**RELATÓRIO DO PROJETO COMPUTACIONAL – Grupo 1**

ALEXANDRE CARDEIRA, CÁTIA ANTUNES, FRANCISCO OLIVEIRA, LUÍS ISIDRO, JOSÉ MARTINS

Este relatório descreve as funções utilizadas para responder às questões (A), (B) e (C) do projeto computacional.

CONTENTS

1.Introdução1

2.Métodos2

3. Resultados16

4. Conclusão21

5. Bibliografia21

ABSTRACT. This report contains a collection of Octave routines

applied to generate and classify magma structures to study some

general properties of the cartesian product on those structures.

1. INTRODUÇÃO

Seja A um conjunto qualquer. Uma estrutura de magma sobre o conjunto *A* é um sistema algébrico com sobre uma função de em *A* que se representa por .

Para , uma estrutura de magma, pode-se analisar as seguintes condições, as quais devem ser válidas para todo

Comutativa:

Associativa:

Cancelativa à direita: se existe um tal que , então

[[1]](#footnote-1)Cancelativa à esquerda: se existe um tal que , então

Medial:

Elemento Neutro: existe tal que

Se e são duas estruturas de magma, um homomorfismo é uma função , se e só se a condição

,

é verificada para todo o .

Se trocarmos os valores de *x* e *y* forem trocados no segundo membro da equação acima, podemos definir a noção de anti-isomorfismo através da condição

,

Quando *f* é uma função bijetiva, ou seja, tem uma inversa, e quando satisfaz a condição de homomorfismo, então *f* diz-se um isomorfismo e os semigrupos e são denominados de isomorfos. Da mesma forma, quando *f* é bijetiva e existe um anti-isomorfismo entre e , então trata-se de um anti-isomorfismo.

Por fim, considera-se que duas estruturas são equivalentes se forem isomorfas ou anti-isomorfas.

1. MÉTODOS

Nesta secção são apresentados todos os métodos utilizados em cada umas das questões do projeto.

Neste trabalho foram considerados os seguintes conjuntos:

tendo sido geradas todas as possíveis estruturas que podem ser definidas nos conjuntos, definindo assim e . Para efeitos deste projeto, definiu-se uma estrutura em A, como uma matriz 2-por-2 com entradas em 1:2 e, em B, como uma estrutura de 3-por-3 com entradas em 1:3.

2.1 Questões

A função *criarMatriz*, é uma função que permite representar e , sendo que estas matrizes representam todas as estruturas que podem ser geradas a partir de *A* e B, respetivamente, cuja cardinalidade é,

,

.

2.1.1 *criarMatriz.m:* Função que recebe um tamanho como paramento e que gera todas as matrizes lineares que podem ser geradas a partir de um dado conjunto.

Uma imagem com texto

Descrição gerada automaticamente

Uma imagem com texto

Descrição gerada automaticamente

2.1.2 *verificaClassess.m:* Função que recebe como parâmetro uma matriz e que devolve as suas classes, de acordo com a característica homológica, imprimindo o resultado.

Uma imagem com texto

Descrição gerada automaticamente

Por forma a testar todas as propriedades de cada matriz linear, foram criadas várias funções, em que cada representa a validação de uma condição e aceita como parâmetro uma linha da matriz linear, devolvendo *true* ou *false*. Estas rotinas são de seguida executadas numa função geral que permite avaliar as propriedades de toda a estrutura, devolvendo uma matriz com informação sobre a validação das diferentes propriedades nas diferentes linhas.

2.1.3 *isComutativa.m:* Função que verifica se uma matriz é comutativa.

Uma imagem com texto

Descrição gerada automaticamente

2.1.4 *isAssociativa.m:* Função que avalia se uma matriz verifica a associatividade.

Uma imagem com texto

Descrição gerada automaticamente

2.1.5 *isMedial.m:* Função que verifica se uma matriz é medial.



2.2.6 *hasElementoNeutro.m*: Função que recebe como parâmetro uma matriz e verifica se esta possui um elemento neutro.

Uma imagem com texto

Descrição gerada automaticamente

2.1.7 *isCancelativaEsquerda.m*: Função que recebe como parâmetro uma linha da matriz linear (matriz 2x2 ou 3x3 linearizada) e compara com um vetor pré-definido de acordo com a propriedade, avaliando se são iguais.

Uma imagem com texto

Descrição gerada automaticamente

2.1.8 isCancelativaDireita.m: Função que recebe como parâmetro uma linha da matriz linear (matriz 2x2 ou 3x3 linearizada) e compara com um vetor pré-definido de acordo com a propriedade, avaliando se são iguais.

