DINÂMICA DOS FLUIDOS COMPUTACIONAL

Guia prático usando o OpenFOAM®

Livia Flavia Carletti Jatobá

Universidade do Estado do Rio de Janeiro, Instituto Politécnico

CAPÍTULO 1

ESCOAMENTO LAMINAR EM UMA CAVIDADE

YURI BRANDÃO DOS SANTOS JOIA

Controle de versão: T01-2.0

Objetivo: estudar a convergência de malha na reprodução qualitativa das linhas de corrente de um escoamento laminar em uma cavidade quadrada.

Os arquivos deste tutorial estão disponíveis no repositório https://github.com/liviajatoba/cfd-openfoam.git, no diretório cavidade/icoFoam/Re-4030-CDS.

1.1 Definição do problema

O problema físico estudado neste tutorial consiste no escoamento de um fluido newtoniano, isotérmico, incompressível em uma geometria bidimensional de uma cavidade quadrada, onde a fronteira superior desloca-se com velocidade conhecida e as demais fronteiras são fixas. A Figura 1.1 mostra a geometria do problema[1]. Neste tutorial, vamos utilizar os utilitários `blockMesh`, `foamLog` e `postProcess`, para as etapas de pré e pós-processamento, e o *solver* `icoFoam`, para a solução do escoamento. O `gnuplot` e o `ParaView®` são as ferramentas utilizada para gerar as imagens.

A Equação da Continuidade e a Equação de Navier-Stokes para um escoamento incompressível de viscosidade constante, onde o termo devido a ação de força de campo gravitacional é desprezável, são as equações que governam este escoamento. A condição de contorno para a velocidade é de primeiro tipo, ou Dirichlet, onde a velocidade nas fronteiras fixas são nulas (condição de não deslizamento) e a velocidade na fronteira superior é conhecida ($U_x = 1m/s$). O *solver* `icoFoam` é o escolhido para a simulação do escoamento.

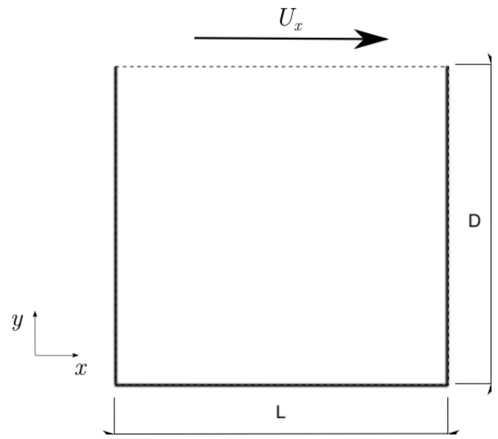


Figura 1.1 Geometria do problema do escoamento em uma cavidade.

A solução do escoamento em uma cavidade é um problema clássico adotado na validação e/ou verificação de métodos numéricos [2, 3, 4, 5]. A solução analítica deste tipo de escoamento pode ser determinada apenas para fluidos invíscidos [6]. Este tipo de escoamento gera linhas de corrente fechadas onde a natureza do vórtice depende da razão de aspecto (razão da altura pela largura) da cavidade e do número de Reynolds [3]. O número de Reynolds para este tipo de escoamento pode ser é definido por,

$$Re_D = \frac{U_x D}{\nu} \tag{1.1}$$

onde U_x é a velocidade da fronteira superior, D é a altura da cavidade e ν é a viscosidade cinemática [3]. Segundo Franco (2015) [7], o escoamento é laminar até $Re_D = 5.000$ e transicional até $Re_D = 10.000$. O estudo experimental deste tipo de escoamento é reportado para diferentes valores de Reynolds e razão de aspecto da cavidade [6, 8, 1, 9]. A Figura 1.2 mostra o resultado da visualização do escoamento para o caso $Re_D = 4030$, onde é possível observar um grande vórtice central e outro vórtice menor no canto inferior da cavidade [1]. O objetivo deste tutorial consiste em reproduzir, qualitativamente, o respectivo resultado.

