
DINÂMICA DOS FLUIDOS COMPUTACIONAL

Guia prático usando o OpenFOAM®

Livia Flavia Carletti Jatobá



Universidade do Estado do Rio de Janeiro, Instituto Politécnico

Licença:

Este trabalho está licenciado sob a licença Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International Public License (CC BY-NC-SA 4.0 - Atribuição-NãoComercial-CompartilhaIgual)

Aviso Legal:

OpenFOAM[®] and OpenCFD[®] são marcas registradas por OpenCFD Limited, que produz o software OpenFOAM[®]. Todas as marcas registradas são de seus proprietários. Este documento não foi aprovado ou endossado por OpenCFD Limited, o produtor do software OpenFOAM[®] e detentor das marcas registradas OPENFOAM[®] and OpenCFD[®].

Agradecimentos:

As autoras agradecem à Franciane F. Rocha e Leib Neubarth pelas contribuições na revisão deste documento.

CAPÍTULO 1

ESCOAMENTO COUETTE ENTRE PLACAS PLANAS

STHEFANY MACHADO SARDINHA

Controle de versão: T01-2.2

Objetivo: realizar verificação de solução numérica com solução analítica.

Os arquivos deste tutorial estão disponíveis no repositório <https://github.com/liviajatoba/cfd-openfoam.git>, no diretório `couettePlacasPlanas/Re-10`.

1.1 Definição do problema

O problema físico estudado neste tutorial consiste no escoamento de um fluido newtoniano, laminar, incompressível e plenamente desenvolvido entre duas placas planas paralelas, onde a placa superior desloca-se com velocidade conhecida e a inferior é fixa [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` é a ferramenta utilizada para gerar gráficos.

A Figura 1.1 mostra um esquema da geometria do problema. A força motriz deste tipo de escoamento é o gradiente de velocidade entre a placa superior e inferior e, portanto, o transporte difusivo de quantidade de movimento linear. Devido a condição de não deslizamento, o fluido em contato com as placas tem a mesma velocidade das placas. Assim, o fluido em contato com a placa superior transfere quantidade de movimento linear para as camadas de fluido adjacentes até a camada de fluido em contato com a placa inferior, que permanece parada. Neste tipo de problema o gradiente de pressão é nulo, ou seja, a pressão é uniforme. Este tipo de escoamento é conhecido na literatura como escoamento Couette [1].

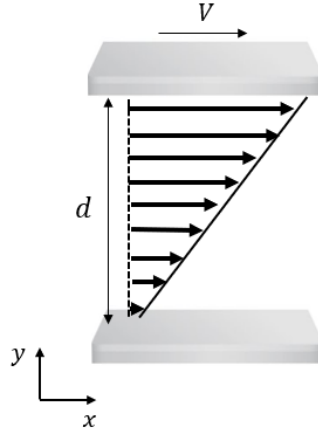


Figura 1.1 Representação simplificada do escoamento entre placas planas.

A Equação da Continuidade e a Equação de Navier-Stokes para um escoamento incompressível e viscosidade constante, onde o termo devido a ação de força de campo gravitacional é desprezível, são as equações que governam este escoamento. Aplicada ao sistema de coordenadas cartesiana, a modelagem pode ser simplificada pelas equações abaixo.

$$\frac{\partial U_x}{\partial x} = 0 \quad (1.1)$$

$$0 = \frac{\mu}{\rho} \left[\frac{\partial^2 U_x}{\partial y^2} \right] \quad (1.2)$$

A velocidade neste escoamento tem componente diferente de zero apenas na direção x , que é função apenas da coordenada y , $U_x(y)$. A condição de contorno para a velocidade é de primeiro tipo, onde a velocidade são conhecidas nas fronteiras superior e inferior e d é a distância entre as placas:

$$U_x(y = 0) = 0 ; U_x(y = d) = V. \quad (1.3)$$

A solução analítica para o perfil de velocidade é

$$U_x(y) = \frac{V}{d} y \quad (1.4)$$

onde V e d são constantes [1].

1.2 Geometria e malha:

A geometria e a malha deste tutorial são 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, conforme mostra trecho do arquivo no Dicionário 1.1. Um caso com dimensões em centímetros ou milímetros, teria o valor unitário substituído por 0.01 ou 0.001, respectivamente.