Uma imagem com texto

Descrição gerada automaticamente

2.1.9 *procCancEsquerda.m:* Função que recebe uma matriz, transformando-a num vetor de n por n de 3 dimensões, comparado se é igual ao vetor predefinido, de acordo com o número de linhas. Devolve o índice da matriz e o total de matrizes que verificam a propriedade.

Uma imagem com texto

Descrição gerada automaticamente

2.1.10 *procCancDireita.m*: Função que recebe uma matriz, transforma essa estrutura num vetor de n por n de 3 dimensões, comparado de seguida se é igual ao vetor predefinido, de acordo com o número de linhas. Devolve o índice da matriz e o total de matrizes que verificam a propriedade.

Uma imagem com texto

Descrição gerada automaticamente

2.1.11 *verificaPropriedades.m*: Função que recebe como parâmetro uma matriz linearizada e que tem como objetivo verificar as propriedades em todas as suas linhas, devolvendo uma estrutura que permitir observar quais é que verificam as propriedades ou não.

Uma imagem com texto

Descrição gerada automaticamente

2.1.12 *isIso.m:* Função que recebe como parâmetros uma matriz linearizada com todas as matrizes possíveis (MLinear), um vetor com uma permutação possível (alpha) e um índice que aponta para uma linha da nossa matriz. Esta rotina calcula a matriz a ser usada pela procura na matriz MLinear através do índice, usando-a então para calcular a matriz que lhe é isomorfa. Posteriormente, é verificado se esta matriz existe dentro da nossa matriz MLinear e é-nos devolvido o seu índice.

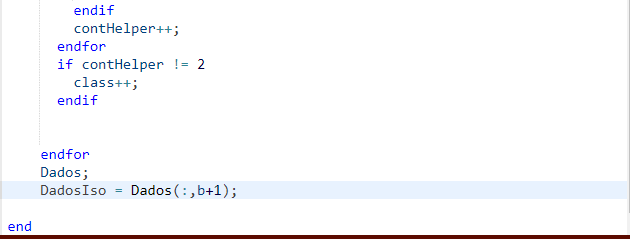
Uma imagem com texto

Descrição gerada automaticamente

2.1.13 *isIso2.m*: Função semelhante à anterior, no entanto recebe como parâmetro apenas a matriz linearizada com todas as matrizes possíveis (MLinear). Neste caso calculamos todas as possibilidades de permutação dentro da própria matriz, devolvendo uma listagem, em coluna, das diferentes classes de isomorfismo.

Uma imagem com texto

Descrição gerada automaticamente



2.1.14 *isAntiIso.m:* Função que recebe como parâmetros uma matriz linearizada com todas as matrizes possíveis (MLinear), um vetor com uma permutação possível (alpha) e um índice que aponta para uma linha da nossa matriz. Esta rotina calcula a matriz a ser usada pela procura na matriz MLinear através do índice, usando-a então para calcular a matriz que lhe é isomorfa como também a que lhe é anti-isomorfa. Posteriormente, é verificado se estas matrizes existem dentro da nossa matriz MLinear e é-nos devolvido os seus índices.

Uma imagem com texto

Descrição gerada automaticamente

2.1.15 *isAntiIso2.m:* Função semelhante à anterior, no entanto recebe como parâmetro apenas a matriz linearizada com todas as matrizes possíveis (MLinear). Neste caso calculamos todas as possibilidades de permutação dentro da própria matriz, devolvendo uma listagem, em coluna, das diferentes classes resultantes da combinação de isomorfismo com anti-isomorfismo.

Uma imagem com texto

Descrição gerada automaticamente

Uma imagem com texto

Descrição gerada automaticamente

2.1.16 *adicionaPropriedades.m:* Esta função tem o objetivo de, dada a matriz linearizada com todas as matrizes possíveis (MLinear), permitir-nos calcular as diferentes classes identificativas das relações de equivalência de homologia, isomorfismo e isomorfismo com anti-isomorfismo. Após o cálculo, é apresentado no ecrã a matriz linear com as várias colunas à direita representativas das classes que foram calculadas.

