OpenFOAM

Anderson de Moura Ribeiro

Programa de Pós-Graduação em
Modelagem Computacional
PPGMC

Programa de Pós-Graduação em
Modelagem Computacional

Link para simulações e arquivos complementares

https://drive.google.com/drive/folders/1nBcm6lJhc64HNRreSUGY3SFt6yApWRtJ?usp=sharing

Palestrante

➢ Formação / Experiências:

- Engenheiro Mecânico (UFSJ / 2021)
- Mestre em modelagem computacional (PPGMC – UFJF / 2023)
  FAPEMIG

  Otimização multiobjetivo do layout de parques eólicos offshore via algoritmos evolucionários utilizando um modelo analítico de esteira de turbulência

- Doutorando em modelagem computacional (PPGMC – UFJF)
  Shell Brasil Petróleo LTDA

  Quantificação de incertezas, análise de sensibilidade e métodos numéricos (FEM/FVM) aplicados à recuperação avançada de petróleo

Palestrante

➢ Formação / Experiências:

- Engenheiro Mecânico (UFSJ / 2021)
- Mestre em modelagem computacional (PPGMC – UFJF / 2023)
  FAPEMIG

  Otimização multiobjetivo do layout de parques eólicos offshore via algoritmos evolucionários utilizando um modelo analítico de esteira de turbulência

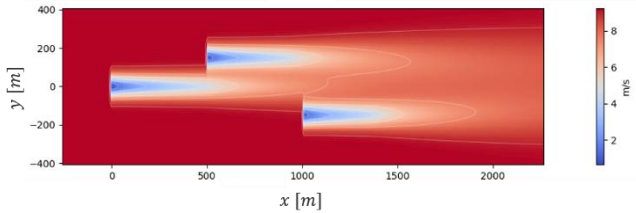


Figura 1 - Efeito de esteira de turbulência com 3 aerogeradores de mesmo modelo

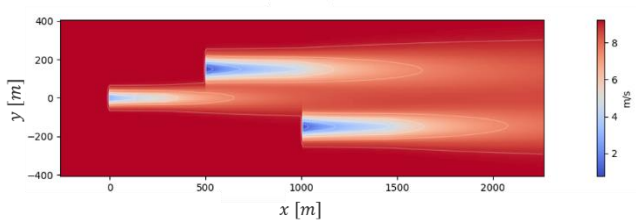
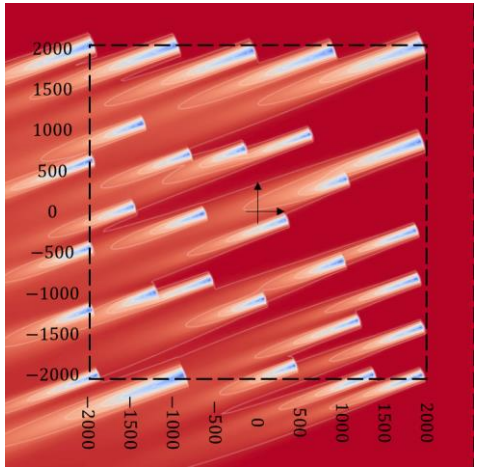


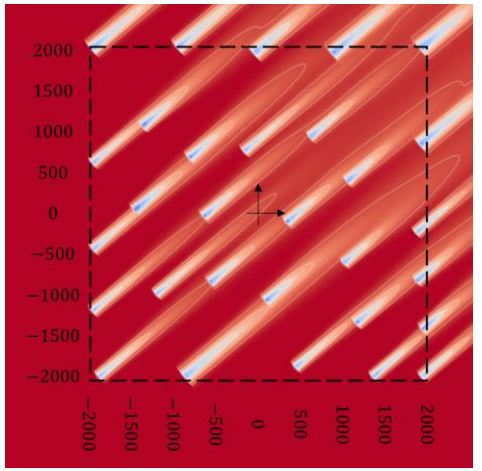
Figura 2 - Efeito de esteira com 1º aerogerador de modelo diferente

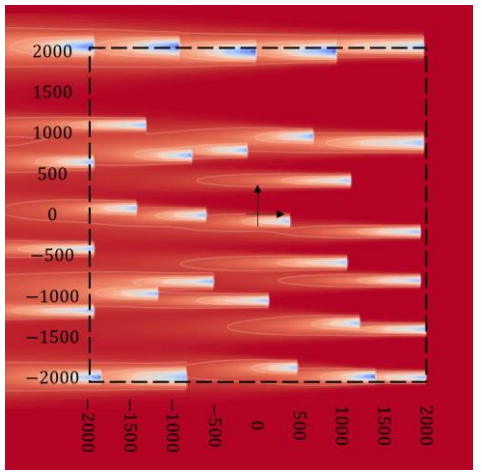
- Doutorando em modelagem computacional (PPGMC – UFJF)
  Shell Brasil Petróleo LTDA

  Quantificação de incertezas, análise de sensibilidade e métodos numéricos (FEM/FVM) aplicados à recuperação avançada de petróleo

Palestrante







➢ Formação / Experiências:

- Engenheiro Mecânico (UFSJ / 2021)
- Mestre em modelagem computacional (PPGMC – UFJF / 2023)
  FAPEMIG

  Otimização multiobjetivo do layout de parques eólicos offshore via algoritmos evolucionários utilizando um modelo analítico de esteira de turbulência

