do monômio $-c_1.b_1.c_1.c_2.b_3$ obtemos $v_1 = [2, 1, 2]$, $v_2 = [2]$, $v_3 = [1]$. Ao ordenar cada v_i , e avaliar seus sinais e expoentes, garantimos a escrita do monômio na forma mais simplificada possível.

Uma verificação da identidade f no maple é seguida dos comandos abaixo

```
w[1]:= comutador([x[1], x[2]]): w[2]:= [distribui(w[1],w[1]),x[2]:
```

```
f:= comutador(w[2]); contf:= nops(f):
```

$$x_1.x_2.x_1.(x_2^2) - x_2.(x_1^2).(x_2^2) - x_1.(x_2^2).x_1.x_2 + x_2^2.(x_1^2).x_2 + x_2.x_1.(x_2^2).x_1 - x_2^2.x_1.x_2.x_1$$

```
for k to contf do
```

```
  v[k] := [];
```

```
  if op(1, op(k, f)) = -1 then h:= -op(k, f); s[k]:=-1; else h:= op(k, f); s[k]:=1; end if;
```

```
  for j while j <= nops(h) do
```

```
    if type(op(j, h), `^`) then v[k] := [op(v[k]), op([j, 1], h), op([j, 1], h)]; else v[k] := [op(v[k]), op(j, h)]; end if;
```

```
  end do:
```

```
end do:
```

```
for k to 2 do x[k]:= Matrix([[a[k], b[k]], [c[k], a[k]]]); end do:
```

```
f:=add(s[k]*ordenamE1E1(prodn(v[k])),k=1..contf);
```

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

De forma análoga, calculamos g como segue

```
w[1]:= comutador([x[3], x[4]]): w[2]:= [x[1],x[2],w[1],x[5]:
```

```
g:= comutador(w[2]); contg:= nops(g):
```

$$x_1.x_2.x_1.(x_2^2) - x_2.(x_1^2).(x_2^2) - x_1.(x_2^2).x_1.x_2 + x_2^2.(x_1^2).x_2 + x_2.x_1.(x_2^2).x_1 - x_2^2.x_1.x_2.x_1$$

```

for k to contg do
    v[k] := [];
    if op(1, op(k, g)) = -1 then h:= -op(k, g); s[k]:=-1; else h:= op(k, g); s[k]:=1; end if;
    for j while j <= nops(h) do
        if type(op(j, h), `^`) then v[k] := [op(v[k]), op([j, 1], h), op([j, 1], h)]; else v[k] :=
[op(v[k]), op(j, h)]; end if;
    end do;
end do;
for k to 5 do x[k]:= Matrix([[a[k], b[k]], [c[k], a[k]]]); end do;
f:=add(s[k]*ordenamE1E1(prodn(v[k])),k=1..contg);

```

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Portanto, verificamos que f e g são, de fato, identidades polinomiais de R .

6 Grau mínimo do polinômio standard para $M_{k,l}(E)$

Nesse capítulo apresentamos uma conjectura de Kemer (ver (KEMER, 2002) pág 860) a cerca do grau mínimo do polinômio standard para matrizes quadradas de ordem n e entradas na álgebra de Grassmann. Antes disso, trataremos das questões propostas por (ALVES et al., 2015) a cerca das identidades standard de $M_n(E)$, onde E denota a álgebra de Grassmann sobre um corpo K de característica diferente de 2.

6.1 O Polinômio Standard para $M_{k,l}(E)$

Em (ALVES et al., 2015) é dada uma condição necessária para que o polinômio standard de grau m seja uma identidade polinomial de $M_n(E)$.

No artigo supracitado são demonstrados os seguintes resultados:

Lema 6.1.1. *Seja $\text{char} K = p$. O polinômio standard St_m é uma identidade polinomial de E se, e somente se, $m \geq p + 1$.*

Lema 6.1.2. *Se a álgebra A satisfaz o polinômio standard St_{2m} então $M_n(A)$ satisfaz o polinômio standard $St_{2mn^2-n^2+1}$.*

Teorema 6.1.3. *Seja St_m uma identidade polinomial para a álgebra $M_n(E)$. Então $m \geq 2n + p - 1$.*

Demonstração. Note que se $n = 1$ o resultado segue do lema 6.1.1, visto que $p + 1 \leq 2 \cdot 1 + p - 1 = p + 1$. Então, suponhamos que $n > 1$ e que St_m é uma identidade polinomial da álgebra $M_n(E)$. Podemos escrever $St_m(x_1, \dots, x_m) = St_{m-2}(x_1, \dots, x_{m-2})x_{m-1}x_m + h(x_1, \dots, x_m)$. Podemos supor que $St_{m-2}(x_1, \dots, x_{m-2}) \notin T(M_n(E))$. Agora, mostraremos que $St_{m-2}(x_1, \dots, x_{m-2}) \notin T(M_{n-1}(E))$. Como St_{m-2} é multilinear é suficiente mostrar que $St_{m-2}(a_1e_1, \dots, a_{m-2}e_{m-2}) = 0$ para cada substituição de $a_i \in E$ e e_i matrizes quadradas unitárias de ordem $n - 1$. As matrizes e_i também serão consideradas matrizes quadradas e'_i de ordem n , adicionando a n -ésima linha e n -ésima coluna de zeros.

Como St_m é uma identidade polinomial de $M_n(E)$, temos que

$$\begin{aligned} 0 &= St_{m-2}(a_1e'_1, \dots, a_{m-2}e'_{m-2}, e_{ln}, e_{nn}) \\ &= St_{m-2}(a_1e'_1, \dots, a_{m-2}e'_{m-2})e_{ln}e_{nn} + h(a_1e'_1, \dots, a_{m-2}e'_{m-2}, e_{ln}, e_{nn}), \end{aligned} \quad (11)$$

onde $1 \leq l < n$ são arbitrários e e_{ij} são matrizes quadradas de ordem n .

