VERIFICAÇÃO DE CÓDIGO PARA SIMULAÇÃO DE ESCOAMENTOS UTILIZANDO A FORMULAÇÃO FUNÇÃO CORRENTE-VORTICIDADE COM O MÉTODO DE ELEMENTOS FINITOS

Rodolfo Rafael Palacios Carrasco

Projeto de Graduação apresentado ao Curso de Engenharia Mecânica da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientador: Gustavo Rabello dos Anjos

Rio de Janeiro

Julho de 2022

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

Departamento de Engenharia Mecânica

DEM/POLI/UFRJ

VERIFICAÇÃO DE CÓDIGO PARA SIMULAÇÃO DE ESCOAMENTOS UTILIZANDO A FORMULAÇÃO FUNÇÃO CORRENTE-VORTICIDADE COM O MÉTODO DE ELEMENTOS FINITOS

Rodolfo Rafael Palacios Carrasco

PROJETO FINAL SUBMETIDO AO CORPO DOCENTE DO DEPARTAMENTO DE ENGENHARIA MECÂNICA DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO MECÂNICO.

Aprovada por:

_____

Prof. Gustavo Rabello dos Anjos, Ph.D.

_____

Prof. Fernando Augusto de Noronha Castro Pinto, Dr.-Ing

_____

Prof. Átila Pantaleão Silva Freire, Ph.D.

RIO DE JANEIRO, RJ – BRASIL

JULHO DE 2022

Palacios Carrasco, Rodolfo Rafael

VERIFICAÇÃO DE CÓDIGO PARA SIMULAÇÃO DE ESCOAMENTOS UTILIZANDO A FORMULAÇÃO FUNÇÃO CORRENTE-VORTICIDADE COM O MÉTODO DE ELEMENTOS FINITOS/ Rodolfo Rafael Palacios Carrasco. – Rio de Janeiro: UFRJ/Escola Politécnica, 2022.

XIII, 67 p.: il.; 29, 7cm.

Orientador: Gustavo Rabello dos Anjos

Projeto de Graduação – UFRJ/ Escola Politécnica/ Curso de Engenharia Mecânica, 2022.

Referências Bibliográficas: p. 54 – 56.

1. Elementos Finitos. 2. Corrente-vorticidade. 3. CFD. 4. Multiprocessamento. I. dos Anjos, Gustavo Rabello. II. Universidade Federal do Rio de Janeiro, UFRJ, Curso de Engenharia Mecânica. III. VERIFICAÇÃO DE CÓDIGO PARA SIMULAÇÃO DE ESCOAMENTOS UTILIZANDO A FORMULAÇÃO FUNÇÃO CORRENTE-VORTICIDADE COM O MÉTODO DE ELEMENTOS FINITOS.

À todas a pessoas que me ajudaram na caminhada pela graduação. À meu pai e minha mãe, Lorenzo e Silvia

Agradecimentos

Agradeço ao meu orientador Gustavo Rabello do Anjos por toda sua paciência de tirar minhas dúvidas, por mais absurdas que sejam. Agradeço a minha mãe Silvia e a meu pai Lorenzo por todo o apoio durante toda minha graduação.

Resumo do Projeto de Graduação apresentado à Escola Politécnica/UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro Mecânico

VERIFICAÇÃO DE CÓDIGO PARA SIMULAÇÃO DE ESCOAMENTOS UTILIZANDO A FORMULAÇÃO FUNÇÃO CORRENTE-VORTICIDADE COM O MÉTODO DE ELEMENTOS FINITOS

Rodolfo Rafael Palacios Carrasco

Julho/2022

Orientador: Gustavo Rabello dos Anjos

Programa: Engenharia Mecânica

Modelos matemáticos para mecânica de fluidos são muito importares para tentar prever seu comportamento. A sua aplicação varia desde oleodutos até a área aeroespacial. Existem diversas maneiras de simular um escoamento, podendo ser empregado o método de diferenças finitas, volumes finitos e elementos finitos. Cada formulação apresenta sua particularidade, não existindo particularmente o melhor método. Sob essa ótica decidimos verificar a validade dos resultados, obtidos pela formulação corrente-vorticidade usando o método de elementos finitos, em certa faixa de número de Reynolds.

Abstract of Undergraduate Project presented to POLI/UFRJ as a partial fulfillment of the requirements for the degree of Mechanical Engineer

CODE VERIFICATION FOR FLOW SIMULATION USING THE STREAMFUNCTION-VORTICITY FORMULATION WITH THE FINITE ELEMENT METHOD

Rodolfo Rafael Palacios Carrasco

July/2022

Advisor: Gustavo Rabello dos Anjos

Department: Mechanical Engineering

Mathematical models for fluid mechanics are very important to try to predict their behavior. Its application ranges from pipelines to aerospace. There are several ways to simulate a flow, using the finite difference, finite volume and finite element method. Each formulation has its particularity, and there is no particular best method. From this point of view, we decided to verify the validity of the results, obtained by the current-votricity formulation using the finite element method, in a certain range of Reynolds number.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Os problemas que envolvem a interação entre um fluido e uma sólido elástico são conhecidos como aeroelasticidade ou Interação Fluido-estrutura. Neste tipo de problema três tipos de forças interagem entre si, a força aerodinâmica ou hidrodinâmica, a elástica e a inercial.

Podemos fazer a interseção dessas forças e obter quatro campos de estudo

1. Aeroelasticidade estática (Aerodinâmica - Forças elásticas)
2. Estabilidade dinâmica (Aerodinâmica - Forças inerciais)
3. Vibrações mecânicas (Forças elásticas - Forças inerciais)
4. Aeroelasticidade dinâmica (Aerodinâmica - Forças elásticas - Forças inerciais)

Nesse trabalho vamos desenvolver um código usando a linguagem de programação Python, para simular o escoamento ao redor de um cilindro com fronteiras rígidas, sem movimentação da malha

Vamos utilizar uma formulação bidimensional como base do projeto. Na simulação é importante saber a cinemática que vai ser utilizada par a formulação. Existem três tipos, euleriana, lagrangiana e lagrangiana-euleriana arbitrária.

A euleriana é utilizada principalmente para a mecânica do fluidos. Os pontos na malha são fixos e o fluido escoa pela malha. O maior problema dessa formulação é de obter a interação com fronteiras moveis ou elásticas e entre diferentes tipos de fluidos.

A lagrangiana é utilizada principalmente para a mecânica dos sólidos. Nesta formulação os pontos da malha tem suas próprias características.

A lagrangiana-euleriana arbitrária é muito útil, pois combina as outras duas formulações, para problemas que envolvem grandes distorções nas fronteiras ou o deslocamento das próprias.

Neste trabalho vamos fazer a formulação baseada na cinemática euleriana.

1.1 Motivação

Na engenharia existem diversos problemas ligados a interação entre dois meios distintos, sejam eles sólido/sólido, fluido/sólido, e fluido/fluido. Entre os diversos problemas estão a troca de calor num leito fluidizado, a oscilação induzida em uma estrutura de perfuração em alto mar e a oscilação induzida na ponta da asa de um avião (*flutter*). Para podermos iniciar esse estudo devemos verificar se a formulação a ser usada tem validade para casos mais simples, se a afirmativa anterior foi verdadeira prosseguiremos, em trabalhos posteriores, para uma formulação mais complexa.

1.2 Objetivo

O principal objetivo a ser alcançado é desenvolver o código verificar-lo, para posteriormente apresentar alguns resultados condizentes com a literatura envolvendo problemas estáticos.

1.3 Organização do trabalho

Em primeiro lugar vamos fazer uma revisão sobre o método de elementos finitos seguido da formulação específica para o problema em questão. Mais a frente será feita uma apresentação sobre o algoritmo de solução com alguns cometários sobre a otimização utilizando ferramentas existentes. Também serão apresentadas as validações numéricas para alguns problemas específicos. Em seguida vamos comparar os resultados obtidos com outras simulações, obtidas na bibliografia especializada.

Capítulo 2

Revisão Bibliográfica

2.1 Método de elementos finitos

Um dos principais desafios de um engenheiro é como modelar em termos matemáticos os fenômenos da natureza. Para esse fim, é muito comum utilizar equações diferenciais parciais. Em muitos casos não existe uma solução analítica generalizada, ela existe apenas para alguns casos específicos, que podem ser utilizados para validar o modelo numérico. Serve para verificar o erro associado para a solução obtida.

Existem algumas maneiras para resolver as equações numericamente; elementos finitos, volumes finitos e diferenças finitas. Dependendo do caso um método é mais conveniente que o outro. Isso pode ser determinado pela natureza matemática do problema relacionado a estabilidade, complexidade e tipo de equação.

De acordo com *Zienkiewicz et al.* [3] desde os anos 1940, métodos intuitivos de discretização, no estudo de mecânica dos sólidos, do meio contínuo se dava através da divisão do mesmo em pequenos elementos elásticos. Usando o mesmo método *Turner et al* [4] mostrou que a substituição direta pode ser feita de maneira mais eficiente, fazendo que cada elemento se comporte de maneira mais simples, comparado ao meio contínuo.

A primeira citação do método de elementos finitos remonta a *Clough et al.* [5]. Desde os anos 1960 a analogia de dividir o problema em pequenas partes, respaldada pela formalidade matemática, de acordo com *Zienkiewicz et al.* [3], convergiam para um procedimento generalizado para a solução de meios contínuos.

Mas apenas em 1976 foi possível utilizar esse método para a dinâmica do fluidos,

antes não existia um problema em alcançar a estabilidade numérica. Isso se devia ao termo convectivo das equações, resultado em oscilações espúrias.

Mais tarde em 1984 *Donea* [6] propôs o a discretização Taylor-Galerkin, seu objetivo é melhorar a derivada temporal utilizando a expansão de Taylor, incluindo os termos de segunda e terceira ordem, os quais são obtidos a partir das equações de governo. Esse será o método escolhido no presente trabalho para melhorar a estabilidade numérica.

2.2 Emissão de vórtices

Quando um cilindro circular é colocado em um fluxo orientado em qualquer direção radial ao cilindro, ele experimentará uma força de sustentação variável devido à formação de vórtices assimétricos. Uma das principais características é a capacidade do sistema de sincronizar a frequência de desprendimento de vórtices (f_V), relacionado com o fluidodinâmica, com a frequência natural (f_N), relacionado com a natureza estrutural, do sistema. A nossa formualção não vai explorar o movimento induzido pelos vortcies, apenas será feita uma analise do escoamento através de estruturas estáticas.

Para iniciar nossa pesquisa, precisamos investigar a relação entre dois números adimensionais bem conhecidos na literatura, o número de Reynolds e o número de Strouhal. Eles podem ser obtidos a partir do teorema de π de Vaschy-Buckingham [7, 8]. Podendo ser definidos da seguinte forma.

$$Re = \frac{U_\infty \cdot D}{v} \tag{2.1}$$

$$Sr = \frac{f \cdot D}{U_\infty} \tag{2.2}$$

Sendo D o comprimento característico, v a viscosidade cinemática, U_∞ a velocidade inicial do fluido e f a frequência de emissão dos vórtices.

Segundo *Fey et al.* [2] a relação entre o numero de Strouhal e o numero de Reynolds pode ser aproximada por uma relação linear, para algumas faixas de Reynolds, se Sr é plotado em função de $1/\sqrt{Re}$. Além disso, podemos escrever a equação linear

da seguinte forma.

$$Sr = Sr^* + \alpha/\sqrt{Re} \tag{2.3}$$

Sendo Sr^* e α as constantes lineares que melhor se adaptam ao intervalo de número de Reynolds de interesse.

Re	Sr^*	α	δSr
$47 < Re < 180$	0.2684	-1.0356	0.001
$180 < Re < 230$	0.2437	-0.8607	0.0015
$230 < Re < 240$	0.4291	-0.36735	0.0015
$240 < Re < 360$	não definido		
$360 < Re < 1300$	0.2257	-0.4402	0.0015

Tabela 2.1: Coeficientes $Sr*$ e m da equação 2.3para alguns intervalos de Reynolds, com estimativa de erro δSr, adaptado de *Fey et al.* [2]

A tabela 2.1 foi obtida por dados experimentais com a relação de $L/D \geq 50$. Apesar de ser muito útil para simulações de escoamentos sobre objetos estáticos, não podemos afirmar que essa relação se manterá com o movimento do cilindro visto que as condições de contorno do problema serão alteradas constantemente. No entanto, podemos usar essas relações para validar o código, verificando se a simulação condiz com os dados experimentais obtidos.

Capítulo 3

Modelo matemático

3.1 Equações de governo

3.1.1 Premissas

Primeiro deduziremos as equações e depois vamos discretiza-las. Inicialmente consideraremos no nosso modelo, que o meio é contínuo em todo o domínio. A seguir está uma lista das premissas iniciais.

- Conservação do momento linear
- Conservação da massa
- Equação da continuidade

3.1.2 Conservação de massa

Para satisfazer a relação entre a densidade e o fluxo de massa no volume de controle, ela pode ser expressa pela seguinte equação.

$$\int_V \frac{\partial}{\partial t} dm = 0 \tag{3.1}$$

Usando um pouco de álgebra para desenvolver a equação chegamos a:

$$\int_V \frac{\partial \rho}{\partial t} dV = 0 \tag{3.2}$$

O fluxo de massa no volume de controle pode ser escrito como:

$$\oint_S \rho \mathbf{v} dS = 0 \tag{3.3}$$

Usando o teorema de Gauss na equação (3.3).

$$\int_V \nabla \cdot \rho \mathbf{v} dA = 0 \tag{3.4}$$

Assumindo que no volume de controle não será criada massa, podemos afirmar que a variação da densidade mais a variação de massa precisa ser igual a zero.

$$\int_V \frac{\partial \rho}{\partial t} dV + \int_V \nabla \cdot \rho \mathbf{v} dA = 0 \tag{3.5}$$

3.1.3 Forma integral da conservação de momento linear

Começamos com a segunda lei do movimento de Newton.

$$\sum \mathbf{F} = \frac{d\mathbf{P}}{dt} \tag{3.6}$$

P é o momento linear. Para um volume de controle, sendo suas fronteiras limitadas por superfícies de controle, a forma integral pode ser escrita da seguinte maneira.

$$\frac{d}{dt} \int_V \rho \mathbf{v} \ dV + \oint_S \mathbf{v} (\rho \mathbf{v} \ \cdot dS) = 0 \tag{3.7}$$

Para o somatório de forças nos contabilizamos as forças de campo e de superfície.

$$\int_V \rho \mathbf{g} dV + \oint_S \sigma \cdot dS = 0 \tag{3.8}$$

Substituindo os termos da equação (3.6).

$$\int_V \frac{\partial}{\partial t} (\rho \mathbf{v}) dV + \oint_S \rho (\mathbf{v}\mathbf{v}) dS) = \int_V \rho \mathbf{g} dV + \oint_S \sigma \cdot dS \tag{3.9}$$

Usando o teorema de Gauss nas integrais de área.

$$\int_V \frac{\partial}{\partial t} (\rho \mathbf{v}) dV + \int_V \nabla \rho (\mathbf{v}\mathbf{v}) dV = \int_V \rho \mathbf{g} dV + \int_V \nabla \sigma dV \tag{3.10}$$

Reorganizando os termos das equações.

$$\int_V \left[\frac{\partial}{\partial t}(\rho \mathbf{v}) + \nabla \rho(\mathbf{v}\mathbf{v}) - \nabla \sigma - \rho \mathbf{g} \right] dV = 0 \tag{3.11}$$

Como $dV \neq 0$, podemos escrever que:

$$\frac{\partial}{\partial t}(\rho \mathbf{v}) + \nabla \rho(\mathbf{v}\mathbf{v}) - \nabla \sigma - \rho \mathbf{g} = 0 \tag{3.12}$$

Usando a regra do produto para as derivadas espacias e temporais.

$$\mathbf{v} \underbrace{\left(\frac{\partial \rho}{\partial t} + \nabla(\rho \mathbf{v}) \right)}_{\text{equação da continuidade}} + \rho \left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \nabla \mathbf{v} \right) - \nabla \sigma - \rho \mathbf{g} = 0 \tag{3.13}$$

Agrupando e reorganizando alguns elementos podemos obter:

$$\rho \left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \nabla \mathbf{v} \right) = \nabla \sigma + \rho \mathbf{g} \tag{3.14}$$

O tensor de tensões σ pode ser reescrito da seguinte maneira.

$$\sigma = -p\mathbf{I} + \tau \tag{3.15}$$

Sendo p o campo de pressões, $\mathbf{I}$ a matriz identidade e τ o tensor de cisalhamento. Para a equação (3.15) assumiremos que o fluido é Newtoniano e incompressível, resultando em:

$$\tau = \mu \left[\nabla \mathbf{v} + \left(\nabla \mathbf{v}^T \right) \right] \tag{3.16}$$

Aplicando o operador ∇ à equação (3.16).

$$\nabla \tau = \mu \nabla^2 \mathbf{v} \tag{3.17}$$

Usando as equações (3.17) e (3.15) e substituindo em (3.14).

$$\rho \left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \nabla \mathbf{v} \right) = -\nabla p + \mu \nabla^2 \mathbf{v} + \rho \mathbf{g} \tag{3.18}$$

Usando a relação $\mu/\rho = \upsilon$, sendo υ a viscosidade cinemática.

$$\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \nabla \mathbf{v} = -\frac{1}{\rho} \nabla p + \upsilon \nabla^2 \mathbf{v} + \mathbf{g} \tag{3.19}$$

3.1.4 Formulação corrente-vorticidade

Neste trabalho usaremos a função corrente vorticidade: ela é muito útil para casos bidimensionais, visto que contorna o problema do acoplamento entre os campos de pressão e velocidade. Inicialmente é necessário introduzir algumas definições.

$$\frac{\partial \Psi}{\partial x} = \mathrm{v}_y, \frac{\partial \Psi}{\partial y} = -\mathrm{v}_x \tag{3.20}$$

$$\omega = \frac{\partial \mathrm{v}_x}{\partial y} - \frac{\partial \mathrm{v}_y}{\partial x} \rightarrow \omega = \nabla \times \mathrm{v} \tag{3.21}$$

Usando a equação (3.20) em (3.21) podemos obter a seguinte relação.

$$-\omega = \frac{\partial^2 \Psi}{\partial^2 y} + \frac{\partial^2 \Psi}{\partial^2 x} \tag{3.22}$$

$$-\nabla^2 \Psi = \emptyset mega \tag{3.23}$$

Sabendo que a seguinte identidade é verdadeira.

$$\mathbf{v} \cdot \nabla \mathbf{v} = \nabla \cdot \frac{\mathrm{v}^2}{2} - \mathbf{v} \times \nabla \times \mathbf{v} \tag{3.24}$$

Usando (3.24) na equação (3.19).

$$\frac{\partial \mathbf{v}}{\partial t} + \nabla \cdot \frac{\mathrm{v}^2}{2} - \mathbf{v} \times \nabla \times \mathbf{v} = -\frac{1}{\rho} \nabla p + \upsilon \nabla^2 \mathbf{v} + \mathbf{g} \tag{3.25}$$

Aplicando o operador ∇ a equação (3.25).

$$\begin{aligned} \nabla \times \left(\frac{\partial \mathbf{v}}{\partial t} \right) + \underbrace{\nabla \times \left(\nabla \cdot \frac{\mathrm{v}^2}{2} \right)}_{=0} - \nabla \times (\mathbf{v} \times \nabla \times \mathbf{v}) \\ = - \underbrace{\nabla \times \left(\frac{1}{\rho} \nabla p \right)}_{=0} + \nabla \times \left(\upsilon \nabla^2 \mathbf{v} \right) + \underbrace{\nabla \times (\mathbf{g})}_{=0} \end{aligned} \tag{3.26}$$

$$\frac{\partial}{\partial t} (\nabla \times \mathbf{v}) - \nabla \times (\mathbf{v} \times \nabla \times \mathbf{v}) = \upsilon \nabla^2 (\nabla \times \mathbf{v}) \tag{3.27}$$

Usando a definição (3.21) na equação (3.27).

$$\frac{\partial}{\partial t} \omega - \underbrace{\nabla \times (\mathbf{v} \times \omega)}_{} = \upsilon \nabla^2 \omega \tag{3.28}$$

O termo destacado na equação (3.28) pode ser expresso da seguinte maneira.

$$\nabla \times (\mathrm{v} \times \omega) = -\mathbf{v} \cdot \nabla\omega + \omega\nabla\mathbf{v} \tag{3.29}$$

Usando-o na equação 3.28).

$$\frac{\partial}{\partial t}\omega + \mathbf{v} \cdot \nabla\omega - \omega\nabla\mathbf{v} = v\nabla^2\omega \tag{3.30}$$

Neste trabalho assumiremos que o fluido é incompressível, sendo assim:

$$\nabla \cdot \mathbf{v} = 0 \tag{3.31}$$

$$\frac{\partial}{\partial t}\omega + \mathbf{v} \cdot \nabla\omega = v\nabla^2\omega \tag{3.32}$$

3.2 Formulação do método de elementos finitos

3.2.1 Discretização do tempo

Neste trabalho usaremos a expansão da série de Taylor, tomando apenas os primeiros termos da série. Usando a equação (3.32) como ponto de partida, e expandido o operador ∇.

$$\frac{\partial\omega}{\partial t} + \mathrm{v}_x\frac{\partial\omega}{\partial x} + \mathrm{v}_y\frac{\partial\omega}{\partial y} = v\frac{\partial^2\omega}{\partial x^2} + v\frac{\partial^2\omega}{\partial y^2} \tag{3.33}$$

Isolando a derivada temporal.

$$\frac{\partial\omega}{\partial t} = -\mathrm{v}_x\frac{\partial\omega}{\partial x} - \mathrm{v}_y\frac{\partial\omega}{\partial y} + v\frac{\partial^2\omega}{\partial x^2} + v\frac{\partial^2\omega}{\partial y^2} \tag{3.34}$$

Para a discretização do tempo utilizaremos o método de diferenças finitas regressivas.

$$\omega_n = \sum_{k=0}^{\infty}(-1)^k\frac{\partial^k\omega_{n+1}}{\partial t^k}\frac{\Delta t^k}{k!} \tag{3.35}$$

Devido a restrições computacionais, somente serão usados os três primeiro termos da série. Poderiamos também selecionar apenas os dois primeiros termos, mas a convergência numérica ficaria comprometica para a faixa de numero de Reynolds

requerida para o problema. Com o terceiro termo o problema de ocilações epúrias e minimizado.

$$\omega_n = \omega_{n+1} - \frac{\partial \omega_{n+1}}{\partial t}\Delta t + \frac{\partial^2 \omega_{n+1}}{\partial t^2}\frac{(\Delta t)^2}{2} \tag{3.36}$$

Usando a equação (3.34) em (3.36).

$$\begin{aligned}\omega_n = \omega_{n+1} - \Delta t \left(-\mathrm{v}_x \frac{\partial \omega_{n+1}}{\partial x} - \mathrm{v}_y \frac{\partial \omega_{n+1}}{\partial y} + v\frac{\partial^2 \omega_{n+1}}{\partial x^2} + v\frac{\partial^2 \omega_{n+1}}{\partial y^2}\right) \\ + \frac{(\Delta t)^2}{2}\frac{\partial}{\partial t}\left(-\mathrm{v}_x \frac{\partial \omega_n}{\partial x} - \mathrm{v}_y \frac{\partial \omega_{n+1}}{\partial y} + v\frac{\partial^2 \omega_{n+1}}{\partial x^2} + v\frac{\partial^2 \omega_{n+1}}{\partial y^2}\right)\end{aligned} \tag{3.37}$$

No último termo podemos mudar a ordem da derivada parcial.