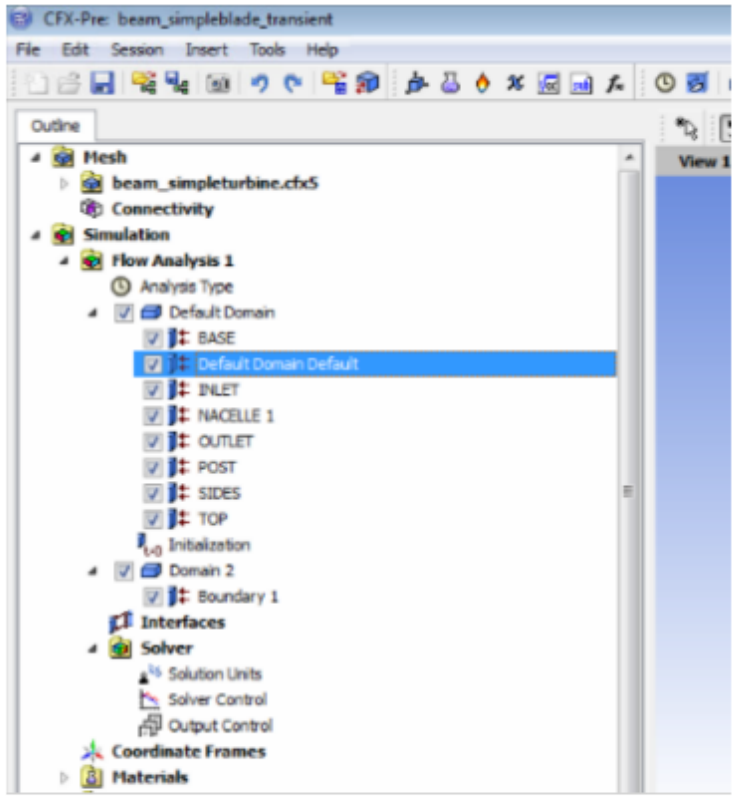
- Doutorando em modelagem computacional (PPGMC – UFJF)
  Shell Brasil Petróleo LTDA

  Quantificação de incertezas, análise de sensibilidade e métodos numéricos (FEM/FVM) aplicados à recuperação avançada de petróleo

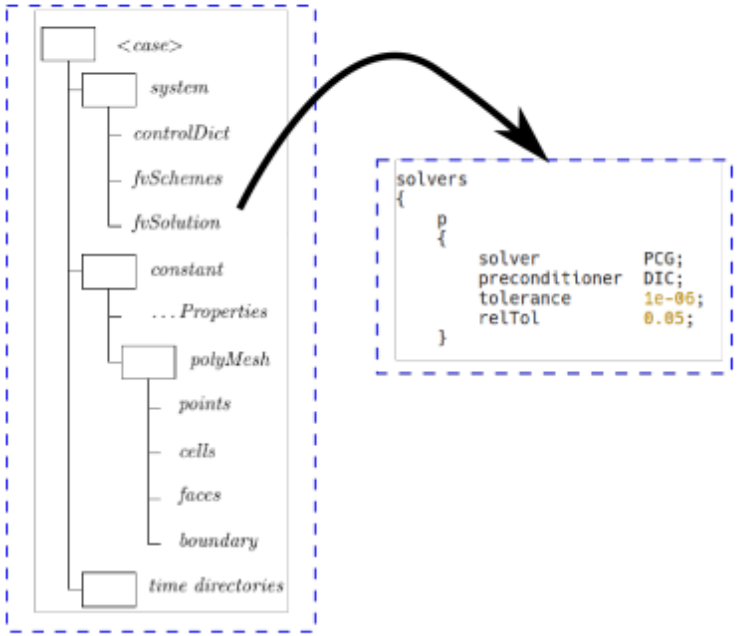
Agenda

- PARTE I
  - Contextualização & Motivação para CFD
  - Apresentação OpenFOAM
  - Fundamentos modelagem de dinâmica dos fluidos e volumes finitos
  - Exemplos OpenFOAM
- PARTE II
  - Aplicação – simulação Vórtice de von Kármán
  - Exercício – simulação tubulação
- PARTE III - Extra
  - Introdução à simulação de turbulência
  - Exercício – simulação aerofólio de turbina eólica

Apresentação OpenFOAM



Ansys CFX



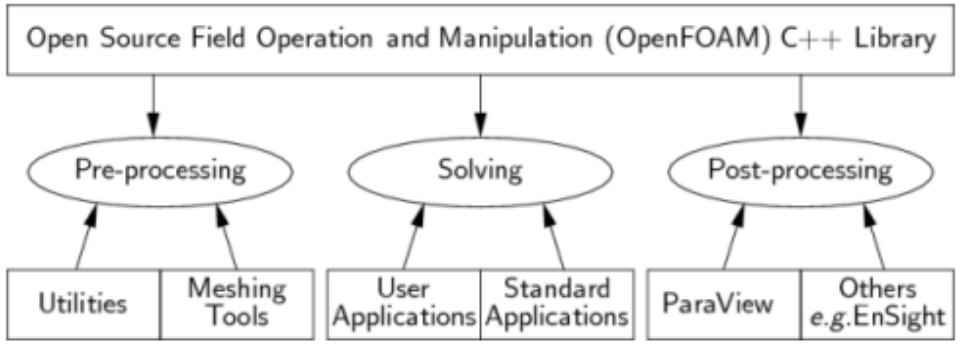
OpenFOAM

https://grupos.moodle.ufsc.br/pluginfile.php/1346781/mod_resource/content/10/Aula00_handout.pdf

Apresentação OpenFOAM

OpenFOAM = **Open** Source **F**ield **O**peration **a**nd **M**anipulation

OpenFOAM é um conjunto de ferramentas para CFD escrito em C++ que contém geradores de malhas, solvers e utilitários de pós-processamento.



https://grupos.moodle.ufsc.br/pluginfile.php/1346781/mod_resource/content/10/Aula00_handout.pdf

Apresentação OpenFOAM

- Criado inicialmente em **1989** por *Henry Weller*[a], no Imperial College com o nome FOAM;
- recebeu contribuições relevantes de *Hrvoje Jasak* durante seu Phd (**1993-1996**);
- em **2000**, *Weller* e *Jasak* criam a empresa *Nabla Ltd* com o intuito de comercializar o FOAM;
- em **2004** a *Nabla Ltd* declara falência, *Weller* juntamente com Chris Greenshields e Mattijs Janssens fundam a OpenCFD Ltd liberando o agora renomeado OpenFOAM como software código aberto sobre a licença GPL.;

[a]Sob supervisão do Prof. David Gosman e do Dr. Radd Issa

https://grupos.moodle.ufsc.br/pluginfile.php/1346781/mod_resource/content/10/Aula00_handout.pdf

Apresentação OpenFOAM

- em **2011**, a OpenCFD é adquirida pela Silicon Graphics International (SGI) e simultaneamente é criado a *OpenFOAM Foundation* sendo para esta transferida o copyright do OpenFOAM;[a]
- em **2012**, o grupo ESI adquire a OpenCFD da SGI;
- em **2014**, *Weller* e *Greenshields* deixam a ESI e continuam o desenvolvimento do OpenFOAM através de uma nova empresa, CFD Direct.