Como em cada monômio de $h(x_1, \dots, x_m)$ pelo menos uma das variáveis x_1, \dots, x_{m-2} aparece à direita de x_{m-1} ou x_m temos que $h(a_1e'_1, \dots, a_{m-2}e'_{m-2}, e_{ln}, e_{nn}) = 0$.

Assim, da equação (11), concluímos que $St_{m-2}(x_1, \dots, x_{m-2}) \notin T(M_{n-1}(E))$. Logo, isso

implica que $St_{m-2(n-1)} \in T(E)$. Pelo lema 6.1.1, temos que $m - 2(n - 1) \leq p + 1$, isto é, $m \leq 2n + p + 1$, como queríamos demonstrar. ■

Pelo lema 6.1.1, temos que E satisfaz a identidade standard St_{p+1} , então pelo lema 6.1.2 $M_n(E)$ satisfaz $St_{(p+1)n^2-n^2+1} = St_{pn^2+1}$.

Dessa forma, questiona-se a existência de alguma identidade polinomial standard para $M_n(E)$ de grau m tal que $m \in [2n + p - 1, pn^2 + 1]$. Em caso afirmativo, a busca pelo grau mínimo passa a ser interessante.

Em (KEMER, 2002) encontramos a seguinte conjectura: Seja d_n o menor número m tal que a álgebra $M_n(E)$ satisfaz a identidade standard de grau m . Então $d_n = (2n - 1)p + 1$.

Uma vez que $M_{k,l}(E) \subseteq M_n(E)$, então toda identidade polinomial de $M_n(E)$ é também uma identidade de $M_{k,l}(E)$, isto é, vale a inclusão $T(M_n(E)) \subseteq T(M_{k,l}(E))$.

Nos fazemos os mesmos questionamentos, porém, para a álgebra graduada definida em 2.2.5. Relembrando, verificamos se a conjectura é válida para a álgebra

$$M_{k,l}(E) = \left\{ \begin{pmatrix} A & B \\ C & D \end{pmatrix} \mid A \in M_k(E_0), B \in M_{k \times l}(E_1), C \in M_{l \times k}(E_1), D \in M_l(E_0) \right\}.$$

Criamos alguns procedimentos no Maple para verificar a conjectura supracitada. É claro que o polinômio standard de grau $m < 4$ não é uma identidade para R , mas também as calculamos a fim de comparar os resultados obtidos com uso dos procedimentos construídos e das funções já implementadas do pacote *Physics*. Para este, bastou usar o procedimento 4.3.1 combinado com o *setup* (ver seção 3), de forma que se defina as entradas das matrizes b e c como anti-comutativas. Assim, usamos os comandos abaixo:

```
n:= [seq(i,i=2..9)];
for i to nops(n) do
  for k to n[i] do
    M[k]:=Matrix(n[i], n[i], [[a[k], b[k],[c[k], d[k]]]);
  end do;
L := [seq(M[k], k = 1 .. n[i])];
tempo[i]:= time[real]();
A[i]:= Stn(L);
tempo[i]:= time[real]()-tempo[i];
end do;
```

Dessa forma conseguimos os resultados da seguinte tabela

Tabela 2 – Tempo (em segundos) para calcular o polinômio standard de grau n

n	Procedimentos próprios	Pacote <i>Physics</i>
2	0.128	0.228
3	0.192	0.406
4	2.594	4.067
5	34.910	34.696
6	479.861	640.708
7	1023.358	
8	21974.312	
9	529054.659	

Note que a partir de $n = 6$, temos uma diferença considerável entre os procedimentos que construímos com os resultados obtidos com o pacote *Physics*. Como o processo para encontrar o polinômio standard torna-se muito lento devido as propriedades do grupo S_n , para $6 < n < 10$, não utilizamos as funções do pacote supracitado, o que justifica os espaços vazios da tabela 2.

Veja que acumulamos cada polinômio standard de grau i na variável $A[i]$. Resta utilizar a característica do corpo nos polinômios para verificar se é, ou não, uma identidade de R . Testamos inicialmente para $\text{char}K = p = 3$ usando a função *mod* do Maple, a qual avalia uma expressão sobre os inteiros módulo p . Por exemplo, $5 \text{ mod } 3 = 2$ e $(-7x + 4y) \text{ mod } 5 = 3x + 4y$. Dessa forma, basta fazer $A[i] \text{ mod } 3$ para cada $i = 2, \dots, 9$. Com isso, tivemos como resultado que o menor grau m tal que St_m seja uma identidade polinomial para $M_{1,1}(E)$ é $m = 6$.

De fato, analisando todos os coeficientes de cada entrada de St_6 , (podemos usar a função *coeffs*) obtemos

```
coeficientesA1:= {};
```

```
for i to 2 do;
```

```
  for j to 2 do;
```

```
    coeficientesA1:= {op(coeficientesA1), coeffs(A[1][i, j]);}
```

```
  end do;
```

```
end do;
```

```
coeficientesA1;
```

```
{-36, 36}
```

```
ifactor(coeficientesA1);
```

$$\{(2)^2(3)^2, -(2)^2(3)^2\}$$

Ou seja, como $36 = 2^2 \cdot 3^2$ (utilizamos a função $ifactor(x)$ para fatorar facilmente um número inteiro x), não há nenhum primo que o divida além do 2 e 3. Note que quando $p = 2$, St_6 também é uma identidade de R , o que é óbvio, pois nesse caso E torna-se comutativa e pelo teorema de Amtsur e Levitzki St_4 é uma identidade polinomial de R .

