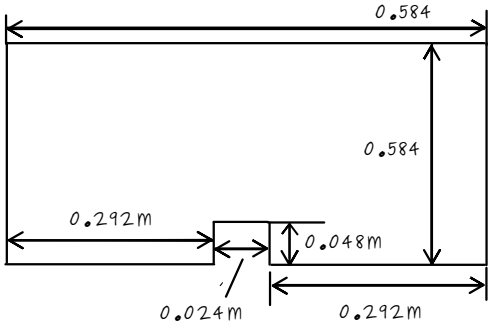
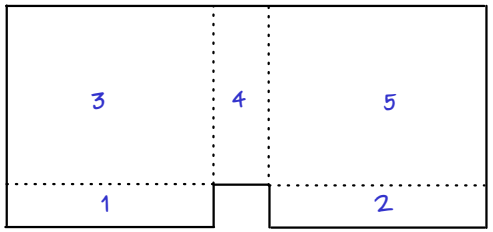
Exemplo de construção de malha no BlockMeshDict (Dicionário de malha em bloco)

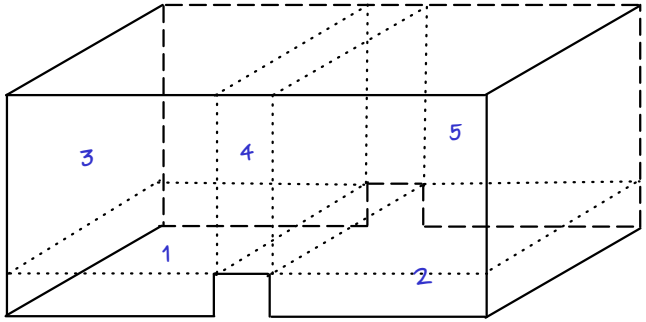
Domínio de cálculo



Blocos

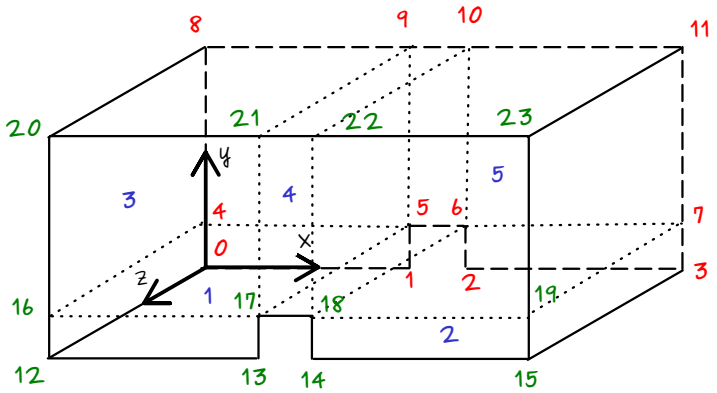


Representação 3D



Bloco	Vértices
1	0, 1, 5, 4, 12, 13, 17, 16
2	2, 3, 7, 6, 14, 15, 19, 18
3	4, 5, 9, 8, 16, 17, 21, 20
4	5, 6, 10, 9, 17, 18, 22, 21
5	6, 7, 11, 10, 18, 19, 23, 22

Vértices



Como isso é representado no arquivo blockMeshDict?

1 – Especificamos as coordenadas (x,y,z) de cada vértice inicialmente;
2 – Posteriormente definimos os vértices contidos em cada bloco, definindo o número de nós e progressão de refinamento em cada direção dentro de cada bloco.

Definindo a posição de cada vértice para o exemplo anterior...