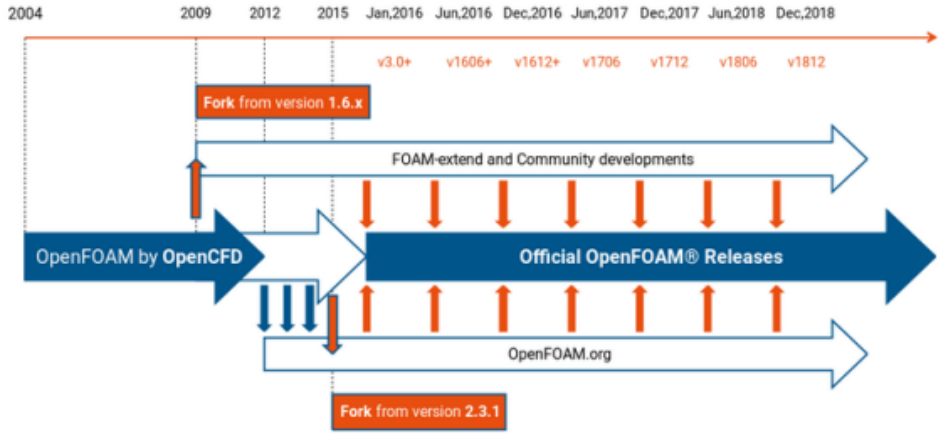
[a] Apesar disso a marca OpenFOAM continua sendo uma propriedade da OpenCFD.

https://grupos.moodle.ufsc.br/pluginfile.php/1346781/mod_resource/content/10/Aula00_handout.pdf

Apresentação OpenFOAM

Observações

- `www.openfoam.com` é a página oficial da empresa OpenCFD;
- `www.openfoam.org` é a página oficial da Fundação OpenFOAM;



https://grupos.moodle.ufsc.br/pluginfile.php/1346781/mod_resource/content/10/Aula00_handout.pdf

Apresentação OpenFOAM

OpenFOAM and The OpenFOAM Foundation

OpenFOAM is free, open source software for CFD from the OpenFOAM Foundation.

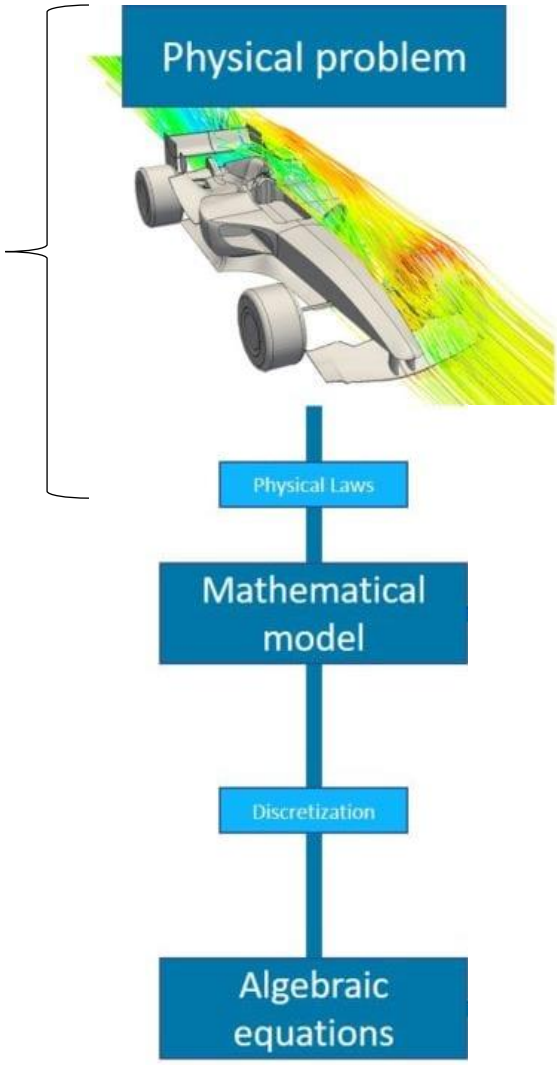
Apresentação OpenFOAM

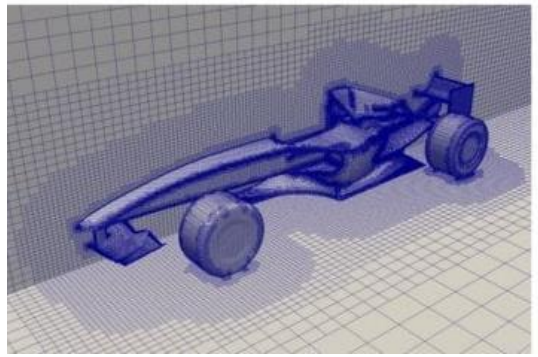
Versão do OpenFOAM utilizada neste minicurso: https://openfoam.org/download/9-linux/

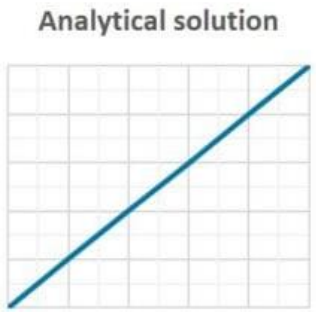
Contextualização & Motivação para CFD

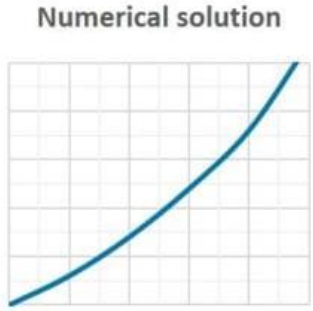
Físicas envolvidas:

- Dinâmica dos fluidos
- Transferência de calor
- Escoamentos multifásicos
- Reações químicas
- Etc.