$$\begin{aligned}\omega_n = \omega_{n+1} - \Delta t \left(-\mathrm{v}_x \frac{\partial \omega_{n+1}}{\partial x} - \mathrm{v}_y \frac{\partial \omega_{n+1}}{\partial y} + v\frac{\partial^2 \omega_{n+1}}{\partial x^2} + v\frac{\partial^2 \omega_{n+1}}{\partial y^2}\right) \\ + \frac{(\Delta t)^2}{2}\left(-\mathrm{v}_x \frac{\partial}{\partial x}\frac{\partial \omega_{n+1}}{\partial t} - \mathrm{v}_y \frac{\partial}{\partial y}\frac{\partial \omega_{n+1}}{\partial t} + v\frac{\partial^2}{\partial x^2}\frac{\partial \omega_{n+1}}{\partial t} + v\frac{\partial^2}{\partial y^2}\frac{\partial \omega_{n+1}}{\partial t}\right)\end{aligned} \tag{3.38}$$

Usando mais uma vez a equação (3.34).

$$\begin{aligned}\omega_{n+1} = \omega_n + \Delta t \left(-\mathrm{v}_x \frac{\partial \omega_{n+1}}{\partial x} - \mathrm{v}_y \frac{\partial \omega_{n+1}}{\partial y} + v\frac{\partial^2 \omega_{n+1}}{\partial x^2} + v\frac{\partial^2 \omega_{n+1}}{\partial y^2}\right) \\ + \frac{(\Delta t)^2}{2}\left[-\mathrm{v}_x \frac{\partial}{\partial x}\left(-\mathrm{v}_x \frac{\partial \omega_{n+1}}{\partial x} - \mathrm{v}_y \frac{\partial \omega_{n+1}}{\partial y} + v\frac{\partial^2 \omega_{n+1}}{\partial x^2} + v\frac{\partial^2 \omega_{n+1}}{\partial y^2}\right)\right. \\ - \mathrm{v}_y \frac{\partial}{\partial y}\left(-\mathrm{v}_x \frac{\partial \omega_{n+1}}{\partial x} - \mathrm{v}_y \frac{\partial \omega_{n+1}}{\partial y} + v\frac{\partial^2 \omega_{n+1}}{\partial x^2} + v\frac{\partial^2 \omega_{n+1}}{\partial y^2}\right) \\ + v\frac{\partial^2}{\partial x^2}\left(-\mathrm{v}_x \frac{\partial \omega_{n+1}}{\partial x} - \mathrm{v}_y \frac{\partial \omega_{n+1}}{\partial y} + v\frac{\partial^2 \omega_{n+1}}{\partial x^2} + v\frac{\partial^2 \omega_{n+1}}{\partial y^2}\right) \\ \left. + v\frac{\partial^2}{\partial y^2}\left(-\mathrm{v}_x \frac{\partial \omega_{n+1}}{\partial x} - \mathrm{v}_y \frac{\partial \omega_{n+1}}{\partial y} + v\frac{\partial^2 \omega_{n+1+1}}{\partial x^2} + v\frac{\partial^2 \omega_{n+1+1}}{\partial y^2}\right)\right]\end{aligned} \tag{3.39}$$

Os termos de terceira ordem contribuem pouco no erro final, ou seja para nosso modelo essa parcela vai assumir o valor igual a zero.

$$\begin{aligned}\omega_n = \omega_{n+1} - \Delta t \left(-\mathrm{v}_x \frac{\partial \omega_{n+1}}{\partial x} - \mathrm{v}_y \frac{\partial \omega_{n+1}}{\partial y} + v\frac{\partial^2 \omega_{n+1}}{\partial x^2} + v\frac{\partial^2 \omega_{n+1}}{\partial y^2}\right) \\ + \frac{(\Delta t)^2}{2}\left[-\mathrm{v}_x \frac{\partial}{\partial x}\left(-\mathrm{v}_x \frac{\partial \omega_{n+1}}{\partial x} - \mathrm{v}_y \frac{\partial \omega_{n+1}}{\partial y}\right) - \mathrm{v}_y \frac{\partial}{\partial y}\left(-\mathrm{v}_x \frac{\partial \omega_{n+1}}{\partial x} - \mathrm{v}_y \frac{\partial \omega_{n+1}}{\partial y}\right)\right]\end{aligned} \tag{3.40}$$

$$\left(\frac{\omega_{n+1}-\omega_n}{\Delta t}\right)+\mathrm{v}_x\frac{\partial\omega_{n+1}}{\partial x}+\mathrm{v}_y\frac{\partial\omega_{n+1}}{\partial y}=v\frac{\partial^2\omega_n}{\partial x^2}+v\frac{\partial^2\omega_n}{\partial y^2}$$
$$+\frac{(\Delta t)}{2}\left[-\mathrm{v}_x\frac{\partial}{\partial x}\left(-\mathrm{v}_x\frac{\partial\omega_{n+1}}{\partial x}-\mathrm{v}_y\frac{\partial\omega_{n+1}}{\partial y}\right)-\mathrm{v}_y\frac{\partial}{\partial y}\left(-\mathrm{v}_x\frac{\partial\omega_{n+1}}{\partial x}-\mathrm{v}_y\frac{\partial\omega_{n+1}}{\partial y}\right)\right] \tag{3.41}$$

Usando uma notação mais compacta.

$$\frac{\omega_{n+1}-\omega_n}{\Delta t}+\mathbf{v}\cdot\nabla\omega_n-\frac{\Delta t}{2}\mathbf{v}\cdot\nabla\cdot(\mathbf{v}\cdot\nabla\omega)=v\nabla^2\omega \tag{3.42}$$

3.2.2 Formulação dos resíduos ponderados

A formulação a seguir consiste em integrar funções de teste w_i a função de governo, para minimizar os resíduos. Usando as equações (3.21), (3.23 e (3.42).

$$\dot{\omega}+\mathbf{v}\cdot\nabla\omega-v\nabla^2\omega-\frac{\Delta t}{2}\mathbf{v}\cdot\nabla\cdot(\mathbf{v}\cdot\nabla\omega)=R_1 \tag{3.43}$$

$$\nabla^2\cdot\psi+\omega=R_2 \tag{3.44}$$

$$\mathbf{v}-(\partial/\partial y,-\partial/\partial x)\,\psi^T=R_3 \tag{3.45}$$

Assumindo que o erro dessas equações é R_i multiplicado por uma função peso w_i no domínio ω, queremos encontrar as funções w_i que minimizam esse erro. Sendo assim:

$$\int_\Omega R_1\cdot\mathrm{w}_1 d\Omega=0 \tag{3.46}$$

$$\int_\Omega R_2\cdot\mathrm{w}_2 d\Omega=0 \tag{3.47}$$

$$\int_\Omega R_3\cdot\mathrm{w}_3 d\Omega=0 \tag{3.48}$$

Usando as equações (3.43), (3.44) e (3.45).

$$\int_\Omega\left[\dot{\omega}+\mathbf{v}\cdot\nabla\omega-v\nabla^2\omega-\frac{\Delta t}{2}\mathbf{v}\cdot\nabla\cdot(\mathbf{v}\cdot\nabla\omega)\right]\mathrm{w}_1 d\Omega=0 \tag{3.49}$$

$$\int_{\Omega} \left(\nabla^2 \cdot \psi + \omega\right) \mathrm{w}_2 d\Omega = 0 \tag{3.50}$$

$$\int_{\Omega} \left[\mathbf{v} - (\partial/\partial y, -\partial/\partial x)\, \psi^T\right] \mathrm{w}_3 d\Omega = 0 \tag{3.51}$$

Como a integral é um operador linear as equações (3.49), (3.50), (3.51) podem ser expandidas.

$$\int_{\Omega} \dot{\omega} \cdot \mathrm{w}_1 d\Omega + \int_{\Omega} (\mathbf{v}\nabla\omega) \cdot \mathrm{w}_1 d\Omega - \underbrace{\int_{\Omega} v\nabla^2\omega \cdot \mathrm{w}_1 d\Omega}_{\text{termo difusivo}} - \underbrace{\frac{\Delta t}{2} \int_{\Omega} \mathbf{v}\nabla(\mathbf{v}\nabla\omega) \cdot \mathrm{w}_1 d\Omega}_{\text{termo numerico difusivo}} = 0 \tag{3.52}$$

$$\underbrace{\int_{\Omega} \nabla^2\psi \cdot \mathrm{w}_2 d\Omega}_{\text{termo difusivo}} + \int_{\Omega} \omega \cdot \mathrm{w}_2 d\Omega = 0 \tag{3.53}$$

$$\int_{\Omega} \mathbf{v} \cdot \mathrm{w}_3 d\Omega - \int_{\Omega} \left[(\partial/\partial y, -\partial/\partial x)\, \psi^T\right] \cdot \mathrm{w}_3 d\Omega = 0 \tag{3.54}$$

Nos termos destacados podemos aplicar o teorema de Gauss para reduzir a ordem da derivada.

$$-\int_{\Omega} v\nabla^2\omega \cdot \mathrm{w}_1 d\Omega = \int_{\Omega} v(\nabla\omega)(\nabla \mathrm{w}_1) d\Omega - \int_{\Gamma} v(\nabla\omega)\mathrm{w}_1 \cdot e_i d\Gamma \tag{3.55}$$

$$-\frac{\Delta t}{2} \int_{\Omega} \mathbf{v}\nabla(\mathbf{v}\nabla\omega) \cdot \mathrm{w}_1 d\Omega = \frac{\Delta t}{2} \int_{\Omega} (\mathbf{v}\nabla\omega)\mathbf{v}\nabla \mathrm{w}_1 d\Omega - \frac{\Delta t}{2} \int_{\Gamma} (\mathbf{v}\nabla\omega)\mathbf{v}\mathrm{w}_1 \cdot e_i d\Gamma \tag{3.56}$$

$$\int_{\Omega} \nabla^2\psi \cdot \mathrm{w}_2 d\Omega = \int_{\Gamma} \nabla\psi \mathrm{w}_2 \cdot e_i d\Gamma - \int_{\Omega} \nabla\psi\nabla \mathrm{w}_2 d\Omega \tag{3.57}$$

Para a condição de contorno de Dirichlet, a função w_i assume o valor zero, equações (3.77), (3.78) e (3.79).

$$-\int_{\Omega} v\nabla^2\omega \cdot \mathrm{w}_1 d\Omega = \int_{\Omega} v(\nabla\omega)(\nabla \mathrm{w}_1) d\Omega \tag{3.58}$$

$$-\frac{\Delta t}{2} \int_{\Omega} \mathbf{v}\nabla(\mathbf{v}\nabla\omega) \cdot \mathrm{w}_1 d\Omega = \frac{\Delta t}{2} \int_{\Omega} (\mathbf{v}\nabla\omega)\mathbf{v}\nabla \mathrm{w}_1 d\Omega \tag{3.59}$$

$$\int_\Omega \nabla^2\psi \cdot \mathrm{w}_2 d\Omega = -\int_\Omega \nabla\psi\nabla\mathrm{w}_2 d\Omega \tag{3.60}$$

Usando as equações (3.58), (3.59) e (3.60).

$$\int_\Omega \dot{\omega}\cdot \mathrm{w}_1 d\Omega + \int_\Omega (\mathbf{v}\nabla\omega)\cdot \mathrm{w}_1 d\Omega + \int_\Omega v(\nabla\omega)(\nabla \mathrm{w}_1)d\Omega + \frac{\Delta t}{2}\int_\Omega (\mathbf{v}\nabla\omega)\mathbf{v}\nabla \mathrm{w}_1 d\Omega = 0 \tag{3.61}$$

$$-\int_\Omega \nabla\psi\nabla\mathrm{w}_2 d\Omega + \int_\Omega \omega\cdot \mathrm{w}_2 d\Omega = 0 \tag{3.62}$$

3.2.3 Discretização do espaço

Usando álgebra para expandir os termos nas equações (3.61), (3.62) e (3.54).

$$\begin{aligned}\int_\Omega \dot{\omega}\cdot \mathrm{w}_1 d\Omega + \int_\Omega \mathrm{v}_x\frac{\partial\omega}{\partial x}\mathrm{w}_1 d\Omega + \int_\Omega \mathrm{v}_y\frac{\partial\omega}{\partial y}\mathrm{w}_1 d\Omega + v\int_\Omega \frac{\partial\omega\partial \mathrm{w}_1}{\partial x\partial x}d\Omega + v\int_\Omega \frac{\partial\omega\partial \mathrm{w}_1}{\partial y\partial y}d\Omega \\ + \frac{\Delta t}{2}\int_\Omega \mathrm{v}_x\frac{\partial \mathrm{w}_1}{\partial x}\left(\mathrm{v}_x\frac{\partial\omega}{\partial x} + \mathrm{v}_y\frac{\partial\omega}{\partial y}\right)d\Omega \\ + \frac{\Delta t}{2}\int_\Omega \mathrm{v}_y\frac{\partial \mathrm{w}_1}{\partial y}\left(\mathrm{v}_x\frac{\partial\omega}{\partial x} + \mathrm{v}_y\frac{\partial\omega}{\partial y}\right)d\Omega = 0\end{aligned} \tag{3.63}$$

$$-\int_\Omega \left(\frac{\partial\psi\partial \mathrm{w}_2}{\partial x\partial x} + \frac{\partial\psi\partial \mathrm{w}_2}{\partial y\partial y}\right)d\Omega + \int_\Omega \omega \mathrm{w}_2 d\Omega = 0 \tag{3.64}$$

$$\int_\Omega \mathrm{v}_x \mathrm{w}_3 - \int_\Omega \frac{\partial\psi}{\partial y}\mathrm{w}_3 d\Omega = 0 \tag{3.65}$$

$$\int_\Omega \mathrm{v}_y \mathrm{w}_3 + \int_\Omega \frac{\partial\psi}{\partial x}\mathrm{w}_3 d\Omega = 0 \tag{3.66}$$

Para cada intervalo no espaço o valor das funções será interpolado usando um polinômio de grau 1.

$$\omega(x,t) \approx \sum_{i=1}^{np}\omega_i(t)N_i(x) \tag{3.67}$$

$$\psi(x,t) \approx \sum_{i=1}^{np}\psi_i(t)N_i(x) \tag{3.68}$$

$$\mathrm{v}_x(x,t) \approx \sum_{i=1}^{np} \mathrm{v}_{(x,i)}(t) N_i(x) \tag{3.69}$$

$$\mathrm{v}_x(x,t) \approx \sum_{i=1}^{np} \mathrm{v}_{(y,i)}(t) N_i(x) \tag{3.70}$$

O mesmo vale para nossas funções peso.

$$\mathrm{w}_1(x,t) \approx \sum_{j=1}^{np} \mathrm{w}_{(1,j)}(t) N_j(x) \tag{3.71}$$

$$\mathrm{w}_2(x,t) \approx \sum_{j=1}^{np} \mathrm{w}_{(2,j)}(t) N_j(x) \tag{3.72}$$

$$\mathrm{w}_3(x,t) \approx \sum_{j=1}^{np} \mathrm{w}_{(3,j)}(t) N_j(x) \tag{3.73}$$

Os conjuntos das funções base podem ser definidos pela seguites relações.

$$\mathbb{W} = \{\omega \in \Omega \to \mathbb{R}^2 : \int_\Omega \omega^2 d\Omega; \omega = \omega_\Gamma\} \tag{3.74}$$

$$\mathbb{P} = \{\psi \in \Omega \to \mathbb{R}^2 : \int_\Omega \psi^2 d\Omega; \psi = \psi_\Gamma\} \tag{3.75}$$

$$\mathbb{V} = \{\mathrm{v} \in \Omega \to \mathbb{R}^2 : \int_\Omega \mathrm{v}^2 d\Omega; \mathrm{v} = \mathrm{v}_\Gamma\} \tag{3.76}$$

Os epaços da função peso podem ser deifinidos pelas seguintes expressões.

$$\mathbb{W}_1 = \{\mathrm{w}_1 \in \Omega \to \mathbb{R}^2 : \int_\Omega \mathrm{w}_1^2 d\Omega < \infty; \mathrm{w}_{1,\Gamma} = 0\} \tag{3.77}$$

$$\mathbb{W}_2 = \{\mathrm{w}_2 \in \Omega \to \mathbb{R}^2 : \int_\Omega \mathrm{w}_2^2 d\Omega < \infty; \mathrm{w}_{2,\Gamma} = 0\} \tag{3.78}$$

$$\mathbb{W}_3 = \{\mathrm{w}_3 \in \Omega \to \mathbb{R}^2 : \int_\Omega \mathrm{w}_3^2 d\Omega < \infty; \mathrm{w}_{3,\Gamma} = 0\} \tag{3.79}$$

Usando as expressões acima nas equações (3.63), (3.64), 3.65 e (3.66).

$$\begin{aligned}
&\int_\Omega \sum_{i=1}^{np} \dot{\omega}_i N_i \sum_{j=1}^{np} \mathrm{w}_{(1,j)} N_j d\Omega \\
&\quad + \mathrm{v}_x \int_\Omega \sum_{i=1}^{np} \frac{\partial \omega_i N_i}{\partial x} \sum_{j=1}^{np} \mathrm{w}_{(1,j)} N_j d\Omega + \mathrm{v}_y \int_\Omega \sum_{i=1}^{np} \frac{\partial \omega_i N_i}{\partial y} \sum_{j=1}^{np} \mathrm{w}_{(1,j)} N_j d\Omega \\
&\quad + v \int_\Omega \left(\sum_{i=1}^{np} \frac{\partial \omega_i N_i}{\partial x} \sum_{j=1}^{np} \frac{\mathrm{w}_{(1,j)} N_j}{\partial x} + \sum_{i=1}^{np} \frac{\partial \omega_i N_i}{\partial y} \sum_{j=1}^{np} \frac{\mathrm{w}_{(1,j)} N_j}{\partial y} \right) d\Omega \\
&\quad + \frac{\Delta t}{2} \int_\Omega \mathrm{v}_x \sum_{j=1}^{np} \frac{\mathrm{w}_{(1,j)} N_j}{\partial x} \left(\mathrm{v}_x \sum_{i=1}^{np} \frac{\partial \omega_i N_i}{\partial x} + \mathrm{v}_y \sum_{i=1}^{np} \frac{\partial \omega_i N_i}{\partial y} \right) d\Omega \\
&\quad + \frac{\Delta t}{2} \int_\Omega \mathrm{v}_y \sum_{j=1}^{np} \frac{\mathrm{w}_{(1,j)} N_j}{\partial y} \left(\mathrm{v}_x \sum_{i=1}^{np} \frac{\partial \omega_i N_i}{\partial x} + \mathrm{v}_y \sum_{i=1}^{np} \frac{\partial \omega_i N_i}{\partial y} \right) d\Omega = 0
\end{aligned} \tag{3.80}$$

$$\begin{aligned}
&- \int_\Omega \left(\sum_{i=1}^{np} \frac{\partial \psi_i N_i}{\partial x} \sum_{j=1}^{np} \frac{\partial \mathrm{w}_{(2,j)} N_j}{\partial x} + \sum_{i=1}^{np} \frac{\partial \psi_i N_i}{\partial y} \sum_{j=1}^{np} \frac{\partial \mathrm{w}_{(2,j)} N_j}{\partial y} \right) d\Omega \\
&\qquad + \int_\Omega \sum_{i=1}^{np} \omega_i N_i \sum_{j=1}^{np} \mathrm{w}_{(2,j)} N_j d\Omega = 0
\end{aligned} \tag{3.81}$$

$$\int_\Omega \sum_{i=1}^{np} \mathrm{v}_{(x,i)} N_i \sum_{j=1}^{np} \mathrm{w}_{(3,j)} N_j d\Omega - \int_\Omega \sum_{i=1}^{np} \frac{\partial \psi_i N_i}{\partial y} \sum_{j=1}^{np} \mathrm{w}_{(3,j)} N_j d\Omega = 0 \tag{3.82}$$

$$\int_\Omega \sum_{i=1}^{np} \mathrm{v}_{(y,i)} N_i \sum_{j=1}^{np} \mathrm{w}_{(3,j)} N_j d\Omega + \int_\Omega \sum_{i=1}^{np} \frac{\partial \psi_i N_i}{\partial x} \sum_{j=1}^{np} \mathrm{w}_{(3,j)} N_j d\Omega = 0 \tag{3.83}$$

Simplificando um pouco.

$$\begin{aligned}
&\sum_{j=1}^{np} \mathrm{w}_{(1,j)} \left\{ \sum_{i=1}^{np} \dot{\omega}_i \int_\Omega N_i N_j d\Omega + \sum_{i=1}^{np} \omega_i \left[\mathrm{v}_x \int_\Omega \frac{\partial N_i}{\partial x} N_j d\Omega + \mathrm{v}_y \int_\Omega \frac{\partial N_i}{\partial y} N_j d\Omega \right.\right. \\
&\qquad + v \int_\Omega \left(\frac{\partial N_i}{\partial x} \frac{N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{N_j}{\partial y} \right) d\Omega \\
&\qquad + \frac{\Delta t}{2} \int_\Omega \mathrm{v}_x \frac{N_j}{\partial x} \left(\mathrm{v}_x \frac{\partial N_i}{\partial x} + \mathrm{v}_y \frac{\partial N_i}{\partial y} \right) d\Omega \\
&\qquad \left.\left. + \frac{\Delta t}{2} \int_\Omega \mathrm{v}_y \frac{N_j}{\partial y} \left(\mathrm{v}_x \frac{\partial N_i}{\partial x} + \mathrm{v}_y \frac{\partial N_i}{\partial y} \right) d\Omega \right] \right\} = 0
\end{aligned} \tag{3.84}$$

$$\sum_{j=1}^{np} \mathrm{w}_{(2,j)} \left\{ \sum_{i=1}^{np} \psi_i \left[- \int_\Omega \left(\frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) d\Omega \right] + \sum_{i=1}^{np} \omega_i \left[\int_\Omega N_i N_j d\Omega \right] \right\} = 0 \tag{3.85}$$

$$\sum_{j=1}^{np} \mathrm{w}_{(3,j)} \left[\sum_{i=1}^{np} \mathrm{v}_{(x,i)} \left(\int_\Omega N_i N_j d\Omega \right) - \sum_{i=1}^{np} \psi_i \left(\int_\Omega \frac{\partial N_i}{\partial y} N_j d\Omega \right) \right] = 0 \tag{3.86}$$

$$\sum_{j=1}^{np} \mathrm{w}_{(3,j)} \left[\sum_{i=1}^{np} \mathrm{v}_{(y,i)} \left(\int_\Omega N_i N_j d\Omega \right) + \sum_{i=1}^{np} \psi_i \left(\int_\Omega \frac{\partial N_i}{\partial x} N_j d\Omega \right) \right] = 0 \tag{3.87}$$

Como $\mathrm{w}_{(1,j)} \neq 0$, $\mathrm{w}_{(2,j)} \neq 0$ and $\mathrm{w}_{(3,j)} \neq 0$

$$\begin{aligned} \sum_{j=1}^{np} \sum_{i=1}^{np} \dot{\omega}_i \int_\Omega N_i N_j d\Omega + \sum_{j=1}^{np} \sum_{i=1}^{np} \omega_i \Bigg[& \mathrm{v}_x \int_\Omega \frac{\partial N_i}{\partial x} N_j d\Omega + \mathrm{v}_y \int_\Omega \frac{\partial N_i}{\partial y} N_j d\Omega \\ & v \int_\Omega \left(\frac{\partial N_i}{\partial x} \frac{N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{N_j}{\partial y} \right) d\Omega \\ & + \frac{\Delta t}{2} \int_\Omega \mathrm{v}_x \left(\mathrm{v}_x \frac{N_j}{\partial x} \frac{\partial N_i}{\partial x} + \mathrm{v}_y \frac{N_j}{\partial x} \frac{\partial N_i}{\partial y} \right) d\Omega \\ & + \frac{\Delta t}{2} \int_\Omega \mathrm{v}_y \left(\mathrm{v}_x \frac{N_j}{\partial y} \frac{\partial N_i}{\partial x} + \mathrm{v}_y \frac{N_j}{\partial y} \frac{\partial N_i}{\partial y} \right) d\Omega \Bigg] = 0 \end{aligned} \tag{3.88}$$

$$\sum_{j=1}^{np} \sum_{i=1}^{np} \psi_i \left[- \int_\Omega \left(\frac{\partial N_i}{\partial x} \frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y} \frac{\partial N_j}{\partial y} \right) d\Omega \right] + \sum_{j=1}^{np} \sum_{i=1}^{np} \omega_i \left[\int_\Omega N_i N_j d\Omega \right] = 0 \tag{3.89}$$

$$\sum_{j=1}^{np} \sum_{i=1}^{np} \mathrm{v}_{(x,i)} \left(\int_\Omega N_i N_j d\Omega \right) - \sum_{j=1}^{np} \sum_{i=1}^{np} \psi_i \left(\int_\Omega \frac{\partial N_i}{\partial y} N_j d\Omega \right) = 0 \tag{3.90}$$

$$\sum_{j=1}^{np} \sum_{i=1}^{np} \mathrm{v}_{(y,i)} \left(\int_\Omega N_i N_j d\Omega \right) + \sum_{j=1}^{np} \sum_{i=1}^{np} \psi_i \left(\int_\Omega \frac{\partial N_i}{\partial x} N_j d\Omega \right) = 0 \tag{3.91}$$