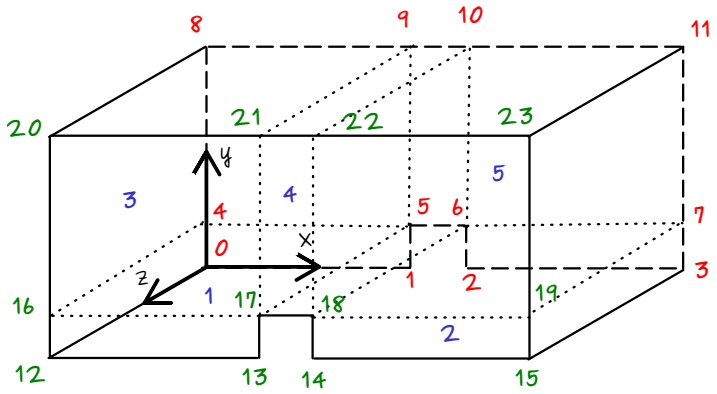
```
convertToMeters 0.146;

vertices
(
    (0 0 0)
    (2 0 0)
    (2.16438 0 0)
    (4 0 0)
    (0 0.32876 0)
    (2 0.32876 0)
    (2.16438 0.32876 0)
    (4 0.32876 0)
    (0 4 0)
    (2 4 0)
    (2.16438 4 0)
    (4 4 0)
    (0 0 0.1)
    (2 0 0.1)
    (2.16438 0 0.1)
    (4 0 0.1)
    (0 0.32876 0.1)
    (2 0.32876 0.1)
    (2.16438 0.32876 0.1)
    (4 0.32876 0.1)
    (0 4 0.1)
    (2 4 0.1)
    (2.16438 4 0.1)
    (4 4 0.1)
);
```

Definindo os vértices contidos em cada bloco junto com o número de nós e progressão de refinamento em cada direção

```
blocks
(
    hex (0 1 5 4 12 13 17 16) (23 8 1) simpleGrading (1 1 1)
    hex (2 3 7 6 14 15 19 18) (19 8 1) simpleGrading (1 1 1)
    hex (4 5 9 8 16 17 21 20) (23 42 1) simpleGrading (1 1 1)
    hex (5 6 10 9 17 18 22 21) (4 42 1) simpleGrading (1 1 1)
    hex (6 7 11 10 18 19 23 22) (19 42 1) simpleGrading (1 1 1)
);
```

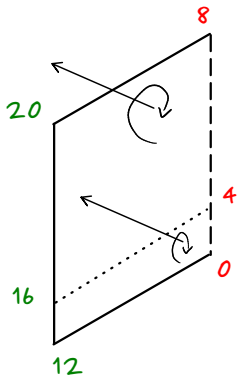
Finalmente, precisamos definir as faces das fronteiras do domínio de cálculo em termo de seus vértices. Isso será útil para a imposição das condições de contorno do problema.



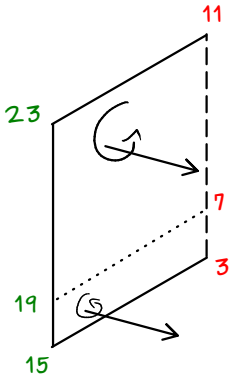
Temos aqui as seguintes fronteiras:

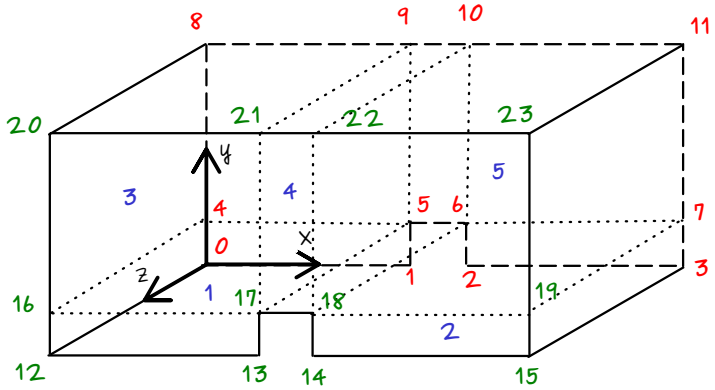
- parede da esquerda
- parede da direita
- parede inferior
- fronteira superior (aberta à atmosfera)

Como isso é representado no arquivo blockMeshDict?

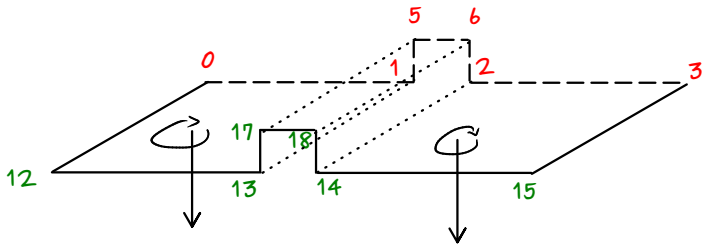



```
boundary
(
    leftWall
    {
        type wall;
        faces
        (
            (0 12 16 4)
            (4 16 20 8)
        );
    }
    rightWall
    {
        type wall;
        faces
        (
            (7 19 15 3)
            (11 23 19 7)
        );
    }
```