Fazendo essa análise para os polinômios de grau $n = 2, \dots, 9$ obtemos os resultados da tabela abaixo:

Tabela 3 – Coeficientes das entradas do polinômio standard de grau n

n	Coeficientes	Fatoração
2	$\{-1, 1\}$	$\{-1, 1\}$
3	$\{-2, -1, 1, 2\}$	$\{-1, 1, -(2), (2)\}$
4	$\{-4, 4\}$	$\{(2)^2, -(2)^2\}$
5	$\{-12, -8, -4, 4, 8, 12\}$	$\{(2)^2, (2)^3, (2)^2(3), -(2)^2, -(2)^3, -(2)^2(3)\}$
6	$\{-36, 36\}$	$\{(2)^2(3)^2, -(2)^2(3)^2\}$
7	$\{-144, -108, -36, 36, 108, 144\}$	$\{(2)^2(3)^2, (2)^2(3)^3, (2)^4(3)^2, -(2)^2(3)^2, -(2)^2(3)^3, -(2)^4(3)^2\}$
8	$\{-576, 576\}$	$\{(2)^6(3)^2, -(2)^6(3)^2\}$
9	$\{-2880, -2304, -576, 2304, 2280\}$	$\{(2)^6(3)^2, (2)^8(3^2), (2)^6(3)^2(5), -(2)^6(3)^2, -(2)^8(3^2), -(2)^6(3)^2(5)\}$

Pela conjectura de (KEMER, 2002), para $n = 2$, devemos ter $d_2 = (2 \cdot 2 - 1)p - 1 = 3p + 1$. Ou seja, se $p = 3$ então $d_2 = 10$ é o grau mínimo da identidade standard para $M_n(E)$. Nos parece um valor alto já que $S_6 \in T(R)$. Sabemos que, se St_n é uma identidade polinomial para uma álgebra A , então $St_{n+1} \in T(A)$. Isso justifica o fato de St_i , com $i = 7, 8, 9$ ser uma identidade para R , uma vez que St_6 o é.

Note que $6 = 2 \cdot 2 + 3 - 1$. Logo, pensando a primeira questão dada em (ALVES et al., 2015), para a álgebra R , é afirmativa. Isto é, $S \neq \emptyset$. Além disso, mostra-se $d_2 = 3 \cdot 3 + 1 = 10$ não é uma identidade minimal, ou seja, a conjectura é falsa para $n = 2$ e $p = 3$ (para a álgebra R). Porém, se $p = 5$, temos que $8 = 2 \cdot 2 + 5 - 1$ e St_8 não é uma identidade para R . Ora, isso implica que St_8 também não é uma identidade polinomial de $M_2(E)$. Podemos concluir que pode-se melhorar o limite inferior $2n + p - 1$ do conjunto S para $M_n(E)$. De fato, pela tabela 3 podemos concluir que nenhum dos polinômios calculados pode ser uma identidade de R para $char K = p > 3$.

6.2 Outros resultados sobre o Polinômio stantard

Da forma como construímos o procedimento `ordenamE1E1` é muito simples tornar qualquer outra variável anticomutativa, acrescentando `or type(op([i, 1, 0], variável))` nas linhas 22, 26, 56 e 62 do código (procedimento A.0.5). Dessa forma, definimos x, y, z, w como variáveis anticomutativas e consideramos os seguintes conjuntos

$$R_1 = \left\{ \begin{bmatrix} a & b \\ c & d \end{bmatrix}; a, b, c, d \in E_0 \right\}; R_2 = \left\{ \begin{bmatrix} a & y \\ z & d \end{bmatrix}; a, d \in E_0, y, z \in E_1 \right\};$$

$$R_3 = \left\{ \begin{bmatrix} x & b \\ c & w \end{bmatrix}; x, w \in E_1, b, c \in E_0 \right\}; R_4 = \left\{ \begin{bmatrix} x & y \\ z & w \end{bmatrix}; x, y, z, w \in E_1 \right\}.$$

Assim, como $E = E_0 \oplus E_1$, temos que a álgebra $M_2(E) = R_2 \oplus R_3 = R_1 \oplus R_4$. Os resultados obtidos até a seção anterior foram em relação as álgebras R_1 e R_2 . Para a álgebra R_1 , vale o teorema de Amitsur e Levitzki. Nesse sentido, calculamos os polinômios standard de grau $3 < m < 9$ apenas para R_3, R_4 e $R_2 + R_3 = \{u + v; u \in R_2 \text{ e } v \in R_3\}$ e obtemos que $St_6 \in T(R_3)$ e $St_6 \in T(R_4)$ quando $charK = 3$. E mais, não obtemos nenhuma outra identidade polinomial para R_i .

Mas, podemos obter algumas identidades polinomiais essencialmente fracas para $M_2(E)$, isto é, identidades polinomiais fracas que não são identidades para $M_2(E)$.

Dessa forma, sejam $u_1, u_2, u_3 \in R_1$. Verificamos que

$$St_4(u_1, u_2, u_3, u_4) = 0 \text{ e } St_6(u_1, u_2, u_3, u_4, u_5, u_6) = 0$$

com $u_4, u_5, u_6 \in R_4$.

