Geração de Malha

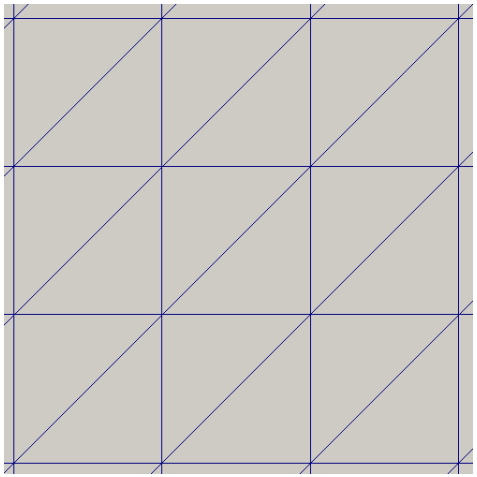
Ramoni Zancanela Sedano
ramoni.zsedano@gmail.com

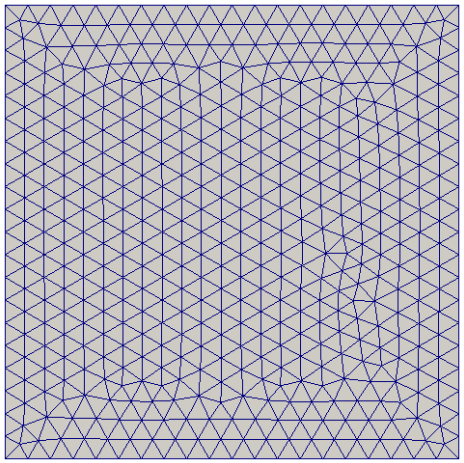
Tópicos Avançados em Elementos Finitos
Programa de Pós-Graduação em Informática
Centro Tecnológico
Universidade Federal do Espírito Santo

Sumário

Tipos de malha

Podemos discretizar o domínio em elementos iguais ou elementos de tamanhos variados. Para a malha com elementos iguais, trabalharemos com elementos triangulares com dois tipos de inclinação, orientada pra direita e esquerda. Para a malha de elementos variados, utilizamos um gerador chamado EasyMesh.

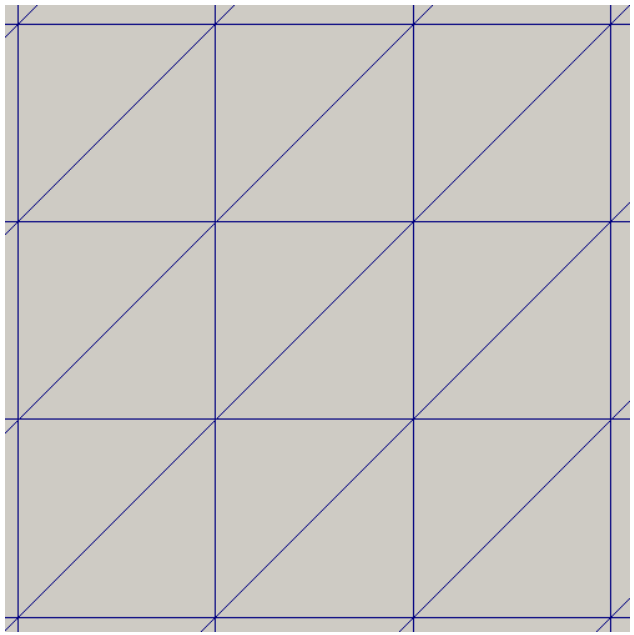




Orientada pra direita

Esse código gera malha orientada para a direita:

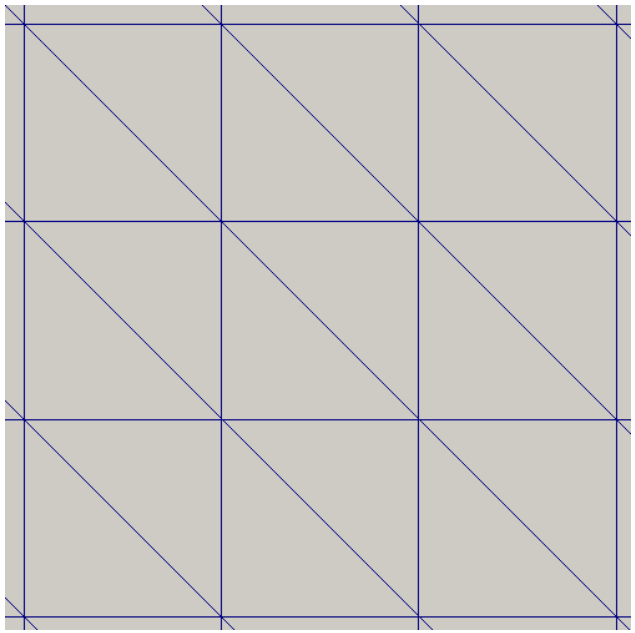
```
no1 = 0;
no2 = 0;
no3 = 0;
j = 0;
k = (2 * nosx) - 2; //elementos em x
for (e = 1; e <= nel; e++){
    j++;
    if (e % 2 == 1){ //elemento impar
        no1++;
        no2 = no1 + 1;
        no3 = no2 + nosx;
    }else{ // elemento par
        no2 = no3;
        no3 = no2 - 1;
    }
    ien[e - 1][0] = no1;
    ien[e - 1][1] = no2;
    ien[e - 1][2] = no3;
    if (j == k){
        no1++;
        j = 0;
    }
}
```



Orientada pra esquerda

Esse código gera malha orientada pra esquerda:

```
no1 = 1;
no2 = 0;
no3 = 0;
j = 0;
k = (2 * nosx) - 2;  //elementos em x
for (e = 1; e <= nel; e++){
    j++;
    if (e \% 2 == 1){ //elemento impar
        no1 = no1;
        no2 = no1 + 1;
        no3 = no1 + nosx;
    }else{  //elemento par
        no1 = no2;
        no2 = no3 + 1;
        no3 = no2 - 1;
    }
    ien[e-1][0] = no1;
    ien[e-1][1] = no2;
    ien[e-1][2] = no3;
    if (j == k){
        no1++;
        j = 0;
    }
}
```



Arquivo de entrada .d

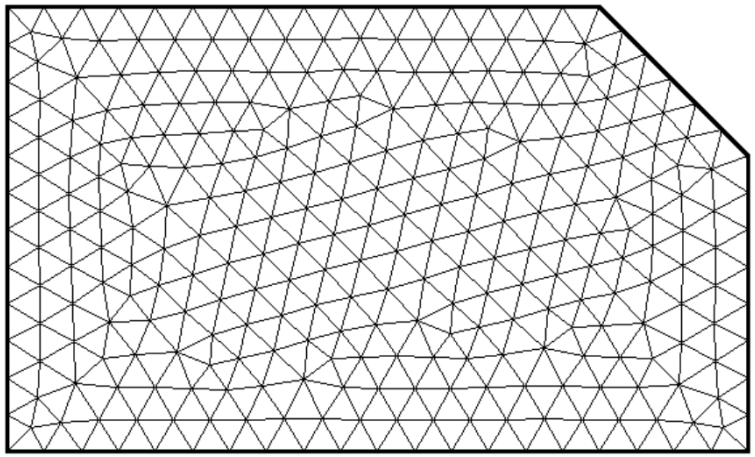
O EasyMesh lê um arquivo de entrada com extensão .d no seguinte formato:

- Primeira linha: <número de nós>
- Linhas seguintes: <número do nó:> <x> <y> <lado do triângulo> <marca do contorno>
- Próxima linha: <número de segmentos>
- Linhas seguintes: <número do segmento:> <ponto inicial> <ponto final> <marca do contorno>

Exemplo

5 # número de nós #
nós definem o contorno
0: 0.0 0.0 0.25 1
1: 5.0 0.0 0.25 2
2: 5.0 2.0 0.25 2
3: 4.0 3.0 0.25 3
4: 0.0 3.0 0.25 3

5 # número de segmentos #
segmentos do contorno
0: 0 1 1
1: 1 2 2
2: 2 3 2
3: 3 4 3
4: 4 0 3



Arquivos de saída

O EasyMesh gera três arquivos de saída com as seguintes extensões:

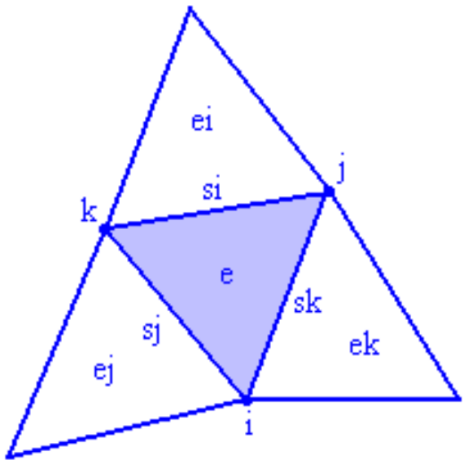
- .n arquivo nó
- .e arquivo elemento
- .s arquivo segmento

O arquivo nó (.n) tem o seguinte formato:

- Primeira linha: <número de nós>
- Linhas seguintes: <número do nó> <x> <y> <marca>
- As duas últimas linhas são comentários inseridos pelo programa para facilitar a leitura do arquivo nó.

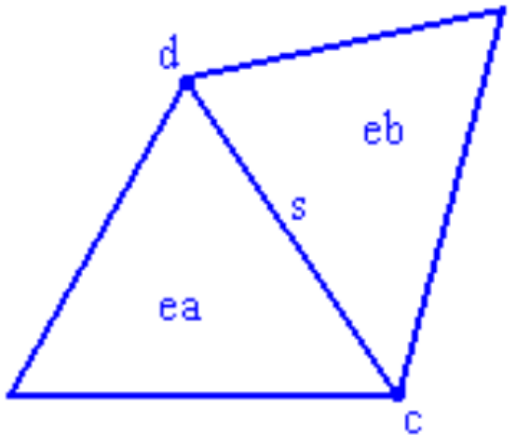
O arquivo elemento (.e) tem o seguinte formato:

- Primeira linha: <número de elementos>
- Linhas seguintes: <número do elemento> <i> <j> <k> <ei> <ej> <ek> <si> <sj> <sk> <xV> <yV> <marca>
- As duas últimas linhas são comentários inseridos pelo programa para facilitar a leitura do arquivo nó.



O arquivo segmento (.s) tem o seguinte formato:

- Primeira linha: <número de segmento>
- Linhas seguintes: <número de segmentos> <c> <d> <ea> <eb> <marca>
- As duas últimas linhas são comentários inseridos pelo programa para facilitar a leitura do arquivo segmento.



Executando EasyMesh e ShowMesh

No terminal de um computador rodando Linux, utilizamos o comando

```
gcc easymesh.c -o easymesh -lm
```

para compilar e o comando

```
./easymesh exemplo.d
```

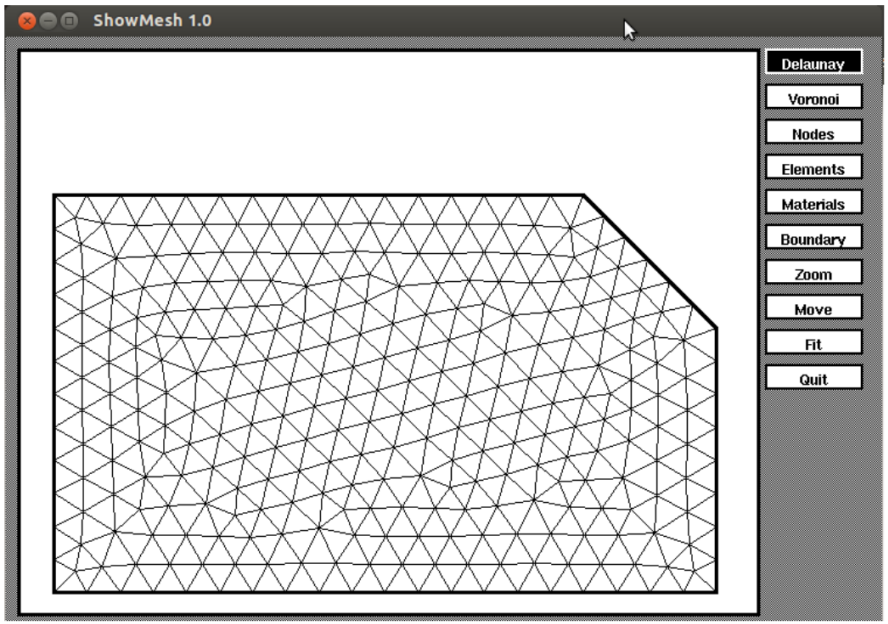
para executar o easyMesh.