Contextualização & Motivação para CFD

1.Aerospace: CFD is crucial for designing and optimizing aircraft and spacecraft, improving aerodynamics, and enhancing safety.

2.Automotive: It's used to improve vehicle aerodynamics, engine performance, and safety in the automotive industry.

3.Energy: CFD helps design wind turbines, study combustion processes, and optimize nuclear reactor cooling systems.

4.Environmental Engineering: It aids in modeling air and water quality and simulating the dispersion of pollutants.

5.HVAC (Heating, Ventilation, and Air Conditioning): CFD is vital for efficient HVAC system design.

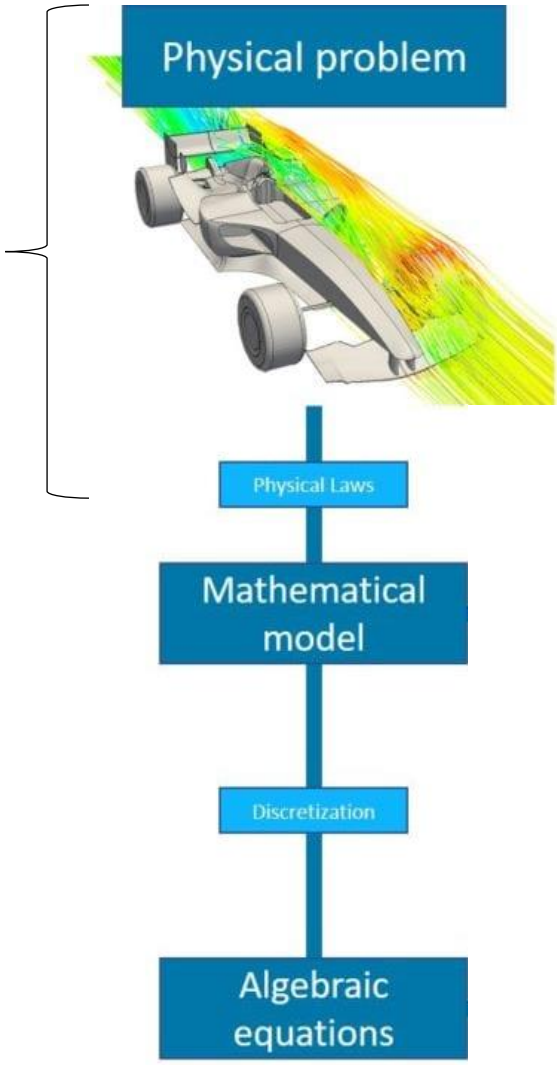
6.Chemical and Process Engineering: It optimizes chemical reactors, mixing processes, and heat exchangers.

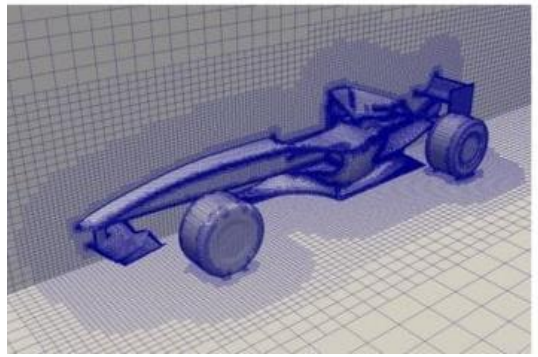
7.Biomedical Engineering: CFD is used for modeling blood flow, respiratory processes, and medical device optimization in the biomedical field.

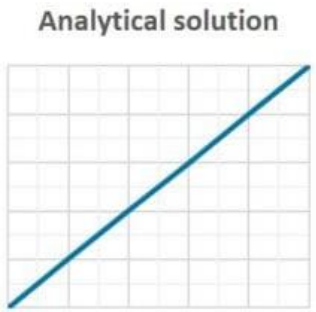
Contextualização & Motivação para CFD

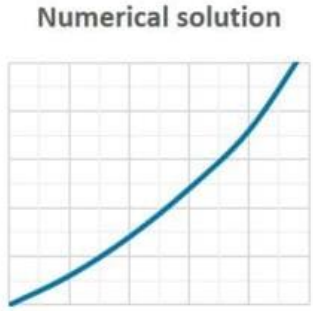
Físicas envolvidas:

- Dinâmica dos fluidos
- Transferência de calor
- Escoamentos multifásicos
- Reações químicas
- Etc.