Usando a forma matricial para reescrever as equações (3.88,3.89, 3.90, 3.91)

$$\begin{aligned} \mathbf{M}\dot{\omega} + \mathrm{v}_x \cdot \mathbf{G}_x \omega + \mathrm{v}_y \cdot \mathbf{G}_y \omega + v \Big(\mathbf{K}_{xx} + \mathbf{K}_{yy} \Big) \omega \\ + \frac{\Delta t}{2} \mathrm{v}_x \Big(\mathrm{v}_x \mathbf{K}_{xx} + \mathrm{v}_y \mathbf{K}_{xy} \Big) \omega + \frac{\Delta t}{2} \mathrm{v}_y \Big(\mathrm{v}_x \mathbf{K}_{yx} + \mathrm{v}_y \mathbf{K}_{yy} \Big) \omega = 0 \end{aligned} \tag{3.92}$$

$$-\Big(\mathbf{K}_{xx}+\mathbf{K}_{yy}\Big)\psi+\mathbf{M}\omega=0 \tag{3.93}$$

$$\mathbf{M}\mathrm{v}_x-\mathbf{G}_y\psi=0 \tag{3.94}$$

$$\mathbf{M}\mathrm{v}_y+\mathbf{G}_x\psi=0 \tag{3.95}$$

As matrizes globais $\mathbf{M}$, $\mathbf{G}_x$, $\mathbf{G}_y$, $\mathbf{K}_{xx}$, $\mathbf{K}_{yy}$, $\mathbf{K}_{xy}$ e $\mathbf{K}_{yx}$ podem ser construídas a partir das matrizes de cada elemento sendo elas:

$$m^e=\int_{\Omega^e}N_i^eN_j^ed\Omega^e \tag{3.96}$$

$$g_x^e=\int_{\Omega^e}\frac{\partial N_i^e}{\partial x}N_j^ed\Omega^e \tag{3.97}$$

$$g_y^e=\int_{\Omega^e}\frac{\partial N_i^e}{\partial y}N_j^ed\Omega^e \tag{3.98}$$

$$k_{xx}^e=\int_{\Omega^e}\frac{\partial N_i^e}{\partial x}\frac{\partial N_j^e}{\partial x}d\Omega^e \tag{3.99}$$

$$k_{xy}^e=\int_{\Omega^e}\frac{\partial N_i^e}{\partial x}\frac{\partial N_j^e}{\partial y}d\Omega^e \tag{3.100}$$

$$k_{yy}^e=\int_{\Omega^e}\frac{\partial N_i^e}{\partial y}\frac{\partial N_j^e}{\partial y}d\Omega^e \tag{3.101}$$

$$k_{yx}^e=\int_{\Omega^e}\frac{\partial N_i^e}{\partial y}\frac{\partial N_j^e}{\partial x}d\Omega^e \tag{3.102}$$

Para elaborar as matrizes globais, um mesmo operador pode ser utilizado, já que os elementos não foram alterados. Na seção (4.3) existe uma explicação de como o

operador $\mathbf{C}$ atua nas matrizes locais.

$$\begin{pmatrix} \mathbf{M} \\ \mathbf{G}_x \\ \mathbf{G}_y \\ \mathbf{K}_{xx} \\ \mathbf{K}_{yy} \\ \mathbf{K}_{xy} \\ \mathbf{K}_{yx} \end{pmatrix} = \mathbf{C} \begin{pmatrix} m^e \\ g_x^e \\ g_y^e \\ k_{xx}^e \\ k_{yy}^e \\ k_{xy}^e \\ k_{yx}^e \end{pmatrix} \tag{3.103}$$

3.2.4 Elementos de malha

Para continuar resolvendo nosso problema, devemos dar prosseguimento a discretização do nosso espaço Ω. Para isso podemos utilizar formas geométricas planares. Elas podem ser estruturadas e não-estruturadas. Sua escolha influencia na estabilidade e precisão do modelo. Como nossa formulação foge do forte acoplamento entre os campos de pressão e velocidade, ela satisfaz o critério de Ladyzhenskaya–Babuška–Brezzi (**LBB**) [9–11], sendo assim, a utilização de uma malha não-estruturada é possível. Em posse dessas informações, utilizaremos o elemento triangular de primeira ordem para nosso domínio.

$$\mathbf{N}(x, y) = [N_i \ N_j \ N_k] \tag{3.104}$$

$$\begin{aligned} N_i &= a_i + b_i x + c_i y \\ N_j &= a_j + b_j x + c_j y \\ N_k &= a_k + b_k x + c_k y \end{aligned} \tag{3.105}$$

Podemos definir os coeficientes por:

$$\begin{aligned} a_i &= x_j y_k - x_k y_j; \ b_i = y_j - y_k; \ c_i = x_k - x_j \\ a_j &= x_k y_i - x_i y_k; \ b_j = y_k - y_i; \ c_j = x_i - x_k \\ a_k &= x_i y_j - x_j y_i; \ b_k = y_i - y_j; \ c_j = x_j - x_i \end{aligned} \tag{3.106}$$

Partindo da equação 3.96 e usando a forma matricial das funções de forma obtemos.

$$m^e = \int_{\Omega^e} N_i^e N_j^e \ d\Omega^e = \int_{\Omega^e} \mathbf{N}^T \mathbf{N} \ d\Omega^e \tag{3.107}$$

Considerando, a definição de coordenadas independentes e de coordenadas triangulares, *Labedev* [12], em função de coordenadas cartesianas obtemos.

$$\int_{\Omega^e} \mathbf{N}^T\mathbf{N}\; d\Omega^e = \int_0^1 \int_0^{1-\varepsilon} \begin{bmatrix} \varepsilon^2 & \varepsilon\eta & \varepsilon(1-\varepsilon-\eta) \\ \varepsilon\eta & \eta^2 & \eta(1-\varepsilon-\eta) \\ \varepsilon(1-\varepsilon-\eta) & \eta(1-\varepsilon-\eta) & (1-\varepsilon-\eta)^2 \end{bmatrix} \left\| \begin{matrix} \frac{\partial\varepsilon}{\partial x} & \frac{\partial\eta}{\partial x} \\ \frac{\partial\varepsilon}{\partial y} & \frac{\partial\eta}{\partial y} \end{matrix} \right\| d\eta d\varepsilon \tag{3.108}$$

Calculando o jacobiano da transposição de coordenadas e resolvendo a integral dupla para todos os termos da matriz.

$$\left\| \begin{matrix} \frac{\partial\varepsilon}{\partial x} & \frac{\partial\eta}{\partial x} \\ \frac{\partial\varepsilon}{\partial y} & \frac{\partial\eta}{\partial y} \end{matrix} \right\| = 2A \tag{3.109}$$

$$\int_0^1 \int_0^{1-\varepsilon} \varepsilon^2 d\eta d\varepsilon = \int_0^1 \varepsilon^2(1-\varepsilon) d\varepsilon = \left(\frac{\varepsilon^3}{3} - \frac{\varepsilon^4}{4}\right)\Big|_0^1 = \frac{1}{12} \tag{3.110}$$

$$\int_0^1 \int_0^{1-\varepsilon} \eta^2 d\eta d\varepsilon = \frac{1}{12} \tag{3.111}$$

$$\int_0^1 \int_0^{1-\varepsilon} \varepsilon\eta\; d\eta d\varepsilon = \int_0^1 \varepsilon \frac{(1-\varepsilon)^2}{2}\; d\varepsilon = \frac{1}{2}\left(\frac{\varepsilon^2}{2} - \frac{2\varepsilon^3}{3} + \frac{\varepsilon^4}{4}\right)\Big|_0^1 = \frac{1}{24} \tag{3.112}$$

$$\int_0^1 \int_0^{1-\varepsilon} \varepsilon(1-\varepsilon-\eta) d\eta d\varepsilon = \int_0^1 \varepsilon\left(1-\varepsilon-\varepsilon(1-\varepsilon) - \frac{(1-\varepsilon)^2}{2}\right) d\varepsilon = \left(\frac{\varepsilon^2}{4} - \frac{\varepsilon^3}{3} + \frac{\varepsilon^4}{8}\right)\Big|_0^1 = \frac{1}{24} \tag{3.113}$$

$$\int_0^1 \int_0^{1-\varepsilon} \eta(1-\varepsilon-\eta) d\eta d\varepsilon = \int_0^1 \frac{(1-\varepsilon)^3}{6} d\varepsilon = \left(-\frac{(1-\varepsilon)^4}{24}\right)\Big|_0^1 = \frac{1}{24} \tag{3.114}$$

$$\int_0^1 \int_0^{1-\varepsilon} (1-\varepsilon-\eta)^2 d\eta d\varepsilon = \frac{1}{12} \tag{3.115}$$

$$m^e = \int_{\Omega^e} \mathbf{N}^T\mathbf{N}\; d\Omega^e = \frac{A}{12} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \tag{3.116}$$

O mesmo raciocínio pode ser usado para as outras matrizes locais, vamos aqui omitir o desenvolvimento.

$$g_x^e = \int_{\Omega^e} \frac{\partial N_i^e}{\partial x} N_j^e d\Omega^e = \int_{\Omega^e} \frac{\partial \mathbf{N}}{\partial x}^T \mathbf{N} \; d\Omega^e = \frac{1}{6} \begin{bmatrix} b_i & b_j & b_k \\ b_i & b_j & b_k \\ b_i & b_j & b_k \end{bmatrix} \tag{3.117}$$

$$g_y^e = \int_{\Omega^e} \frac{\partial N_i^e}{\partial y} N_j^e d\Omega^e = \int_{\Omega^e} \frac{\partial \mathbf{N}}{\partial y}^T \mathbf{N} \; d\Omega^e = \frac{1}{6} \begin{bmatrix} c_i & c_j & c_k \\ c_i & c_j & c_k \\ c_i & c_j & c_k \end{bmatrix} \tag{3.118}$$

$$k_{xx}^e = \int_{\Omega^e} \frac{\partial N_i^e}{\partial x} \frac{\partial N_j^e}{\partial x} \; d\Omega^e = \int_{\Omega^e} \frac{\partial \mathbf{N}^T}{\partial x} \frac{\partial \mathbf{N}}{\partial x} \; d\Omega^e = \frac{1}{4A} \begin{bmatrix} b_i^2 & b_i b_j & b_i b_k \\ b_j b_i & b_j^2 & b_j b_k \\ b_k b_i & b_j b_k & b_k^2 \end{bmatrix} \tag{3.119}$$

$$k_{xy}^e = \int_{\Omega^e} \frac{\partial N_i^e}{\partial x} \frac{\partial N_j^e}{\partial x} \; d\Omega^e = \int_{\Omega^e} \frac{\partial \mathbf{N}^T}{\partial x} \frac{\partial \mathbf{N}}{\partial y} \; d\Omega^e = \frac{1}{4A} \begin{bmatrix} b_i c_i & b_i c_j & b_i c_k \\ b_j c_i & b_j c_j & b_j c_k \\ b_k c_i & b_j c_k & b_k c_k \end{bmatrix} \tag{3.120}$$

$$k_{yy}^e = \int_{\Omega^e} \frac{\partial N_i^e}{\partial y} \frac{\partial N_j^e}{\partial y} \; d\Omega^e = \int_{\Omega^e} \frac{\partial \mathbf{N}^T}{\partial y} \frac{\partial \mathbf{N}}{\partial y} \; d\Omega^e = \frac{1}{4A} \begin{bmatrix} c_i^2 & c_i c_j & c_i c_k \\ c_j c_i & c_j^2 & c_j c_k \\ c_k c_i & c_j c_k & c_k^2 \end{bmatrix} \tag{3.121}$$

3.3 Matrizes globais

Para fazer a montagem das matrizes globais precisamos colocar os valores das matrizes locais triangulares no lugar. A matriz global e uma matriz $n \times n$, sendo n o número total de nós do espaço discretizado. No código IEN é uma matriz $nx3$, composta por todos os elementos triangulares, Em cada linha estão os identificadores de cada nó.

$$\begin{bmatrix} & \vdots & \\ 7 & 8 & 9 \\ 10 & 11 & 12 \\ & \vdots & \end{bmatrix} \tag{3.122}$$

Sabendo que cada elemento da matriz local $a_{(i,j)} \rightarrow A_{(IEN[k,i],IEN[k,j])}$ sendo a a matriz local e A a matriz global, podemos implementar isso no nosso código.

3.4 Forças aerodinâmicas

Para calcular as forças de arrasto e sustentação utilizaremos as transformações derivadas do momento (DMT) que deriva do inglês*"derivative-moment transformations"*. Assim como em *Wu et al.* [13], vamos utilizar três expressões para as forças tuando no corpo. São elas:

$$\mathbf{F_V} = -\frac{\mu}{k}\int_{V_f} \mathbf{P} \times \nabla^2 \omega dV \tag{3.123}$$

$$\mathbf{F}_B = \rho \int_{\partial B} \mathbf{P} \times \left(\mathbf{n}_{(\partial B)} \times \frac{\partial^2 \mathbf{P}}{\partial t^2}\right) dS \tag{3.124}$$

$$\mathbf{F}_\Gamma = -\mu \int_\Gamma \mathbf{P} \times (\mathbf{n}_\Gamma \times (\nabla \times \omega)) dS \tag{3.125}$$

Sendo V_f o domínio com dimensões $n = 2, 3$ ao redor de um corpo B com fronteira delimitado por Γ. Sendo $k = n - 1$.

$$\mathbf{F} = \mathbf{F}_{V_f} + \mathbf{F}_B + \mathbf{F}_\Gamma \tag{3.126}$$

A força resultante sobre o corpo pode ser obtida somando às três expressões. Expandido os componentes e utilizando a dimensão do problema, $n = 2$ e separando as forças nas direções das componentes x e y obtemos:

$$-\mu \int_{V_f} \mathbf{P} \times \nabla^2 \omega dV_f = \int_{V_f} (x_i \nabla^2 \omega \cdot \hat{j} - y_i \nabla^2 \omega \cdot \hat{i}) dV_f \tag{3.127}$$

$$\rho \int_{\partial B} \mathbf{P} \times \left(\mathbf{n}_{(\partial B)} \times \ddot{\mathbf{P}}\right) dS = \rho \int_{\partial B} y_i(\ddot{y}_i - \ddot{x}_i)\hat{i} - x_i(\ddot{y}_i - \ddot{x}_i)\hat{j} ds \tag{3.128}$$

$$\mathbf{F}_\Gamma = \mu \int_\Gamma y_i \left(\frac{\partial \omega}{\partial x} + \frac{\partial \omega}{\partial y}\right)\hat{i} - x_i \left(\frac{\partial \omega}{\partial x} + \frac{\partial \omega}{\partial y}\right)\hat{j} dS \tag{3.129}$$

Expandido os componentes e utilizando a dimensão do problema, $n = 2$ e separando as forças nas direções das componentes i e j, sendo F_L força de sustentação e F_D a forças de arrasto.

$$\mathbf{F}_L = -\mu \int_{V_f} y\nabla^2\omega dV_f + \rho \int_{\partial B} y(\ddot{y} - \ddot{x})dS + \mu \int_{\Gamma} y\left(\frac{\partial\omega}{\partial x} + \frac{\partial\omega}{\partial y}\right)dS \tag{3.130}$$

$$\mathbf{F}_D = \mu \int_{V_f} x\nabla^2\omega dV_f - \rho \int_{\partial B} x(\ddot{y} - \ddot{x})dS - \mu \int_{\Gamma} x\left(\frac{\partial\omega}{\partial x} + \frac{\partial\omega}{\partial y}\right)dS \tag{3.131}$$

De acordo com *Wu et al.* [13], as integrais em B e Γ rapidamente tendem a zero conforme a superficie V_f, de integração, aumenta, isso é válido para numeros de Reynolds superiores a 300. Sendo assim as expressões se reduzem para:

$$\mathbf{F}_L = -\mu \int_{V_f} y\nabla^2\omega dV_f \tag{3.132}$$

$$\mathbf{F}_D = \mu \int_{V_f} x\nabla^2\omega dV_f \tag{3.133}$$

Para comparar os valores obtidos na simulação com aqueles na literatura é necessário obter os coeficientes, tanto de arrasto como de sustentação. Como estamos trabalhando em um caso bidimensional o coeficiente será obtido em função de uma unidade de comprimento l, na direção perpendicular ao problema e D_c o diâmetro do circulo.

$$C_D = \frac{F_D}{\frac{1}{2}\rho U_\infty^2 D_c} \tag{3.134}$$

$$C_L = \frac{F_L}{\frac{1}{2}\rho U_\infty^2 D_c} \tag{3.135}$$

Utilizando a formulação de elementos finitos desenvolvida anteriormente, podemos afirmar que as integrais podem ser reescritas da seguinte forma.

$$\mathbf{F}_L = -\mu y(\mathbf{K}_{xx} + \mathbf{K}_{yy})\omega^T \tag{3.136}$$

$$\mathbf{F}_D = \mu x(\mathbf{K}_{xx} + \mathbf{K}_{yy})\omega^T \tag{3.137}$$

Capítulo 4

Métodos Computacionais

Uma vez tendo feito a discretização das equações de governo, precisamos encontrar um modo em que a máquina a ser utilizada possa fazer os calculos automáticamente. Para isso vamos fazer uma programa utilizando a linguagem python, com alguns módulos extras adcionais, para importar e simplificar os cálculos.

Vamos justificar detalhadamente o uso de cada ferramenta, e caso seja de interesse, os codigos podem ser consultados nos anexos (citar).

A primeira biblioteca a ser utilizada será a API do Gmsh, *Geuzaine* [14] (versão 4.9.5), com python. A API é a maneira como um programa interage com outro programa, sem a necessidade de traduzir todo o código para a linguagem utilizada. Com ela podemos automatizar o processo de criação das malhas, não sendo necessário recorer a interface gráfica do programa para desenhar as geometrias. Além de simplificar a etapa, ganhamos a facilidade de mudar rapidamente os parâmetros da malha.

O módulo para computação cientifica chamado *Scipy* possui diversas funções específicas para computação científica. Deste extrairemos a parte de matrizes esparsas, o método de resolução para solução de sistemas lineares, sendo inlcuso o précondicionador. Falaremos dele mais a frente.

Outra bilbioteca é a que possibilita que possamos fazer calculos utilizando todos os núcleos do processador disponíveis na máquina. O módulo *Multiprocessing* possibilita que o usuário tenha essa possibilidade.

O ultimo módulo importate é o módulo *numpy*. Ele otimiza o cálculo de operações com vetores, além de conter diversas ferramentes para analise de dados.

Uma delas sendo a trasformada discreta de fourier.

4.1 Gmsh API

Com o API do Gmsh não dependemos mais da parte gráfica, podendo parametrizar ela por completo. Isso é muito ultil, uma vez que o operador pode programar diversos casos de geometris e integrar-los com o algrotimo de solução, sendo apenas necessário fazer uma verificação simples para checar se os ressultados estão saindo como esperado. Uma vez feito isso, podemos deixar as simulações e nos preocupar com outras partes do problema.

A parte mais complexa nesse processo é a aprendizagem de como usar os camandos específicos para desenhar os objetos. Pode se dizer que a parte gráfica é muito mais intutiva, uma vez que não envolve nenhum conhecimento prévio com programação. Sendo assim, se apenas uma única geometria será usada no problema e a malha for de tamanho constante para todos casos, é mais conveniente utilizar apenas a parte gráfica. Em qualquer outro caso é mais conveniente usar a API com o python.

Uma pequena explicação de como foi feita a geometria para (caso específico) está contida no anexo (citar).

4.2 Scipy

Essa biblioteca contém divesas funções para métodos matemáticos, algumas já estão otimizadas com outro módulo do python, o *Cython*, é uma bilblioteca que faz possivel utilizar a linguagem C, para otimizar o tempo de computo para diversas operações.

Deste módulo vamos utilizar a parte de matrizes esparsas. Elas são conveninetes para o tipo de sistema linear que vamos enfrentar. Como veremos mais a frente na seção (citar), as matrizes para resolução do problema contém muitos zeros, ou seja a maior parte dos elementos contém uma informação igual, sendo possivel assim reduzir-la a apenas ao valores que são são nulos, caso fosse utilizada a matriz cheia, o computador usado não seria capaz de armazenar na memória RAM, impossibilitando a resolução rápida do problema. Nesse módulo também estão contidas as operações de algebra linear para trabalhar com esse tipo de matriz.

4.3 Módulo para computação paralela

Para que se possa utilizar mais de um processador com o python, precisamos importar a bilbioteca *multiprocessing*. Com ela podemos organizar nosso programa de maneira a executar tarefas em conjunto.

Umas das primeiras ideias para otimizar o programa seria a otimização da montagem das matrizes globais. Como visto em (3.92) e (3.103), a montagem das matrizes globais somente necessita dos parâmetros iniciais da simulação, porém a matriz de estabilização demanda sua montagem a cada iteração, pois depende da velocidade obtidas nos pontos da malha.

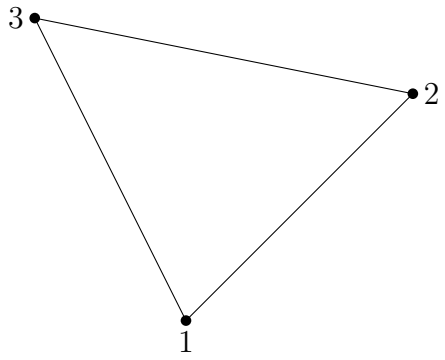


Figura 4.1: Representação fantasia de um elemento triangular da malha, com numeração local

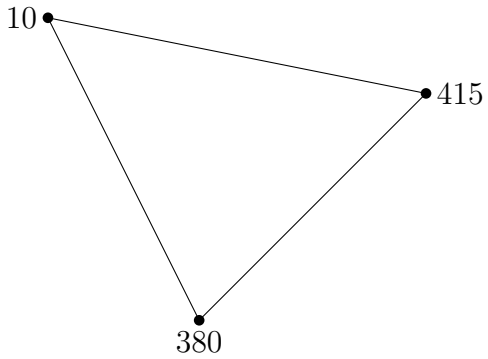


Figura 4.2: Representação fantasia de um elemento triangular da malha, com numeração global

Como dito anteriormente, temos um vetor chamado *IEN*, que armazena todas as informações sobre como os pontos estão ligados globalmente. O vetor *IEN* possui dimesões,tal que a linha representa um elemento, para nosso caso ele é tringular,

portanto deve conter três elementos na linha, logo a matriz tem dimesões , (elementos totais $\times$ 3). Usando a matriz massa local (3.116) para exemplificar.

$$m^e = \frac{A}{12}\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$$

Os elementos da matriz local representam como os pontos interagem um com outro. Usando as figuras (4.1) e (4.2) para ilustrar como o processo de montagem. O ponto 1 se transforma no ponto 380, o ponto 2 se transforma no ponto 415 e o ponto 3 se transforma no ponto 10. O elemento da matriz local ($m_{1,1}$) será adicionado ao elemento da matriz global ($M_{380,380}$).