Dicionário 1.1 Definição do comprimento da geometria no blockMeshDict.

```
convertToMeters 1;
```

Em seguida, são declaradas os valores das variáveis adotadas neste tutorial, conforme o trecho do arquivo no Dicionário 1.2. A distância $d = 0.1 \text{ m}$ é definida através da variável dCanal. As demais distâncias são os comprimentos nas direções x e z, respectivamente, definidas como xCanal e zCanal. A solução numérica para o escoamento entre placas planas é realizada para uma discretização com nY divisões em y. Assim, a solução é para em uma seção x e z qualquer da placa plana, ou seja, esses comprimentos foram arbitrariamente definidos como 0.4 m e 0.1 m , respectivamente, e possuem apenas uma única divisão.

Dicionário 1.2 Definição das variáveis no blockMeshDict.

```
xCanal 0.4;      //comprimento do canal na direcao x
dCanal 0.1;      //distancia entre as placas na direcao y
zCanal 0.1;      //distancia na direcao z
nY 5;           //numero de divisoes em y
```

A próxima etapa consiste em listar as coordenadas de todos os vértices. A Figura 1.2 mostra a geometria da placa plana e seus respectivos vértices. O Dicionário 1.3 é o trecho do blockMeshDict onde a declaração dos vértices é feita, onde as variáveis xCanal, dCanal e zCanal definidas anteriormente usando a notação \$.

Dicionário 1.3 Definição dos vértices no blockMeshDict

```
vertices
(
    (0          0          0)          //vertice 0
    ($xCanal    0          0)          //vertice 1
    ($xCanal    $hCanal    0)          //vertice 2
    (0          $hCanal    0)          //vertice 3
    (0          0          $zCanal)     //vertice 4
    ($xCanal    0          $zCanal)     //vertice 5
    ($xCanal    $hCanal    $zCanal)     //vertice 6
    (0          $hCanal    $zCanal)     //vertice 7
);
```

A construção da malha é dada a partir da definição de blocos dentro da geometria. Neste caso, foi construído um único bloco hexaédrico com uma única divisão nas direções x e z, e nY na direção y. O Dicionário 1.4 mostra o trecho do blockMeshDict para criar um bloco, 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). Por fim, é declarado uma opção de refino de malha em cada direção. Não foi adotado refino de malha nas fronteiras e, por esse motivo, os parâmetros do simpleGrading são unitários.

Dicionário 1.4 Definição de um bloco no blockMeshDict.

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

A última etapa de construção da malha consiste na declaração das fronteiras do domínio. O Dicionário 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 identificá-la. Este nome será utilizado para definir as condições de contorno para velocidade e pressão. Em seguida utilizaremos a regra da mão direita para determinar os vértices que serão utilizados para construir uma fronteira.

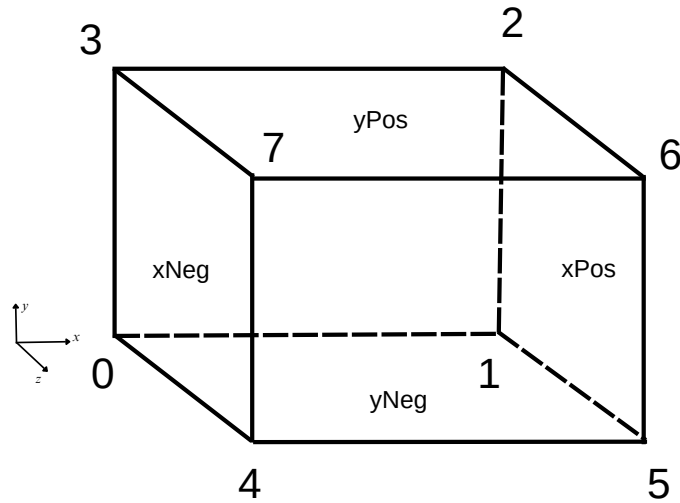


Figura 1.2 Construção da geometria usando o `blockMesh`.