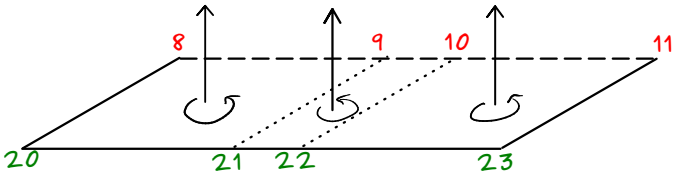





```
lowerWall
    {
        type wall;
        faces
        (
            (0 1 13 12)
            (1 5 17 13)
            (5 6 18 17)
            (2 14 18 6)
            (2 3 15 14)
        );
```




```
atmosphere
    {
        type patch;
        faces
        (
            (8 20 21 9)
            (9 21 22 10)
            (10 22 23 11)
        );
```



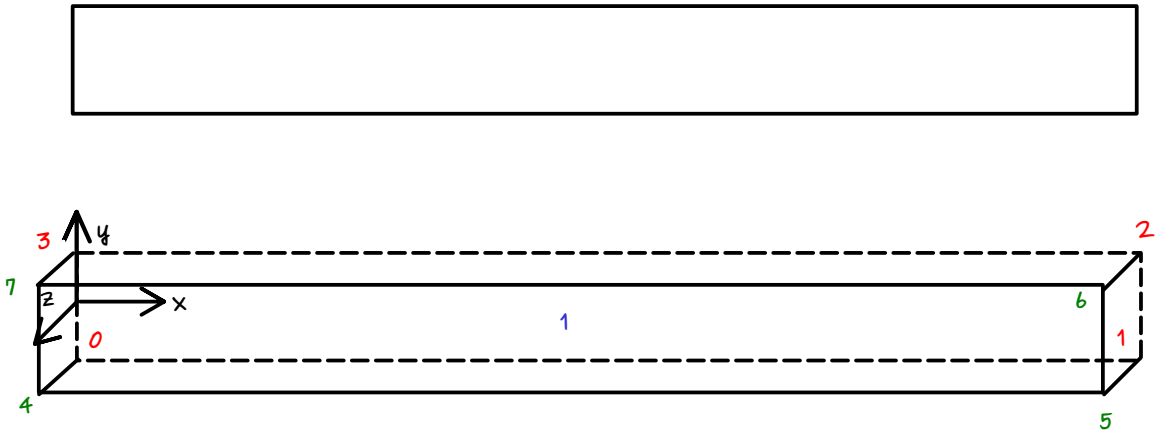
Regra de numeração

1 – O sentido da ordem dos vértices em cada face deve ser tal que quando usada a regra da mão de direita projeta um vetor apontando para fora do domínio.

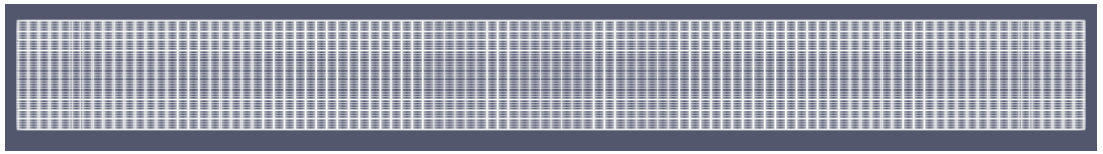
Exemplo de aplicação

Para fins de aplicação, vamos brincar com a alteração de uma malha de um dos tutoriais do OpenFoam. O tutorial em questão consiste no problema de Hartmann, no qual o escoamento de um fluido condutor no interior de um canal entre placas paralelas sob a ação de um campo magnético aplicado é simulado. Esse caso encontra-se na pasta tutorials/electromagnetics/mhdFoam/hartmann. A seguir são apresentados o domínio de cálculo, a malha e parte do arquivo blockMeshDict da pasta original do tutorial (sem edição).