Sendo assim, como explicado, para completar toda a matriz global seria necessário percorrer todo o vetor (IEN) e fazer as operações de soma. Para otimizar esse processo foi decidido separar esse vetor pelo número de processadores disponíveis e obter "Matrizes globais".

Devido a que o modulo de *multiprocessing* não oferece uma maneira de trabalhar com matrizes esparsas com endereço de memória compartilhada, de maneira nativa, foi decidido criar matrizes globais parciais, para cada processador. Isso encarece um pouco o algoritmo visto que é necessário fazer outra operação de soma após percorrer o vetor (IEN).

Nem sempre com a implementação da computação em paralelo podemos observar a melhora no desempenho do código, mas no nosso caso se observa uma melhora considerável.

Como podemos observar na figura 4.3, temos uma diminuição do tempo de montagem das matrizes é da ordem de $2/n$, para quatro processadores, sendo n o número de processadores utilizados para a paralelização. Como as operações matemáticas para a montagem das matrizes, são simples, poderíamos utilizar núcleos de GPU, ao invés dos núcleos do processador. Para este caso, seria necessário utilizar o endereço de memória compartilhado.

Para mais informações sobre o código implementação do código um exemplo pode ser contemplado no Apêndice C.2.

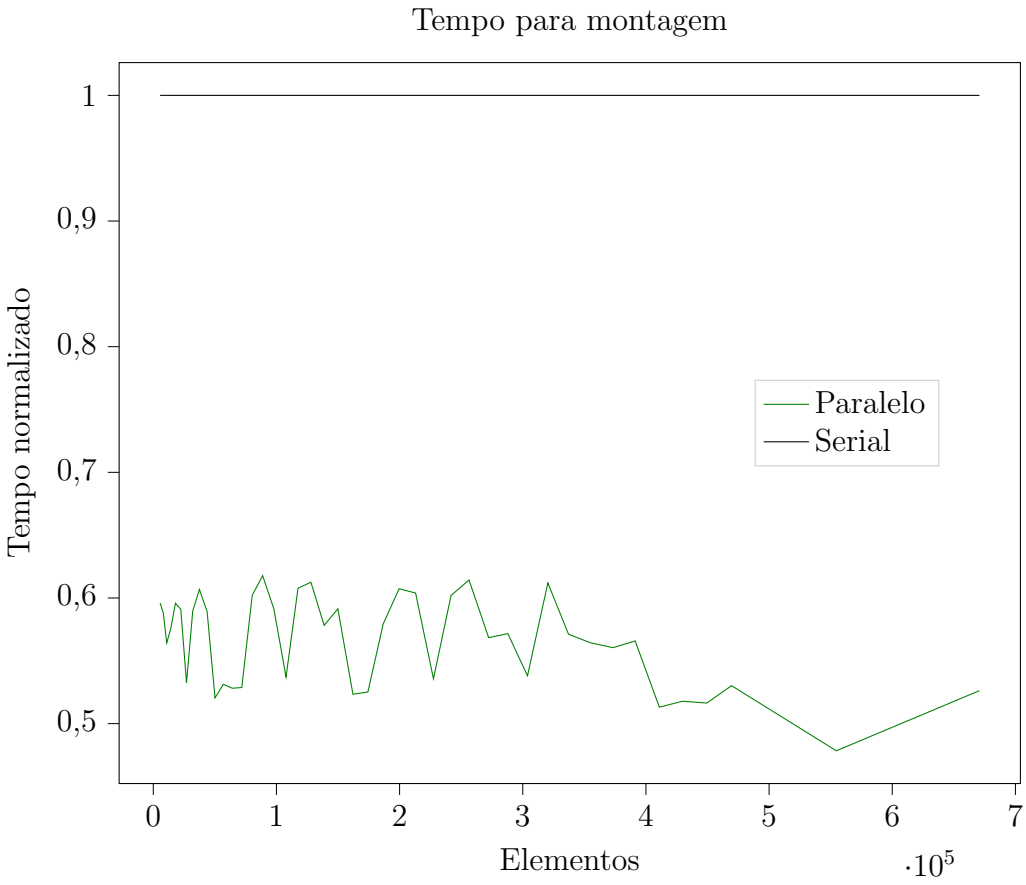


Figura 4.3: Tempo de uso da CPU normalizado

Capítulo 5

Validações

5.1 Escoamento Poiseuille

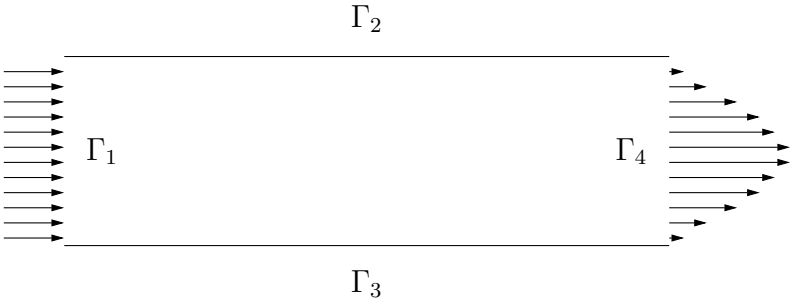


Figura 5.1: Especificação dos contornos para o problema dos escoamento plano de Poiseuille

O escoamento de Poiseuille é um dos primeiro casos que podem ser usados para validar o código numérico. Consiste no escoamento de um fluido confinado por uma fronteira, cuja velocidade na interface é zero. Para o caso 2D consideramos esse fluxo como escoamento de plano de Poiseuille.

Para o valor de $Re = 1$, esse escoamento tem solução analítica conhecida, sendo assim perfeito para verificar se o código tem o comportamento esperado.

5.1.1 Condições de contorno

Vamos usar os contornos da figura (5.1) para impor as condições necessárias para execução do programa.

$$\begin{cases} U(x,y) = U_\infty \mid x \; \varepsilon \; \Gamma_1 \\ U(x,y) = 0 \mid (x,y) \; \varepsilon \; (\Gamma_2, \Gamma 4) \\ \psi(x,y) = f(y)\big|_0^1 \mid (x,y) \; \varepsilon \; \Gamma_1 \\ \psi(x,y) = 1 \mid (x,y) \; \varepsilon \; \Gamma_2 \\ \psi(x,y) = 0 \mid (x,y) \; \varepsilon \; \Gamma_3 \\ \nabla\psi(x,y) \cdot \mathbf{n} = 0 \mid (x,y) \; \varepsilon \; \Gamma_4 \\ \mathbf{U}_\infty = 1 \; \hat{i} + 0 \; \hat{j} \end{cases} \tag{5.1}$$

Como estamos trabalhando com as equações de maneira dimensional, precisamos também garantir que as dimensões utilizadas sejam compativeis para que o escomento tenha o $Re = 1$. Para isso o comprimento de Γ_1 deve ser de $1m$ e a velocidade $U_inf = 1m/s$, com a viscosidade cinemática sendo igual a $1m^2/s$.

5.1.2 Resultados

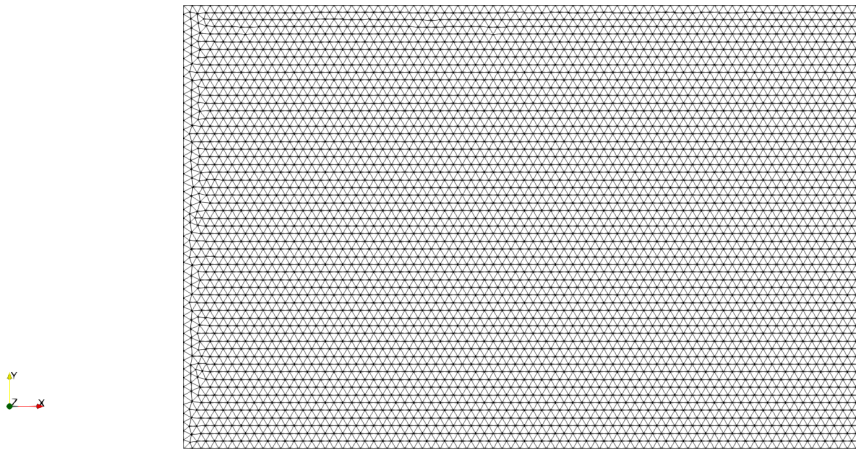


Figura 5.2: Detalhe da malha utilizada no escoamento plano de Poiseuille

Usando uma malha com 14.814 pontos e 29.026 elementos, pode ser vista na figura (5.2), obtivemos o gráfico da figura (5.3). Para o critério de parada foi utilizado

o erro entre as velocidades das iterações $N+1$ e N sobre o valor em módulo de U_∞, representado pela equação (5.2), sendo $[V]$ a velocidade para cada ponto.

$$\frac{[V]_{(N+1)} - [V]_{(N)}}{|U_\infty|} \leqslant 10^{-5} \tag{5.2}$$

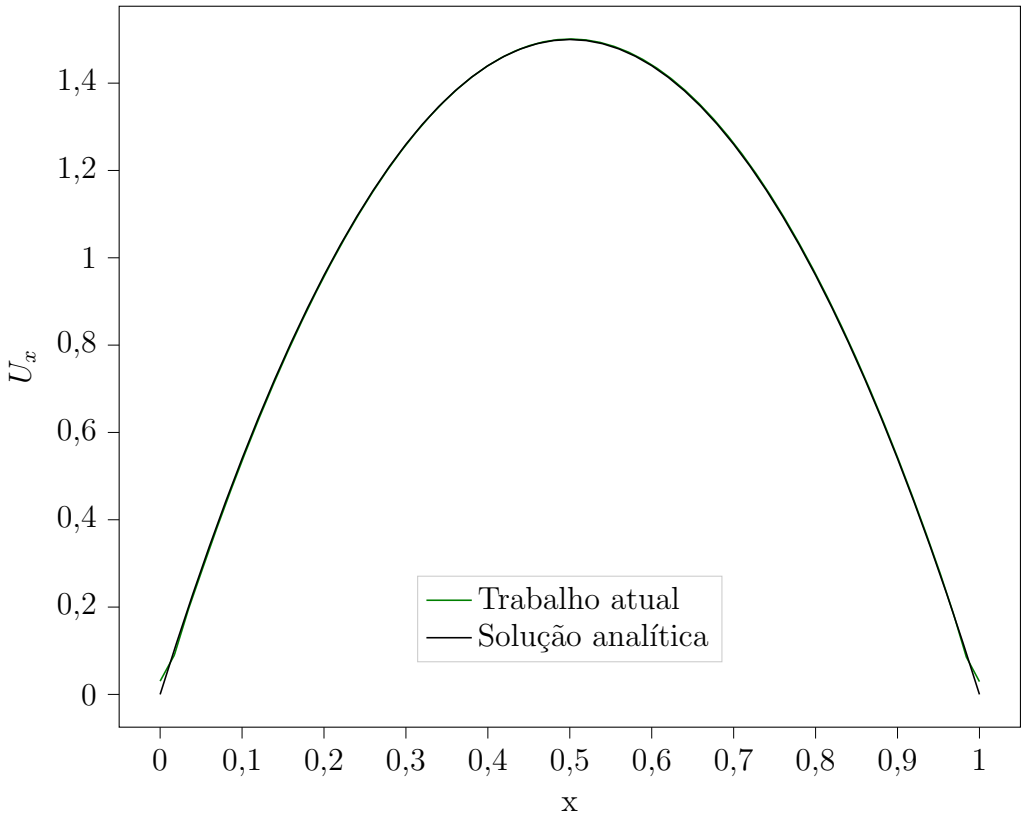


Figura 5.3: Comparação entre a solução analítica do problema da cavidade, com a solução obtida pelo programa. O gráfico representa o valor da velocidade na direção paralela ao escoamento, ao longo do eixo y, que é a direção perpendicular ao escoamento.

5.2 Cavidade

Um problema clássico da mecânica dos fluidos é o escoamento em uma cavidade. Consiste, para o caso bidimensional, em uma caixa quadrada em que apenas uma parede tem a velocidade determinada não nula, sendo as outras, com velocidade corresponde a de estagnação.

5.2.1 Condições de contorno

Para a formulação de corrente vorticidade-vorticidade, utilizando os contornos da figura 5.4, as condições de contorno precisam ser aplicadas segundo os critérios a seguir.

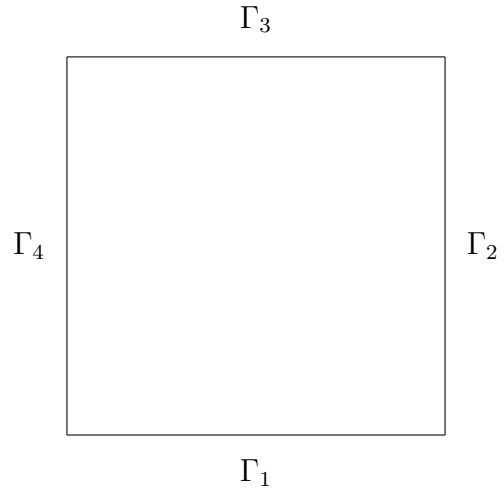


Figura 5.4: Contornos para o problema da cavidade.

$$\begin{cases} U(x,y) = U_\infty \mid x \; \varepsilon \; \Gamma_3 \\ U(x,y) = U_{x,y} \mid (x,y) \; \varepsilon \; (\Gamma_1, \Gamma_2, \Gamma 4) \\ \psi(x,y) = 1 \mid (x,y) \; \varepsilon \; \Gamma_3 \\ \psi(x,y) = 0 \mid (x,y) \; \varepsilon \; (\Gamma_1, \Gamma_2, \Gamma 4) \\ \mathbf{U}_\infty = U_\infty \hat{x} + 0\hat{y} \end{cases} \tag{5.3}$$

Como a formulação sendo dimensional precisamos definir que a velocidade $U_\infty = 1m/s$ e as dimenões da lateral do quadrado possui $1m$.

5.2.2 Resultados

Para compararmos os resultados obtidos nessas simulações com o trabalho de *Ghia et al.* [15] utilizaremos o parâmetro adimensional $\mathbf{U}^* = U(x,y)/|U_\infty|$, isso se faz necessário, pois a velocidade inicial U_∞ usada na simulação difere da bibliografia.

Para valores até $Re = 400$ podemos evidenciar que os resultados obtidos estão em consonância com os observados por *Ghia et al.* [15], como constatado nas figuras (5.5), (5.6), (5.7) e (5.8). Para valores de $Re = 1000$ a simulação apresenta

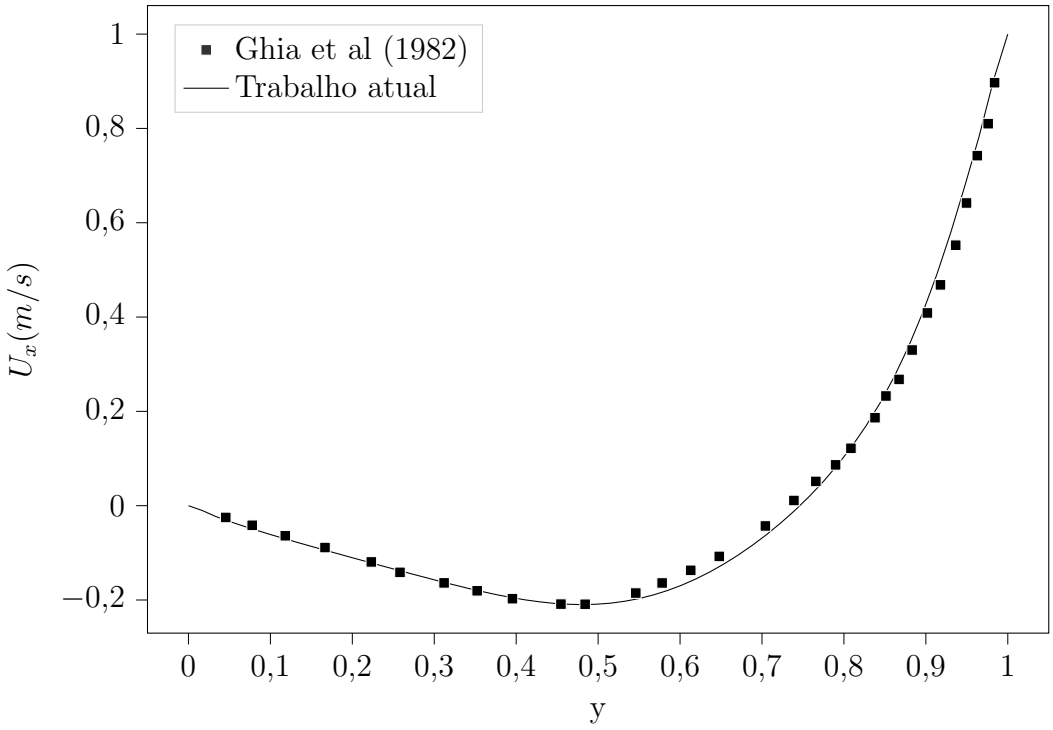


Figura 5.5: Variação da componente de velocidade na direção horizontal. ao longo do eixo $x = 0,5$, para $Re = 100$

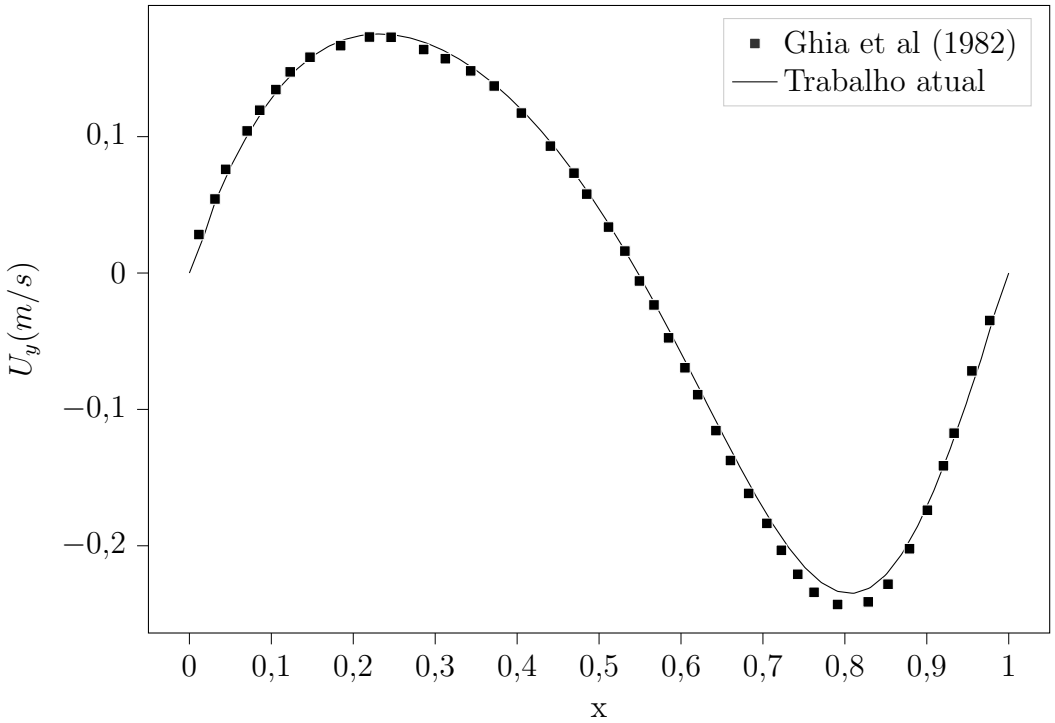


Figura 5.6: Variação da componente da velocidade na direção vertical ao longo do eixo $y = 0,5$, para $Re = 100$

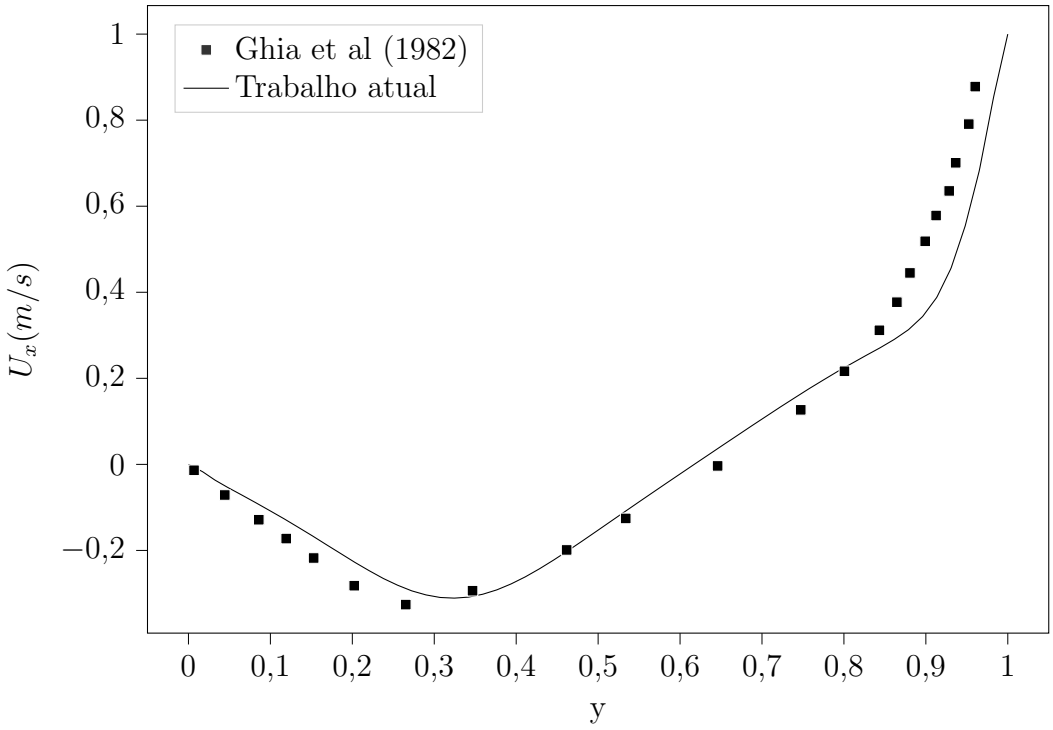


Figura 5.7: Variação da componente da velocidade na direção horizontal. ao longo do eixo $x = 0,5$, para $Re = 400$

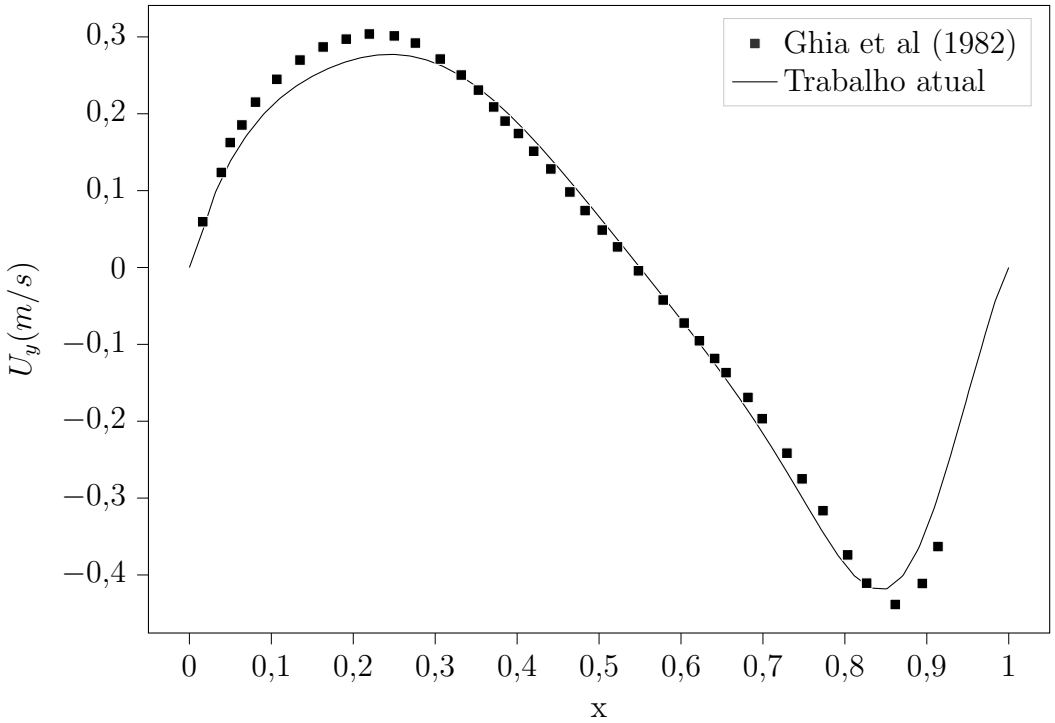


Figura 5.8: Variação da componente da velocidade na direção vertical ao longo do eixo $y = 0,5$, para $Re = 400$