A declaração é feita no sub-dicionário `boundary` onde a ordem de cada vértice é tal qual a regra da mão direita, com a normal sempre apontando para fora do volume. Nesta malha, cinco fronteiras são criadas: `yPos`, `yNeg`, `xPos`, `xNeg` e `zPlan`. Em seguida, cada fronteira é identificada através de uma característica topológica `type` e os vértices das faces que a compõem, na ordem consistente com a normal apontando para fora da geometria.

Dicionário 1.5 Definição das fronteiras no `blockMeshDict`.

```
boundary
(
    yPos
    {
        type patch;
        faces
        (
            (3 7 6 2)
        );
    }
    yNeg
    {
        type patch;
        faces
        (
            (0 1 5 4)
        );
    }
    xNeg
    {
        type cyclic;
        neighbourPatch xPos;
        faces
        (
            (0 4 7 3)
        );
    }
    xPos
    {

```



```

        type cyclic;
        neighbourPatch xNeg;
        faces
        (
            (1 2 6 5)
        );
    }
    zPlan
    {
        type empty;
        faces
        (
            (3 2 1 0)
            (4 5 6 7)
        );
    }
};

```

As faces que compõem as fronteiras são identificadas através dos vértices que a constroem. Assim, através da Figura 1.2, é possível identificar que a fronteira superior é formada pela face de vértices (3 7 6 2), que recebeu o nome de `yPos`. A escolha da característica topológica da fronteira, ou `type`, depende do tipo de condição de contorno das variáveis. Neste caso, a fronteira superior e inferior (`yPos` e `yNeg`) são todas do tipo `patch`, que é uma característica topológica genérica para cálculos diversos. As fronteiras laterais (`xPos` e `xNeg`) são do tipo `cyclic`, ou seja, os valores das variáveis na direção `x` são uniformes. Esta característica topológica é utilizada para reproduzir matematicamente o escoamento desenvolvido em `x` e a sua configuração é feita em pares, através da declaração do `neighbourPatch`, conforme mostra o Dicionário 1.5. Por fim, os planos no eixo `z`, são do tipo `empty`. Essa característica topológica existe pois, no OpenFOAM®, todas as geometrias são tridimensionais. Assim, quando um problema precisa ser simplificado para bidimensional, os planos normais ao plano cuja solução é desejada recebe a característica topológica `empty`. Neste caso, busca-se uma solução no plano `xy` e, por esse motivo, as fronteiras `zPlan` recebem o `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 do caso, execute o comando:

```
$ blockMesh
```

A visualização da geometria e malha são feitas usando o aplicativo ParaView. Assim, em um terminal no diretório do 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, conforme a Figura 1.3.

```
$ paraFoam
```

1.3 Propriedades, condições de contorno e inicial

A próxima etapa de pré-processamento consiste na definição das propriedades do fluido, do regime de escoamento (laminar ou turbulento) e das condições de contorno para velocidade e pressão. Este tutorial tem escoamento laminar com número de Reynolds igual a 10.

As propriedades do fluido são especificadas no dicionário `physicalProperties` no diretório `constant`. É necessário especificar a viscosidade cinemática ($\nu = \text{nu}$), conforme mostra o trecho do arquivo no Dicionário 1.6. Observe que a propriedade viscosidade cinemática é declarada através

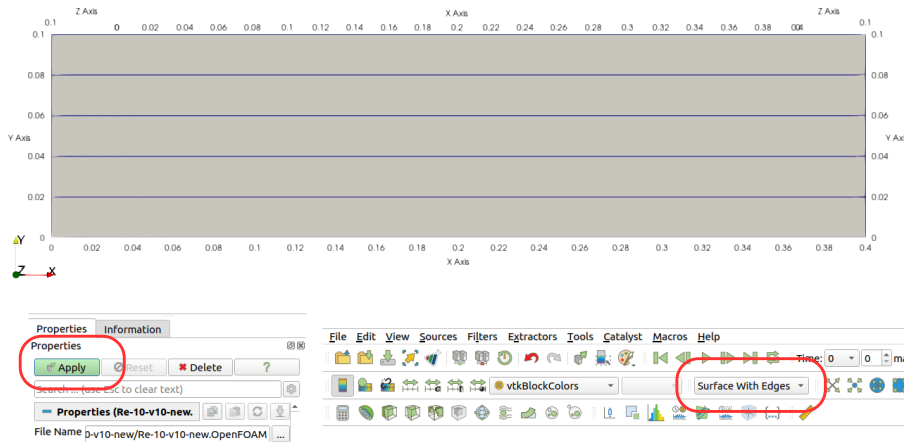


Figura 1.3 Visualização da malha usando o ParaView.