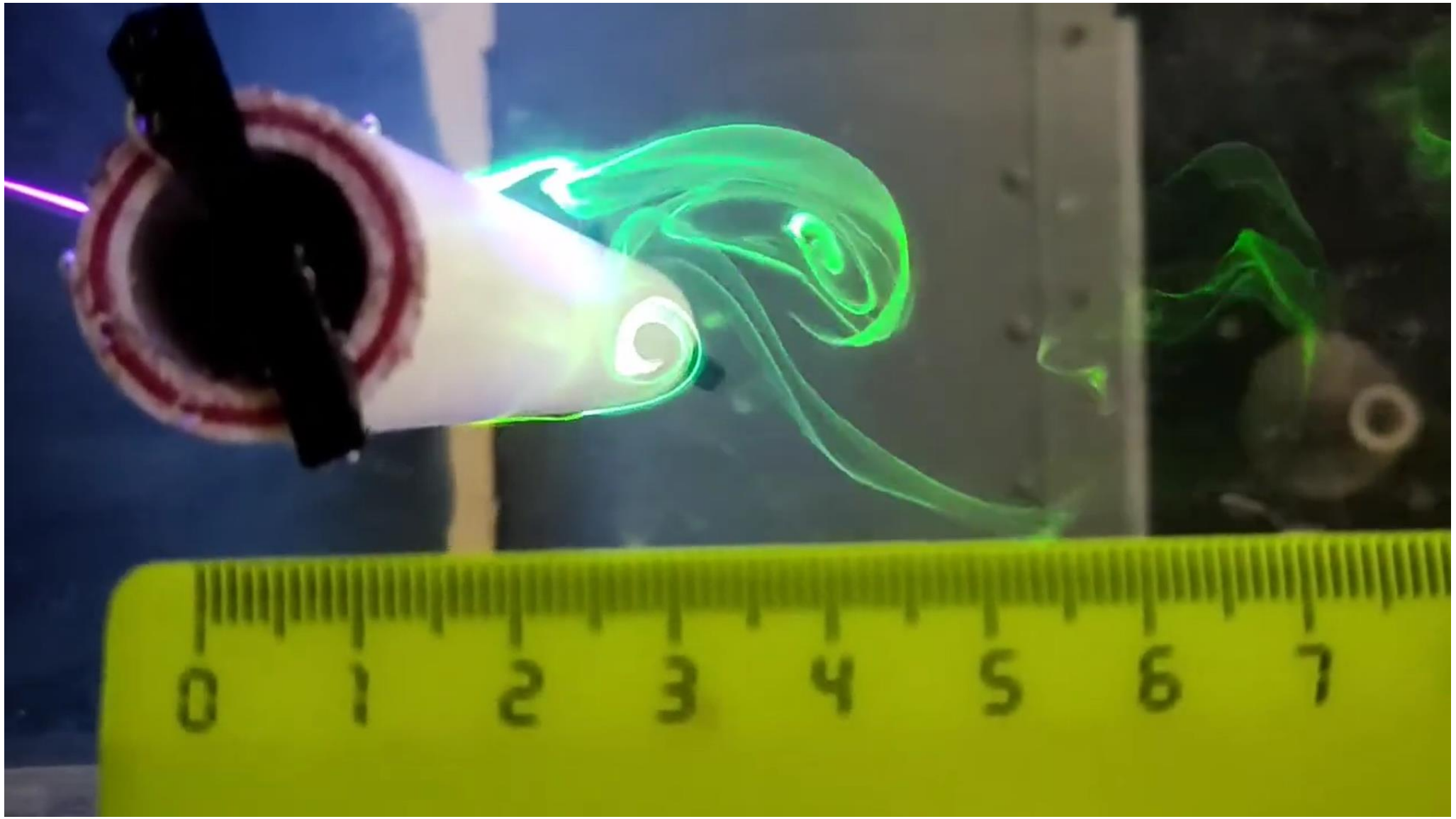




Contextualização & Motivação para CFD

https://www.youtube.com/watch?v=SawKLWT1bDA

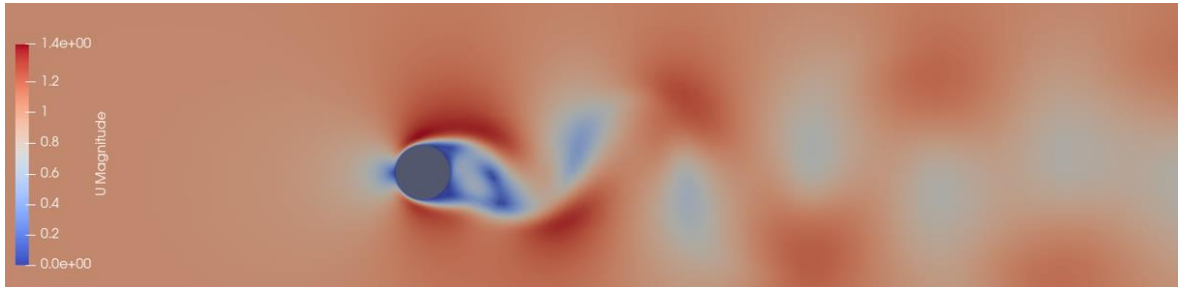
Contextualização & Motivação para CFD



https://www.youtube.com/watch?v=2LUpHjuRZbw

Agenda

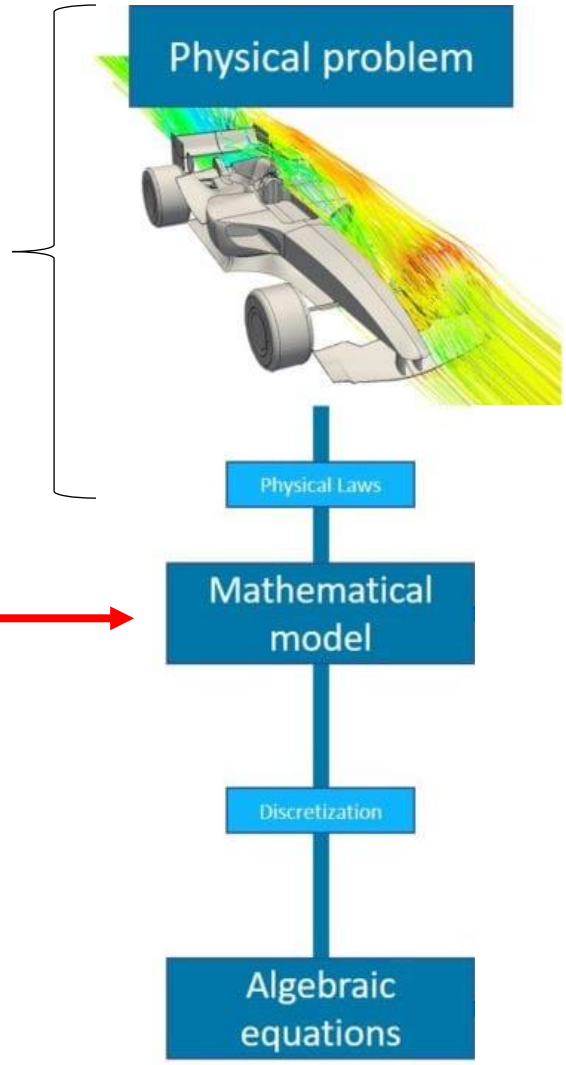
- PARTE I
  - Contextualização & Motivação para CFD
  - Apresentação OpenFOAM
  - Fundamentos modelagem de dinâmica dos fluidos e volumes finitos
  - Exemplos OpenFOAM
- PARTE II
  - Aplicação – simulação Vórtice de von Kármán
  - Exercício – simulação tubulação
- PARTE III - Extra
  - Introdução à simulação de turbulência
  - Exercício – simulação aerofólio de turbina eólica

