

EasyMesh, ShowMesh e ParaView

Ramoni Zancanela Sedano

ramoni.zsedano@gmail.com

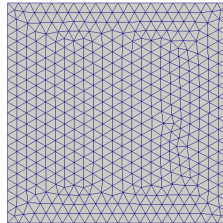
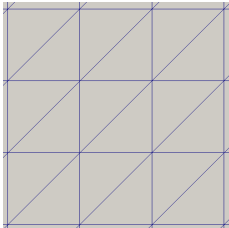
Elementos Finitos
Programa de Pós-Graduação em Informática
Centro Tecnológico
Universidade Federal do Espírito Santo

Sumário

- 1 Introdução
- 2 Malha Estruturada
 - Orientada à direita
 - Orientada à esquerda
- 3 Malha Não Estruturada
 - EasyMesh e ShowMesh
- 4 Visualizando solução

Tipos de malha

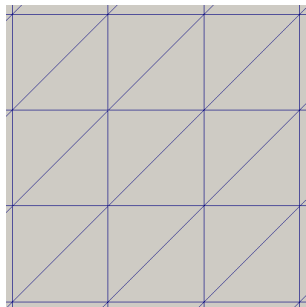
Podemos discretizar o domínio em elementos iguais ou elementos de tamanhos variados. Para a malha com elementos iguais, trabalharemos com elementos triangulares com dois tipos de inclinação, orientada pra direita ou esquerda. Para a malha de elementos variados, utilizamos um gerador chamado EasyMesh.



Orientada à direita

Esse código, em linguagem C, gera malha orientada para a direita:

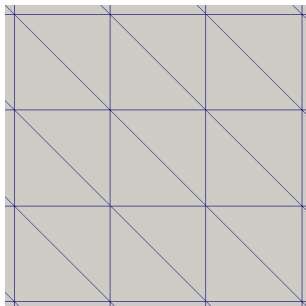
```
no1 = 0;
no2 = 0;
no3 = 0;
j = 0;
k = (2 * nosx) - 2; // Número de elementos em x
for (e = 1; e <= nel; e++){
    j++;
    if (e % 2 == 1){ // Elemento de número ímpar
        no1++;
        no2 = no1 + 1;
        no3 = no2 + nosx;
    }else{ // Elemento de número par
        no2 = no3;
        no3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



Orientada à esquerda

Esse código, em linguagem C, gera malha orientada para a esquerda:

```
no1 = 1;
no2 = 0;
no3 = 0;
j = 0;
k = (2 * nosx) - 2; // Número de elementos em x
for (e = 1; e <= nel; e++){
    j++;
    if (e % 2 == 1){ // Elemento de número ímpar
        no1 = no1;
        no2 = no1 + 1;
        no3 = no1 + nosx;
    }else{ // Elemento de número 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;
    }
}
```



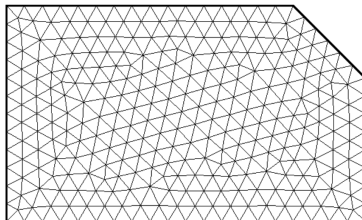
Arquivo de entrada .d

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 que 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.

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
```

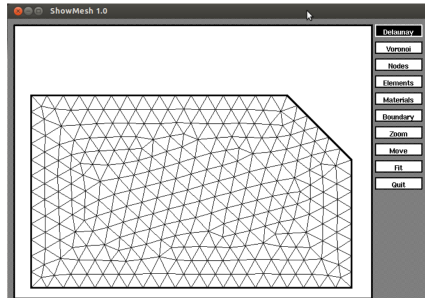
para executar o easyMesh.

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
```



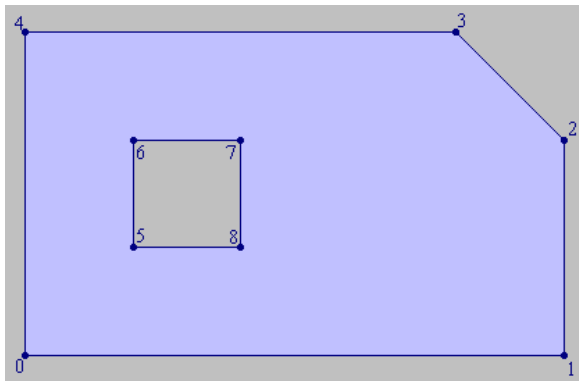
Pode ocorrer um erro ao compilar o ShowMesh

```
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$
```

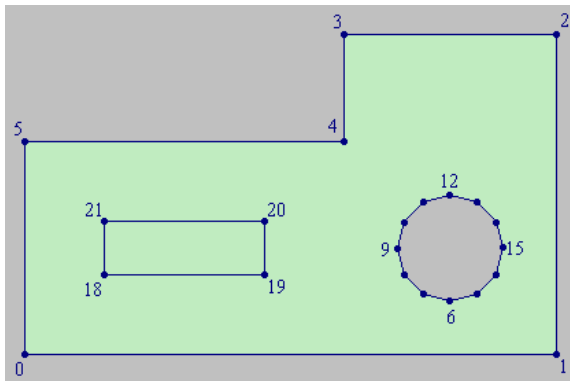
Isso ocorre por que o pacote X11 não está instalado, então executamos o seguinte comando no terminal

```
sudo apt-get install libX11-dev
```

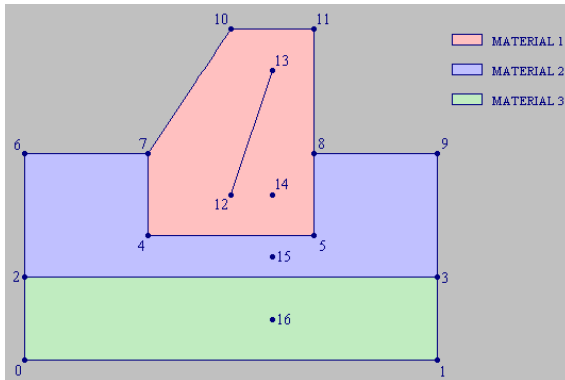
Exemplo 1



Exemplo 2



Exemplo 3



ParaView

O ParaView é um aplicativo 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 extensão .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.