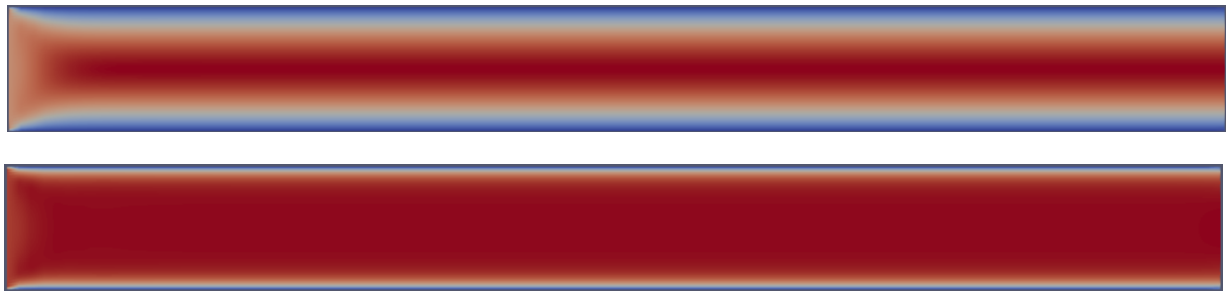
```
vertices
(
    0 (0 -1 0)
    1 (20 -1 0)
    2 (20 1 0)
    3 (0 1 0)
      (0 -1 0.1)
    4 (20 -1 0.1)
    5 (20 1 0.1)
    6 (0 1 0.1)
); 7
```




```
blocks
(
    hex (0 1 2 3 4 5 6 7) (100 40 1) simpleGrading (1 1 1)
);
```



A solução numérica pelo mhdFoam desse problema já foi comparada com a solução exata proposta por Shercliff (1953) e validada!



B = 0.1 T

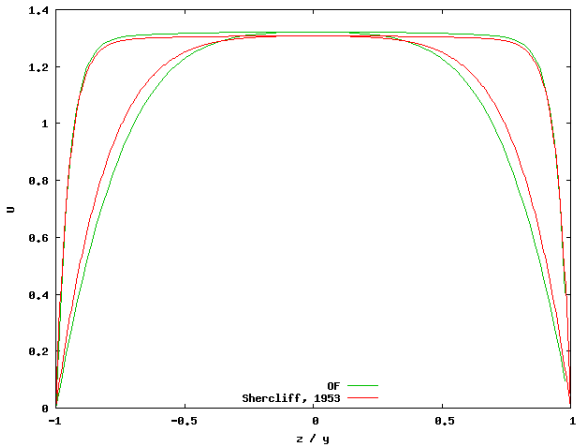
B = 10 T

STEADY MOTION OF CONDUCTING FLUIDS IN PIPES UNDER TRANSVERSE MAGNETIC FIELDS

By J. A. SHERCLIFF

Communicated by G. K. Batchelor

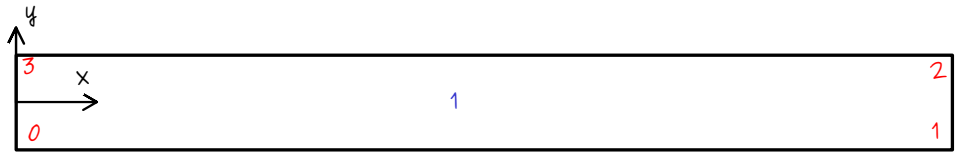
Received 4 June 1952



Percebemos pela soluções (numérica e exata) que a adição de um campo magnético no movimento do fluido condutor no canal leva a um achatamento do perfil de velocidade, acompanhado de uma diminuição na espessura da camada limite do problema.

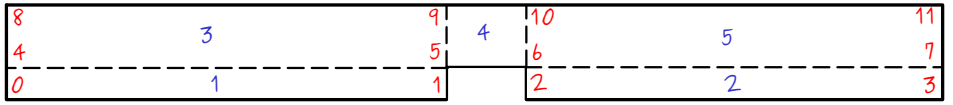
Gostaríamos agora de investigar como a presença de um pequeno obstáculo no meio do caminho altera essa dinâmica.

Caso original



0: (0 −1 0)
1: (20 −1 0)
2: (20 1 0)
3: (0 1 0)

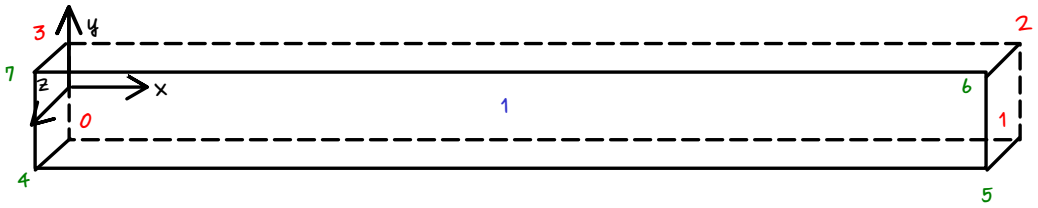
Caso modificado



0: (0 −1 0)
1: (9.5 −1 0)
2:(10.5 −1 0)
3:(20 −1 0)
4: (0 −0.5 0)
5: (9.5 −0.5 0)
6: (10.5 −0.5 0)
7: (20 −0.5 0)
8: (0 1 0)
9: (9.5 1 0)
10: (10.5 1 0)
11: (20 1 0)

Precisamos agora estender a geometria modificada para o caso tridimensional, dado que o OpenFOAM entende que todos os problemas (mesmo aqueles que serão tratados de maneira bidimensional) são tridimensionais.