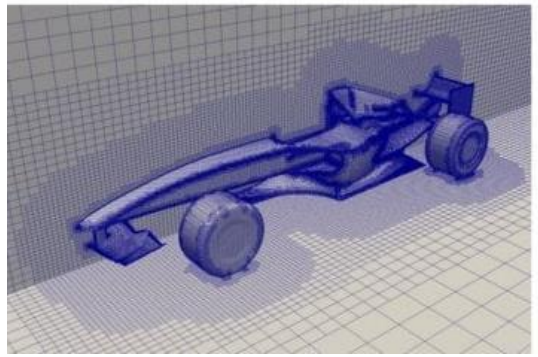


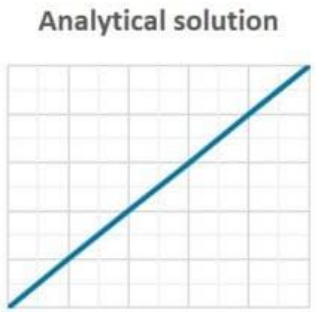
Contextualização & Motivação para CFD

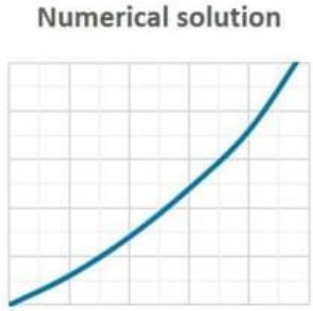
Físicas envolvidas:

- Dinâmica dos fluidos
- Transferência de calor
- Escoamentos multifásicos
- Reações químicas
- Etc.









Contextualização & Motivação para CFD

The Rise of the Navier-Stokes Equations

The central mathematical description for all theoretical fluid dynamics models is given by the Navier-Stokes equations, which describe the motion of viscous fluid domains. The history of their discovery is quite interesting.

It is a bizarre coincidence that the famous equation of Navier-Stokes has been generated by Claude-Louis Navier (1785-1836) and Sir George Gabriel Stokes (1819-1903), who had never met. At first, Claude-Louis Navier conducted studies on a partial section of equations up until 1822. Later, Sir George Gabriel Stokes adjusted and finalized the equations in 1845[9].

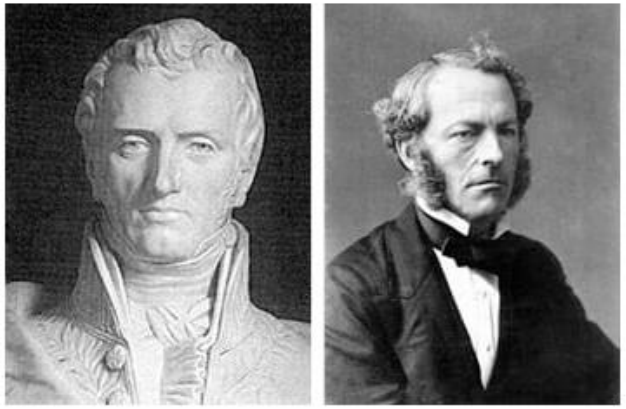


Figure 2: Claude-Louis Navier[14] (left) and Sir George Gabriel Stokes[15] (right)

Governing Equations of CFD

The main structure of thermo-fluids examination is directed by governing equations that are based on the conservation law of fluid's physical properties. The basic equations are the three laws of conservation[10,11]:

1. **Conservation of Mass: Continuity Equation**
2. **Conservation of Momentum: Newton's Second Law**
3. **Conservation of Energy: First Law of Thermodynamics or Energy Equation**

These principles state that mass, momentum, and energy are stable constants within a closed system. In short, everything must be conserved.

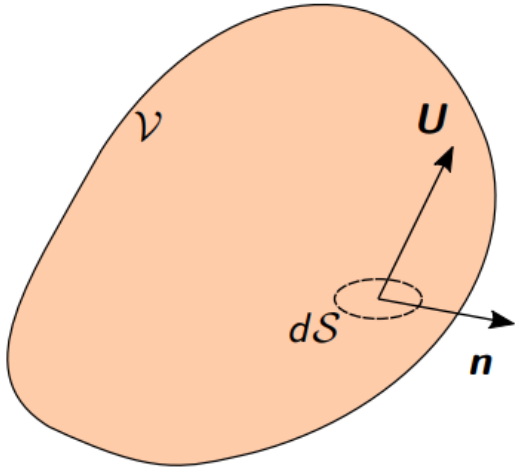
https://www.simscale.com/docs/simwiki/cfd-computational-fluid-dynamics/what-is-cfd-computational-fluid-dynamics/

Equações de governo

Conservação de massa

- Na ausência de fontes de massa ou de locais pelos quais a massa possa desaparecer (sorvedouros), toda a massa que entra no sistema deve sair e/ou se acumular no sistema

Variação temporal da quantidade de massa no elemento	=	Descarga resultante através das fronteiras do elemento