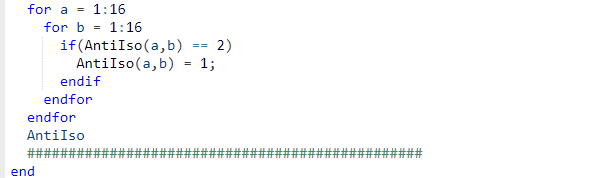
Uma imagem com texto

Descrição gerada automaticamente

2.1.17 *imprimeIsoAnti.m:* Esta função tem também como objetivo de mostrar as relações de equivalência ao nível do isomorfismo e anti-isomorfismo (como as combinações) de uma forma matricial, que permite ser mais fácil visualização para o utilizador. De notar que esta função está apenas preparada para as matrizes com tamanho 2x2, pelo que, para as matrizes 3x3 achámos que esta função se iria tornar inviável, tanto pela ineficiência do cálculo, como a difícil visualização da matriz gerada.

Uma imagem com texto

Descrição gerada automaticamente



2.2.18 *main.c:* É um ficheiro com o procedimento utilizado pelo grupo para executar as rotinas acima mencionadas.

Uma imagem com texto

Descrição gerada automaticamente

1. RESULTADOS

Nesta secção apresentamos os resultados obtidos para cada uma das funções do projeto computacional.

3.1.1*criarMatriz.m*

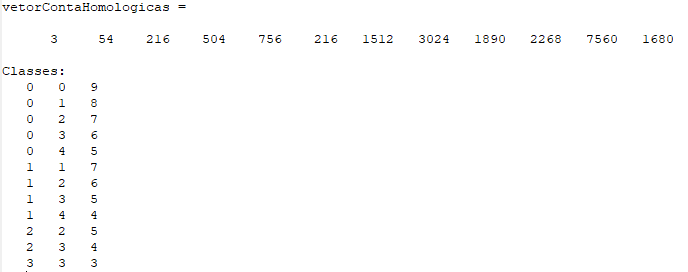
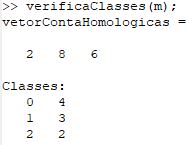
Nas matrizes 2x2, também devolve uma estrutura que na qual as quatro primeiras posições representam uma matriz, a quinta posição é o índice que a permite identificar na estrutura, a sexta e sétima representam a frequência dos elementos na matriz e na oitava posição está a classe homológica a que cada pertence.

**Uma imagem com mesa

Descrição gerada automaticamenteUma imagem com mesa

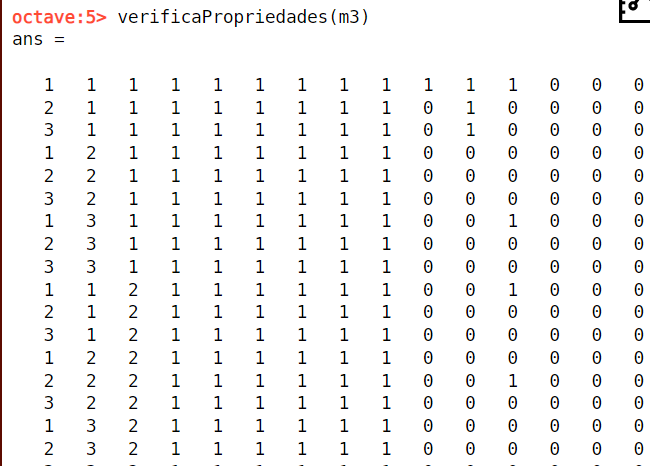
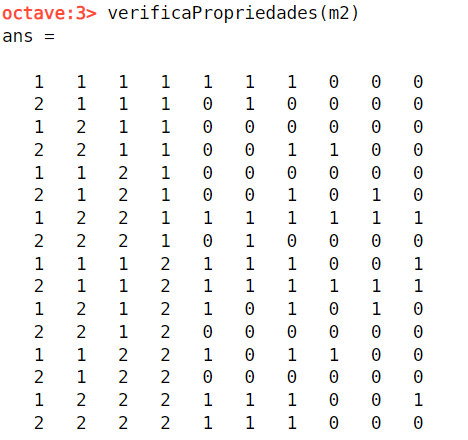
Descrição gerada automaticamente**

3.1.2*vetorClassess.m*

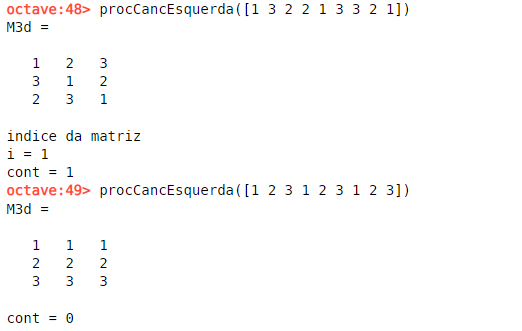
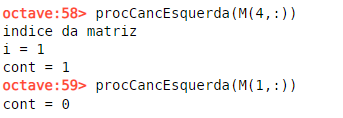


