

UNIVERSIDADE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação
Programa de Ciências de Computação e Matemática Computacional

Relatório de Trabalho 3
Disciplina Geração de Malhas

Aluna: Rosalía Taboada Leiva
Docentes: Dr. Prof. Luis Gustavo Nonato
Dr. Antonio Castelo Filho

29 de Setembro de 2017

1 Introdução

Este trabalho cria um código para gerar uma corner table de um arquivo vtk. Neste caso daremos 3 exemplos para ver como funciona o código que se está propondo.

2 Procedimento

O código principal, que na verdade é uma função, tem como nome `corner_table.m` e dá como resultado a tabela requerida em forma de matriz.

A primeira coisa que precisamos para fazer este código é ter o arquivo vtk e poder ler no Matlab, para isso tem sido feito o código chamado `read_vtk.m`, com isso já temos os dados do arquivo vtk como duas matrizes, a primeira chamada ‘Vert’ que guarda os Vertices da discretização e a segunda chamada ‘Face’ que guarda a distribuição dos vertices em cada triângulo.

Sabendo que a distribuição da estrutura Corner - Table é como se vê na tabela 1.

Tabela 1: Corner - Table

Corn	Vertice	Face	c.n	c.p	c.o	c.r	c.l
c_1	0	1	2	3	...		
c_2	1	1	3	1	...		
\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots	\vdots

A proposta de código consiste em construir as 5 primeiras colunas da corner table usando a matriz ‘Face’ assim:

- A primeira coluna é um vetor de 1 até $3 \times$ quantidade de faces, que neste caso quantidade de faces será igual ao número de linhas da matriz Faces.
- Para preencher as 4 colunas restantes começaremos pela coluna 3 e as três primeiras células desta coluna colocamos o número 1 que corresponde ao primeiro triângulo (e depois as três seguintes com o número 2 e assim sucessivamente) ou face e com isso as primeiras células das colunas 2, 4 e 5 já podem ser preenchidas pois o primeiro vértice de face 1 é o elemento (1,2) da matriz Face, assim consequentemente o corner posterior será o elemento (1,3) da matriz Face e o anterior será o elemento (1,4) da matriz Face. Y fazemos de forma similar com a segunda e terceira linha das colunas 2, 4 e 5.
- Para obter o corner oposto, se criou uma função chamada `c_oposto.m` que devolve a face onde se encontra o corner oposto, que em forma geral o que faz é identificar a face onde este corner está e encontra os vértices v_1 e v_2 que acompanham ele nessa face, e depois encontra a face na qual coincidem v_1 e v_2 como vértices; e o vértice restante seria o corner oposto.
- Para preencher a sétima e oitava coluna é somente usar o fato que o corner da direita de c_j é o oposto do corner posterior de c_j e o corner da esquerda de c_j é o oposto do corner anterior de c_j .

3 Resultados

Os resultados que obtive foi aplicando o código para tres exemplos diferentes:

A. Exemplo 1

Para este caso, temos o arquivo vtk chamado exemplosimple.vtk e obtive como resultado a seguinte corner-table (tabela 2):