Let us recall the Gauss or Divergence theorem,

$$\int_V \nabla \cdot \mathbf{a} dV = \oint_{\partial V} d\mathbf{S} \cdot \mathbf{a}$$

$$\int_{\mathcal{V}} \frac{\partial \rho}{\partial t} d\mathcal{V} = -\int_{\mathcal{V}} \nabla \cdot (\rho \mathbf{V}) d\mathcal{V} \iff \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{V}) = 0$$

Equações de governo

Conservação de momento linear

- Aplicação da segunda lei de Newton: Força = massa × aceleração

Taxa de variação temporal do momento de uma partícula	=	Resultantes das forças que agem sobre essa partícula

- Elemento de fluido que se desloca com o escoamento
- A massa é constante $\delta m = \rho(\delta x)(\delta y)$, que no limite $\delta x, \delta y \to 0$, $dm = \rho dx dy$

Equações de governo

- Considera-se inicialmente a direção x do escoamento, componente da velocidade é $u = u(x, y, t)$
- A variação da velocidade entre dois pontos do escoamento:

$$\delta u = \frac{\partial u}{\partial x}\delta x + \frac{\partial u}{\partial y}\delta y + \frac{\partial u}{\partial t}\delta t$$

- Dividindo-se a expressão anterior por δt:

$$\frac{\delta u}{\delta t} = \frac{\partial u}{\partial x}\frac{\delta x}{\delta t} + \frac{\partial u}{\partial y}\frac{\delta y}{\delta t} + \frac{\partial u}{\partial t}$$

- No limite, obtém-se a aceleração do elemento de fluido na direção x:

$$\frac{Du}{Dt} = \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}$$

- A aceleração do elemento de fluido é a derivada substantiva de u:

$$\frac{Du}{Dt} = \frac{\partial u}{\partial t} + (\mathbf{V} \cdot \nabla)u,$$

  em que $\mathbf{V} = (u, v)$.

- Pela segunda lei de Newton, a componente da força resultante $\mathbf{F}$ na direção x que age sobre o elemento fluido:

$$(\rho dxdy)\frac{Du}{Dt} = F_x$$

Equações de governo

Forças que agem sobre o fluido

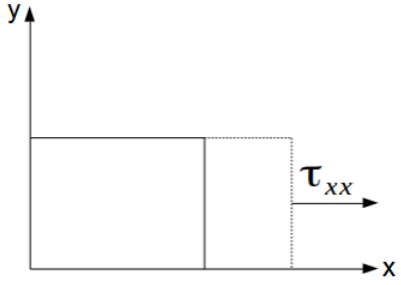
- Forças de campo: agem sobre a massa de fluido como um todo, isto é, sobre cada ponto de um elemento de fluido
  - Exemplos dessas forças: gravidade, eletromagnética
  - Tem a forma geral $\rho \mathbf{f} dxdy$, em que $\mathbf{f}$ é um vetor que representa a força exercida no elemento de fluido por unidade de massa (aceleração)
  - Adicionadas como termos auxiliares (fontes) das equações de momento
- **Forças de superfície**: agem sobre a superfície do elemento de fluido
  - Decorrem da pressão exercida sobre o fluido por um elemento exterior
  - Decorrem também das tensões viscosas normais e de cisalhamento devido ao atrito com os elementos de fluido adjacentes em movimento
  - Intrínsecas ao fluido, elas aparecem como *termos constitutivos* das equações de momento

Equações de governo

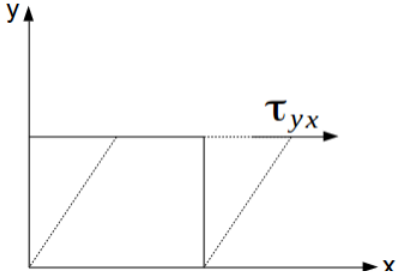
Nomenclatura das tensões viscosas τ_{ij}

- Tensão τ_{ij}: os índices i e j indicam que a tensão age na direção j sobre a superfície normal à direção i
- Exemplo: tensão τ_{yx} age na direção x sobre a superfície normal à direção y
- As tensões na direção positiva dos eixos x e y terão sinal positivo; caso contrário, serão negativas

- Tensões normais tendem a esticar ou comprimir o elemento de fluido
  - Proporcionais à variação temporal do volume do elemento