1.2 Geometria e malha

A geometria e a malha deste tutorial serão construídas simultaneamente usando o utilitário `blockMesh`. O dicionário `blockMeshDict`, localizado no diretório `system` do caso, contém as instruções necessárias para criar a geometria e malha. A primeira instrução no `blockMeshDict` determina a unidade de comprimento da geometria em relação ao metro. Neste caso, a geometria está em metros, conforme mostra o Código 1.2:

Código 1.1 Definição do comprimeto da geometria no blockMeshDict

```
convertToMeters 1;
```

Em seguida, vamos definir os valores das variáveis da geometria, onde L é o comprimento na direção x, D é o comprimento na direção y e S é uma distância z arbitrária, uma vez que a solução será apenas para o plano x, y.

Código 1.2 Definição do comprimeto da geometria no blockMeshDict

```
L 0.1;   // comprimento em x
```

Figura 1.2 Resultado experimental da visualização do escoamento em uma cavidade com $Re_D = 4030$ [1].

```
D 0.1;   //comprimento em y
S 0.01;  //comprimento em z
```

A próxima etapa é definir as coordenadas dos vértices. A Figura 1.3 mostra a geometria da cavidade e seus respectivos vértices. O Código 1.3 é o trecho do `blockMeshDict` onde a declaração dos vértices é feita. Note que a coordenada de cada vértice retratado na Figura 1.3 é definida no trecho do dicionário `blockMeshDict` no Código 1.3.

Código 1.3 Definição dos vértices no blockMeshDict

```
vertices
(
    (0  0  0)           //vertice 0
    ($L 0  0)           //vertice 1
    ($L $D 0)           //vertice 2
    (0  $D 0)           //vertice 3
    (0  0  $S)          //vertice 4
    ($L 0  $S)          //vertice 5
    ($L $D $S)          //vertice 6
    (0  $D $S)          //vertice 7
);
```

Neste tutorial, o escoamento será resolvido para duas malhas, ou seja, será necessário a configuração de dois casos (ou simulações). Desta forma, serão necessários dois diretórios, chamados `m1` e `m2`, com os arquivos para as simulações, onde a diferença é dada pelo tamanho da malha construída. A construção da malha é dada a partir da definição de blocos dentro da geometria. A malha do primeiro caso, foi construída em um único bloco hexaédrico com 20 divisões nas direção x, 20 na direção y, 1 na direção z e sem refino de malha nas fronteiras. O Código 1.4 mostra o trecho do dicionário para criar um bloco:

Código 1.4 Definição de um bloco no blockMeshDict

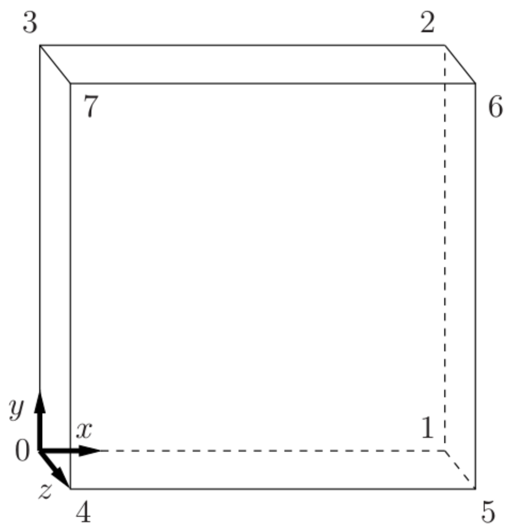


Figura 1.3 Esquema para construção da geometria usando o `blockMesh` [10].

```
 blocks
(
    hex (0 1 2 3 4 5 6 7) (20 20 1) simpleGrading (1 1 1)
);
```

onde primeiro é definido o tipo da malha (`hex` = hexaédrica), depois os vértices que constroem o bloco são declarados, seguidos do número de divisões em cada direção (x, y, z) e, por fim, é declarado uma opção de refino de malha em cada direção `simpleGrading)`. A Tabela 1.1 mostra o número de divisões para a construção da malha de cada caso deste tutorial.