de uma lista de expoentes de unidades (`dimensionSet`), $[0 \ 2 \ -1 \ 0 \ 0 \ 0 \ 0]$, e um escalar 0.01. Valor 0.01 é aquele calculado para Reynolds 10 ($Re = dV/\nu$), onde a velocidade é $V = 1\text{m/s}$ e a distância entre as placas é 0.1m. A lista de expoentes são para as unidades primárias, na seguinte ordem: massa, comprimento, tempo, temperatura, quantidade de matéria, corrente e intensidade luminosa. Assim, a viscosidade cinemática tem unidade em m^2/s .

Dicionário 1.6 Definição das propriedades do fluido no `physicalProperties`.

```
nu          [0 2 -1 0 0 0 0] 0.01;
```

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 Dicionário 1.7 apresenta a especificação da condição de contorno e inicial do campo de velocidade. A primeira definição do arquivo é a lista `dimensionSet` que representa a unidade da grandeza (m/s), em seguida o valor da velocidade no campo interno, ou seja, nos centros dos volumes de controle do domínio computacional (`internalField`). Neste caso, o campo interno 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. Recorde que as fronteiras criadas neste caso foram: `yPos`, `yNeg`, `xPos`, `xNeg` e `zPlan` (Dicionário 1.5).

A condição de contorno para velocidade na fronteira `yPos` e `yNeg` é do tipo valor fixo, declarada como `fixedValue`. A fronteira `yPos` é aquela que tem velocidade na direção x igual a 1m/s. Já na fronteira `yNeg` a velocidade é igual a zero. Os planos `xPos` e `xNeg` terão os valores uniformes em x , portanto, a condição de contorno é do tipo `cyclic`. Por fim, a fronteira `zPlan` é do tipo `empty` para que a solução do escoamento seja apenas no plano xy .

Dicionário 1.7 Arquivo de condição de contorno/inicial para velocidade no diretório 0.

```
dimensions  [0 1 -1 0 0 0 0];
```

```
internalField  uniform (0 0 0);
```

```
boundaryField
{
    yPos
    {
```

```

        type          fixedValue;
        value          uniform (1 0 0);
    }
    yNeg
    {
        type          fixedValue;
        value          uniform (0 0 0);
    }
    xNeg
    {
        type          cyclic;
    }
    xPos
    {
        type          cyclic;
    }
    zPlan
    {
        type          empty;
    }
}

```

O Dicionário 1.8 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 no OpenFOAM[®] 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 yPos, yNeg é do tipo zeroGradient, onde o valor da fronteira é igual ao valor do volume vizinho. A condição de contorno dos planos xPos, xNeg são do tipo cyclic, que repetem os valores na direção x para reproduzir o escoamento desenvolvido. Por fim, a fronteira zPlan é do tipo empty para que a solução do escoamento seja apenas no plano xy.

Dicionário 1.8 Arquivo de condição de contorno/inicial para pressão relativa no diretório 0

```

dimensions          [0 2 -2 0 0 0 0];

internalField        uniform 0;

boundaryField
{
    yPos
    {
        type          zeroGradient;
    }
    yNeg
    {
        type          zeroGradient;
    }
    xNeg
    {
        type          cyclic;
    }
    xPos
    {
        type          cyclic;
    }
    zPlan
    {
        type          empty;
    }
}

```

}

1.4 Configurações da solução numérica

As configurações da solução numérica são as escolhas dos esquemas de interpolação, métodos de solução dos sistemas algébricos, tolerâncias, número de iterações no acoplamento pressão-velocidade, tempo inicial, final e passo de tempo da simulação. Estas definições são feitas nos dicionários no diretório `system` do caso.

Os esquemas de interpolação utilizados neste tutorial foram: método implícito de Euler na integração temporal e linear nas discretizações dos gradientes, divergentes e laplacianos. As demais operações foram discretizadas usando uma interpolação linear. O Dicionário 1.9 mostra as declarações no dicionário `fvSchemes`.