Caso original




```
0: (0 −1 0)
1: (20 −1 0)
2: (20 1 0)
3: (0 1 0)
```

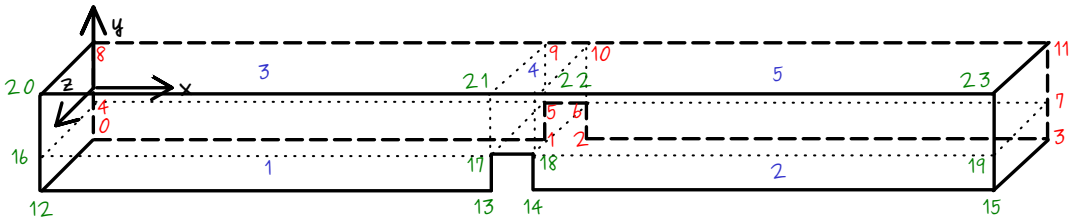
Nós do fundo (reais)

```
4: (0 −1 0.1)
5: (20 −1 0.1)
6: (20 1 0.1)
7: (0 1 0.1)
```

Nós da frente (projeções)

Bloco	Vértices
1	0, 1, 2, 3, 4, 5, 6, 7

Caso modificado




```
0: (0 −1 0)      4: (0 −0.5 0)      8: (0 1 0)
1: (9.5 −1 0)    5: (9.5 −0.5 0)    9: (9.5 1 0)
2:(10.5 −1 0)    6: (10.5 −0.5 0)   10: (10.5 1 0)
3:(20 −1 0)      7: (20 −0.5 0)     11: (20 1 0)
```

Nós do fundo

```
12: (0 −1 0.1)     16: (0 −0.5 0.1)     20: (0 1 0.1)
13: (9.5 −1 0.1)   17: (9.5 −0.5 0.1)   21: (9.5 1 0.1)
14:(10.5 −1 0.1)   18: (10.5 −0.5 0.1)  22: (10.5 1 0.1)
15:(20 −1 0.1)     19: (20 −0.5 0.1)    23: (20 1 0.1)
```

Nós da frente

Bloco	Vértices
1	0, 1, 5, 4, 12, 13, 17, 16
2	2, 3, 7, 6, 14, 15, 19, 18
3	4, 5, 9, 8, 16, 17, 21, 20
4	5, 6, 10, 9, 17, 18, 22, 21
5	6, 7, 11, 10, 18, 19, 23, 22

Como todas essas diferenças são então representadas no arquivo blockMeshDict?

Caso original

```
vertices
(
    (0 -1 0)
    (20 -1 0)
    (20 1 0)
    (0 1 0)
    (0 -1 0.1)
    (20 -1 0.1)
    (20 1 0.1)
    (0 1 0.1)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) (100 40 1) simpleGrading (1 1 1)
);
```

```
vertices
(
    (0 -1 0)
    (9.5 -1 0)
    (10.5 -1 0)
    (20 -1 0)
    (0 -0.5 0)
    (9.5 -0.5 0)
    (10.5 -0.5 0)
    (20 -0.5 0)
    (0 1 0)
    (9.5 1 0)
    (10.5 1 0)
    (20 1 0)
    (0 -1 0.1)
    (9.5 -1 0.1)
    (10.5 -1 0.1)
    (20 -1 0.1)
    (0 -0.5 0.1)
    (9.5 -0.5 0.1)
    (10.5 -0.5 0.1)
    (20 -0.5 0.1)
    (0 1 0.1)
    (9.5 1 0.1)
    (10.5 1 0.1)
    (20 1 0.1)
);
```