Para visualizar das malhas geradas pelo EasyMesh utilizamos o ShowMesh, que pode ser compilado utilizando o comando

gcc showmesh.c -o showmesh -lX11

e executado com o comando

./showmesh exemplo



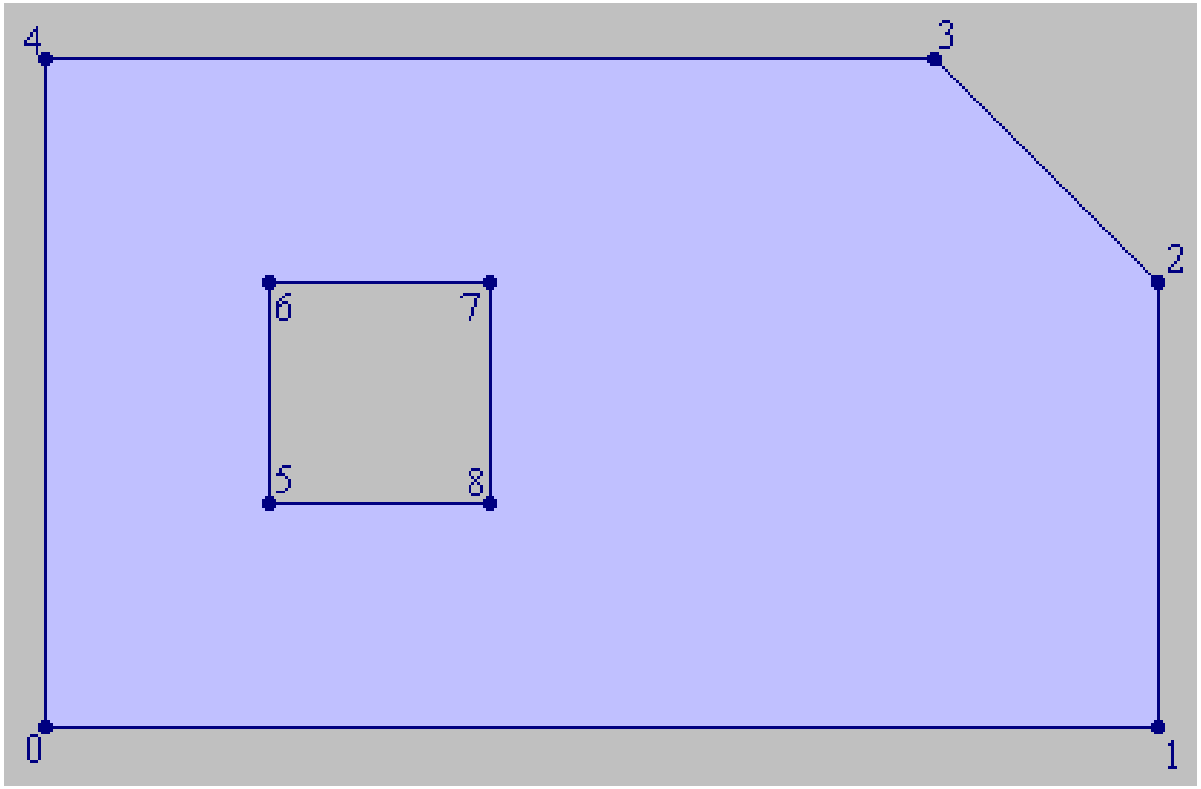
Pode ocorrer um erro ao compilar o ShowMesh

```
ramoni@ramoni:~/Dropbox/Apresentação MEF$ gcc showmeshCERTO.c -o show -lX11
showmeshCERTO.c:34:22: fatal error: X11/Xlib.h: Arquivo ou diretório não encontrado
 #include <X11/Xlib.h>
                      ^
compilation terminated.
ramoni@ramoni:~/Dropbox/Apresentação MEF$
```