Dicionário 1.9 Definição dos esquemas de interpolação no `fvSchemes` no diretório `system`

```
ddtSchemes
{
    default          Euler;
}

divSchemes
{
    default          Gauss linear;
}

gradSchemes
{
    default          Gauss linear;
}

laplacianSchemes
{
    default          Gauss linear corrected;
}
```

Os métodos para a solução dos sistemas algébricos foram: Gauss-Seidel e Gradiente Conjugado para a velocidade e pressão, respectivamente. Um pré-condicionador diagonal foi configurado na solução da pressão. As tolerâncias de 10^{-6} e 10^{-7} foram configuradas para a velocidade e pressão, respectivamente. O Dicionário 1.10 mostra o dicionário `fvSolutions` com as configurações citadas.

Dicionário 1.10 Definição dos parâmetros solução no `fvSolution` no diretório `system`

```
solvers
{
    U
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance        1e-06;
        relTol          0;
    }
    pFinal
    {
        solver          PCG;
        preconditioner  DIC;
    }
}
```

```

        tolerance      1e-07;
        relTol          0;
    }
}
PISO
{
    pRefCell            0;
    pRefValue           0;
}

```

Por fim, os parâmetros de controle da simulação, como tempo inicial, final, passo de tempo e instantes de tempo salvos, são configurados no dicionário `controlDict`, conforme trecho do arquivo no Dicionário 1.11. O passo de tempo (`deltaT`), configurado para o caso foi calculado considerando o critério de Courant unitário para o caso.

Dicionário 1.11 Definição dos parâmetros solução no `controlDict` no diretório `system`

```

application      icoFoam;
startFrom        startTime;
startTime        0;
stopAt           endTime;
endTime          1;
deltaT           0.02;
writeControl     adjustableRunTime;
writeInterval    0.1;
purgeWrite       0;
writeFormat      ascii;
writePrecision   6;
writeCompression off;
timeFormat       general;
timePrecision    6;

```

1.5 Solução do escoamento

A solução do escoamento 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` e malha do caso. Assim, em um terminal no diretório do caso execute o comando:

```
$ icoFoam > log
```

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

1.6 Convergência Numérica

Uma vez que a execução do comando no terminal está concluída, a próxima etapa consiste em avaliar a convergência da solução numérica. A análise da convergência numérica consiste em acompanhar os resíduos das soluções dos sistemas algébricos e uma variável de interesse. A característica da convergência numérica depende das características do escoamento e, neste caso, o escoamento atinge o regime estacionário. Neste tutorial, apenas o resíduo para U_x é relevante, uma vez que a pressão é uniforme e as demais componentes da velocidade são nulas.

Os valores dos resíduos iniciais e finais para solução de cada sistema algébrico é resultado da execução do comando `icoFoam`, que foi armazenado no arquivo `log`. O Dicionário 1.12 abaixo mostra a informação no início da simulação, primeiro passo de tempo e, no instante de tempo final. A linha que mostra o resultado da solução do sistema algébrico para U_x foi colocada em negrito para destaque. Note que, no primeiro instante de tempo, o resíduo inicial (`Initial residual`) para U_x tem valor unitário, o resíduo final (`Final residual`) tem ordem de grandeza semelhante a tolerância para U especificada no `fvSolution` e 6 iterações foram necessárias para esta convergência. Já no último instante de tempo, o resíduo inicial é da ordem de grandeza de 10^{-5} . Assim, é possível observar uma convergência numérica da ordem de 10^{-5} no último instante de tempo.

Dicionário 1.12 Resultado da solução do escoamento usando o `icoFoam`.

```
...

Starting time loop
Time = 0.02s

Courant Number mean: 0 max: 0
smoothSolver: Solving for Ux, Initial residual = 1, Final residual = 2.83
               e-07, No Iterations 6
...

Time = 1s

Courant Number mean: 0.024996 max: 0.0449981
smoothSolver: Solving for Ux, Initial residual = 1.74885e-05, Final
               residual = 1.32559e-07, No Iterations 2
...

End
```

Apesar do valor do resíduo inicial no último instante de tempo já ser um parâmetro válido para análise da convergência numérica e regime estacionário, um gráfico do resíduo inicial ao longo do tempo é um importante resultado para análise da convergência numérica. O OpenFOAM® possui um utilitário que pode ser utilizado para capturar as informações de resíduos do arquivo `log`, o `foamLog`. Assim, em um terminal, execute o comando a seguir.

```
$ foamLog log
```