Caso modificado

```
blocks
(
    hex (0 1 5 4 12 13 17 16) (48 10 1) simpleGrading (1 1 1)
    hex (2 3 7 6 14 15 19 18) (48 10 1) simpleGrading (1 1 1)
    hex (4 5 9 8 16 17 21 20) (48 30 1) simpleGrading (1 1 1)
    hex (5 6 10 9 17 18 22 21) (4 20 1) simpleGrading (1 1 1)
    hex (6 7 11 10 18 19 23 22) (48 30 1) simpleGrading (1 1 1)
);
```

Note que precisamos alterar a quantidade de nós em cada block para mantermos o mesmo refinamento de malha do caso original. Se somarmos o número de nós em y por exemplo dos blocos 1 e 3 do caso modificado teremos: 40 nós. Note que especial atenção deve ser dada ao bloco 4, que por possuir uma dimensão menor em y deve ter um número menor de nós para mantermos a proporção.

Precisamos agora definir as fronteiras

Caso original

```
boundary
(
    inlet
    {
        type patch;
        faces
        (
            (0 4 7 3)
        );
    }
    outlet
    {
        type patch;
        faces
        (
            (2 6 5 1)
        );
    }
    lowerWall
    {
        type patch;
        faces
        (
            (1 5 4 0)
        );
    }
    upperWall
    {
        type patch;
        faces
        (
            (3 7 6 2)
        );
    }
    frontAndBack
    {
        type empty;
        faces
        (
            (0 3 2 1)
            (4 5 6 7)
        );
    }
);
```

Caso modificado

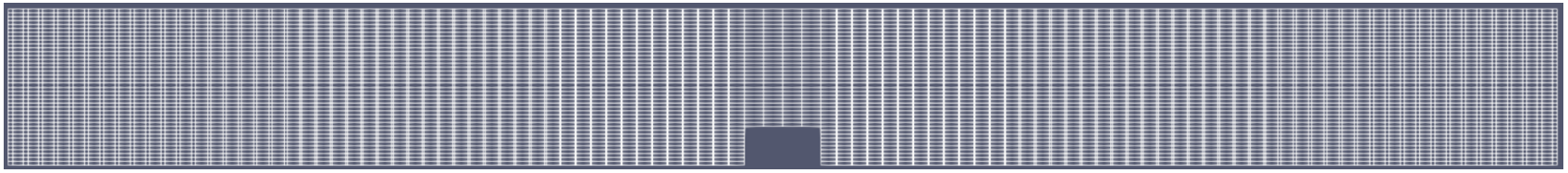
```
boundary
(
    inlet
    {
        type patch;
        faces
        (
            (0 12 16 4)
            (4 16 20 8)
        );
    }
    outlet
    {
        type patch;
        faces
        (
            (7 19 15 3)
            (11 23 19 7)
        );
    }
    lowerleftWall
    {
        type wall;
        faces
        (
            (0 1 13 12)
        );
    }
```

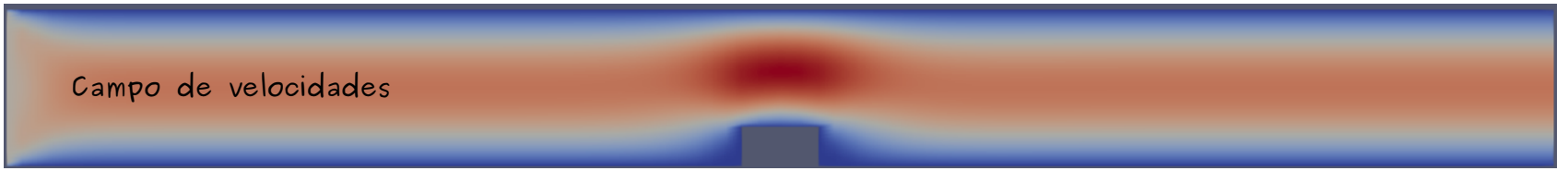
```
    obstacle
    {
        type wall;
        faces
        (
            (1 5 17 13)
            (5 6 18 17)
            (2 14 18 6)
        );
    }
    lowerrightWall
    {
        type wall;
        faces
        (
            (2 3 15 14)
        );
    }
    upperWall
    {
        type wall;
        faces
        (
            (8 20 21 9)
            (9 21 22 10)
            (10 22 23 11)
        );
    }
```

```
    frontAndBack
    {
        type empty;
        faces
        (
            (0 4 5 1)
            (4 8 9 5)
            (5 9 10 6)
            (6 10 11 7)
            (2 6 7 3)
            (12 13 17 16)
            (16 17 21 20)
            (17 18 22 21)
            (14 15 19 18)
            (18 19 23 22)
        );
    }
);
```

