FEPiC++

Felipe Montefuscolo

6 de julho de 2011

1 Numeração dos elementos

1.1 Tetraedro

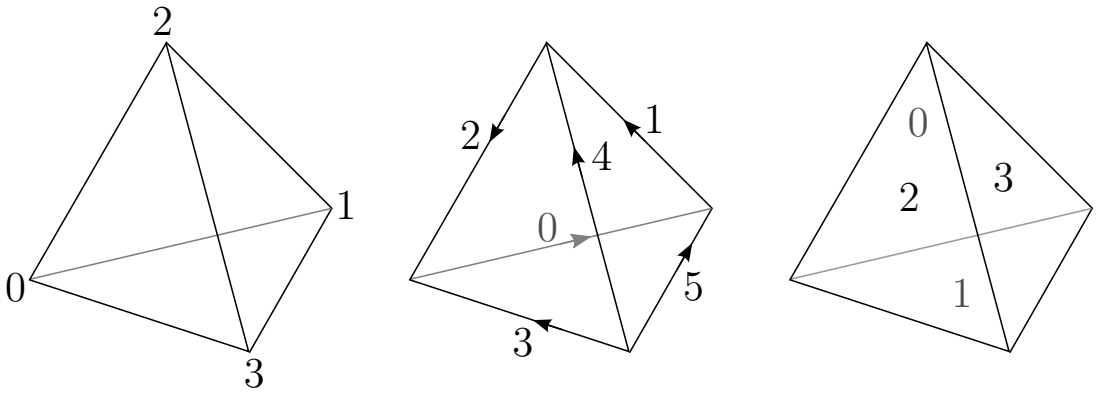

Figura 1

edge	facet-in	facet-out	pos facet-in	pos facet-out
0	0	1	-0	+0
1	0	3	-2	+2
2	0	2	-1	+1
3	2	1	-2	+2
4	3	2	-0	+0
5	1	3	-1	+1

(a) `edges_x_facets`

facet	edge 0	edge 1	edge 2
0	-0	-2	-1
1	+0	-5	+3
2	+4	+2	-3
3	-4	+5	+1

(b) `facets_x_edges`

Tabela 1

vertex	face-0	face-1	face-2	pos-f0	pos-f1	pos-f2
0	0	1	2	1	0	2
1	0	1	3	0	1	2
2	0	2	3	2	1	0
3	1	2	3	2	0	1

Tabela 2: `vertices_x_facets`

1.2 Hexaedro

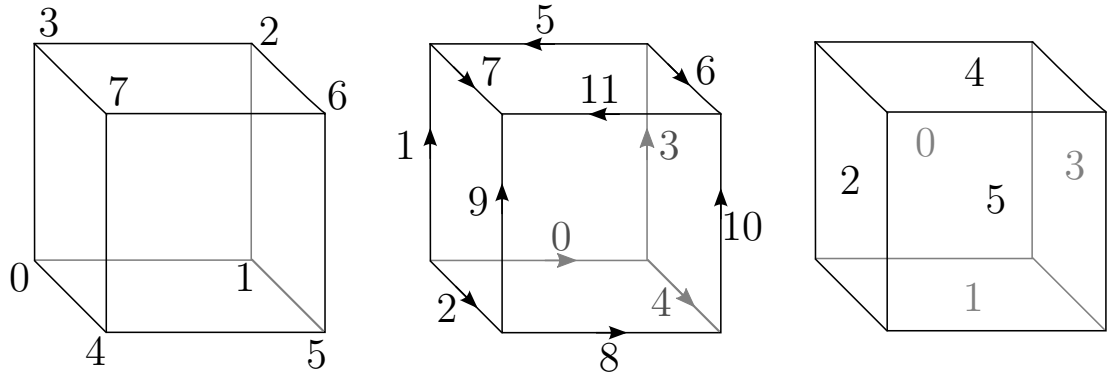

Figura 2

edge	facet-in	facet-out	pos facet-in	pos facet-out
0	0	1	-3	+0
1	2	0	-3	+0
2	1	2	-3	+0
3	0	3	-2	+0
4	3	1	-3	+1
5	0	4	-1	+0
6	4	3	-3	+1
7	2	4	-2	+1
8	1	5	-2	+0
9	5	2	-3	+1
10	3	5	-2	+1
11	4	5	-2	+2

(a) `edges_x_facets`

face	edge 0	edge 1	edge 2	edge 3
0	+1	-5	-3	-0
1	+0	+4	-8	-2
2	+2	+9	-7	-1
3	+3	+6	-10	-4
4	+5	+7	-11	-6
5	+8	+10	+11	-9