Figura 1: Exemplo1

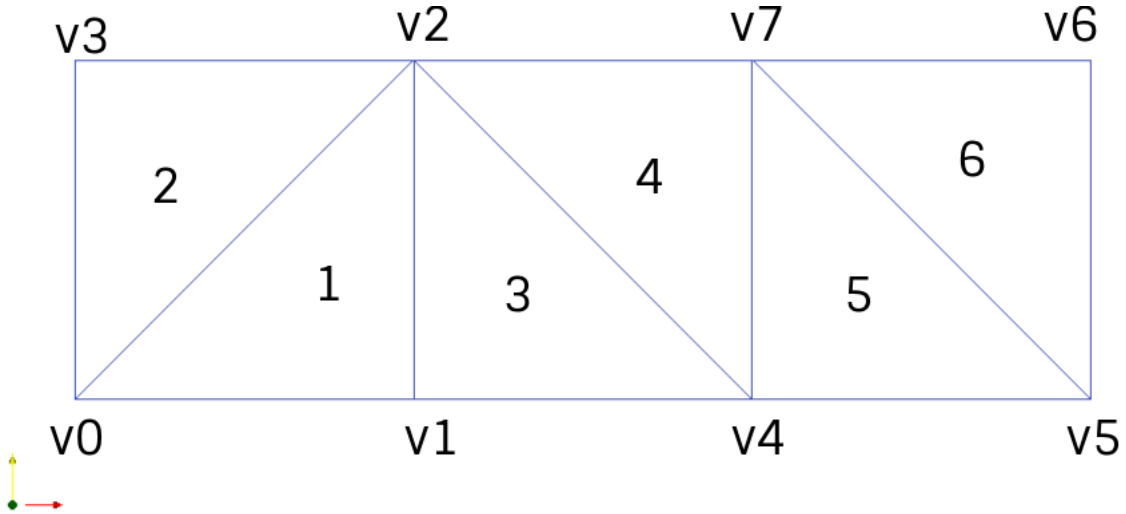


Tabela 2: Corner-Table para exemplo 1

Corner	Vert	Face	C.n	C.p	C.o	C.r	C.l
1	0	1	2	3	8	6	0
2	1	1	3	1	6	0	8
3	2	1	1	2	0	8	6
4	0	2	5	6	0	0	2
5	2	2	6	4	0	2	0
6	3	2	4	5	2	0	0
7	1	3	8	9	11	1	0
8	4	3	9	7	1	0	11
9	2	3	7	8	0	11	1
10	4	4	11	12	0	7	14
11	7	4	12	10	7	14	0
12	2	4	10	11	14	0	7
13	4	5	14	15	17	12	0
14	5	5	15	13	12	0	17
15	7	5	13	14	0	17	12
16	5	6	17	18	0	13	0
17	6	6	18	16	13	0	0
18	7	6	16	17	0	0	13

Onde o 0 nas colunas 6, 7 e 8 representa \emptyset .

- B. **Exemplo 2** Para este caso, se tem o arquivo vtk com nome exemplo2.vtk como segue na figura 2, aplicamos tambem o código corner_table.m e obtive a tabela 3

Figura 2: Exemplo2

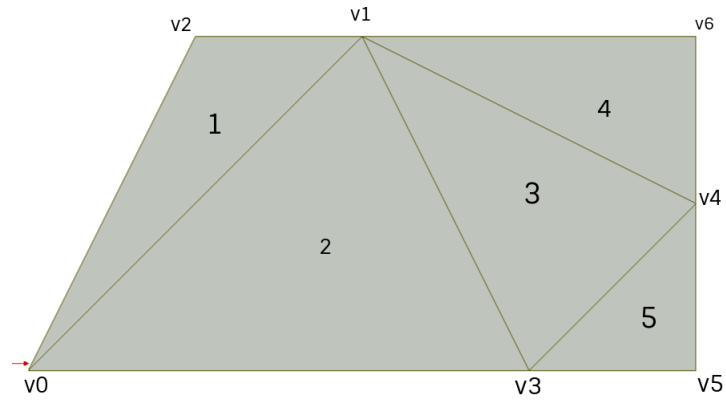
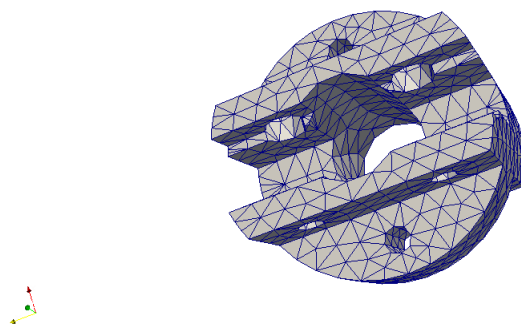


Tabela 3: Corner-Table do exemplo 2

Corner	Vert	Face	C.n	C.p	C.o	C.r	C.l
1	0	1	2	3	0	0	6
2	1	1	3	1	0	6	0
3	2	1	1	2	6	0	0
4	1	2	5	6	0	9	3
5	0	2	6	4	9	3	0
6	3	2	4	5	3	0	9
7	1	3	8	9	14	12	5
8	3	3	9	7	12	5	14
9	4	3	7	8	5	14	12
10	1	4	11	12	0	0	8
11	4	4	12	10	0	8	0
12	6	4	10	11	8	0	0
13	3	5	14	15	0	7	0
14	5	5	15	13	7	0	0
15	4	5	13	14	0	0	7

- C. **Exemplo 3** Para este exemplo, se tem como dado inicial o arquivo socket.vtk dado pelo professor cuja figura está dado por:

Figura 3: Exemplo 3



Aplicando o código obtemos a tabela que pode ser vista no documento pdf com nome cuadroct_trabalh3.pdf

3.1 Segunda Parte do Trabalho

A segunda parte do trabalho é que dado um vértice qualquer, identificar e mostrar o anél e as faces (ou triângulos) que o tem como vértice.