Tabela 1.1 Número de divisões para as malhas dos casos.

caso	x, y, z
m1	$20, 20, 1$
m2	$40, 40, 1$

A última etapa de construção da malha consiste na declaração das fronteiras do domínio. O Código 1.5 é o trecho do `blockMeshDict` onde é feita a declaração das fronteiras. A declaração de uma fronteira começa pela escolha de um nome (qualquer) para identifica-la. Este nome será usado para definir as condições de contorno da velocidade e pressão nestas faces do domínio. Esta geometria é composta por três fronteiras: `fSuperior`, `paredes` e `zPlanos`. Em seguida, cada fronteira deverá ser identificada através de uma característica topológica, chamada `type`, e a lista de vértices para construir as faces que a compõem.

Código 1.5 Definição das fronteiras no blockMeshDict

```
boundary
(
    fSuperior
    {
        type wall;
```

```
        faces
        (
            (3 7 6 2)
        );
    }
    paredes
    {
        type wall;
        faces
        (
            (0 4 7 3)
            (2 6 5 1)
            (1 5 4 0)
        );
    }
    zPlanos
    {
        type empty;
        faces
        (
            (0 3 2 1)
            (4 5 6 7)
        );
    }
);
```

As faces que compõem as fronteiras são identificadas através dos vértices que a constrói. Assim, através da Figura 1.3, é possível identificar que a fronteira superior é formada pela face de vértices `(3 7 6 2)`, que recebeu o nome de `fSuperior`. A escolha do `type` da fronteira depende das características do problema. Neste caso, a fronteira superior, laterais e inferior são todas do tipo `wall`, ou seja, representam uma parede. A diferença entre a face superior e as demais será feita através da condição de contorno para a velocidade. Note que, apesar do problema ser bidimensional, todas as geometrias no OpenFOAM® são tridimensionais. Desta forma, para definir o problema com aproximação bidimensional é necessário adotar um característica topológica particular para as fronteiras que não serão discretizadas (eixo z com 1 divisão). Essas fronteiras foram identificas como `zPlanos` e são `type empty`.

Uma vez editado o arquivo `blockMeshDict`, a geometria e malha podem ser construídas através da execução do comando **`blockMesh`** no diretório do caso. Assim, em um terminal no diretório de cada caso, execute o comando.

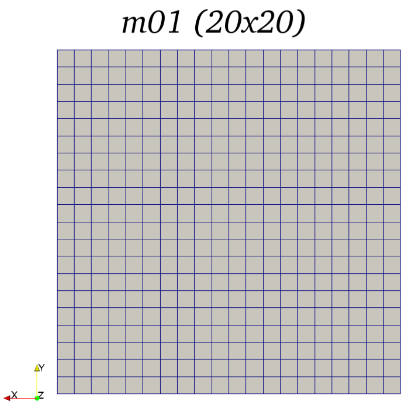
```
$ blockMesh
```

A visualização da geometria e malha são feitas usando o aplicativo ParaView®. Assim, em um terminal no diretório de cada caso, execute o comando `paraFoam` para visualizar a sua geometria e malha. O aplicativo ParaView® será carregado e, escolha a opção *Apply* e *Surface with Edges*, para visualizar a malha de cada caso, conforme a Figura 1.4.

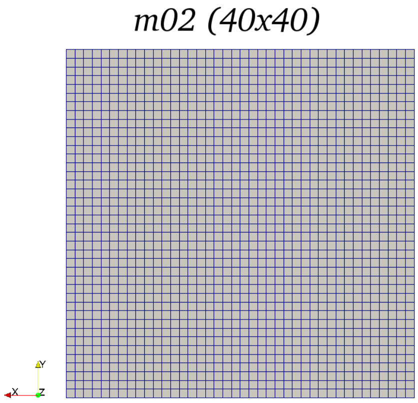
```
$ paraFoam
```

1.3 Propriedades, condições de contorno e inicial

A próxima etapa de preprocessamento consiste na definição das propriedades do fluido, do regime de escoamento (laminar ou turbulento) e das condições de contorno para velocidade e pressão. As propriedades do fluido são especificadas no dicionário `physicalProperties` no diretório `constant`. É necessário especificar o modelo do fluido (`Newtonian`) e a viscosidade cinemática (ν = `nu`), que foi calculada para o valor de $Re_D = 4030$.