Como $M_2(E) = R_1 \oplus R_4$, podemos dizer que St_4 e St_6 são identidades polinomiais essencialmente fracas para $M_2(E)$. Podemos dizer ainda que estas são identidades polinomiais 2-graduadas de $M_2(E)$ (termo não definido no presente texto, para mais detalhes ver (AZEVEDO, 2003)). O curioso é que o mesmo não acontece para $St_5(u_1, u_2, u_3, u_4, u_5)$ e $St_7(u_1, u_2, u_3, u_4, u_5, u_6, u_7)$ com $u_4, u_5, u_6, u_7 \in R_4$, isto é, quando o grau do polinômio standard é ímpar.

Se $u_i \in R_1$, com $i = 1, \dots, 4$, temos que todos os polinômios de grau maior que 4 se anulam independentemente da característica do corpo, o que é esperado devido a linearidade do polinômio standard e o teorema de Amitsur e Levitzki.

7 Considerações Finais

Após a leitura de vários trabalhos sobre as álgebras com identidades polinomiais, começamos a construção de modelos computacionais para trabalhar com álgebras não comutativas. Dentre esses modelos, construímos procedimentos, com uso do software Maple, para satisfazer a propriedade distributiva da multiplicação sobre a soma, inerente ao estudo de álgebras em geral, bem como para satisfazer propriedades de álgebras específicas tais como a álgebra de Gassmann.

Como nossos principais objetos de pesquisa foram as álgebras matriciais $M_n(E)$ e $M_{k,l}(E)$, as quais, em geral, apresentam dificuldades em encontrar suas identidades, buscamos encontrar e verificar as identidades polinomiais de tais álgebras em seus casos particulares como, por exemplo, $n = 2$ e $k = l = 1$. De posse dos procedimentos construídos, verificamos a validade das identidades polinomiais fracas encontradas por Di Vincenzo e La Scala para a álgebra $M_{1,1}(E)$, sendo elas $c = [x_1, x_2, x_3]$ e $p = [x_4, x_1][x_3, x_1][x_2, x_1]$. Verificamos também as identidades $f = [[x_1, x_2]^2, x_2]$ e $g = [x_1, x_2, [x_3, x_4], x_5]$ para a álgebra $M_{1,1}(E)$, ambas encontradas por Popov. Como resultado, também verificamos algumas outras identidades clássicas de PI-álgebras, como a identidade da álgebra de Gassmann dada por $[[x_1, x_2], x_3]$, a identidade de Hall para a álgebra $M_2(K)$ dada por $[[x_1, x_2]^2, x_2]$ e o bem conhecido teorema de Amitsur e Levitzki, o qual afirma que o polinômio standard de grau $2n$ dado por $st_{2n}(x_1, \dots, x_{2n}) = \sum_{\sigma \in S_{2n}} (-1)^\sigma x_{\sigma(1)} \dots x_{\sigma(2n)}$ é uma identidade para a álgebra $M_n(K)$.

Devido a boas propriedades do polinômio standard, implementamos um procedimento no Maple para encontrar identidades standard para a álgebra $M_{k,l}(E)$, em especial quando $k = l = 1$. A partir daí, identificamos que o polinômio standard de grau 6 é uma identidade de $M_{1,1}(E)$, quando a característica do corpo é $p = 3$. Quando o número primo $p > 3$ não encontramos identidades polinomiais de grau menor que 10, visto que não conseguimos calcular polinômios standard de grau maior que 9 devido ao processamento ser muito grande. Com isso, apesar de construirmos procedimentos genéricos, não conseguimos encontrar identidades para álgebra das matrizes $M_{k,l}(E)$, com $k, l > 1$ e $p > 3$.

Porém, pudemos encontrar identidades polinomiais essencialmente fracas de $M_n(E)$, isto é, polinômios que não são identidades de $M_n(E)$, mas são identidades para algum subespaço da mesma. Verificamos, por exemplo, que $st_4(x_1, x_2, x_3, x_4) = 0$ e $st_6(x_1, x_2, x_3, x_4, x_5, x_6) = 0$ com $x_1, x_2, x_3 \in M_2(E_0)$ e $x_4, x_5, x_6 \in M_2(E_1)$, onde E_0, E_1 são subespaços da álgebra de Grassmann E .

Além disso, percebemos que calcular o polinômio standard de grau 10 para $M_n(E)$ seria interessante a ponto de verificar a validade ou não da conjectura de Kemer quando $p = 3$. Porém, ficou inviável computacionalmente, devido ao tempo de processamento ser alto. Foram realizadas algumas tentativas de paralelização do procedimento para obter o polinômio standard. Porém, os

resultados não foram satisfatórios, sendo que o tempo de processamento, por razões relativas ao acesso de memória, foi maior que o esperado.

Estamos revisando esse procedimento, tendo em vista que a programação paralela é uma das saídas que temos para o cálculo do polinômio standard de grau $n > 9$.

A paralelização dos procedimentos construídos aqui fica como sugestão de trabalhos futuros com uso do Maple ou com outra linguagem de programação simbólica como o Mathematica. Inclusive este último apresenta uma aparente vantagem por ter implementado o produto exterior (da álgebra de Grassmann) denotado com o símbolo \wedge .