O utilitário irá gerar um novo diretório `logs`, contendo diversos arquivos com dados da solução numérica. Temos interesse em construir um gráfico do resíduo inicial de U_x em relação ao tempo. Neste tutorial, vamos utilizar o aplicativo `gnuplot` para construir o gráfico. O Dicionário 1.13 mostra o arquivo utilizado para armazenar os comandos para construção do gráfico usando o `gnuplot`. Note que o comando do `gnuplot` acessa o arquivo U_x dentro do diretório `logs` e traça o gráfico usando a primeira (\$1) e segunda (\$2) colunas do arquivo como eixos x e y , respectivamente.

Dicionário 1.13 Arquivo com comandos do `gnuplot` para construção do gráfico de convergência numérica.

```
reset

set term png
set style func linespoints

set autoscale
set lmargin 13
set bmargin 4

set output "residuals.png"
```

```

set ylabel"/*1.5{/Italic residuo inicial}" offset 1,0
set xlabel"/*1.5{/Italic t(s)}" offset 0,0

set logscale y
set format y "%.1e"

plot "logs/Ux_0" using ($1):($2) title"{U_x}" with lines lt 1 lw 5

```

Para gerar a imagem do gráfico, execute no terminal o seguinte comando:

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

A Figura 1.4 mostra a análise da convergência numérica através do resíduo inicial de U_x ao longo da simulação. Note que, no início da simulação os valores dos resíduos são altos e, ao longo do tempo, reduzem até a ordem de grandeza de 10^{-5} . Como o valor especificado para a tolerância na solução da velocidade foi de 10^{-6} , a convergência do resíduo inicial para 10^{-5} indica que o escoamento atingiu o estado estacionário. A análise de uma variável de interesse ao longo do tempo, a fim de confirmar a convergência para o regime estacionário e comparação com a solução analítica é feita a seguir.

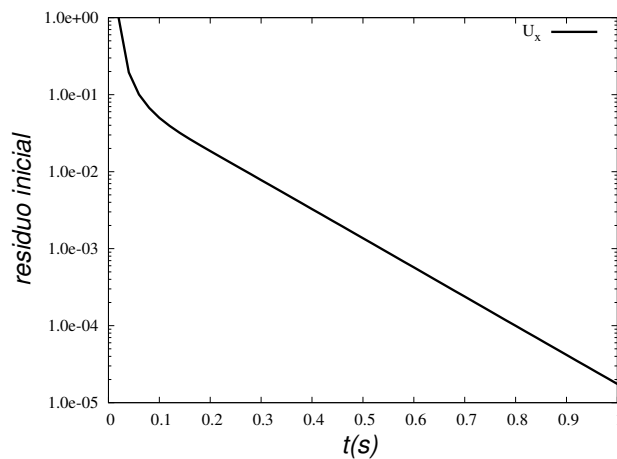


Figura 1.4 Resíduo inicial ao longo da simulação.

1.7 Análise dos resultados

Uma vez verificada a convergência numérica da solução, a próxima etapa consiste em avaliar a variável de interesse no campo. Neste tutorial, a variável de interesse é U_x ao longo da distância y . O utilitário `postProcess` do OpenFOAM® será utilizado para capturar os valores de U_x ao longo de y . Um arquivo adicional, chamado `sampleU` e localizado no diretório `system`, é necessário para especificar o local da geometria onde os valores de U_x devem ser capturados. O Dicionário 1.14 mostra o trecho do dicionário `sampleU`.