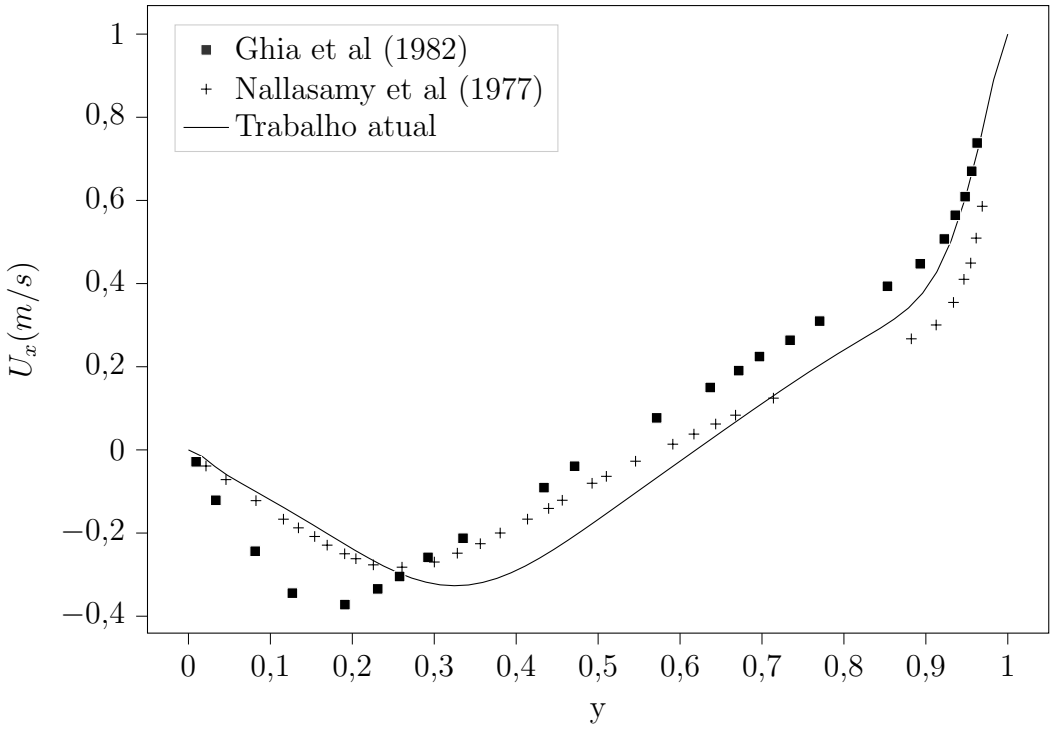


Figura 5.9: Variação da componente da velocidade na direção horizontal. ao longo do eixo $x = 0,5$, para $Re = 1000$

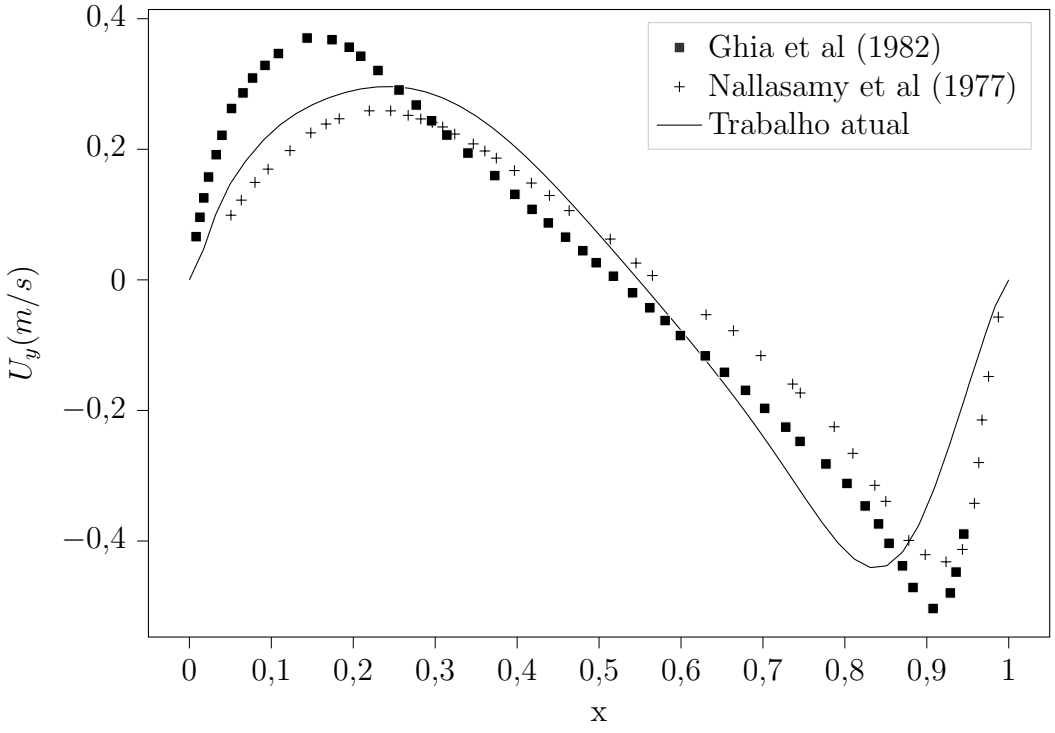


Figura 5.10: Variação da componente da velocidade na direção vertical ao longo do eixo $y = 0,5$, para $Re = 1000$

certas limitações, sendo aconselhado diminuir o intervalo de tempo das interações e diminuir também o tamanho do elemento. Isso resultaria num tempo muito grande para realizar as simulações, sendo aconselhado algum outro método de estabilização números mais elevados de Reynolds.

5.3 Degrau (*"backward-facing step"*)

O problema do degrau é outro bem descrito na bibliografia e com vários resultados experimentais conhecidos, sendo assim um ótimo problema para validar o código. A seguir apresentamos as condições de contorno para esse problema na Figura 5.11.

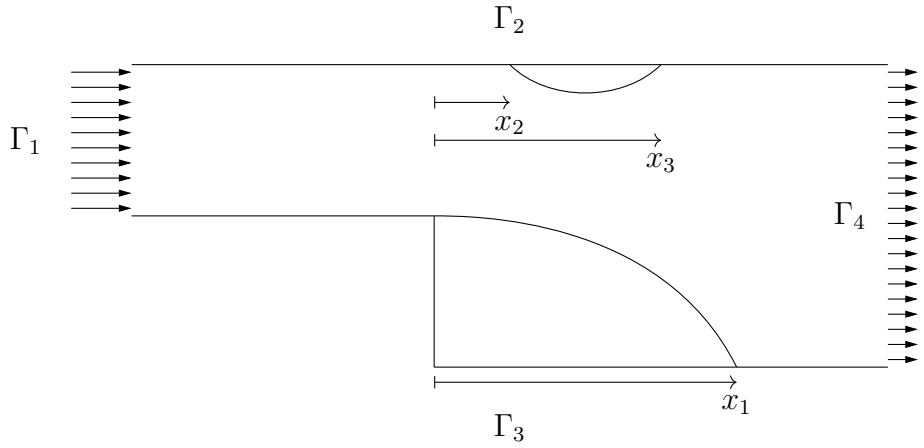


Figura 5.11: Contorno para o problema do degrau.

Definido o contorno podemos aplicar as condições iniciais para resolver o problema.

$$\begin{cases} U(x,y) = 0 \mid (x,y)\ \varepsilon\ (\Gamma_2, \Gamma_3) \\ U(x,y) = U_\infty \mid (x,y)\ \varepsilon\ \Gamma_1 \\ \psi(x,y) = f(y)\big|_0^1 \mid (x,y)\ \varepsilon\ \Gamma_1 \\ \psi(x,y) = 1 \mid (x,y)\ \varepsilon\ \Gamma_2 \\ \psi(x,y) = 0 \mid (x,y)\ \varepsilon\ \Gamma_3 \\ \nabla\psi(x,y) \cdot \mathbf{n} = 0 \mid (x,y)\ \varepsilon\ \Gamma_4 \\ \mathbf{U}_\infty = U_\infty \hat{x} + 0\hat{y} \end{cases} \tag{5.4}$$

Compararemos nossos resultados com os obtidos experimentalmente por *Armaly et al.* [16] usando o parâmetro x_1/S, sendo S a altura do degrau e x_1 o ponto onde

o escoamento se reconecta com a parede.

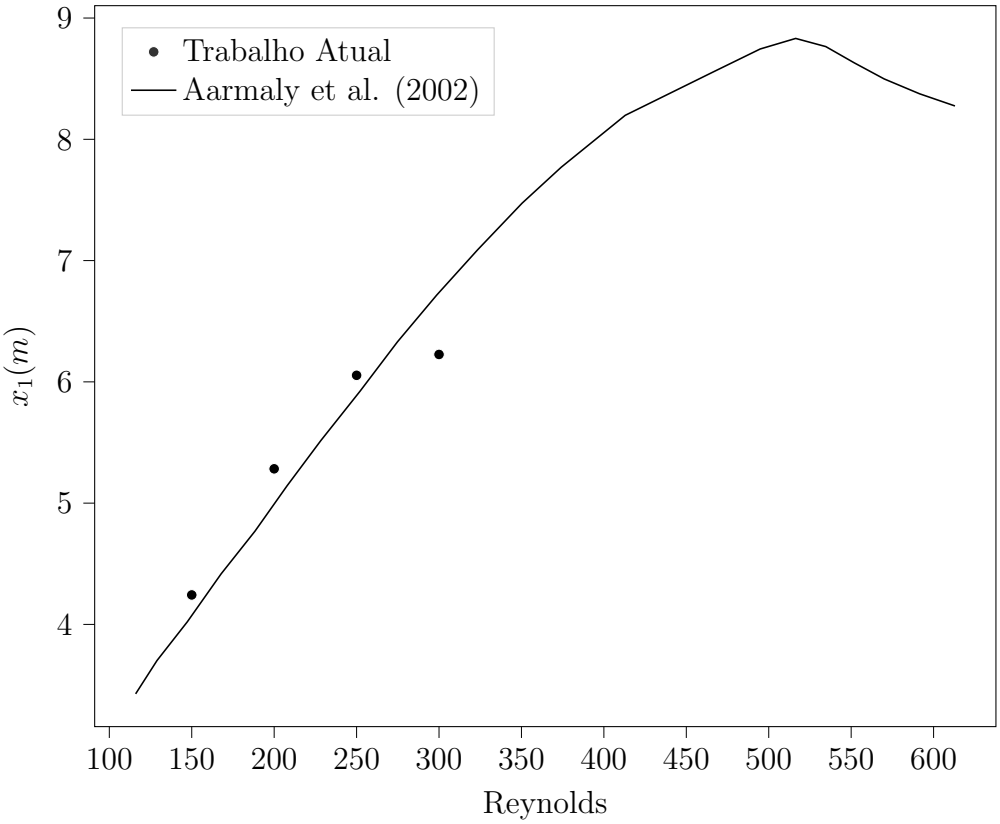


Figura 5.12: Comparação entre os resulados obtidos pelo modélo e a bibliografia

5.4 Cilindro estático

As condições de contorno para um cilindro estático são muito similares àquelas utilizadas no escoamento em um duto. Para garantir que o número de Reynolds seja aquele esperado no escoamento ao redor do corpo de interesse, imporemos, nas paredes superior e inferior, a velocidade de entrada de fluido no canal.

Como falamos no capítulo 2, compararemos a frequência de descolamento dos vórtices pode ser relacionado com o número de Reynolds, segundo a equação (2.3).

Para essa simulação obteremos o coeficiente de sustentação (C_L) usando a equação (3.135). Como o coeficiente tem oscilações ao longo do tempo, utilizaremos o valor eficaz (rms) segundo a equação 5.5.

$$C_L{}^{rms} = \sqrt{\frac{1}{n}({C_{L_1}}^2 + {C_{L_1}}^2 + \cdots + {C_{L_n}}^2)} \tag{5.5}$$

Sabendo que o C_L é diretamente influenciado pelo descolamento dos vórtices, usamos a trasnformada de fourier discreta para obter as frequências de ocilação.

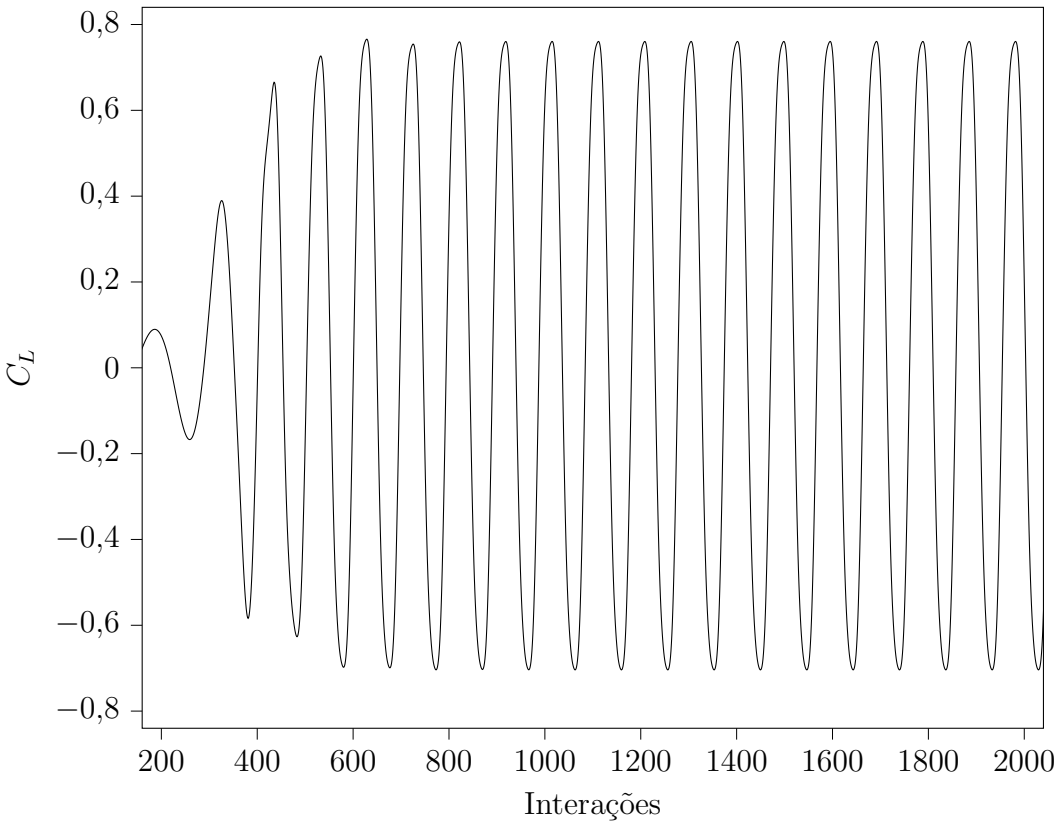


Figura 5.13: Variação do coeficiente de sustentação, C_l, em função das interações

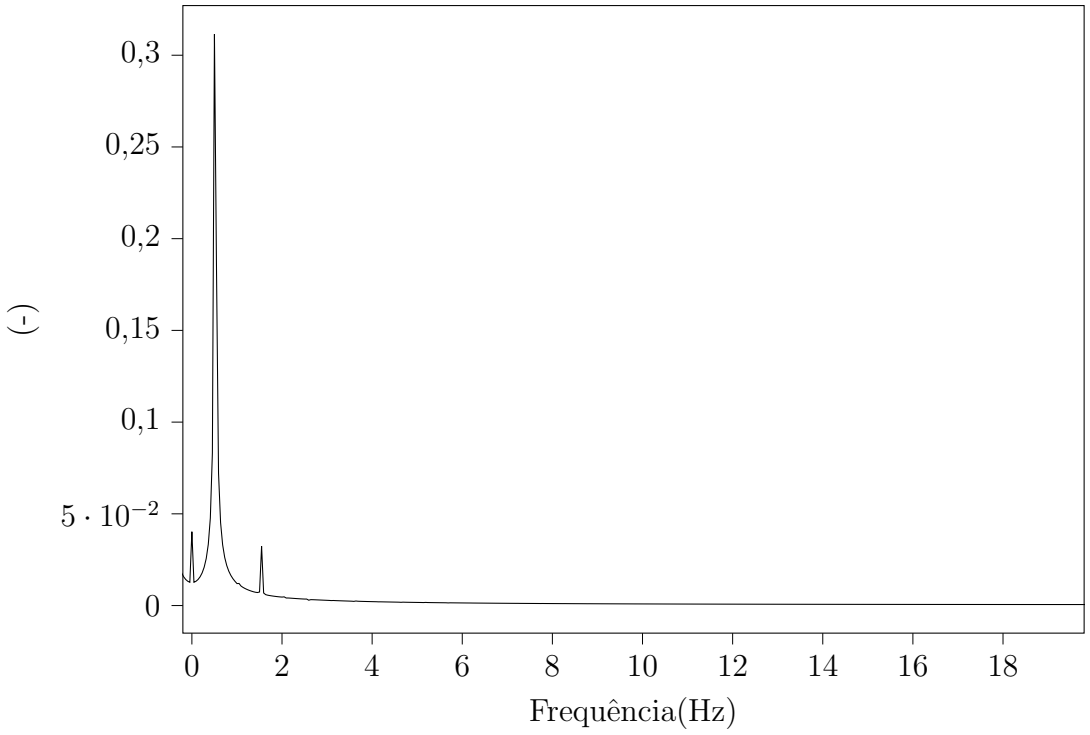


Figura 5.14: Transformada rápida de fourier discreta intervalos de interação $\Delta t = 0,018s$, para $Re = 300$

5.4.1 Malha

Para este caso foi utilizada uma malha com 19.197 pontos e 37.945 elementos triangulares. Visando obter uma maior precisão para o coeficiente de sustentação nos locais próximos à fronteira do círculo foram refinados, com a área de integração.

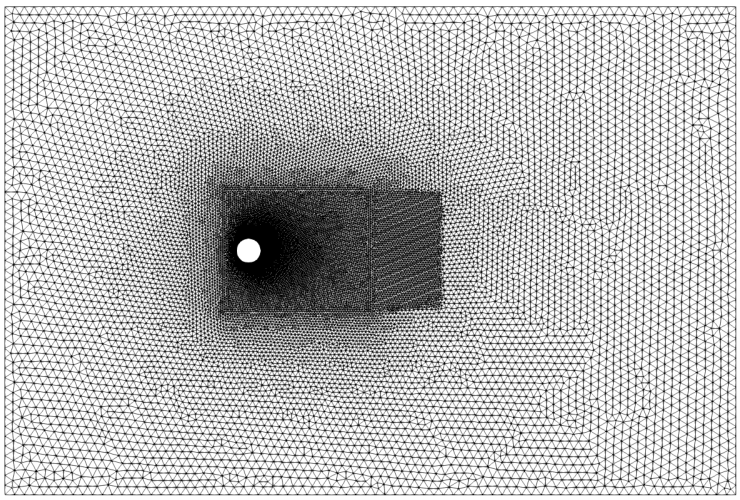


Figura 5.15: Malha no interior do dominio para o caso do escoamente ao redor do cilindro

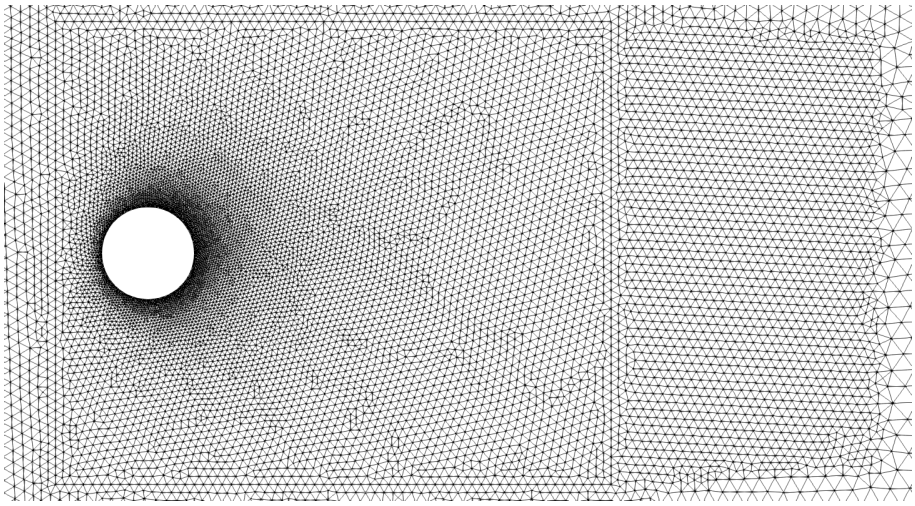


Figura 5.16: Malha próxima da área de integração para obeteção dos valores de C_L

5.4.2 Resultados

Caculando o coeficiente e o numero de Strouhal para cada numero de reynolds no intervalo, $180 \leqslant Re \leqslant 325$, com passo de 5.