(a) Malha 20x20



(b) Malha 40x40

Figura 1.4 Imagens das malhas geradas.

Código 1.6 Definição das propriedades do fluido no transportProperties

```
transportModel  Newtonian;
nu              nu [ 0 2 -1 0 0 0 0 ] 2.5e-05;
```

O regime do escoamento é definido no dicionário `turbulenceProperties` no diretório `constant`. O Código 1.7 mostra como é feita a escolha do regime de escoamento, através da definição do da variável `simulationType`, que foi configurada para uma escoamento laminar.

Código 1.7 Definição do regime do escoamento no turbulenceProperties

```
simulationType  laminar;
```

A condição de contorno e inicial dos campos de velocidade (`U`) e pressão (`p`) são especificadas em um arquivo para cada variável no diretório `0` do caso. O Código 1.8 apresentada a especificação da condição de contorno e inicial do campo de velocidade. A primeira definição do arquivo é a lista que representa o expoente de cada unidade primária, que no caso da velocidade é m/s. Em seguida, o valor da velocidade dentro do domínio é inicializado como uniforme e nulo. As condições de contorno são especificadas para cada fronteira a partir do nome escolhido na etapa de construção de malha. As três fronteiras criadas neste caso foram: `fSuperior`, `paredes` e `zPlanos`. Cada fronteira será especificadas através de um tipo (`type`) e um valor (`value`). O tipo `fixedValue` é uma condição de contorno de primeiro tipo onde o valor da variável é prescrita. O valor da velocidade na fronteira superior (`fSuperior`) é $U_x = 1\ m/s$ e nas laterais (`paredes`) é igual a zero. Por fim, o problema é bidimensional e portanto a fronteira `zPlanos` é do tipo `empty`.

Código 1.8 Arquivo de condição de contorno/inicial para velocidade (U) no diretório 0.

```
dimensions      [0 1 -1 0 0 0 0];
internalField   uniform (0 0 0);
boundaryField
{
    fSuperior
    {
        type            fixedValue;
        value           uniform (1 0 0);
    }
```

```
    paredes
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

    zPlanos
    {
        type            empty;
    }
}
```

O Código 1.9 apresenta a especificação da condição de contorno e inicial do campo de pressão. Note que o campo de pressão especificado na solução do escoamento incompressível pelo `icoFoam` é um campo de pressão relativa (manométrica) dividido pela massa específica do fluido, ou seja, a pressão é na verdade p/ρ. Por este motivo, a unidade do campo de pressão é m^2/s^2. A condição de contorno adotada para a pressão nas fronteiras `fSuperior` e `paredes` é do tipo `zeroGradient`. Este tipo de condição de contorno atribui o valor da variável do volume vizinho para a face da fronteira. Isto significa que que o gradiente da pressão normal à face da fronteira é nulo. A fronteira `zPlanos` é do tipo `empty` pois o problema é bidimensional.

Código 1.9 Arquivo de condição de contorno/inicial para a pressão relativa (p) no diretório 0.

```
dimensions      [0 2 -2 0 0 0 0];
internalField   uniform 0;
boundaryField
{
    fSuperior
    {
        type            zeroGradient;
    }

    paredes
    {
        type            zeroGradient;
    }

    zPlanos
    {
        type            empty;
    }
}
```

1.4 Configurações da solução numérica

As configurações de controle de passo de tempo, solução dos sistemas algébricos e esquemas de interpolação são realizadas através dos dicionários no diretório `system`.

O arquivo `fvSchemes` é aquele onde os esquemas de interpolação são selecionados. Neste tutorial, será utilizado o método de Euler implícito para o tempo e o método das diferenças centrais de segunda ordem para as discretizações das operações de divergente e laplaciano. Já as demais, foram realizadas usando a interpolação linear. O Código 1.10 mostra a respectiva configuração no arquivo `fvSchemes`.