Referências

- ALVES, S. M.; SARTORI, K. K.; ARAKAWA, V. A. The Standard Identity on $M_n(E)$ in Characteristic $p > 2$. **Journal of Algebra**, v. 9, n. 4, p. 155–158, 2015.
- AMITSUR, S. A.; LEVITZKI, J. Minimal Identities for Algebras. **Math. Soc.** **1**, p. 449–463, 1950.
- ANDRADE, L. N. **Introdução à Computação Algébrica com o Maple**. [S.l.]: SBM, 2004.
- ARMSTRONG, M. A. **Groups and Symmetry**. [S.l.]: Springer-Verlag, Cota 20F/ARM, 1988.
- AZEVEDO, S. S. **Identidades Graduadas para Álgebras de Matrizes**. Dissertação (Tese de Doutorado) — Instituto de Matemática, Estatística e Computação Científica (UNICAMP), 2003.
- BENTIN, R. M.; ALVES, S. M. Building Grassmann Numbers from PI-Algebras. **ArXiv e-prints, abr**, 2012.
- DEHN, M. Über die Grundlagen der Projektiven Geometrie und Allgemeine Zahlssysteme. **Math. Ann.****85**, p. 184–194, 1922.
- DRENSKY, V. New Central Polynomials for the Matrix Algebra. **J. Math.**, Israel, n. 92, p. 235–248, 1995.
- DRENSKY, V.; CATTANEO, G. P. A Central Polynomial of low Degree for 4×4 Matrices. **J. Algebra**, n. 168, p. 469–478, 1995.
- DRENSKY, V.; RASHKOVA, T. G. Wear Polynomial Identities for the Matrix Algebras. **Commun. in Algebra**, n. 21, p. 3779–3795, 1993.
- FORMANEK, E. Central Polynomials for Matrix Rings. **J. Algebra**, n. 23, p. 129–132, 1972.
- FORMANEK, E. The Polynomial Identities and Invariants of $n \times n$ Matrices. **CBMS Regional Conf. Series in Math.** **78**, Published for the Confer. Board of the Math. Sci, Washington DC, AMS, Province RI, n. Math. 77, 1991.
- GALVÃO, A. T. **PI-Álgebras**. Dissertação (Dissertação de Mestrado) — Universidade Estadual de Campinas, 2003.
- GIAMBRUNO, A.; VALENTI, A. Central Polynomials and Matrix Invariants. **Israel J. Math.**, n. 96, p. 281–297, 1996.
- HALPIN, P. Central and Wear Identities for Matrices. **Commun. in Algebra**, n. 11, p. 2237 – 2248, 1983.
- JAMIOŁKOWSKI, A. On Applications of PI-Algebras in the Analysis of Quantum Channels. **International Journal of Quantum Information**, v. 10, n. 08, 2013.
- KAPLANSKY, I. Problems in the Theory of Rings. **Report of a conference on linear algebras**, Washington DC, BuBl., n. 502, p. 1–3, 1957.

- KAPLANSKY, I. Problems in the Theory of Rings, Revisited. **The American Mathematical Monthly**, Mathematical Association of America, v. 77, n. 5, p. 445–454, 1970. ISSN 00029890, 19300972.
- KEMER, A. Matrix Type of Some Algebras over a Field of Characteristic p . **Journal of Algebra**, v. 251, n. 2, p. 849–863, 2002. ISSN 00218693.
- MUNDIN, K. C.; MUNDIN, M. S. P. Álgebra de Grassmann e a Teoria Quântica. **Revista Brasileira de Ensino de Física**, v. 19, n. 02, 1997.
- POPOV, A. Identities of the Tensor Square of a Grassmann Algebra. **Algebra i Logika**, n. 4, p. 442–471, 1982.
- PORTUGAL, R. Introdução ao Maple. Laboratório Nacional de Computação Científica (LNCC). 2002.
- RAZMYSLOV, Y. P. Finite Basing of the Identities of Matrix Algebra Second Order Over a Field of Characteristic Zero. Translation: *Algebra and Logica* 12, p. 47–73, 1973.
- RAZMYSLOV, Y. P. On a Problem of Kaplansky. Translation: *Math. USSR, Izv.*, p. 479–496, 1973.
- RAZMYSLOV, Y. P. Identities of Algebras and Their Representations. Translation: *Translations of Math. Monographs*, AMS, Providence, R. I., 1994.
- ROWEN, L. Polynomial Identities of Rings Theory. **Acad. Press.**, 1980.
- VINCENZO, O. M.; SCALA, R. Robinson-Schensted-Knuth Correspondence and Wear Polynomial Identities of $M_{1;1}(E)$. **Algebra Colloquium**, n. 12:2, p. 333–349, 2005.
- WAGNER, W. Über die Grundlagen der Projektiven Geometrie und Allgemeine Zahlssysteme. **Math. Ann.**113, p. 528–567, 1936.
- ZIMMERMANN, H. **PI-Algebras: An Introduction**. Berlin - New York: Lectures Notes in Math., Springer-Verlag, 1975.

Anexos

ANEXO A – Procedimentos em Maple

Segue os procedimentos construídos (apenas os que não aparecem no corpo do texto) usando a linguagem Maple.

Para usar a propriedade $b_i b_j = -b_j b_i$, criamos o procedimento a seguir.

Procedimento A.0.1. *Função para ordena os $x_i \in E_1$ usando a regra anti-comutativa: $x_i x_j = -x_j x_i$.*