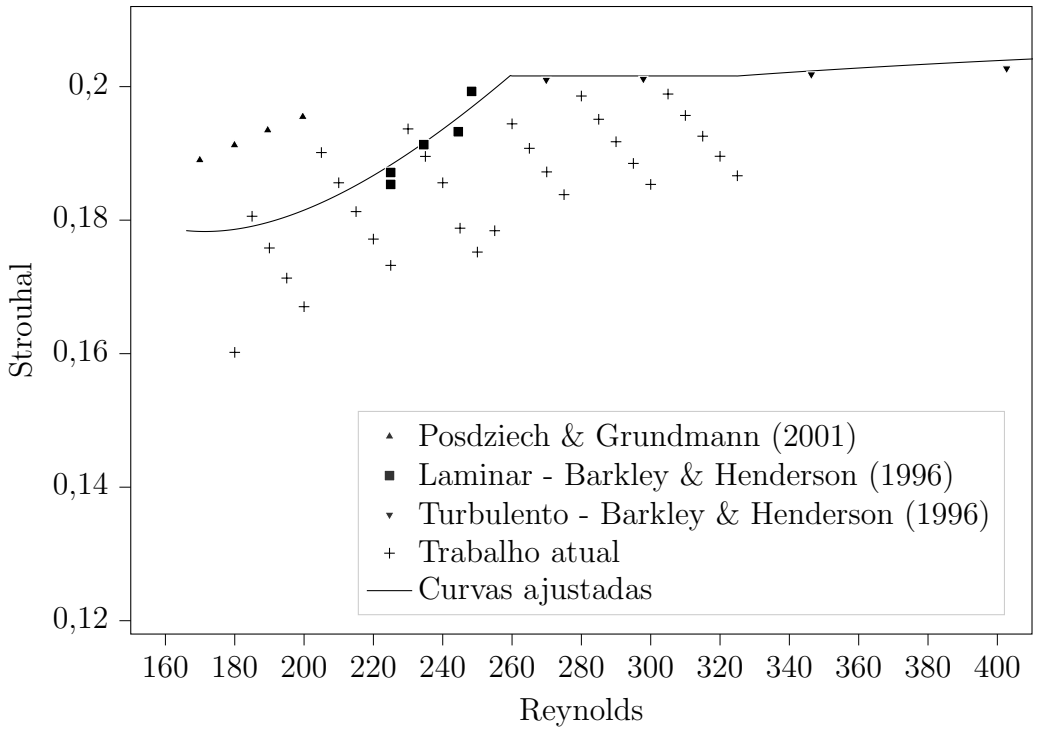


Figura 5.17: Comparação do numero de Strouhal com a bibliografia

Assim como descrito por *Fiabane et al.* [17], obtivemos resultados com valores semelhantes à bibliografia, não podemos deixar de comentar que na faixa de número de *Re* escolhida existe uma transição no tipo de desprendimentos de vórtices, passando

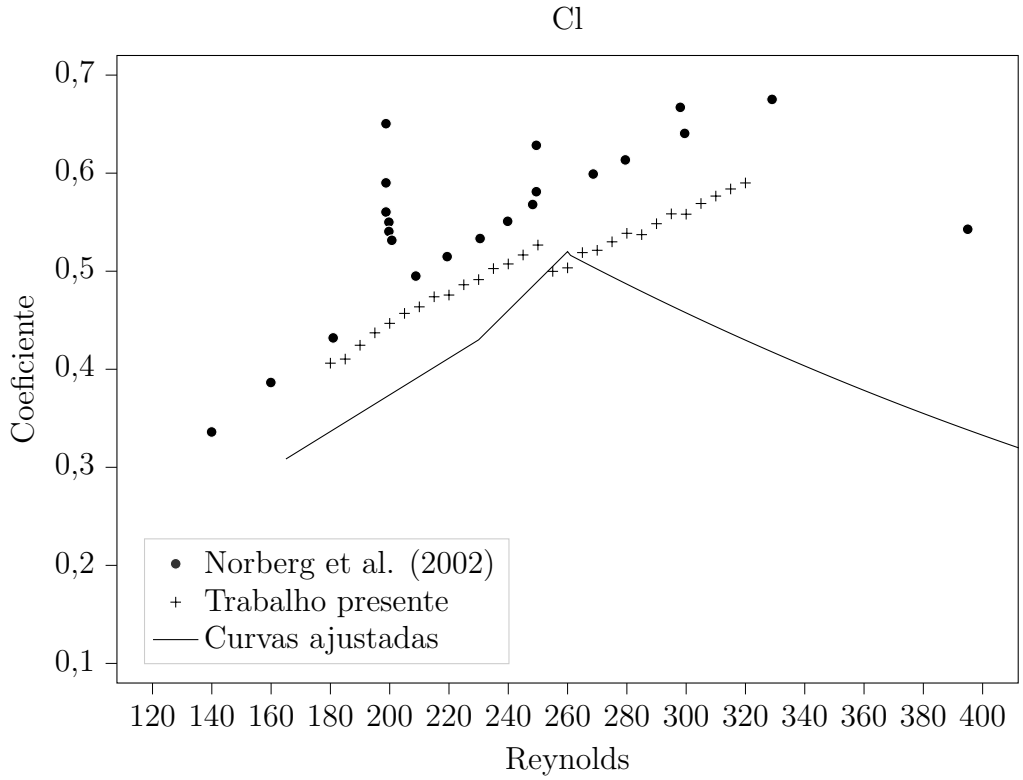


Figura 5.18: Comparação do coeficiente de sustentação C_L com a bibliografia

do desprendimento laminar para o turbulento, o que poderia explicar o afastamento dos resultados do ajuste da curva descrito pela bibliografia, o modelo de calculo de coeficiente de sustentação não inclui as parcelas de integração de linha no contorno da região arbitrada. Porém, os resultados estão próximos do trabalho de *Norberg C.* [18].

5.5 Elipse estática

Na simulação com o círculo observamos que as ocilações no coeficiente de arrasto são consideraveis. Para verificar se com a subtituição desse círculo por uma elipse diminuimos a variação do coeficiente, fizemos uma bateria de simulações com o Re variando de 240 a 340, também variado a geometria da elipse.

A geometria de interesse pode ser definida pela seguinte equação matématica.

$$\left(\frac{x}{a}\right)^2 + \left(\frac{x}{b}\right)^2 = 1 \tag{5.6}$$

Para vermos como o comportamento muda, conforme alteramos a relação de

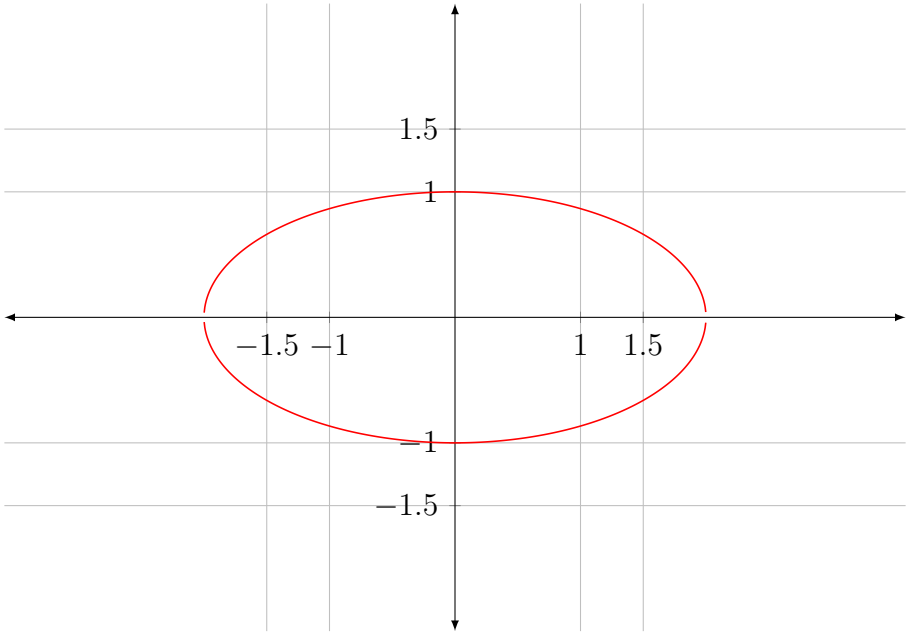


Figura 5.19: Parâmetros da equação (5.6), $a = 2$, $b = 1$.

a/b. Como estamos utilizando a fomulação dimensional, tanto a como b, estão em metros. Na figura (5.21) podemos ver como a malha de elementos está distribuida. Fizemos um refinamento próximo da elipse para melhorar a precisão do cálculo de coeficiente de sustentação. A malha possuir um total 38933 células triangulares. O tempo necessário para concluir cada iteração do programa foi de aproximadamente 4s. Levando em conta que para cada numero de Reynolds precisamos de 2400 iterações para garantir que as ocilações tenham um periodo e amplitude fixa, o tempo obter cada ponto da figura (5.20) leva aproximadamente 2h40m. Para cada razão de aspecto foram feitas simulações variando o Reynolds de 240 a 340, isso nós deu um tempo aproximado de 27h para cada.

Como podemos ver, conforme a relação de aspecto diminui, tentendo a geometria de um círculo, o coeficiente de sustentanção, em valor rms, aumenta. Isso se deve a que a esteira em que ocorre o desprendimento dos vortices é maior em relação as geometria com maior razão de aspecto.

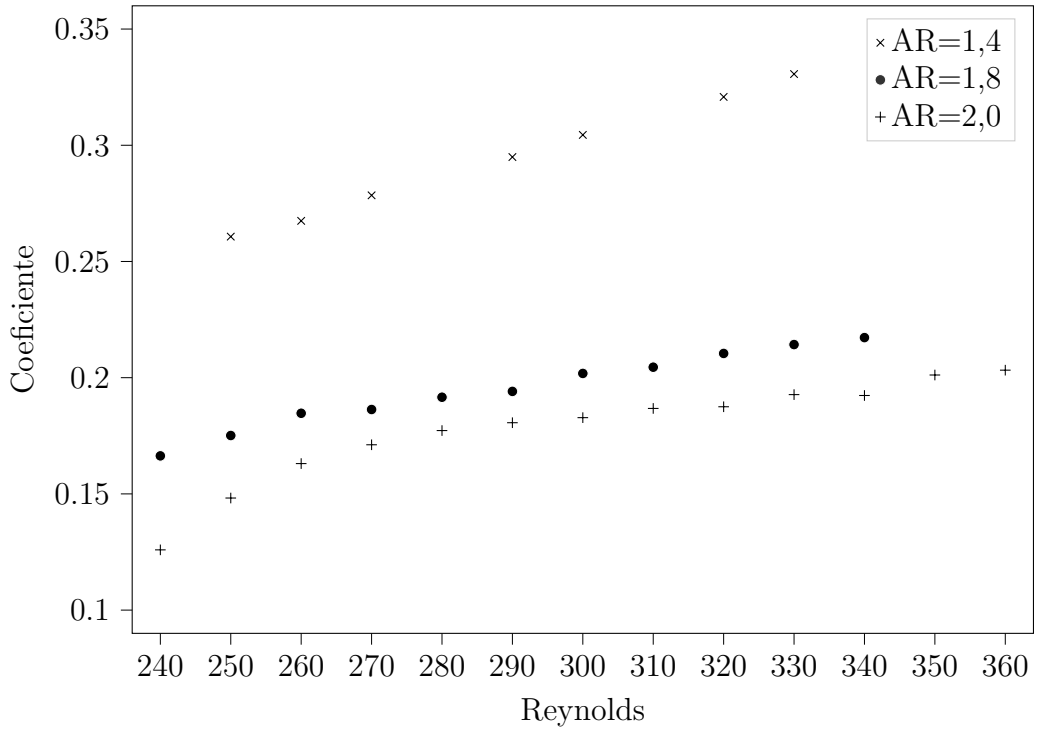


Figura 5.20: Grafico mostrando a variação do coeficiente de sustentação C_L em função do numero de Reynolds. Chamamos a relação de aspecto, $a/b = AR$.

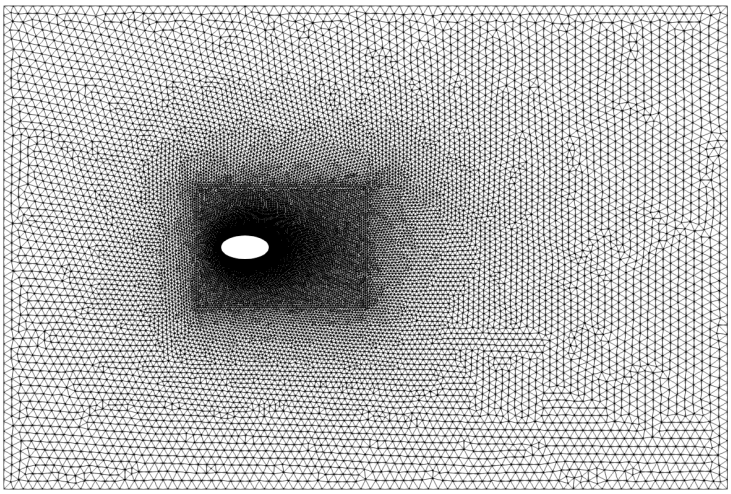
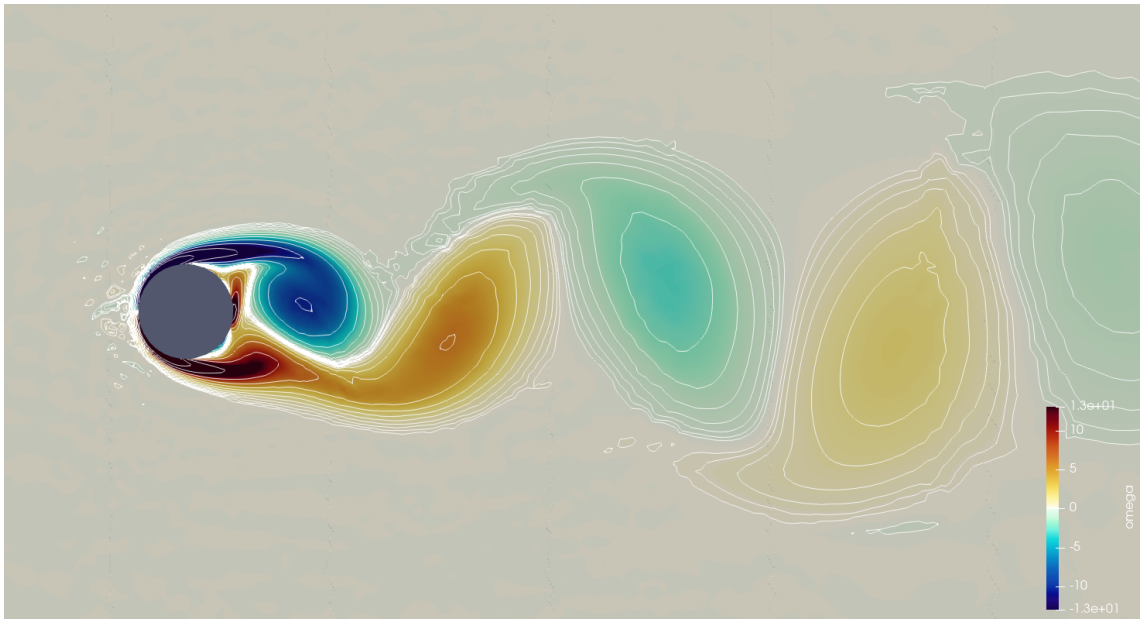
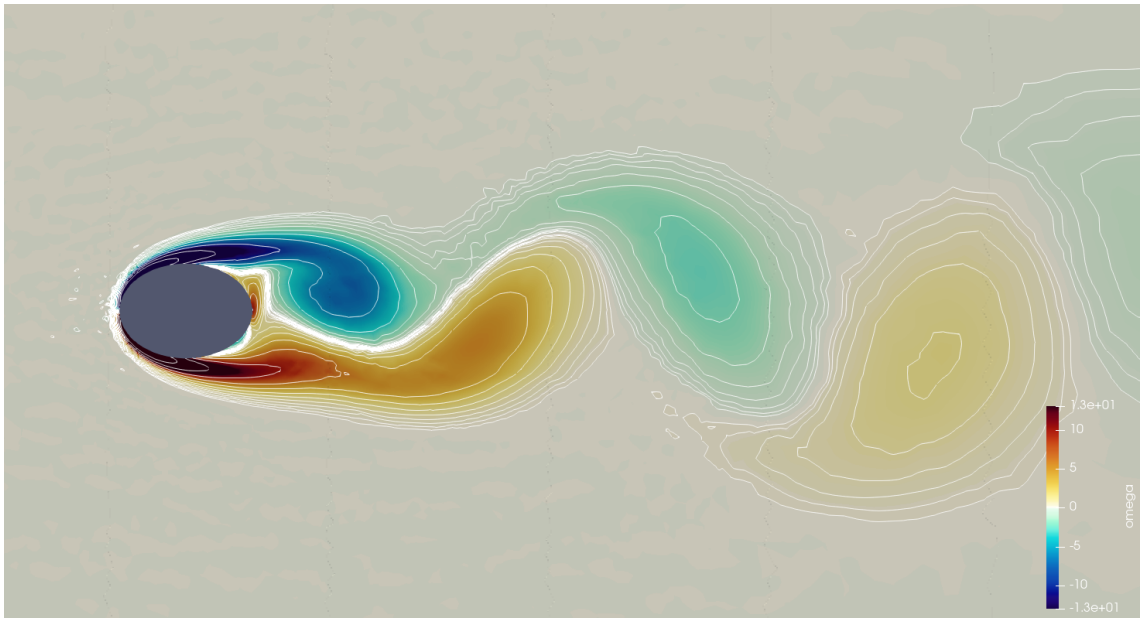


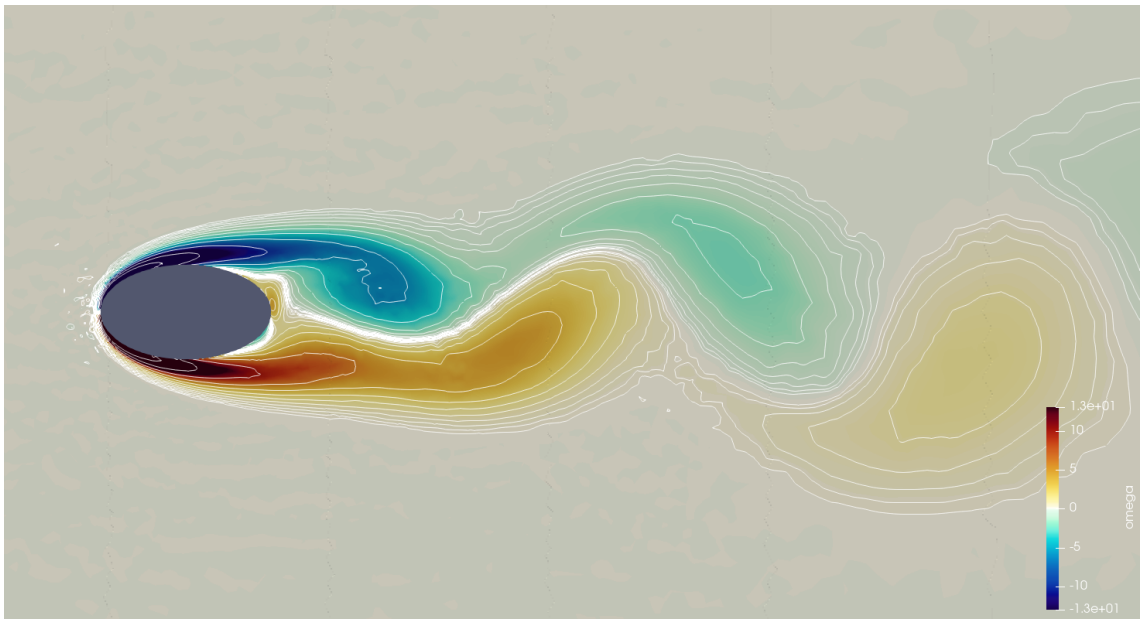
Figura 5.21: Malha para simulação com a geometria da elipse



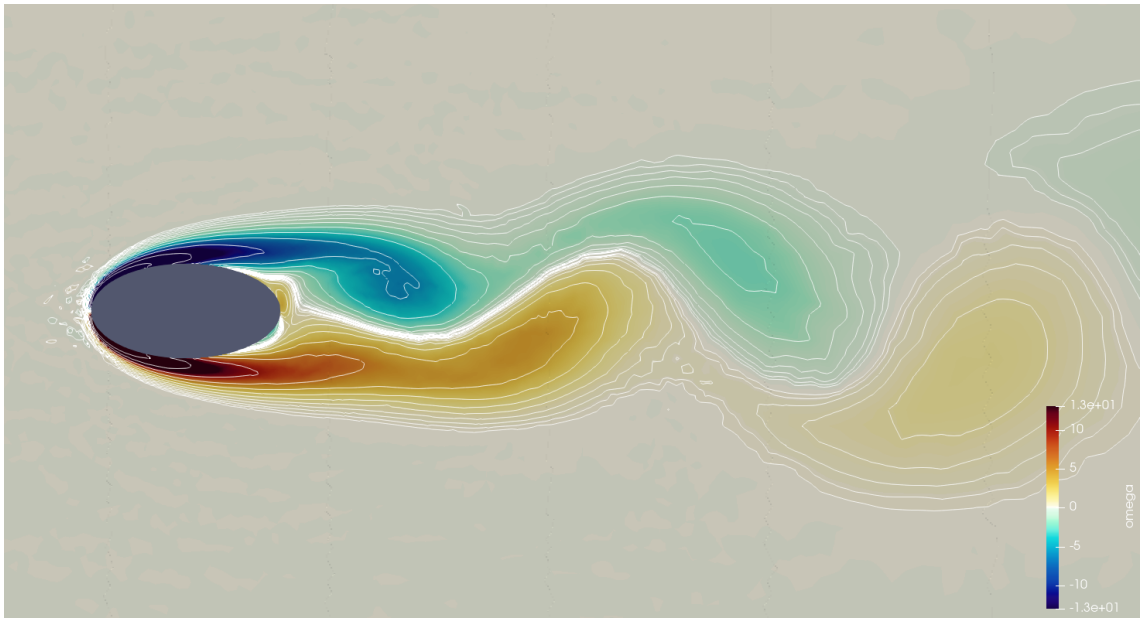
(a)



(b)

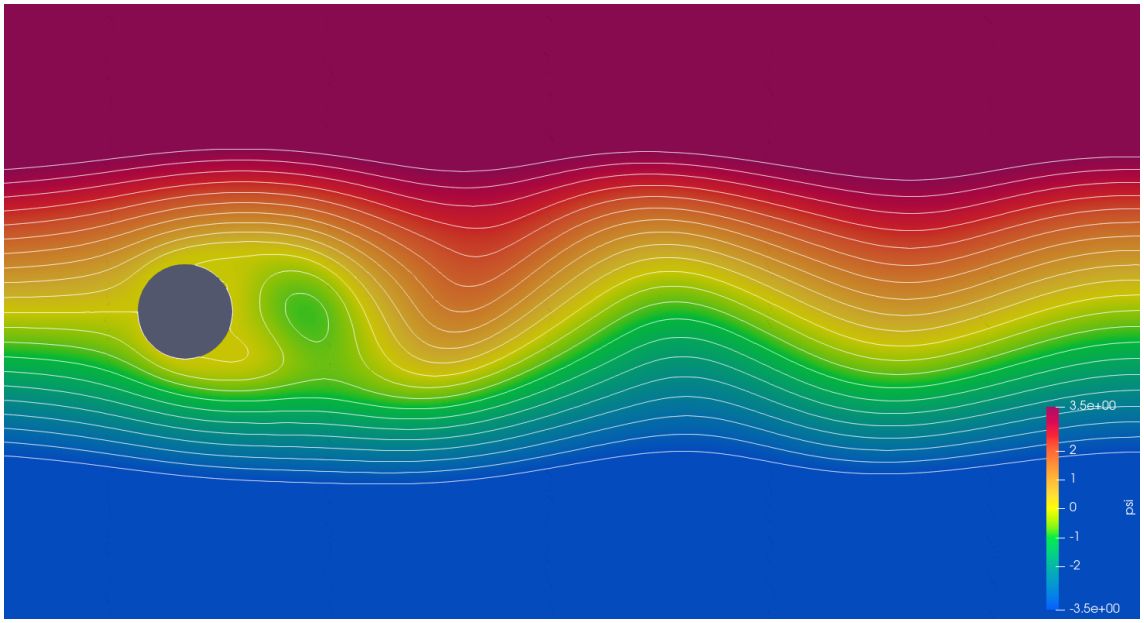


(c)