Isso ocorre por que o pacote X11 não está instalado, então executamos o seguinte comando no terminal

sudo apt-get install libX11-dev

Exemplo 1



Exemplo 1

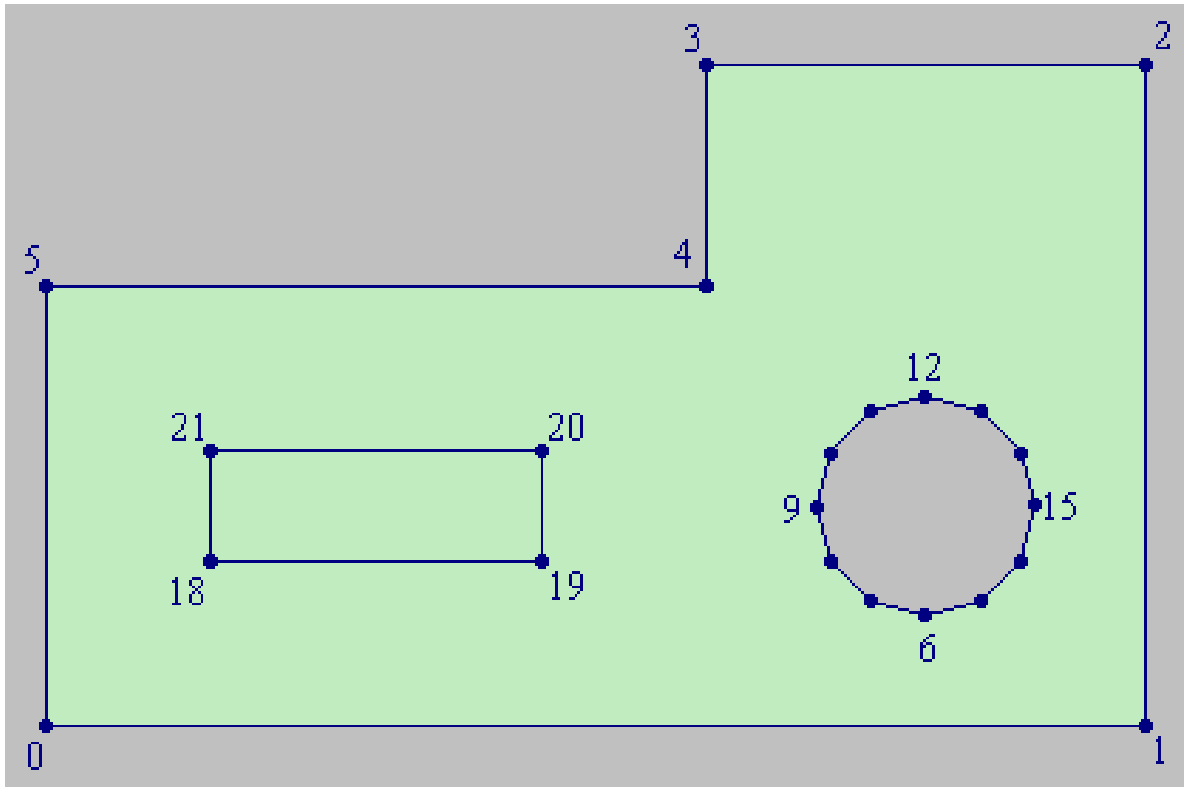
9 # numero de pontos #
Nós definem o contorno
0: 0.0 0.0 0.25 1
1: 5.0 0.0 0.25 2
2: 5.0 2.0 0.25 2
3: 4.0 3.0 0.25 3
4: 0.0 3.0 0.25 3

Nós definem o furo
5: 1.0 1.0 0.1 4
6: 1.0 2.0 0.1 4
7: 2.0 2.0 0.1 4
8: 2.0 1.0 0.1 4

9 # Numero de segmentos #
Segmentos do Contorno
0: 0 1 1
1: 1 2 2
2: 2 3 2
3: 3 4 3
4: 4 0 3