Código 1.10 Arquivo de métodos de discretização no fvSchemes.

```
ddtSchemes
{
    default         Euler;
}
gradSchemes
{
    default         Gauss linear;
}
divSchemes
{
    default                         none;
    div(phi,U)                      Gauss linear;
    div((nuEff*dev2(T(grad(U)))))   Gauss linear;
}
laplacianSchemes
{
    default         Gauss linear corrected;
}
```

O arquivo `fvSolutions` é o dicionário onde são feitas as configurações da solução do sistema algébrico. O método de Gauss-Seidel foi o selecionado para o sistema algébrico da velocidade, com uma tolerância de 10^{-5}. Já o sistema algébrico para a pressão é resolvido usando um método Multi-grid com pré-condicionador Gauss-Seidel. Além disso, para a pressão foi estabelecido o número de 2 corretores para o acoplamento pressão-velocidade.

Código 1.11 Arquivo de métodos de solução dos sistemas algébricos no fvSolutions.

```
solvers
{
    U
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-05;
        relTol          0;
    }
    p
    {
        solver          GAMG;
        tolerance       1e-06;
        relTol          0.1;
        smoother        GaussSeidel;
    }
    pFinal
    {
        $p;
        tolerance       1e-06;
        relTol          0;
    }
}
PISO
{
    nCorrectors     2;
    nNonOrthogonalCorrectors 0;
    pRefCell        0;
    pRefValue       0;
}
```

Por fim, o arquivo `controlDict` é o dicionário onde são feitas as configurações do passo de tempo, tempo inicial, final e definição de quais os instantes de tempo serão salvos. No *solver* icoFoam, o passo de tempo, representado por deltaT no controlDict, deve ser configurado de acordo com o critério de Courant unitário. O número de Courant é definido como,

$$Co = \frac{\delta t|\mathbf{U}|}{\delta x} \tag{1.2}$$

onde $|\mathbf{U}|$ é a magnitude da velocidade, δt é o passo de tempo e δx é o o tamanho da malha na direção x. O valor do δt deve ser calculado na condição mais crítica do escoamento, ou seja, velocidade máxima e menor elemento de malha. Neste tutorial, a malha é uniforme e a Tabela 1.2 apresenta os valores de δt para cada caso.

Tabela 1.2 Passo de tempo para cada caso.

caso	δt
m01	0.005
m02	0.0025

Note, o objetivo deste tutorial consiste em comparar a solução numérica do escoamento com o resultado experimental da visualização do escoamento, ou seja, a trajetória. Assim, precisamos definir quais as variáveis de interesse devem ser calculadas ao longo da simulação para alcançar este objetivo. Neste caso, o cálculo das linhas de corrente do escoamento permite a visualização do comportamento do campo de velocidade semelhante ao resultado do experimental. Note ainda, que no estacionário, as linhas de corrente são idênticas às trajetórias.

Assim, vamos utilizar o sub-dicionário, functions no controlDict para configurar o calculo de linhas de corrente ao longo da simulação. O Código 1.12 mostra como é feita a configuração desta ferramenta, que começa com escolha de nome para a função, chamada de linhaDeCorrente. O OpenFOAM® possui diversos tipos de funções, ou cálculos, de pós-processamento já programadas, como o caso do cálculo da linha de corrente. Assim, o type da função é streamLine. Cada tipo de função tem um conjunto de entrada de dados, que são as demais declarações no Código 1.12. Neste caso, vamos salvar o calculo da linha de corrente nos mesmos intervalos de tempo que foi configurado a gravação da solução transiente e, o tipo de arquivo será o vtk. A linha de corrente será extrapolada em ambas as direções da velocidade e a linha que passa pelos pontos start e end é a região de amostragem.