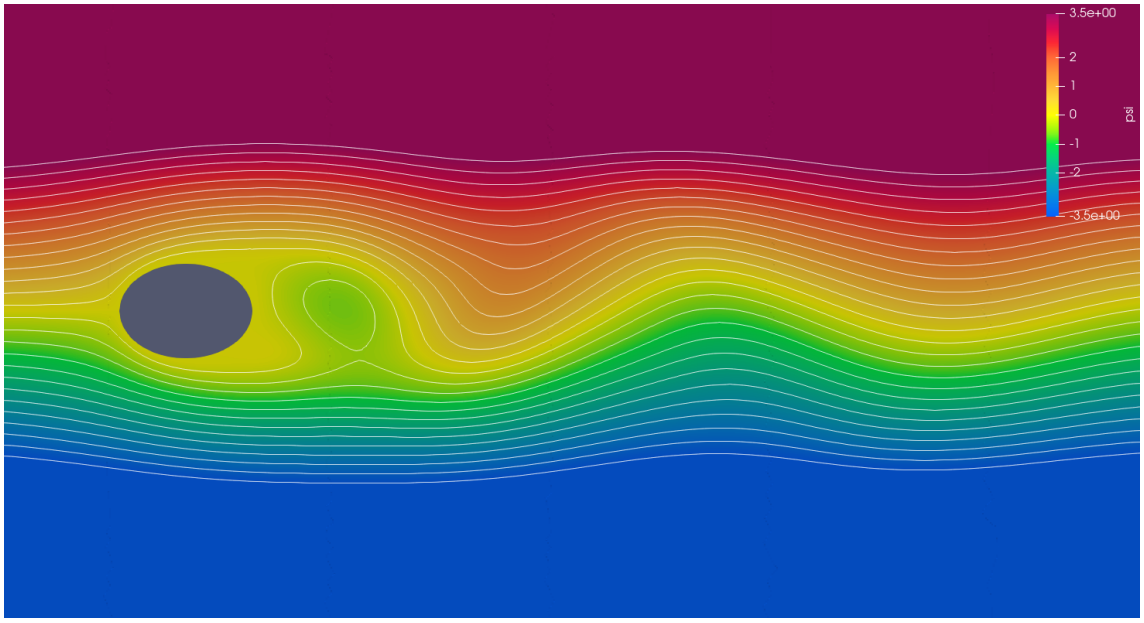


(d)

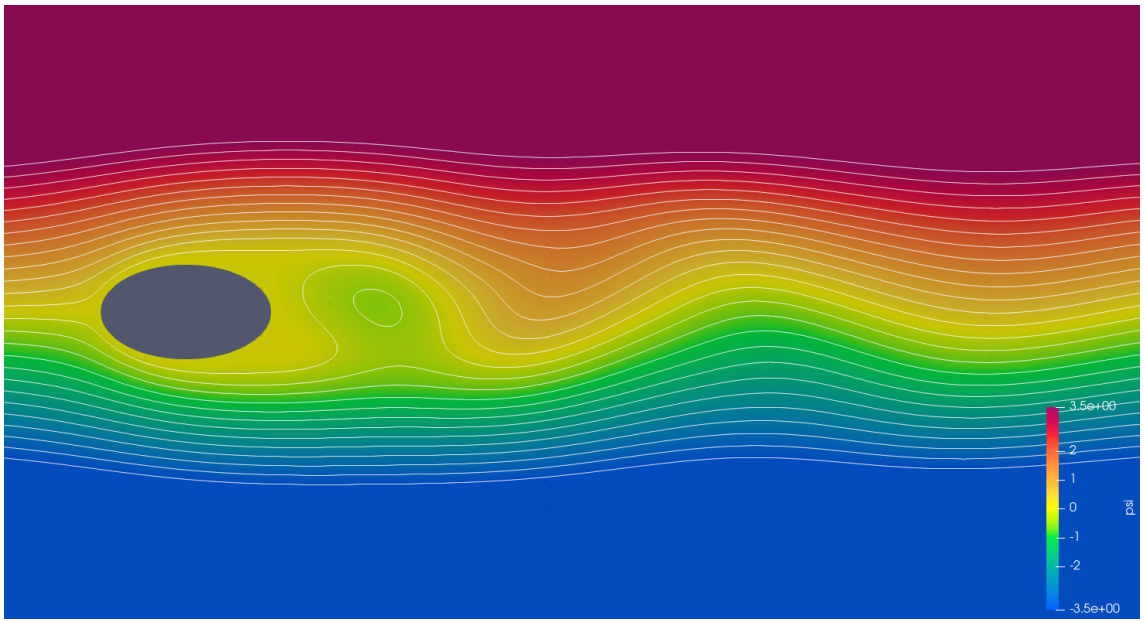
Figura 5.22: Representação gráfica da vorticidade, ω, para número de Reynolds igual a 250 com as seguintes razões de aspecto (**AR**). (a) AR = 1,0; (b) AR = 1,4; AR = 1,8; AR = 2,0. Os limites de representação de cores são, -13 e 13.



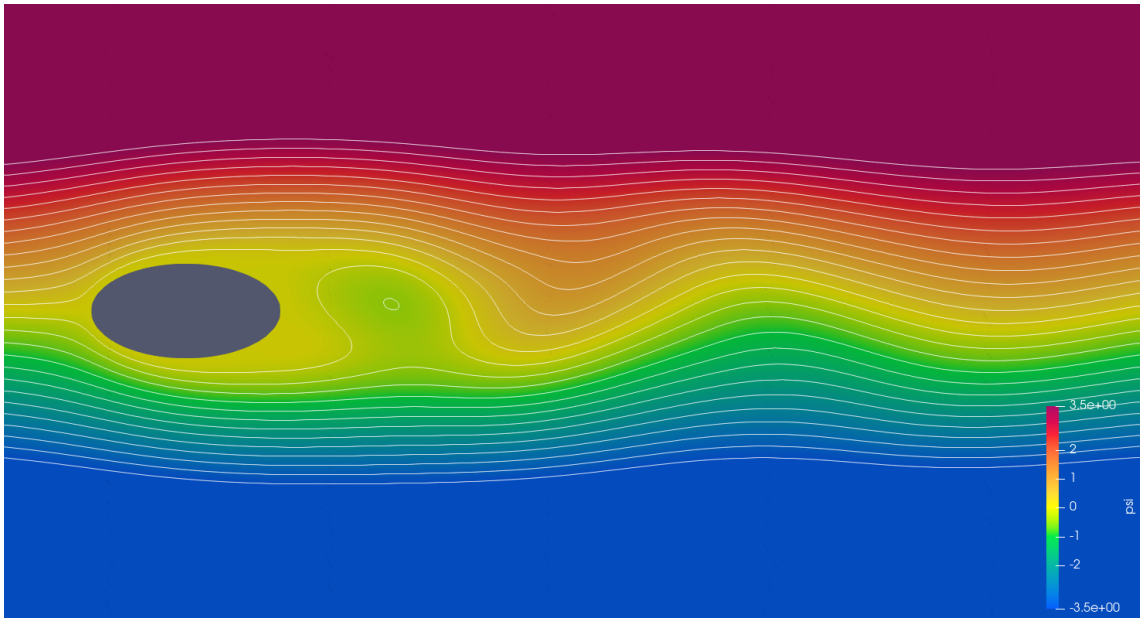
(a)



(b)

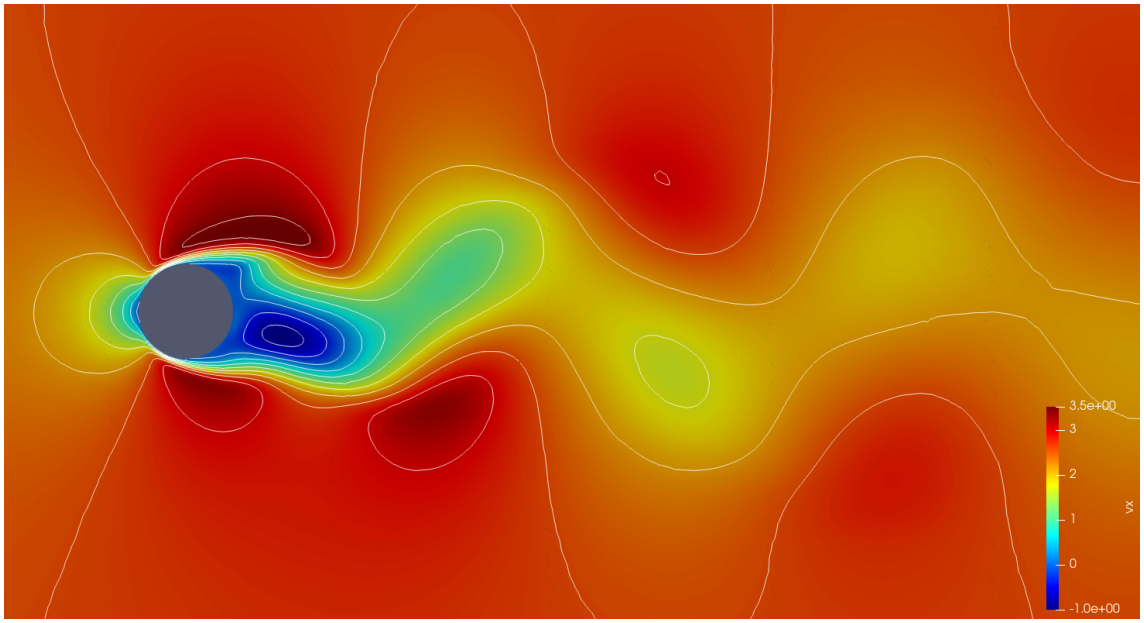


(c)

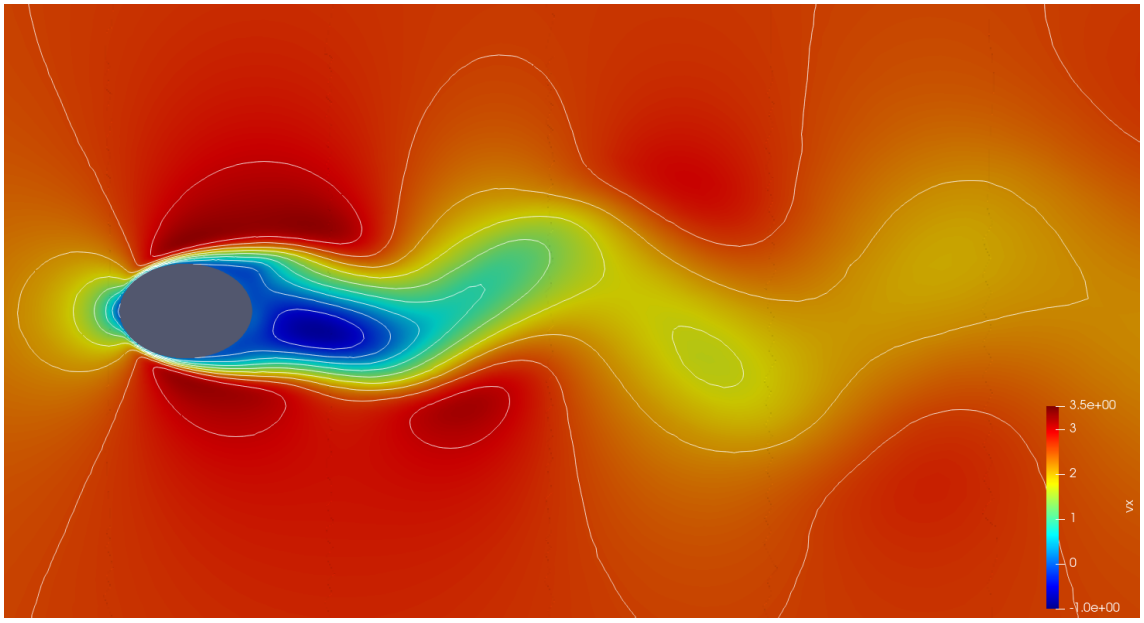


(d)

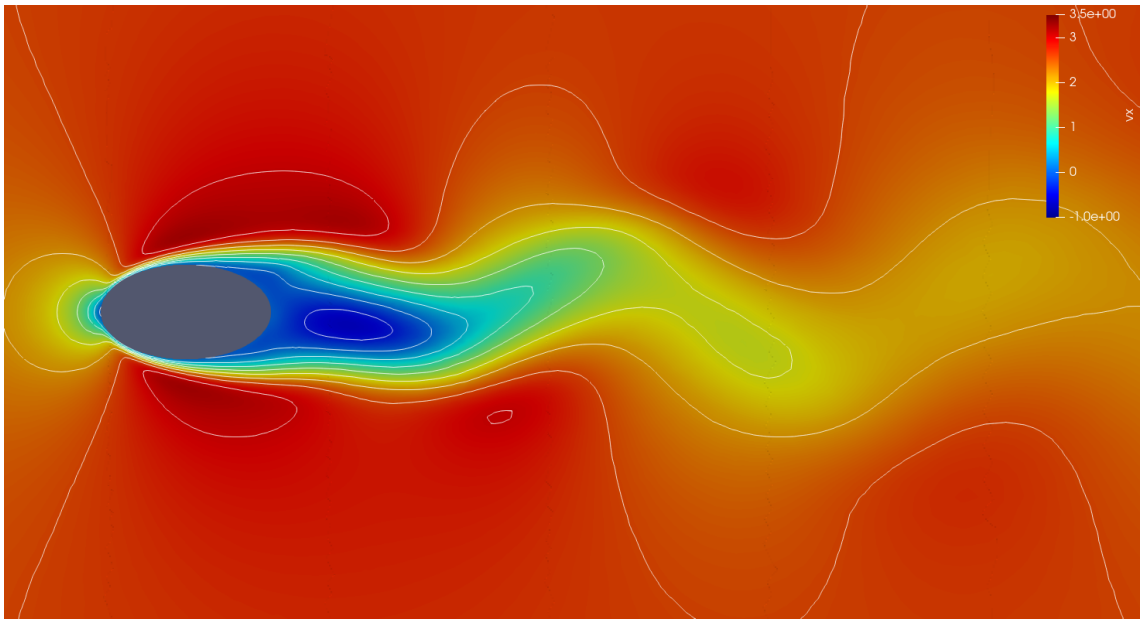
Figura 5.23: .Representação da função corrente, ψ, para número de Reynolds igual a 250 com as seguintes razões de aspecto (**AR**). (a) AR = 1,0; (b) AR = 1,4; AR = 1,8; AR = 2,0. Os limites de representação de cores são, -3,5 e 3,5.



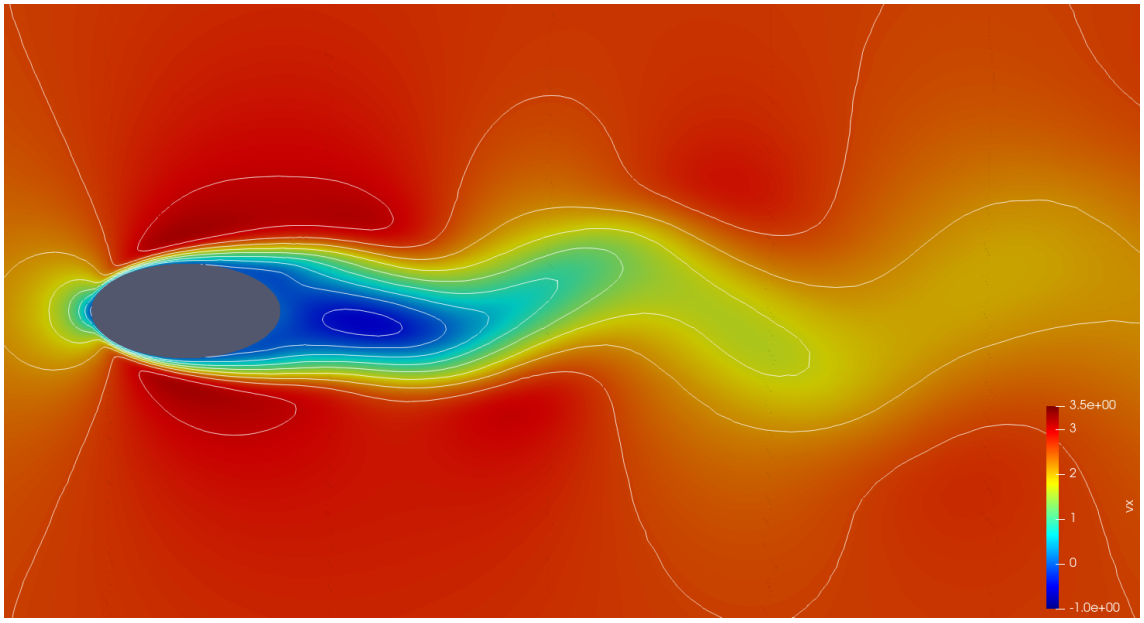
(a)



(b)

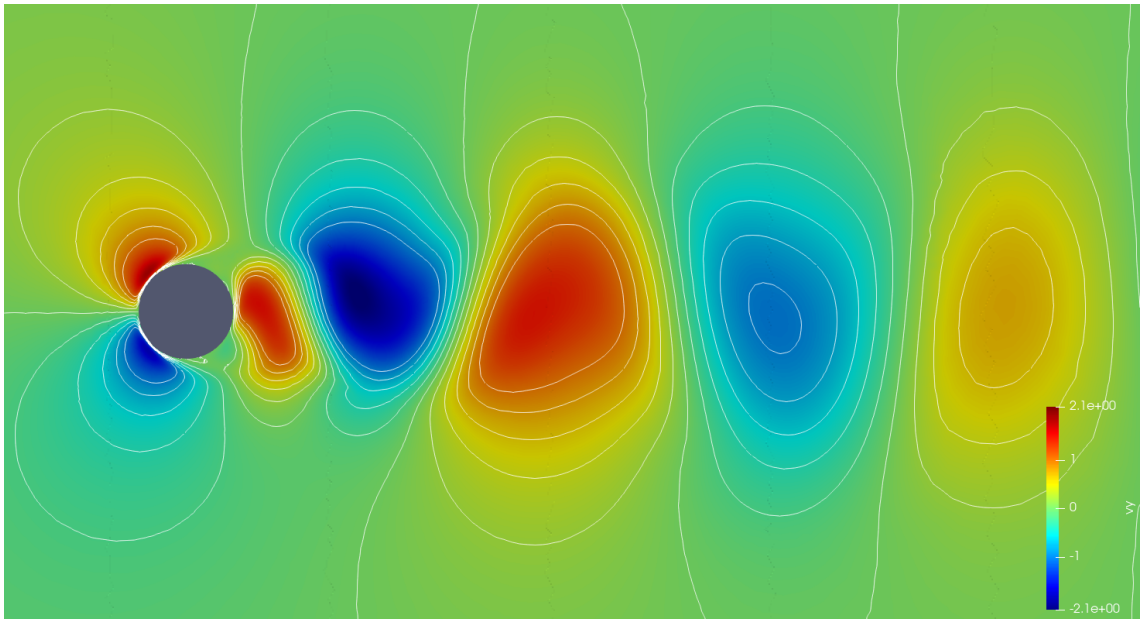


(c)

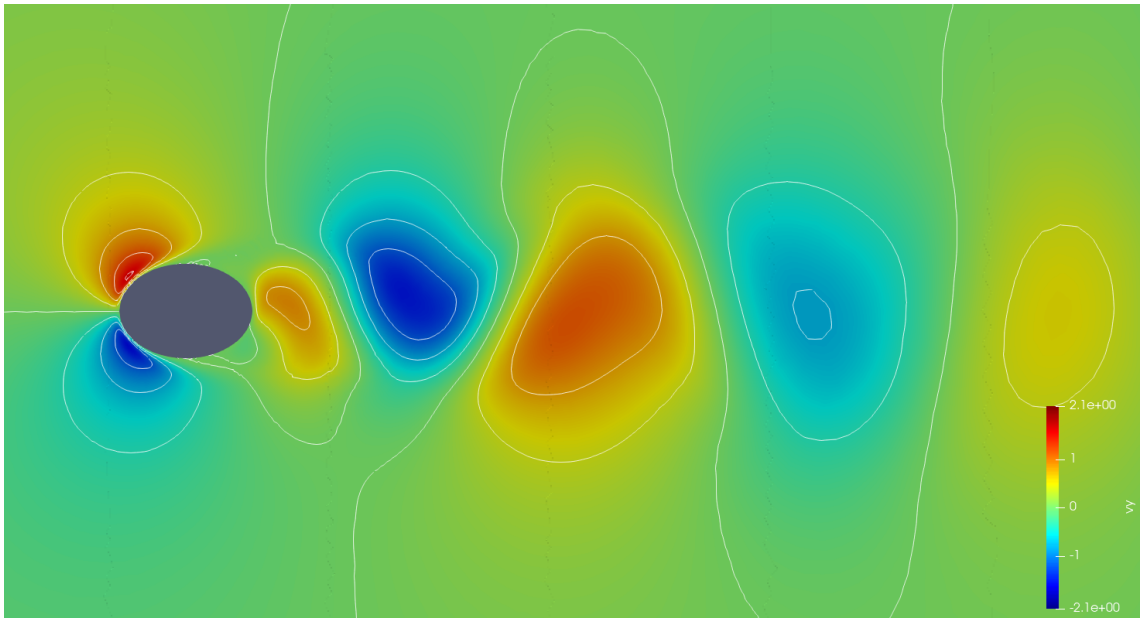


(d)

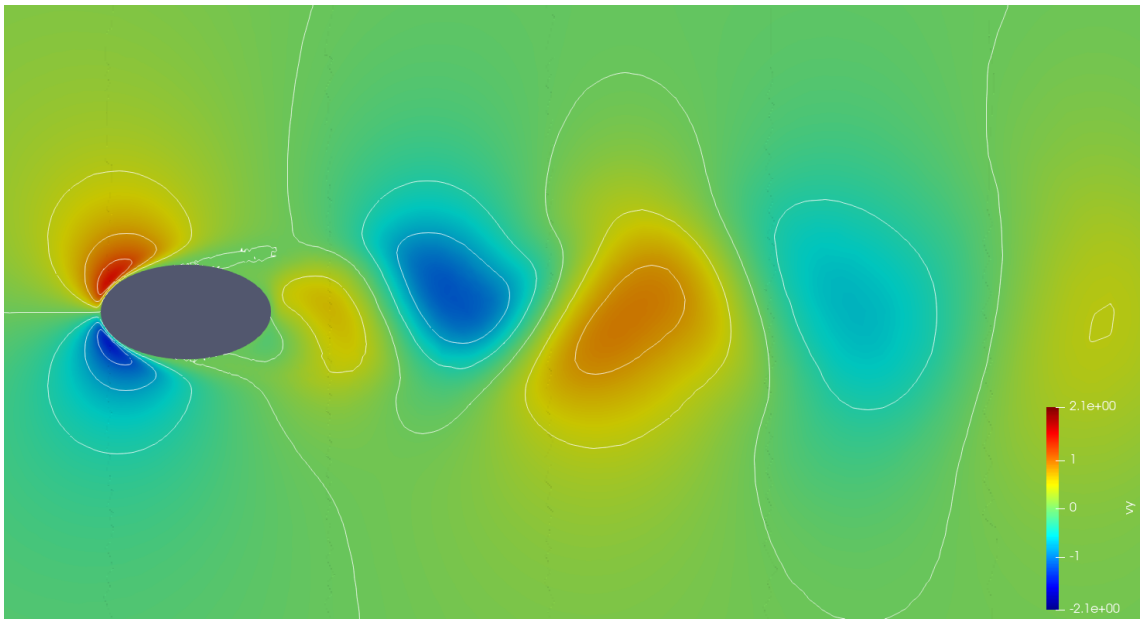
Figura 5.24: Representação gráfica do campo de velocidades na direção horizontal, para número de Reynolds igual a 250 com as seguintes razões de aspecto (**AR**). (a) AR = 1,0; (b) AR = 1,4; AR = 1,8; AR = 2,0. Os limites de representação de cores são, $-1,0m/s$ e $3,5m/s$.