Segmentos do furo
5: 5 6 4
6: 6 7 4
7: 7 8 4
8: 8 5 4

Exemplo 2

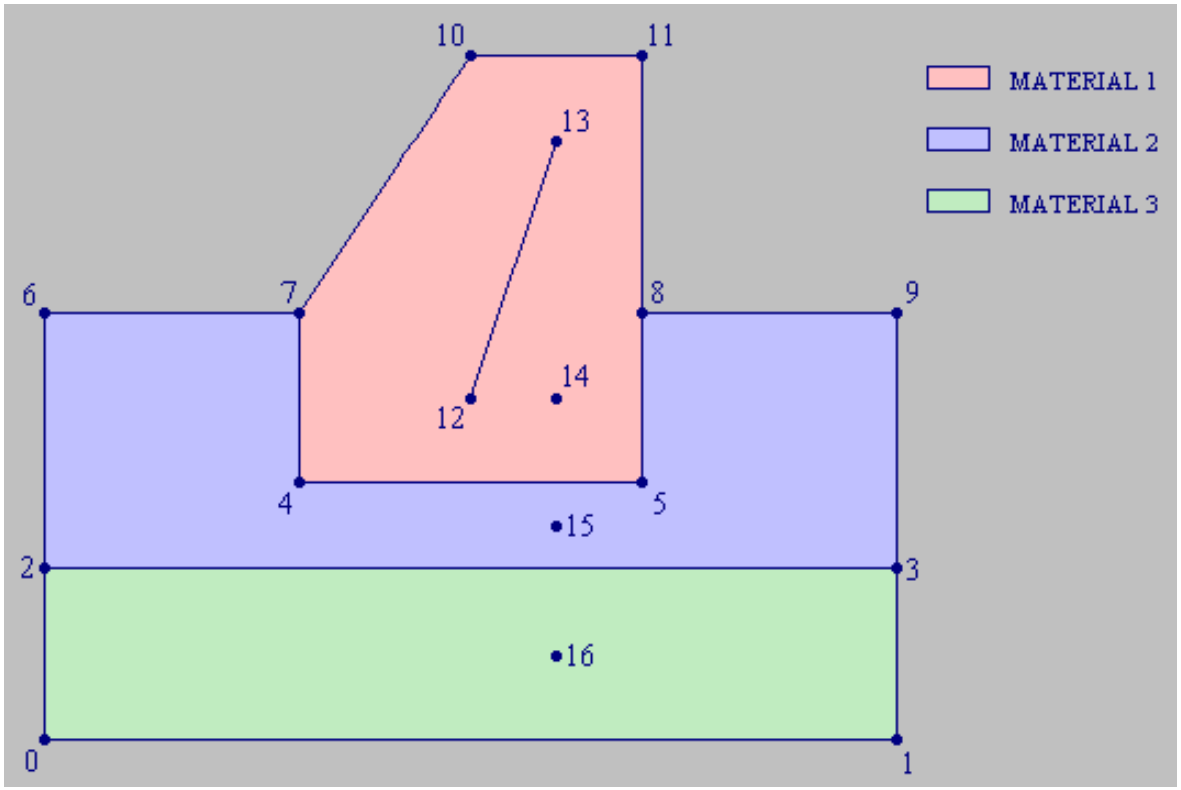


Exemplo 2

```
22 # numero de pontos #
# Nós definem o contorno #
0: 0 0 0.2 1
1: 5 0 0.2 1
2: 5 3 0.2 1
3: 3 3 0.2 1
4: 3 2 0.2 1
5: 0 2 0.2 1

# Nós definem o furo #
6: 4.00 0.50 10.0 2
7: 3.75 0.567 10.0 2
8: 3.567 0.75 10.0 2
9: 3.50 1.00 10.0 2
10: 3.567 1.25 10.0 2
11: 3.75 1.433 10.0 2
12: 4.00 1.50 10.0 2
13: 4.25 1.433 10.0 2
14: 4.433 1.25 10.0 2
15: 4.50 1.00 10.0 2
16: 4.433 0.75 10.0 2
17: 4.25 0.567 10.0 2

# Nós definem o falso furo #
18: 0.75 0.75 0.1 0
19: 2.25 0.75 0.1 0
20: 2.25 1.25 0.1 0
21: 0.75 1.25 0.1 0
```

```
22 # Numero de segmentos #
# Segmentos do Contorno #
0: 0 1 1
1: 1 2 1
2: 2 3 1
3: 3 4 1
4: 4 5 1
5: 5 0 1

# Segmentos do furo #
6: 6 7 2
7: 7 8 2
8: 8 9 2
9: 9 10 2
10: 10 11 2
11: 11 12 2
12: 12 13 2
13: 13 14 2
14: 14 15 2
15: 15 16 2
16: 16 17 2
17: 17 6 2

# Segmentos falso furo #
18: 18 19 0
19: 19 20 0
20: 20 21 0
21: 21 18 0
```