(b) `facets_x_edges`

Tabela 3

vertex	face-0	face-1	face-2	pos-f0	pos-f1	pos-f2
0	0	1	2	0	0	0
1	0	1	3	3	1	0
2	0	3	4	2	1	0
3	0	2	4	1	3	1
4	1	2	5	3	1	0
5	1	3	5	2	3	1
6	3	4	5	2	3	2
7	2	4	5	2	2	3

Tabela 4: `vertices_x_facets`

1.3 Triângulo/Quadrângulo

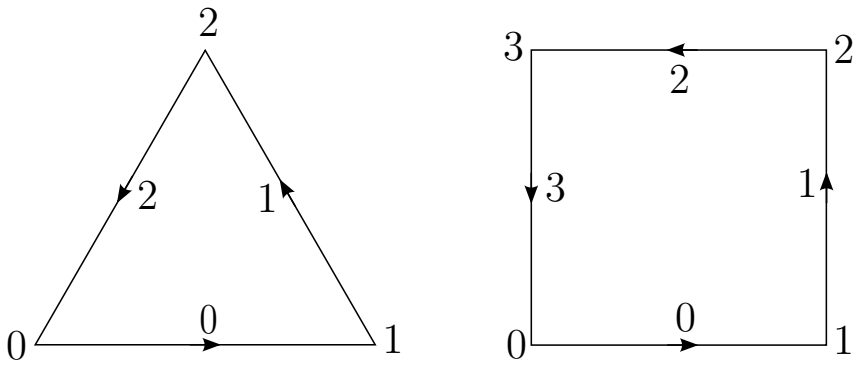

Figura 3

vertex	face-0	face-1
v	v	(`n_facets` + v − 1) % `n_facets`

ou

vertex	face-s
v	(`n_facets` + v − s) % `n_facets`

Tabela 5: `vertices_x_facets`

facet	vertex-0	vertex-1
f	f	(f+1) % `n_facets`

ou

facet	vertex-q
f	(f+q) % `n_facets`

Tabela 6: `facets_x_vertices`

2 Algoritmos

lid = local id
gid = global id

2.1 Edge star

```
RotateAroundEdge(C, eC, s, &D, &eD, &q)
input:
  C: cell global id
  eC: local id of edge in C
  s: wich direction to rotate (face-in: 0; face-out: 1)
output:
  D: cell global id
  eD: local id of edge in D
  q: wich direction to rotate (face-in: 0; face-out: 1)
/* Obs: RotateAroundEdge(D, eD, q, ...) don't return C, eC and S. Never. */

f = edges_x_facets(eC, s) //f é a facet de C adjacente a D
(D, g, anc) = i(C, f) // g é a facet de D adjacente a C;
se D==-1 retorn // não tem célula por esse sentido
eCf = edges_x_facets(eC, s+2) // posição de eC em relação a f
eDg = (6-eCf-anc)%4   (se hexaedro)
    = (4-eCf-anc)%3   (se tetraedro)
eD  = faces_x_edges(g, eDg)
if (edges_x_facets(eD,0)==g)
  q=1
else
  q=0

fim
================================================================================

V = EdgeStar(C, eC)
input:
  C: cell global id
  eC: local id of edge in C
output
  V: matriz com as células incidentes à aresta e suas posições com relação a elas.

D=C, eD=eC, q=0, s=0, V.push(D, eD)
do
  RotateAroundEdge(D, eD, q, &D, &eD, &q)
  if (D==C)
    return V
  if (D<0 && s==0)
    D=C, eD=eC, q=1, s=1, continue
  V.push(D, eD)
while (true)

================================================================================
para 3D
VertexStarStep0(C, vC, &M, &counter)
input:
```

```
  C: cell global id
  vC: local id of the vertex in C
output:
  M: matriz com as células incidentes ao vértice e as posições.
  counter: contador de células

marca a célula C;
M.push(C, vC);
counter++;
para cada face f de C incidente ao vértice
  (D, g, anc) = i(C,f)
  se D>=0
    vCf = vertices_x_facets(vC, f+3) // posição do vértice em relação a f
    vDg = (5-vCf-anc)%3 (tet)
        = (7-vCf-anc)%4 (hex)
    vD  = facets_x_vertices(g, vDg)
    VertexStarTemp(D, vD, &M, &counter)

end
================================================================================
para 3D
V = VertexStar(C, vC)

M = zeros(2,50);
counter=0;
VertexStarStep0(C, vC, &M, &counter)
M.resize(2,counter);
return M;

================================================================================
para 2D
VertexStarSetp0(C, vC, &M, &counter)
M.push(C, vC)
counter++
q=0;
D=C, vD=vC
do
  g=(n_facets + vD - q)%n_facets;  // face por onde se vai girar
  (D,g) ⟵ i(D,g) // gira
  se D<0           // se não tem célula
    se q==1 break;
    se q==0
      D=C, vD=vC, q=1, continue
  vD = (g + 1 - q) % n_facets
  M.push(D, vD)
  counter++
while (true)
```