3.1.3 *verificaPropridades.m*

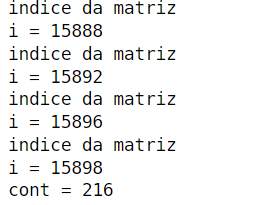
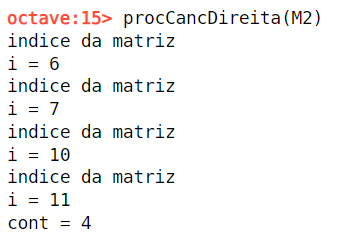
Devolve uma estrutura, que no caso das matrizes *2x2*, as quatro primeiras posições representam uma matriz e as seguintes posições representam a presença das propriedades associativa, comutativa, medial, cancelamento à esquerda, cancelamento à direita e se tem elemento neutro, respetivamente. Nas matrizes *3x3*, as nove primeiras representam a matriz, e as seguintes posições são iguais à matriz *2x2.*



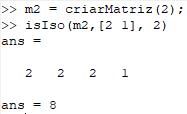
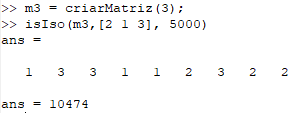
3.1.4 *procCancEsquerda.m*



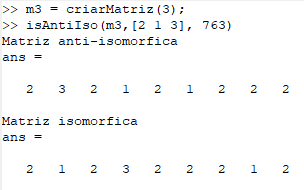
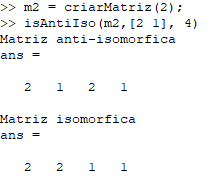
3.1.5 *procCancDireita.m*



3.1.6*isIso.m*



3.1.7 *isAntiIso.m*



3.1.8 *adicionaPropriedades.m*

Devolve uma estrutura, que no caso das matrizes *2x2*, as quatro primeiras posições representam uma matriz, e nas *3x3* são as primeiras nove. No caso, das 2x2, a quinta posição é o índice que a permite identificar na estrutura, a sexta e sétima representam a frequência dos elementos na matriz, na oitava posição está a classe homológica a que cada pertence, a penúltima coluna é a classe resultante do isomorfismo e última coluna representa as classes por anti-isomorfismo e isomorfismo

Uma imagem com mesa

Descrição gerada automaticamente

**Uma imagem com mesa

Descrição gerada automaticamente**

3.1.9 *adicionaPropriedades.m*

Além de devolver uma estrutura igual à da função acima mencionada, também devolve a contagem de classes de acordo com o isomorfismo, anti-isomorfismo e homologia. Os valores são mostrados abaixo para as matrizes 2x2 e 3x3 respetivamente. Os valores dizem respeito à relação de equivalência por homologia, isomorfismo e isomorfismo com anti-isomorfismo respetivamente.





3.1.10 *imprimeAntiIso.m*

Uma imagem com mesa

Descrição gerada automaticamente

4.CONCLUSÃO

Este projeto computacional foi desenvolvido com o objetivo de explorar as potencialidades da programação do *Octave* no que diz respeito à teoria sobre as estruturas de magma e as suas propriedades, bem como, a classificação das estruturas de semigrupos e a existência de isomorfismo e anti-isomorfismo.

Uma das limitações do projeto é o facto de as funções implementadas só aceitarem estruturas de *2x2* ou *3x3*. A rotina *CriarMatriz.m* apresenta o código para implementar *4x4,* no entanto acorre um transbordo de memória. Além disso, são apresentadas algumas rotinas que apenas podem ser utilizadas nas estruturas *2x2*, uma vez que nas matrizes *3x3* iriam provocar um transbordo na memória ou iriam consumir demasiado tempo a executar.

No geral, apesar do projeto não responder a todas as questões do enunciado, permitiu ao grupo adquirir conhecimentos sobre a construção e manipulação de estruturas no *Octave*, permitindo pôr em prática a teoria sobre a caracterização de todas as estruturas magmas até nove elementos.

1. BIBLIOGRAFIA

N. Martins-Ferreira, M. Belbut, *Guide To The Project on the Classification of Magmas and Smigroup-Like Structures*, CDRSP-IPLeiria Technical Report, GTLab(Monoids-9.1) 115 (2022)

1. *Date: May* 31, 2022; Matemática Discreta EI-PL; Grupo 1 [↑](#footnote-ref-1)