Exemplo 3


10
11
13
MATERIAL 1
MATERIAL 2
MATERIAL 3
6
7
8
9
14
12
4
5
15
2
3
16
0
1

Exemplo 3

```
17 # numero de pontos #
# Nós definem o contorno #
0: 0 0 0.5 1
1: 5 0 0.5 1
2: 0 1 0.1 1
3: 5 1 0.1 1
4: 1.5 1.5 0.1 4
5: 3.5 1.5 0.1 4
6: 0 2.5 0.25 1
7: 1.5 2.5 0.1 2
8: 3.5 2.5 0.1 2
9: 5 2.5 0.25 1
10: 2.5 4 0.1 2
11: 3.5 4 0.1 2

# linha de engrossamento ou refinamento #
12: 2.5 2 0.4 0
13: 3 3.5 0.4 0

# marca do material #
14: 3 3 0 1 # material 1 #
15: 3 1.25 0 2 # material 2 #
16: 3 0.5 0 3 # material 3 #
```

```
15 # Numero de segmentos #
# Segmentos do Contorno #
0: 0 1 1
1: 1 3 1
2: 3 9 1
3: 9 8 1
4: 8 11 2
5: 11 10 2
6: 10 7 2
7: 7 6 1
8: 6 2 1
9: 2 0 1

# fronteira entre verde e azul #
10: 3 2 3

# fronteira entre azul e vermelho #
11: 8 5 4
12: 5 4 4
13: 4 7 4
14: 12 13 0
```

ParaView

O ParaView é um aplicando livre para a visualização e análise de conjuntos de dados científicos, principalmente àqueles que são definidos em um espaço de duas ou três dimensões, incluindo os que se estendem na dimensão temporal.

Para visualização da solução, o ParaView executa um script com extenção .vtu da seguinte forma:

```
<VTKFile type="UnstructuredGrid" version="0.1" byte_order="BigEndian">
  <UnstructuredGrid>
    <Piece NumberOfPoints="nnos" NumberOfCells="nel">
      <PointData Scalars="scalars">
        <DataArray type="Float32" Name="solucao" Format="ascii">
        . . .
        </DataArray>
      </PointData>
      <Points>
        <DataArray type="Float32" NumberOfComponents="3" Format="ascii">
        . . .
        </DataArray>
      </Points>
      <Cells>
        <DataArray type="Int32" Name="connectivity" Format="ascii">
        . . .
        </DataArray>
        <DataArray type="Int32" Name="offsets" Format="ascii">
        . . .
        </DataArray>
        <DataArray type="Int32" Name="types" Format="ascii">
        . . .
        </DataArray>
      </Cells>
    </Piece>
  </UnstructuredGrid>
</VTKFile>
```

Onde

- *NumberOfPoints* é o número de nós da malha e *NumberOfCells* é o número de elementos;
- *PointData Scalars="scalars"* indica que um valor escalar está sendo associado aos nós e

  Name="solucao"é o nome dos dados encontrados, pode ser densidade, temperatura, pressão.

  Aqui é colocado a solução associada a cada nó.
- Em *Points*, *NumberOfComponents="3"* indica quantas componentes estão envolvidas, no caso bidimensional teremos 3 componentes, os valores de x, y e solução.
- Em *Cells*, *connectivity* será preenchida com a conectividade dos elementos, a matriz IEN

  offsets é preenchido com números múltiplos de 3 para cada elemento;

  types é preenchido com 5 para cada elemento.