Entrada: *Um produto com elementos indexados x_i .*

Saída: *Um produto de x_i com índices organizados em ordem crescente e o sinal.*

```

1  ordenaE1:=proc(expr)
2      local sinal, aux, vet, expaux, expaux2, i, j, cont;
3      if op(1,expr)=-1 then
4          expaux:= -expr;
5          sinal:=-1;
6      else
7          expaux:= expr;
8          sinal:=1;
9      end if;
10     if (type(expaux,indexed) or type(expaux, '^')) then
11         return expr;
12     end if;
13     # // formar uma lista com os indices de expaux.
14     vet:=[];
15     for i to nops(expaux) do
16         if (type(op(i,expaux), '^')) then
17             for j to op([2, 1], op(i, expaux)) do
18                 vet:= [op(vet),op([1, 1], op(i, expaux))];
19             end do;
20         else
21             vet:= [op(vet),op([i, 1], expaux)];
22         end if;
23     end do;
24     cont:= 0;
25     # // ordenar vet em ordem crescente.
26     j:= 1;
27     while j>0 do
28         j:=0;
```

```

29     for i to nops(vet)-1 do
30         if vet[i]>vet[i+1] then
31             aux:= vet[i];
32             vet[i]:= vet[i+1];
33             vet[i+1]:= aux;
34             cont:= cont + 1;
35             j:= j+1;
36         end if;
37     end do;
38 end do;
39 # // verificar o sinal a depender da quantidade de mudanca
    na posicao do termo
40 if (cont mod 2 <> 0) then
41     sinal:= -1 .~ sinal;
42 end if;
43 expaux2:= 1;
44 for i to nops(vet) do
45     for j to nops(expaux) do
46         if (type(op(j,expaux), '^')) then
47             if (evalb(op([1, 1], op(j, expaux))=vet[i]))
                then
48                 expaux2:= expaux2 .~ op(j, expaux);
49                 expaux:= subsop(j = vet[nops(vet)]+1, expaux
                    );
50                 break;
51             end if;
52             elif (evalb(op([j, 1], expaux)=vet[i])) then
53                 expaux2:= expaux2 .~ op(j,expaux);
54                 expaux:= subsop(j = vet[nops(vet)]+1, expaux);
55                 break;
56             end if;
57         end do;
58     end do;
59     return sinal .~ expaux2;
60 end proc;

```

┘

Usamos a variável *sinal* para armazenar o sinal do monômio que a função *ordenaEI* recebe de parâmetro. A expressão *type(op(i,expaux), '^')* retorna verdadeiro se *expaux* está escrito como potência e falso caso contrário. Isso é válido quando tratamos de polinômios que não são multilineares. Note que o monômio é reorganizado de acordo com os índices, mas não é levado


```

27         if (evalb(op([0, 1], expaux[j])=b)) then
28             vet2[i]:=[op(vet2[i]),1];
29             bs[i]:=bs[i] .~ expaux[j];
30         else
31             vet2[i]:=[op(vet2[i]),2];
32             cs[i]:=cs[i] .~ expaux[j];
33         end if;
34     end if;
35 end if;
36 end do;
37 end do;
38 cont:= 0;
39 expaux2:= 1;
40 for k to op([1, 1], expaux[nops(expaux)]) do
41     # ordena vet2[k]
42     j:= 1;
43     while j>0 do
44         j:=0;
45         for i to nops(vet2[k])-1 do
46             if vet2[k][i]>vet2[k][i+1] then
47                 aux:= vet2[k][i];
48                 vet2[k][i]:= vet2[k][i+1];
49                 vet2[k][i+1]:= aux;
50                 cont:= cont + 1;
51                 j:= j+1;
52             end if;
53         end do;
54     end do;
55     expaux2:= expaux2 .~ bs[k] .~ cs[k];
56 end do;
57 for i to nops(expaux2) do
58     if type(op(i,expaux2), '^') then
59         return 0;
60     end if;
61 end do;
62 # verificar o sinal a depender da quantidade de mudanca na
63     posicao do termo
64 if (cont mod 2 <> 0) then
65     sinal:= -1 .~ sinal;
66 end if;
67 return sinal*expaux2;
68 end proc;

```



```

24         else
25             expbc:= expbc .~ op(i,exp1);
26         end if;
27     else
28         if (op([i, 0, 1],exp1)=a or op([i, 0,
29             1],exp1)=d or type(op(i,exp1), '
30             numeric')) then
31                 expad:= expad * op(i,exp1);
32             else
33                 expbc:= expbc .~ op(i,exp1);
34             end if;
35         end if;
36     end do;
37     expaux[k,t]:= expad * expbc;
38     if versinal=-1 then
39         expaux[k,t]:= -expad * expbc;
40     end if;
41 else # caso 3: exp1 recebe uma soma de termos do
42     tipo a*a*b + b*(a.b) + ...
43     expaux[k,t]:= 0;
44     for i to nops(exp1) do
45         expaux2:= op(i,exp1); # simplificar o
46         trabalho com os indices e usa o processo
47         anterior com exp1=expaux2.
48     versinal:=op(1,expaux2);
49     # // caso 1: expaux2 recebe apenas um
50     elemento a,b,c,d ou a^2, etc.
51     if (type(expaux2, '^') or type(expaux2, '
52     indexed') or type(expaux2, 'numeric'))
53     then
54         expaux[k,t]:= expaux[k,t] + expaux2;
55     else # caso 2: expaux2 recebe
56         apenas um elemento como produto: a.a.b.
57         if type(versinal, 'numeric') then
58             expaux2:=expaux2/versinal;
59         end if;
60         expad:=1;
61         expbc:=1;
62         for j to nops(expaux2) do
63             if (type(op(j,expaux2), '^')) then
64                 if (op([j, 1, 0], expaux2)=a or
65                     op([j, 1, 0], expaux2)=d)

```



```

56         then
57             expad:= expad * op(j,expaux2
58                 );
59         else
60             expbc:= expbc .~ op(j,
61                 expaux2);
62         end if;
63     else
64         if (op([j, 0, 1], expaux2)=a or
65             op([j, 0, 1], expaux2)=d or
66             type(op(j,expaux2), 'numeric')
67             ) then
68             expad:= expad * op(j,expaux2
69                 );
70         else
71             expbc:= expbc .~ op(j,
72                 expaux2);
73         end if;
74     end if;
75 end do;
76 if type(versinal, 'numeric') then
77     expaux[k,t]:= expaux[k,t] + versinal
78         * expad * expbc;
79 else
80     expaux[k,t]:= expaux[k,t] + expad *
81         expbc;
82     end if;
83 end if;
84 end do;
85 end if;
86 end do;
87 expaux:= Matrix(m,n, (i,j)-> expaux[i,j]); # Escreve o
88     resultado no formato usual de matriz.
89 return expaux;
90 end proc;
```

┘

Procedimento A.0.4. *Função para usar a regra de que a e d podem comutar em E.*

Entrada: Uma matriz com entradas em E.