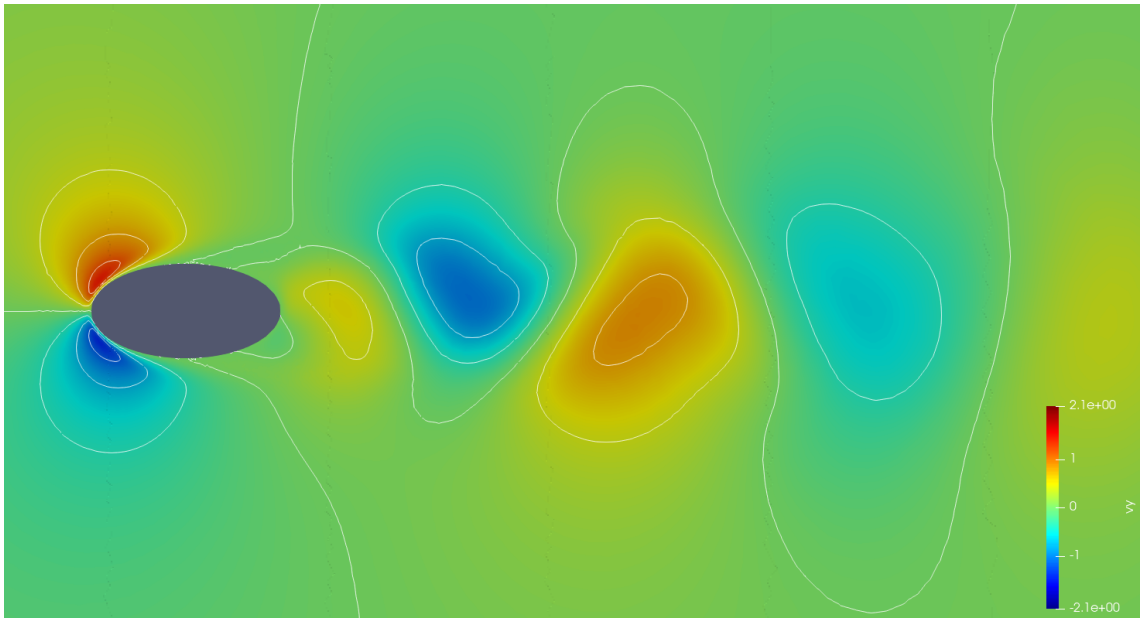
(a)



(b)



(c)



(d)

Figura 5.25: Representação gŕafica do campo de velocidades, na direção vertical, para número de Reynolds igual a 250 com as seguintes razões de aspecto (**AR**). (a) AR = 1,0; (b) AR = 1,4; AR = 1,8; AR = 2,0. Os limites de representação de cores são, $-2,1m/s$ e $2,1m/s$.

As figuras (5.22), (5.23), (5.24) e (5.25) não estão em regime permanente, apenas a amplitude de oscilação é constante. Elas foram obtidas na iteração de número 2000. Podemos observar que existe o desprendimento de vórtices, assim como uma zona

de recirculação a jusante das geometrias.

Capítulo 6

Conclusões

Para os casos de validações obtivemos resultados satisfatórios até o número $Re = 300$, para números maiores, foi necessário diminuir o tamanho da malha e também diminuir o passo de tempo. Mesmo com esses artifícios para a validação da cavidade de $Re = 1000$ os resultados não foram satisfatórios, isso era esperado, pois a formulação não consegue lidar com tais instabilidades numéricas. Para números maiores de reynods é recomendado outra forma de estabilização.

Para trabalhos futuros é possível melhorar o código implementado a resolução de sistemas lineares usando PyCUDA, consiste num ambiente de desenvolvimento disponibilizado pela Nvidia, para a programação de GPUs para cálculos matemáticos.

Com isso estaríamos otimizando o método de cálculo, atualmente limitado pelo número de núcleos de processadores da máquina. Com o PyCUDA poderíamos explorar a resolução do problema com um número muito maior de núcleos trabalhando em paralelo. Isso é interessante visto que o custo de operação e aquisição de um núcleo de GPU é muito menor que um núcleo de CPU.

Referências Bibliográficas

[1] SARPKAYA, T., "Vortex-induced oscillations: a selective review", 1979.

[2] FEY, U., KÖNIG, M., ECKELMANN, H., "A new Strouhal–Reynolds-number relationship for the circular cylinder in the range $47 < Re < 2 \times 10^5$", *Physics of Fluids*, v. 10, n. 7, pp. 1547–1549, 1998.

[3] ZIENKIEWICZ, O. C., TAYLOR, R. L., ZHU, J. Z., *The finite element method: its basis and fundamentals*. Elsevier, 2005.

[4] TURNER, M. J., CLOUGH, R. W., MARTIN, H. C., et al., "Stiffness and deflection analysis of complex structures", *journal of the Aeronautical Sciences*, v. 23, n. 9, pp. 805–823, 1956.

[5] CLOUGH, R. W., "The finite element method in plane stress analysis". In: *Proceedings of 2nd ASCE Conference on Electronic Computation, Pittsburgh Pa., Sept. 8 and 9, 1960*, 1960.

[6] DONEA, J., "A Taylor–Galerkin method for convective transport problems", *International Journal for Numerical Methods in Engineering*, v. 20, n. 1, pp. 101–119, 1984.

[7] VASCHY, A., "Sur les lois de similitude en physique". In: *Annales télégraphiques*, v. 19, pp. 25–28, 1892.

[8] BUCKINGHAM, E., "On physically similar systems; illustrations of the use of dimensional equations", *Physical review*, v. 4, n. 4, pp. 345, 1914.

[9] KALANTAROV, V. K., LADYZHENSKAYA, O. A., "The occurrence of collapse for quasilinear equations of parabolic and hyperbolic types", *Journal of Soviet Mathematics*, v. 10, n. 1, pp. 53–70, 1978.

[10] BABUŠKA, I., "Error-bounds for finite element method", *Numerische Mathematik*, v. 16, n. 4, pp. 322–333, 1971.

[11] BREZZI, F., "On the existence, uniqueness and approximation of saddle-point problems arising from Lagrangian multipliers", *Publications mathématiques et informatique de Rennes*, , n. S4, pp. 1–26, 1974.

[12] LEBEDEV, L. P., CLOUD, M. J., EREMEYEV, V. A., *Advanced engineering analysis: the calculus of variations and functional analysis with applications in mechanics*. World Scientific, 2012.

[13] WU, J.-Z., LU, X.-Y., ZHUANG, L.-X., "Integral force acting on a body due to local flow structures", *Journal of Fluid Mechanics*, v. 576, pp. 265–286, 2007.

[14] GEUZAINE, C., REMACLE, J.-F., "Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities", *International journal for numerical methods in engineering*, v. 79, n. 11, pp. 1309–1331, 2009.

[15] GHIA, U., GHIA, K. N., SHIN, C., "High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method", *Journal of computational physics*, v. 48, n. 3, pp. 387–411, 1982.

[16] ARMALY, B. F., DURST, F., PEREIRA, J., et al., "Experimental and theoretical investigation of backward-facing step flow", *Journal of fluid Mechanics*, v. 127, pp. 473–496, 1983.

[17] FIABANE, L., GOHLKE, M., CADOT, O., "Characterization of flow contributions to drag and lift of a circular cylinder using a volume expression of the fluid force", *European Journal of Mechanics-B/Fluids*, v. 30, n. 3, pp. 311–315, 2011.

[18] NORBERG, C., "Fluctuating lift on a circular cylinder: review and new measurements", *Journal of Fluids and Structures*, v. 17, n. 1, pp. 57–96, 2003.

[19] CHRISTIE, I., GRIFFITHS, D. F., MITCHELL, A. R., et al., "Finite element methods for second order differential equations with significant first derivatives", *International Journal for Numerical Methods in Engineering*, v. 10, n. 6, pp. 1389–1396, 1976.

[20] SHIH, T., TAN, C., HWANG, B., "Effects of grid staggering on numerical schemes", *International Journal for numerical methods in fluids*, v. 9, n. 2, pp. 193–212, 1989.

[21] LIBERZON, A., FELDMAN, Y., GELFGAT, A. Y., "Experimental observation of the steady-oscillatory transition in a cubic lid-driven cavity", *Physics of fluids*, v. 23, n. 8, pp. 084106, 2011.

[22] NALLASAMY, M., PRASAD, K. K., "On cavity flow at high Reynolds numbers", *Journal of Fluid Mechanics*, v. 79, n. 2, pp. 391–414, 1977.

[23] WU, J.-Z., MA, H.-Y., ZHOU, M.-D., *Vorticity and vortex dynamics*. Springer Science & Business Media, 2007.

[24] KHALAK, A., WILLIAMSON, C. H., "Motions, forces and mode transitions in vortex-induced vibrations at low mass-damping", *Journal of fluids and Structures*, v. 13, n. 7-8, pp. 813–851, 1999.

Apêndice A

Códigos Gmsh API

Para instalar a API do Gmsh os seguintes comandos podem ser executados. São comandos em Bash, sendo assim, é recomendado ter um sistema operacional baseado em Linux.

```
pip install --upgrade gmsh
pip install meshio[all]
```

Listing A.1: Instalando dependências do meshio via comandos do terminals

Tendo instalado os módulos podemos escrever a rotina configuração inicial do problema, dizendo quais os pontos de interesse, também podendo espeficificar a formas geométricas variadas.

```
import gmsh
import sys

gmsh.initialize() #Inicializar a API
gmsh.model.add("Cylinder") #Nome do arquivo
```

Listing A.2: Comandos iniciais básicos

Nesta primeira parte apenas iniciamos a API e demos um nome ao aquivo. Vamos utilizar como exemplo a geometria descrita pela imagem a seguir.(Não esquecer de colocar a figura).

Para criar linhas precisamos criar pontos iniciais e finais. O comando para criação de pontos pode ser visto nas linhas 16 a 20 e 22 a 25 (Listing A.4). Os parâmetros de entrada de `gmsh.model.occ.addPoint(x,y,z, mesh)`, sendo "x","y"e "z"as respectivas coordenadas dos pontos e "mesh"sendo o tamanho de

malha desejado nas proximidades desses pontos. Para criar as linhas podemos executar o comando visto nas linhas 27 a 30 (Listing A.4). Os parâmetros iniciais de `gmsh.model.occ.addLine(P1, P2)`, sendo "P1"o ponto inicial da linha e "P2"o ponto final. Para criar um circulo, utilizamos o comando da linha 37 ((Listing A.4). Os parâmetros iniciais são (x,y,z, raio), sendo "x", "y"e "z"as coordenadas do centro do circulo e "raio"sendo o raio do circulo.

```
1
2  #Outside Box
3  x1_b1 = -10
4  x2_b1 = 20
5  y1_b1 = -10
6  y2_b1 = 10
7
8  #Inside Box
9  x1_b2 = -1
10 x2_b2 = 5
11 y1_b2 = -2.5
12 y2_b2 = 2.5
13
14 target_mesh_out = 0.4 #target mesh size
15 target_mesh_in = 0.1
16
17 p1_b1 = gmsh.model.occ.addPoint( x1_b1, y1_b1, 0, target_mesh_out)
18 p2_b1 = gmsh.model.occ.addPoint( x2_b1, y1_b1, 0, target_mesh_out)
19 p3_b1 = gmsh.model.occ.addPoint( x2_b1, y2_b1, 0, target_mesh_out)
20 p4_b1 = gmsh.model.occ.addPoint( x1_b1, y2_b1, 0, target_mesh_out)
21
22 p1_b2 = gmsh.model.occ.addPoint( x1_b2, y1_b2, 0, target_mesh_in)
23 p2_b2 = gmsh.model.occ.addPoint( x2_b2, y1_b2, 0, target_mesh_in)
24 p3_b2 = gmsh.model.occ.addPoint( x2_b2, y2_b2, 0, target_mesh_in)
25 p4_b2 = gmsh.model.occ.addPoint( x1_b2, y2_b2, 0, target_mesh_in)
26
27
28 line1_b1 = gmsh.model.occ.addLine(p1_b1, p2_b1)
29 line2_b1 = gmsh.model.occ.addLine(p2_b1, p3_b1)
30 line3_b1 = gmsh.model.occ.addLine(p3_b1, p4_b1)
31 line4_b1 = gmsh.model.occ.addLine(p4_b1, p1_b1)
32
```

```
line1_b2 = gmsh.model.occ.addLine(p1_b2, p2_b2)
line2_b2 = gmsh.model.occ.addLine(p2_b2, p3_b2)
line3_b2 = gmsh.model.occ.addLine(p3_b2, p4_b2)
line4_b2 = gmsh.model.occ.addLine(p4_b2, p1_b2)
```

Listing A.3: Desenho de geometria da imagem (refereência)

Nesta primeira parte apenas iniciamos a API e demos um nome ao aquivo. Vamos utilizar como exemplo a geometria descrita pela imagem a seguir.(Não esquecer de colocar a figura).

Para criar linhas precisamos criar pontos iniciais e finais. O comando para criação de pontos pode ser visto nas linhas 16 a 20 e 22 a 25 (Listing A.4). Os parâmetros de entrada de `gmsh.model.occ.addPoint(x,y,z, mesh)`, sendo "x","y"e "z"as respectivas coordenadas dos pontos e "mesh"sendo o tamanho de malha desejado nas proximidades desses pontos. Para criar as linhas podemos executar o comando visto nas linhas 27 a 30 (Listing A.4). Os parâmetros iniciais de `gmsh.model.occ.addLine(P1, P2)`, sendo "P1"o ponto inicial da linha e "P2"o ponto final. Para criar um circulo, utilizamos o comando da linha 37 ((Listing A.4). Os parâmetros iniciais são (x,y,z, raio), sendo "x", "y"e "z"as coordenadas do centro do circulo e "raio"sendo o raio do circulo.

```
#Outside Box
x1_b1 = -30
x2_b1 = 30
y1_b1 = -20
y2_b1 = 20

#Inside Box
x1_b2 = -0.75
x2_b2 = 0.75
y1_b2 = -1
y2_b2 = 1

target_mesh_out = 1 #target mesh size
target_mesh_in = 0.01

p1_b1 = gmsh.model.occ.addPoint( x1_b1, y1_b1, 0, target_mesh_out)
p2_b1 = gmsh.model.occ.addPoint( x2_b1, y1_b1, 0, target_mesh_out)
```

```
p3_b1 = gmsh.model.occ.addPoint( x2_b1, y2_b1, 0, target_mesh_out)
p4_b1 = gmsh.model.occ.addPoint( x1_b1, y2_b1, 0, target_mesh_out)

p1_b2 = gmsh.model.occ.addPoint( x1_b2, y1_b2, 0, target_mesh_in)
p2_b2 = gmsh.model.occ.addPoint( x2_b2, y1_b2, 0, target_mesh_in)
p3_b2 = gmsh.model.occ.addPoint( x2_b2, y2_b2, 0, target_mesh_in)
p4_b2 = gmsh.model.occ.addPoint( x1_b2, y2_b2, 0, target_mesh_in)

line1_b1 = gmsh.model.occ.addLine(p1_b1, p2_b1)
line2_b1 = gmsh.model.occ.addLine(p2_b1, p3_b1)
line3_b1 = gmsh.model.occ.addLine(p3_b1, p4_b1)
line4_b1 = gmsh.model.occ.addLine(p4_b1, p1_b1)

line1_b2 = gmsh.model.occ.addLine(p1_b2, p2_b2)
line2_b2 = gmsh.model.occ.addLine(p2_b2, p3_b2)
line3_b2 = gmsh.model.occ.addLine(p3_b2, p4_b2)
line4_b2 = gmsh.model.occ.addLine(p4_b2, p1_b2)

cylinder1 = gmsh.model.occ.addCircle(0, 0, 0, 0.5)
```

Listing A.4: Desenho de geometria da imagem (refereência)

Tendo criado os pontos e linhas precisamos delimitar as superfícies de interesse. Para juntar curvas utilizamos o comando `gmsh.model.occ.addCurveLoop([list_line])`, sendo "list_line"a lista de linhas ou curvas. Para a superfície é necessário especificar quais serão as fronteiras, fazemos isso com o comando `msh.model.occ.addPlaneSurface([list_curveLoop])`, sendo "list_curveLoop"a lista de curvas delimitantes. Ao final executamos o comando `gmsh.model.occ.synchronize()`.

```
cylinder1 = gmsh.model.occ.addCircle(0, 0, 0, 0.5)

curve_loop1 = gmsh.model.occ.addCurveLoop([line1_b1, line2_b1,
                                           line3_b1, line4_b1])

curve_loop2 = gmsh.model.occ.addCurveLoop([line1_b2, line2_b2,
                                           line3_b2, line4_b2])

curve_loop3 = gmsh.model.occ.addCurveLoop([cylinder1])

```

```
surface1 = gmsh.model.occ.addPlaneSurface([curve_loop1,
                                           curve_loop2])

surface2 = gmsh.model.occ.addPlaneSurface([curve_loop2,
                                           curve_loop3])

gmsh.model.occ.synchronize()
```

Listing A.5: Desenho de geometria da imagem (refereência)

Tendo já desenhado a geometria completa do problema podemos dar nomes as linhas e superfícies de interesse. Fazemos isso com os comandos `gmsh.model.addPhysicalGroup (dim,[list_curve])` e `gmsh.model.setPhysicalName(dim, tag, Name)`, sendo "dim" a dimensão espacial do grupo, 1 para linhas e 2 para superfícies, "tag" sendo utilizado para fazer a referência ao grupo criado e "Name"o nome que queremos no grupo criado.

```
bot_boudary = gmsh.model.addPhysicalGroup(   1,    [line1_b1])
outlet_boudary = gmsh.model.addPhysicalGroup(1, [line2_b1])
top_boudary = gmsh.model.addPhysicalGroup(   1,    [line3_b1])
inlet_boudary = gmsh.model.addPhysicalGroup( 1,  [line4_b1])
cylinder_boudary = gmsh.model.addPhysicalGroup( 1,  [cylinder1])
inside_mesh_line = gmsh.model.addPhysicalGroup( 1,  [line1_b2,
   line2_b2,
                                                     line3_b2,
   line4_b2])

outside_mesh = gmsh.model.addPhysicalGroup(2, [surface1])
inside_mesh = gmsh.model.addPhysicalGroup(2, [surface2])

gmsh.model.setPhysicalName(1, bot_boudary, "Bottom")
gmsh.model.setPhysicalName(1, outlet_boudary, "Outlet")
gmsh.model.setPhysicalName(1, top_boudary, "Top")
gmsh.model.setPhysicalName(1, inlet_boudary, "Inlet")
gmsh.model.setPhysicalName(1, cylinder_boudary, "Cylinder_boudary")
gmsh.model.setPhysicalName(1, inside_mesh_line, "Inside_mesh_line")
gmsh.model.setPhysicalName(2, outside_mesh, "Outside_mesh")
gmsh.model.setPhysicalName(2, inside_mesh, "Inside_mesh")
```

Listing A.6: Desenho de geometria da imagem (refereência)

Para salvar o malha basta executar o seguinte comando.

```
gmsh.write("cylinder_test.msh")
```

Listing A.7: Salvar a malha gerada

Apêndice B

Importação de malha

Para fazer a importação da malha precisamos instalar também o módulo *meshio*. Com ele podemos trabalhar com maior facilidade com os arquivos gerados pela API do Gmsh. Para a instalação basta seguir o comando Bash a seguir, é altamente recomendado ter um sistema operacional baseado em Linux.

```
pip install meshio[all]
```

Listing B.1: Instalando dependências do meshio via comandos do terminal

Para ler o arquivo e organizar as informações, executamos os seguintes comandos.

```
msh = meshio.read("cylinder_test.msh")
IEN = msh.cells_dict["triangle"]
cells = [('triangle',IEN)]
cc_all = msh.cell_data_dict["gmsh:physical"]["line"]
```

Listing B.2: Instalando dependências do meshio via comandos do terminal

Apêndice C

Códigos computação paralela

C.1 Montagem das matrizes globais

Para fazer a montagem das matrizes globais precisamos colocar os valores das matrizes locais triangulares no lugar. A matriz global e uma matriz $n \times n$, sendo n o número total de nós do espaço discretizado. No código IEN é uma matriz $nx3$, composta por todos os elementos triangulares, Em cada linha estão os identificadores de cada nó.

$$\begin{bmatrix} & \vdots & \\ 7 & 8 & 9 \\ 10 & 11 & 12 \\ & \vdots & \end{bmatrix} \tag{C.1}$$

Sabendo que cada elemento da matriz local $a_{(i,j)} \rightarrow A_{(IEN[k,i],IEN[k,j])}$ sendo a a matriz local e A a matriz global, podemos implementar isso no nosso código. A seguir apresentamos como seria o laço para construir essas matrizes.

```
class Matrix_Global:
    def __init__(X, Y, IEN):
        self.X = X
        self.Y = Y
        self.IEN = IEN
    def matrix_mouting(self, inicial, final):
        M = lil_matrix((self.npoints,self.npoints), dtype='float')
        for e in range(inicial, final):
            v = self.IEN[e]
```

```
            # area do elemento
            det = self.X[v[2]]*( self.Y[v[0]]-self.Y[v[1]]) \
                  + self.X[v[0]]*( self.Y[v[1]]-self.Y[v[2]]) \
                  + self.X[v[1]]*(-self.Y[v[0]]+self.Y[v[2]])
            area = det/2.0
            m = (area/12.0) * np.array([ [2.0, 1.0, 1.0],
                                         [1.0, 2.0, 1.0],
                                         [1.0, 1.0, 2.0] ])
            for i in range(0,3):
                ii = self.IEN[e,i]
                for j in range(0,3):
                    jj = self.IEN[e,j]
                    # montagem (assembling) das matrizes K e M
                    M[ii,jj] = M[ii,jj] +  m[i,j]
        return M
```

Listing C.1: Salvar a malha gerada

Esse algoritmo pode ser executado serialmente ou paralelamente. Como as matrizes são criadas a partir de uma lista de elementos, podemos dividir essa lista pelo número de processadores disponíveis na máquina para agilizar o processo. Essa abordagem é conveniente para a otimização do processo de montagem. No modelo atual o movimento da malha é nulo, logo as matrizes originais não se alteram, sendo apenas necessário sua montagem uma vez. Se a malha tiver algum movimento o algoritmo paralelo se torna ainda mais interessante, visto que precisaríamos montar as matrizes a cada iteração.

C.2 Implementação de rotina com *multiprocessing*

Para poder implementar o método paralelo precisamos importar uma biblioteca chamada `multiprocessing`, ela já vem incluída na versão do python 3.8.10. Apresentaremos um código dando explicações posteriores de cada elemento de programação.

```
import multiprocessing as mp
cpu_cores = mp.cpu_count()
list_cores = []
```

```
tuple_int = ()
ne = len(IEN) #numero de elementos triangulares
Matrices_Object = Matrix_Creation(X, Y, IEN)
for i in range(0, cpu_cores):#Do not repeat this step
    if i < cpu_cores-1:
        tuple_int = ((i * (ne // cpu_cores)),
                          (i+1) * (ne // cpu_cores))
        list_cores.append(tuple_int)
    else:
        tuple_int = (i * (ne // cpu_cores) ,
            (i+1) * (ne // cpu_cores) + ne%cpu_cores)
        list_cores.append(tuple_int)
def MatricesInParallel():
    with mp.Pool(cpu_cores) as p:
        a = p.starmap(Matrices_Global.matrix_mouting, list_cores)
    return a
a = MatricesInParallel()
M    = lil_matrix( (npoints,npoints), dtype='float')
for i in a:
    M    = np.add(M,i)
```

Listing C.2: Salvar a malha gerada

`mp.cpu_count()`: Esse comando retorna o numero de núcleos do processador
`list_cores`: É uma lista composta por tuplas
`tuple_int`: É uma tupla composta pelas entradas da função a ser paralelizada
`Matrices_Object`: É a classe que contem os parâmetros para criar as matrizes
`mp.Pool(cpu_cores)`: Esse comando cria um local onde o existe um numero definido de trabalhadores capazes de realizar tarefas em paralelo. O numero máximo de trabalhadores que podem trabalhar em paralelo é limitado pelo numero de núcleos.
`p.starmap(function(), entrys)`: Esse comando retorna uma lista com o resultado da função `function()`, com entradas `entrys`.

Como dito anteriormente dividimos a lista de elementos pelo número de processadores. É de suma importância dizer que nesse algoritmo trabalhamos com cada processo tendo seu endereço de memória separado. Isto é, para cada divisão da lista de elementos, obteremos uma matriz esparsa diferente, sendo obrigatório, no final, juntar todas as partes. O algoritmo seria mais rápido se o endereço de memória

fosse compartilhado, não sendo preciso juntar as matrizes no final.