Dicionário 1.14 Dicionário `sampleU` no diretório `system`

```

type sets;
libs ("libsampling.so");

```

12 ESCOAMENTO COUETTE ENTRE PLACAS PLANAS

```
interpolationScheme cellPointFace;
setFormat raw;
sets
(
    linhaA
    {
        type          lineUniform;
        nPoints        5;
        axis           y;
        start          (0.2 0 0.05);
        end            (0.2 0.1 0.05);
    }
);
fields
(
    U
);
```

Note que uma linha chamada de `linhaA`, do ponto `start` ao ponto `end`, captura os valores de U e da coordenada y ao longo de 5 pontos (`nPoints`). Em um terminal no diretório do caso, execute o comando abaixo para capturar os valores da velocidade.

```
$ postProcess -func sampleU
```

Um vez concluída a execução do comando, um novo diretório `postProcess` será criado no diretório do caso. Dentro deste diretório `postProcess`, os resultados do dicionário `sampleU` são armazenados em um diretório de mesmo nome, para cada instante de tempo. O `gnuplot` será utilizado novamente para construir o gráfico de U_x em função de y para diferentes instantes de tempo e, ainda, comparar com a solução analítica. O Dicionário 1.15 mostra o arquivo com os comandos do `gnuplot` para traçar o gráfico de U_x vs y . Note que a função $f(x)$ descreve o perfil de velocidade obtido pela solução analítica e, os resultados da simulação, em diferentes instantes de tempo, são lidos diretamente dentro do diretório `postProcess/sampleU`.

Dicionário 1.15 Arquivo com comandos do `gnuplot` para construção do gráfico U_x

```
reset

set term png
set style func linespoints

set autoscale
set lmargin 13
set bmargin 4

set output "Ux.png"

set key left
set xrange [0:0.1]
set xlabel "{/*1.5{/Italic y(m)}" offset 1,0
set ylabel "{/*1.5{/Italic U_x (m/s)} }" offset 0,0

V = 1
h = 0.1
a = V/h
f(x) = a*x

plot f(x) title "analitico" with linespoints lt 1, \
    "postProcessing/sampleU/1/linhaA.xy" using ($1):($2) title "{1s}" \
    with lines lt 2 lw 3.5, \
```



```
"postProcessing/sampleU/0.4/linhaA.xy" using ($1):($2) title"{0.4s}"
with lines lt 3 lw 3.5, \
"postProcessing/sampleU/0.2/linhaA.xy" using ($1):($2) title"{0.2s}"
with lines lt 4 lw 3.5
```

Para gerar a imagem do gráfico, execute no terminal o seguinte comando:

```
$ gnuplot plot-Ux.plt
```

O resultado deste comando é o gráfico, conforme a Figura 1.5(a). Observe que, no instante de tempo 0.2s o perfil de velocidade ainda está bem distante da solução no estado estacionário. Já no instante de tempo 0.6s, não é possível fazer distinção visual entre o perfil de velocidade simulado e o analítico.

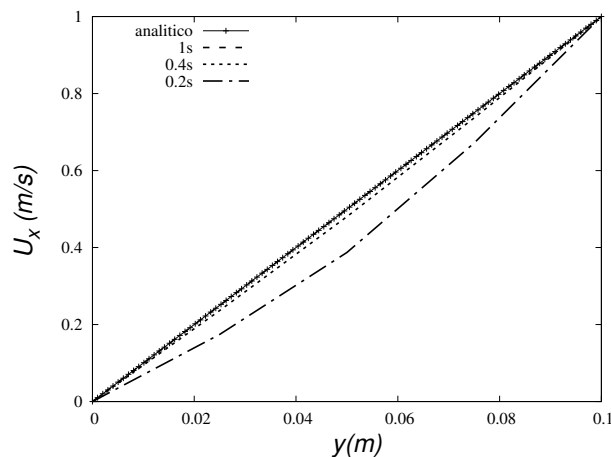


Figura 1.5 Comparação do perfil de velocidade.

EXERCÍCIOS

1.1 Execute o tutorial novamente porém, altere o número de Reynolds. Utilize a viscosidade cinemática ν , no dicionário `physicalProperties` no diretório `constant`, para alterar o valor do Reynolds do escoamento. Verifique a convergência numérica, faça a comparação com a solução analítica e informe qual foi o tempo final necessário para atingir o estado estacionário.

- Reynolds = 100;
- Reynolds = 1000;

Referências Bibliográficas

- [1] Donald F. Young, Bruce R. Munson, Theodore H. Okiishi, A Brief Introduction to Fluid Mechanics, Fifth Edition, John Wiley Sons, Inc., pg 225, 2011.
- [2] Christopher J. Greenshields, OpenFOAM, The OpenFOAM Foundation, User Guide, version 7, 2019.