Saída: Uma matriz com a e $d \in E_0$.

```

1 ordenamE1:= proc(M::Matrix)
2     local i, j, expad, k, t, expbc, expaux, m, n, expl, expaux2,
3         versinal;
4     uses LinearAlgebra; # carrega o pacote liberando funcoes que
5         permite trabalhar com matrizes.
6     m, n := Dimension(M); # Ler e nomeia as dimensoes da matriz
7         M;
8     for k to m do
9         for t to n do
10            expl:= M[k,t]; # // expr recebe uma expressao de
11                cada entrada de M.
12            versinal:=op(1,expl);
13            # // associando os a's e b's.
14            # // caso 1: expl recebe apenas um elemento a,b,c,d
15                ou a^2, etc.
16            if (type(expl, '^') or type(expl,'indexed') or type(
17                expl,'numeric')) then
18                expaux[k,t]:= expl;
19            # // caso 2: expl recebe apenas um elemento como
20                produto: a.a.b ou 2*(a.b.a)
21            elif (type(expl, '.') or type(versinal,'numeric'))
22                then
23                if type(versinal,'numeric') then
24                    expl:= expl/versinal;
25                end if;
26                expad:=1;
27                expbc:=1;
28                for i to nops(expl) do
29                    if (type(op(i,expl), '^')) then
30                        if (op([i, 1, 0], expl)=b or op([i, 1,
31                            0], expl)=c) then
32                            expbc:= expbc .~ op(i,expl);
33                        else
34                            expad:= expad * op(i,expl);
35                        end if;
36                    else
37                        if (op([i, 0, 1],expl)=b or op([i, 0,
38                            1],expl)=c ) then
39                            expbc:= expbc .~ op(i,expl);
40                        else

```

```

31             expad:= expad * op(i,expl);
32         end if;
33     end if;
34 end do;
35 expbc:= ordenaE1(expbc);
36 expaux[k,t]:= expad * expbc;
37 if versinal=-1 then
38     expaux[k,t]:= -expad * expbc;
39 end if;
40 else # caso 3: expl recebe uma soma de termos do
41     tipo  $a*a*b + b*(a.b) + \dots$ 
42     expaux[k,t]:= 0;
43     for i to nops(expl) do
44         expaux2:= op(i,expl); # simplificar o
45         trabalho com os indices e usa o processo
46         anterior com expl=expaux2.
47         versinal:=op(1,expaux2);
48         # // caso 1: expaux2 recebe apenas um
49         elemento a,b,c,d ou a^2, etc.
50         if (type(expaux2, '^') or type(expaux2, '
51         indexed') or type(expaux2, 'numeric'))
52         then
53             expaux[k,t]:= expaux[k,t] + expaux2;
54         else # caso 2: expaux2 recebe
55         apenas um elemento como produto: a.a.b.
56         if type(versinal, 'numeric') then
57             expaux2:=expaux2/versinal;
58         end if;
59         expad:=1;
60         expbc:=1;
61         for j to nops(expaux2) do
62             if (type(op(j,expaux2), '^')) then
63                 if (op([j, 1, 0], expaux2)=b or
64                     op([j, 1, 0], expaux2)=c)
65                 then
66                     expbc:= expbc .~ op(j,
67                     expaux2);
68                 else
69                     expad:= expad * op(j,expaux2
70                     );
71                 end if;
72             else
73                 else

```

```

62         if (op([j, 0, 1], expaux2)=b or
63             op([j, 0, 1], expaux2)=c)
64             then
65                 expbc:= expbc .~ op(j,
66                     expaux2);
67             else
68                 expad:= expad * op(j,expaux2
69                     );
70             end if;
71         end if;
72     end do;
73     expbc:= ordenaE1(expbc);
74     if type(versinal,'numeric') then
75         expaux[k,t]:= expaux[k,t] + versinal
76             * expad * expbc;
77     else
78         expaux[k,t]:= expaux[k,t] + expad *
79             expbc;
80     end if;
81 end do;
82     expaux:= Matrix(m,n, (i,j)-> expaux[i,j]); # Escreve o
83         resultado no formato usual de matriz.
84     return expaux;
85 end proc;

```

┘

Procedimento A.0.5. Função para usar a regra de que a e d podem comutar em E .

Entrada: Uma matriz com entradas em E .