Para essa tarefa se fez uma função em Matlab chamado `anel_vert.m` onde o usuário ingresa o vértice a consultar e a função entrega dois vetores, um com os índices dos vértices do anél e outro vetor com a numeração das faces que o tem como vértice. O que basicamente faz este código é dada um vértice v usando a `corner table` encontrada anteriormente, e procura na coluna dos vértices que tantas vezes se repete v , depois procura em cada na qual pertence v os `corners` posteriores e anteriores a v ; por consequência pode-se encontrar os vértices associados a cada `corner`.

A. Para o exemplo 1, demos como vertice a analizar o v2, então teriamos como resultado

Figura 4: Aplicando `anel_vert(2)`

```
Command Window
>> [Anel, Triangulos]=anel_vert(2)

Anel =

     0
     1
     3
     4
     7

Triangulos =

     1
     2
     3
     4
```

E isso pode ser verificado com a mesma `corner-table` deste exemplo (tabela 2).

B. Para o exemplo 2, demos como vertice a analizar o v3, então teriamos como resultado E isso pode ser

Figura 5: Aplicando `anel_vert(3)`

```
Command Window
>> [Anel, Triangulos]=anel_vert(3)

Anel =

     0
     1
     4
     5

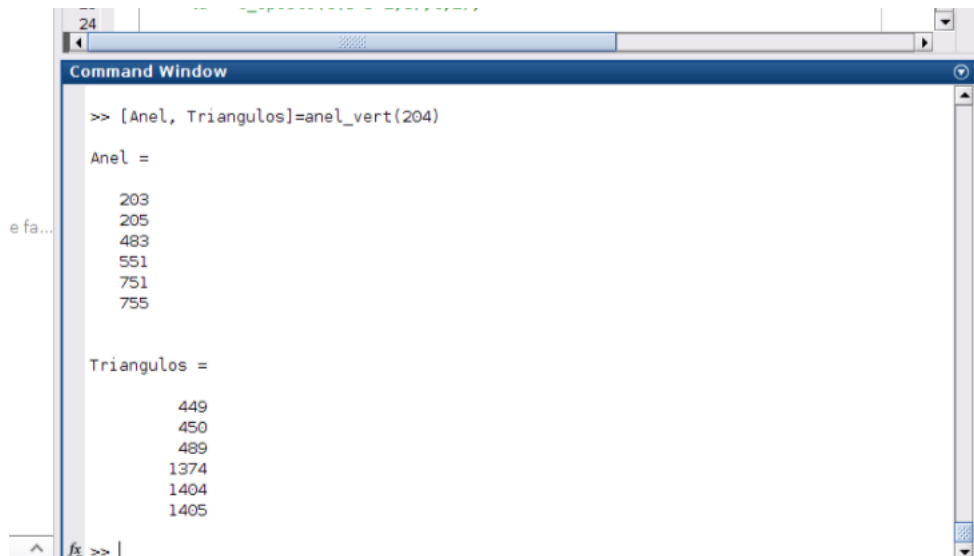
Triangulos =

     2
     3
     5
```

verificado com a mesma `corner-table` deste exemplo (tabela 3).

C. Para o exemplo 3, demos como vertice a analizar o v204, então teriamos como resultado

Figura 6: Aplicando `anel_vert(204)`



The screenshot shows a MATLAB Command Window with the following text:

```
>> [Anel, Triangulos]=anel_vert(204)

Anel =

    203
    205
    483
    551
    751
    755

Triangulos =

    449
    450
    489
   1374
   1404
   1405
```

4 Conclusão

- A corner-table é possível criar, com um pouco de trabalho, mas depois de ter sido criada é fácil fazer as consultas acerca de anel e dos triângulos (ou faces) que contem algum vertice especifico.
- Deve-se tratar de criar o código o mais otimizado possível pois quando a quantidade de vertices é muita como no exemplo 3, demora mais tempo para calcular a corner-table e os dados requeridos na segunda parte do trabalho.