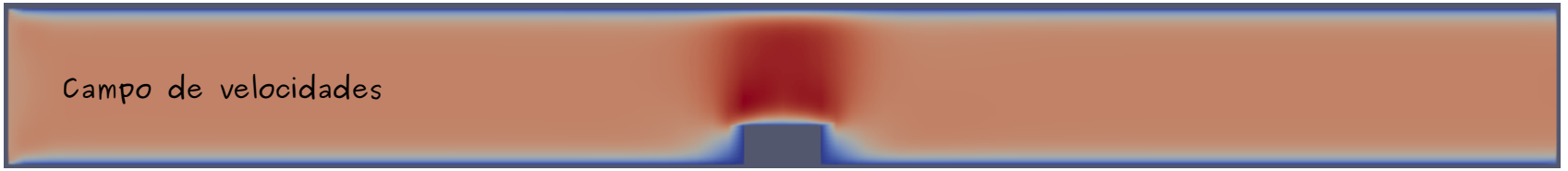
Note que quebramos a parede de baixo em 3 paredes separadas, uma do lado esquerdo, outra do lado direito e uma representando o obstáculo.

Alguns resultados típicos...

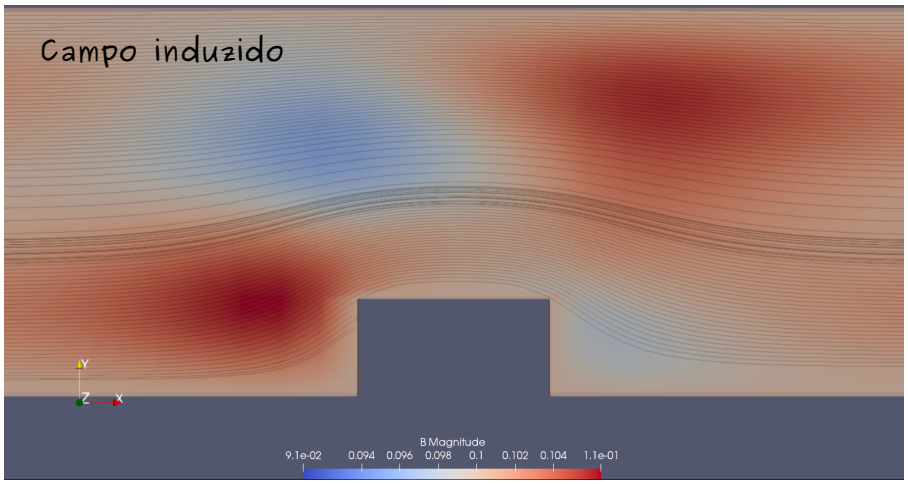




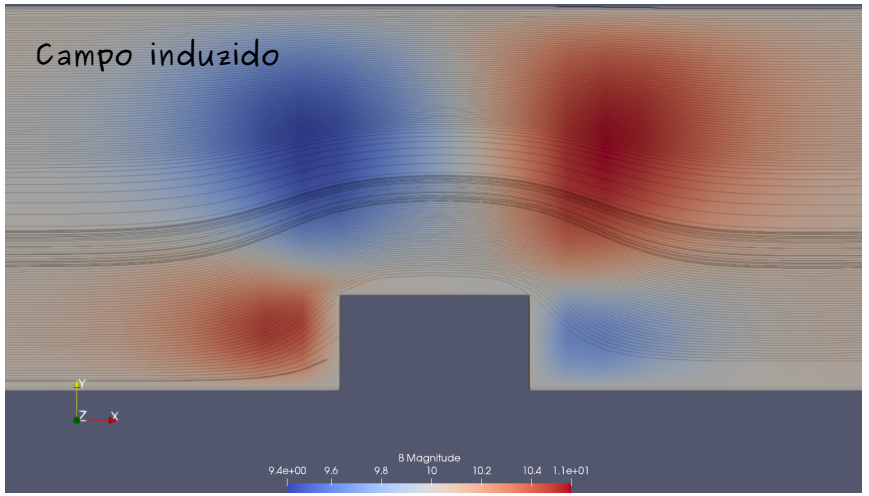
B = 0.1 T



B = 10 T



B = 0.1 T



B = 10 T