Código 1.12 Trecho do arquivo controlDict com a declaração da função para o cálculo das linhas de corrente.

```
functions
{
 linhaDeCorrente
 {
    type            streamLine;
    libs            ("libfieldFunctionObjects.so");

    executeControl  writeTime;
      writeControl    writeTime;
    setFormat       vtk;

    lifeTime        1000;
      nSubCycle       5;
    U               U;
      direction       both;
    fields
      ( U );
    seedSampleSet
    {
        type        lineUniform;
```

```
        axis        x;
        start       (0.1 0   0.005);
        end         (0   0.1 0.005);
        nPoints     500;
    }
 }
}
```

1.5 Solução do escoamento

A solução do escoamento em uma cavidade é dada pela execução do *solver* escolhido, que no caso foi o `icoFoam`. A execução do *solver* irá montar o sistema algébrico da solução numérica das equações que governam o escoamento, considerando os esquemas de interpolação escolhidos no `fvSchemes`, a malha do caso e os métodos de solução escolhidos no `fvSolutions`. Neste tutorial, temos dois casos e, portanto, em um terminal no diretório de cada caso, execute o comando:

```
$ icoFoam > log
```

Este comando irá executar o *solver* do `icoFoam` e armazenar os dados de saída no terminal no arquivo `log`. Os diretórios para os passos de tempo serão criados e os respectivos campos de velocidade e pressão salvos, até a simulação atingir o seu tempo final configurado no dicionário `controlDict` no diretório `system`.

1.6 Convergência Numérica

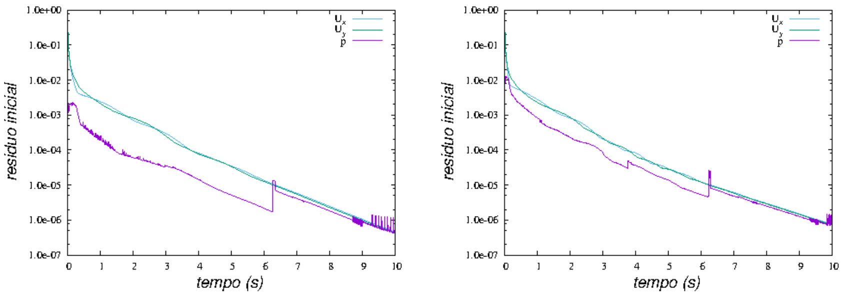
A primeira etapa de análise dos resultados consiste na verificação da convergência numérica. A convergência numérica de uma simulação é realizada através da análise dos resultados do resíduo inicial dos sistemas algébricos por passo de tempo. O OpenFOAM® tem um utilitário, o **`foamLog`** que extrai os valores dos resíduos iniciais e finais do arquivo `log`. A chamada deste comando deve ser feita para cada caso, em um terminal no diretório do caso, da seguinte forma:

```
$ foamLog log
```

Um novo diretório `logs` será criado, contendo diversos arquivos. A análise da convergência numérica é feita através da construção de um gráfico de resíduos inicias por tempo. Neste tutorial, vamos utilizar o **`gnuplot`** como ferramenta para construção dos gráficos. Um arquivo previamente configurado, chamado `plot-residuo.plt` contém as instruções para criar o gráfico de resíduo inicial para as componentes da velocidade e pressão. A construção do gráfico é feita através do comando no terminal:

```
$ gnuplot plot-residuo.plt
```

A Figura 1.5 mostra os gráficos, de resíduo inicial em função do tempo para pressão e componentes da velocidade, gerados para os casos do tutorial. Observe que, ao longo da simulação, os valores do resíduos iniciais, para cada passo de tempo e de todas as variáveis da simulação, apresentam um padrão de decaimento. Ao final da simulação, e possível observar que os resíduos iniciais de todas as variáveis ficaram abaixo de 10^{-5}. Este tipo de comportamento dos resíduos inciais ao longo da simulação demonstra que, ao final da simulação, a diferença entre o campo das variáveis no início do passo de tempo e no final do passo de tempo é da ordem de 10^{-5}. Como a ordem de grandeza das variáveis é de 10^{0}, pode-se concluir que as variáveis não mudam mais com o tempo e, portanto, a simulação alcançou o regime estacionário.