Saída: Uma matriz com a e $d \in E_0$.

```

1 ordenaE1:= proc(M::Matrix)
2     local i, j, expad, k, t, expbc, expaux, m, n, expl, expaux2,
3         versinal;
4     uses LinearAlgebra; # carrega o pacote liberando funcoes que
5         permite trabalhar com matrizes.
6     m, n := Dimension(M); # Ler e nomeia as dimensoes da matriz
7     M;

```

```

5   for k to m do
6       for t to n do
7           expl:= M[k,t];  # // expr recebe uma expressao de
                           cada entrada de M.
8           versinal:=op(1,expl);
9           # // associando os a's e b's.
10          # // caso 1: expl recebe apenas um elemento a,b,c,d
              ou a^2, etc.
11          if (type(expl, '^') or type(expl, 'indexed') or type(
              expl, 'numeric')) then
12              expaux[k,t]:= expl;
13          # // caso 2: expl recebe apenas um elemento como
              produto: a.a.b ou 2*(a.b.a)
14          elif (type(expl, '.' ) or type(versinal, 'numeric'))
              then
15              if type(versinal, 'numeric') then
16                  expl:= expl/versinal;
17              end if;
18              expad:=1;
19              expbc:=1;
20              for i to nops(expl) do
21                  if (type(op(i,expl), '^')) then
22                      if (op([i, 1, 0], expl)=b or op([i, 1,
                          0], expl)=c) then
23                          expbc:= expbc .~ op(i,expl);
24                      else
25                          expad:= expad * op(i,expl);
26                      end if;
27                  else
28                      if (op([i, 0, 1],expl)=b or op([i, 0,
                          1],expl)=c ) then
29                          expbc:= expbc .~ op(i,expl);
30                      else
31                          expad:= expad * op(i,expl);
32                      end if;
33                  end if;
34              end do;
35              expbc:= ordenaE1E1(expbc);
36              expaux[k,t]:= expad * expbc;
37              if versinal=-1 then
38                  expaux[k,t]:= -expad * expbc;
39              end if;

```



```

67         end if;
68     end do;
69     expbc:= ordenaE1E1(expbc);
70     if type(versinal,'numeric') then
71         expaux[k,t]:= expaux[k,t] + versinal
            * expad * expbc;
72     else
73         expaux[k,t]:= expaux[k,t] + expad *
            expbc;
74     end if;
75     end if;
76     end do;
77     end if;
78     end do;
79 end do;
80 expaux:= Matrix(m,n, (i,j)-> expaux[i,j]); # Escreve o
        resultado no formato usual de matriz.
81 return expaux;
82 end proc:

```

┘

Para calcular o produto entre duas matrizes de forma que as entradas não comutem, foi criado o procedimento a seguir concatenado com a função *distribui* (procedimento ??).

Procedimento A.0.6. *Função para calcular o produto de duas matrizes sem comutar as entradas.*

Entrada: Duas matrizes x_1, x_2 .

Saída: Uma matriz produto $x_1.x_2$ sem comutar as entradas.

```

1 prod_mat_grass := proc (M::Matrix, N::Matrix)
2     local m, n, r, s, MN, i, j, k;
3     uses LinearAlgebra;
4     m, n := Dimension(M); # // Ler e nomeia as dimensoes da
        matriz M;
5     r, s := Dimension(N); # // Ler e nomeia as dimensoes da
        matriz N;
6     if (n <> r) then
7         error "Dimensoes incompativeis.";
8     end if;
9     for i to m do

```

```

10     for j to s do
11         MN[i, j] := 0;
12         for k to n do
13             MN[i, j] := MN[i, j] + distribui(M[i, k], N[k, j]);
14         end do;
15     end do;
16 end do;
17 MN:= Matrix(m, s, (i, j)-> MN[i, j]);
18 return MN; # // retorna uma matriz.
19 end proc:

```

┘

A fim de simplificar a entrada dos dados, criamos a função *prodn* para calcular o produto entre n matrizes de ordens compatíveis.

Procedimento A.0.7. *Função para calcular o produto de uma sequência finita de matrizes.*

Entrada: Uma lista de matrizes $L := [x_1, x_2, \dots, x_n]$ com $n \geq 2$.

Saída: Uma matriz produto $x_1 \cdot x_2 \cdot \dots \cdot x_n$.

```

1 prodn := proc (L::list)
2     local i, M;
3     uses LinearAlgebra; # // carrega o pacote liberando funcoes
4     M := L;
5     if (nops(L) = 1) then
6         return M;
7     end if;
8     M := prod_mat_grass(L[1], L[2]);
9     for i from 3 to nops(L) do
10        M := prod_mat_grass(M, L[i]);
11    end do;
12    return M;
13 end proc:

```

┘

Para o cálculo do polinômio standard para matrizes com entradas na álgebra de Grassmann, bastou utilizar vários procedimentos concatenados como segue

Procedimento A.0.8. *Função para calcular o polinômio standard de $M_{k;l}(E)$.*

Entrada: Uma lista de matrizes $L := [x_1, x_2, \dots, x_n]$ com $n \geq 2$.

Saída: Uma matriz (polinômio standard de grau n).

```

1 Stn := proc (L::list)
2     local i, j, Perm, T, pol, standn, sinal;
3     uses combinat, group;          # // Libera as funcoes numbperm
4     , partition, permute, parity etc.
5     if nops(L) < 2 then
6         error "A lista deve ter mais que 2 elementos";
7     end if;
8     Perm := permute([seq(i, i = 1 .. nops(L))], nops(L));
9     standn := 0;
10    for i to nops(Perm) do
11        T:= [seq(L[Perm[i][t]],t=1..nops(Perm[i]))]
12        pol := prodn(T);
13        pol:= ordenamEl(pol);
14        sinal:= parity(convert(Perm[i], 'disjyc')); # //
15        Reescreve a permutacao como ciclos dijuntos.
16        pol:= sinal*pol;      # // parity retorna 1 se a
17        permutacao for par ou -1 se impar.
18        standn := standn + pol;
19    end do;
20    return standn;
21 end proc;
```

┘

Note que o parâmetro da função *Stn* é uma lista de matrizes, as quais são as variáveis do polinômio standard, de forma que fica uma notação bem parecida com a convencional. De forma muito parecida com o procedimento 4.3.1, nesse caso apenas criamos listas auxiliares *T* para usar as demais funções de forma simples.