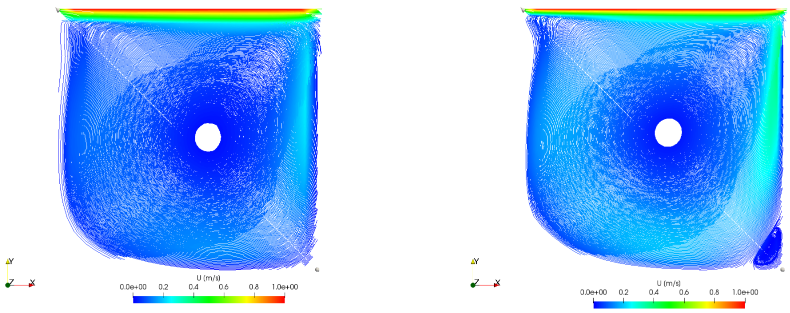
(a) Malha 20x20 (b) Malha 40x40

Figura 1.5 Gráficos dos resultados de resíduo inicial ao longo da simulação para análise da convergência numérica.

1.7 Análise dos resultados

Uma vez confirmada que a solução numérica convergiu para o estado estacionário em todos as malhas simuladas, a próxima etapa do tutorial consiste em analisar as variáveis no campo. O objetivo do tutorial consiste em comparar as linhas de corrente da solução numérica com o resultado experimental da visualização do escoamento. Configuramos o calculo da linha de corrente no sub-dicionario `functions` do `controlDict`, conforme o trecho no Código 1.12. Os resultados, para cada instante de tempo, foram armazenados no diretório `posProcessing/sets/linhaDeCorrente`. A visualização dos resultados pode ser feita abrindo o arquivo `vtk` do último instante de tempo usando o ParaView® e, em seguida, selecionado o campo de velocidade para o gradiente de cores.

A Figura 1.6 mostra os resultados para as linhas de corrente dos casos simulados. A comparação das Figuras 1.2 e 1.6 mostra que a malha 20x20 não capturou o vórtice menor no canto inferior da cavidade, enquanto que o presença deste vórtice é observada na solução do escoamento para malha 40x40. Assim o número de volumes de controle da malha é um importante fator a ser considerado na análise de solução numéricas do escoamento de fluidos.



(a) Malha 20x20 (b) Malha 40x40

Figura 1.6 Visualização das linhas de corrente para as malhas.

EXERCÍCIOS

1.1 Execute um novo caso com uma geometria $L/D = 0.5$ onde $D = 0.1\ m$ e $Re_D = 1.000$. Avalie a convergência numérica e os resultados de linha de corrente para diferentes malhas.

Referências Bibliográficas

[1] Faure, T., Adrianos, P., Lusseyran, F., Pastur, L.; Visualizations of the flow inside an open cavity at medium range Reynolds numbers, Exp Fluids, 42:169–184, 2007.

[2] Burggraf, O.; Analytical and numerical studies of the structure of steady separeted flows, Journal of Fluid Mechanics, Volume 24, Issue 01, pp 113 151, 1966.

[3] Bozeman, J. and Dalton, C.; Numerical Study of Viscous Flow in a Cavity, Journal of Computational Physics, V. 12, p. 348-363, 1973.

[4] Ghia, U.; Ghia, K.; Shin, C., High-Re Solutions For Incompressible-Flow Using The Navier Stokes Equations And A Multigrid Method, Journal Of Computational Physics Volume: 48 Issue: 3, p. 387-411, 1982.

[5] O'Brien, V.; Closed Streamlines Associated with Channel Flow over a Cavity, Physics of Fluids 15, p.2089, 1972.

[6] Frank Pan and Andreas Acrivos, Steady flows in rectangular cavities, Journal of Fluid Mechanics, Volume 28, Issue 04, pp 643 655, 1967.

[7] Franco, E. and Barrera, M. and Laín, S.; 2D lid-driven cavity flow simulation using GPU-CUDA with a high-order finite difference scheme, Journal of the Brazilian Society of Mechanical Sciences and Engineerings, Volume 37, 2015.

[8] Leo F. Donovan, A, A Numerical Solution of Unsteady Flow in a Two-Dimensional Square Cavity, AIAA JOURNAL, 1969.

[9] Tanja Siegmann-Hegerfeld and Stefan Albensoeder and Hendrik C. Kuhlmann, Three-dimensional flow in a lid-driven cavity with width-to-height ratio of 1.6, Exp Fluids, 54:1526, 2013.

[10] Christopher J. Greenshields, OpenFOAM, The Open Source CFD Toolbox, User Guide, 2015.