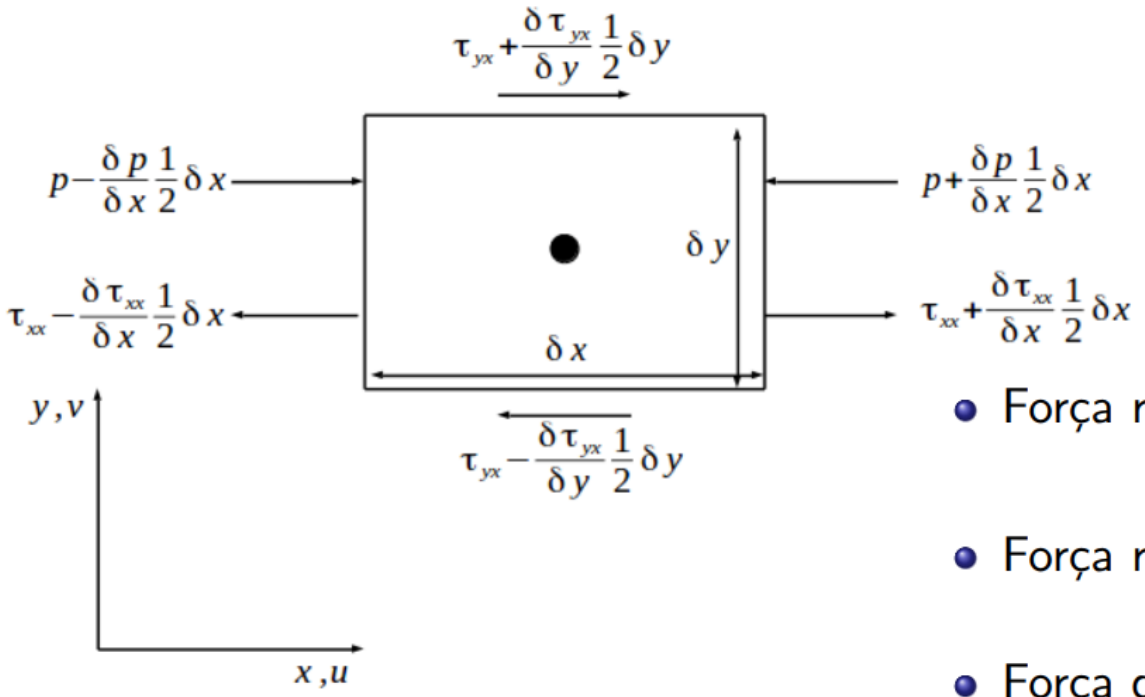


- Tensões de cisalhamento: tendem a deformar o elemento, sendo proporcionais sua taxa de deformação



Equações de governo

- Tensões normais e tangenciais na direção x
- Tensões na direção positiva dos eixos x e y terão sinal positivo; caso contrário, serão negativas



- Força resultante nas faces esquerda e direita:
$$\left(-\frac{\delta p}{\delta x}+\frac{\delta\tau_{xx}}{\delta x}\right)\delta x\delta y$$
- Força resultante nas faces superior e inferior:
$$\frac{\delta\tau_{yx}}{\delta y}\delta x\delta y$$
- Força de superfície resultante na direção x é:
$$\left(-\frac{\delta p}{\delta x}+\frac{\delta\tau_{xx}}{\delta x}\right)\delta x\delta y+\frac{\delta\tau_{yx}}{\delta y}\delta x\delta y$$
- No limite $\delta x, \delta y \to 0$, essa força na direção x vale
$$F_x=\left(-\frac{\partial p}{\partial x}+\frac{\partial\tau_{xx}}{\partial x}\right)dxdy+\frac{\partial\tau_{yx}}{\partial y}dxdy$$

Equações de governo

- Escrevendo a segunda lei de Newton na direção x:

$$(\rho dxdy)\frac{Du}{Dt} = \left(-\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x}\right) dxdy + \frac{\partial \tau_{xy}}{\partial y} dxdy + \rho f_x dxdy$$

  - Força de superfície: $\left(-\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x}\right) dxdy + \frac{\partial \tau_{xy}}{\partial y} dxdy$
  - Força de campo: $\rho f_x dxdy$

- Conservação de momento linear na direção x:

$$\boxed{\rho \frac{Du}{Dt} = -\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \rho f_x}$$

Obs.: $\frac{Du}{Dt} = \frac{\partial u}{\partial t} + (\mathbf{V} \cdot \nabla) u$

Equações de governo

Conservation of Momentum which can be referred to as the Navier-Stokes Equation is given by:

$$\overbrace{\frac{\partial}{\partial t}(\rho\vec{v})}^{I} + \overbrace{\nabla\cdot(\rho\vec{v}\vec{v})}^{II} = \overbrace{-\nabla p}^{III} + \overbrace{\nabla\cdot\left(\bar{\bar{\tau}}\right)}^{IV} + \overbrace{\rho\vec{g}}^{V} \qquad (6)$$

where p is static pressure, $\bar{\bar{\tau}}$ is viscous stress tensor and $\rho\vec{g}$ is the gravitational force per unit volume. Here, the roman numerals denote:

I: Local change with time
II: Momentum convection
III: Surface force
IV: Diffusion term
V: Mass force

https://www.simscale.com/docs/simwiki/cfd-computational-fluid-dynamics/what-is-cfd-computational-fluid-dynamics/

$$\rho\frac{Du}{Dt} = -\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \rho f_x$$

$$\frac{Du}{Dt} = \frac{\partial u}{\partial t} + (\mathbf{V}\cdot\nabla)u$$

Equações de governo

Viscous stress tensor $\bar{\bar{\tau}}$ can be specified as below in accordance with Stoke's Hypothesis:

$$\tau_{ij} = \mu \frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} - \frac{2}{3}(\nabla \cdot \vec{v})\delta_{ij} \qquad (7)$$

If the fluid is assumed to be incompressible with constant viscosity coefficient μ the Navier-Stokes equation simplifies to:

$$\rho \frac{D\vec{v}}{Dt} = -\nabla p + \mu \nabla^2 \vec{v} + \rho \vec{g} \qquad (8)$$

Conservação da massa:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{V}) = 0$$

Overview

- Category: **Incompressible**
- transient
- incompressible

Solver: icoFOAM

OpenFOAM

Equations

The solver uses the **PISO** algorithm to solve the continuity equation:

$$\nabla \cdot \mathbf{u} = 0$$

Where:

$\mathbf{u}$ = Velocity

p = Kinematic pressure

and momentum equation:

$$\frac{\partial}{\partial t}(\mathbf{u}) + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) - \nabla \cdot (\nu \nabla \mathbf{u}) = -\nabla p$$

Solvers

basic	potential flow, Laplace and transport equations
combustion	combustion fluid flows
compressible	compressible fluid flows
discreteMethods	molecule movement
DNS	direct numerical simulations
electromagnetics	electrostatics, magnetohydrodynamics
financial	Black-Scholes equation to price commodities
heatTransfer	heat transfers flows
incompressible	incompressible flows
lagrangian	Lagrangian approach to CFD
multiphase	multiphase flows
stressAnalysis	stress analysis in a solid body

https://www.openfoam.com/documentation/guides/latest/doc/openfoam-guide-applications-solvers.html

Capability matrix

Solver	transient	compressible	turbulence	heat-transfer	buoyancy	combustion	multiphase	particles	dynamic mesh	multi-region	fvOptions
boundaryFoam											
buoyantPimpleFoam	✓	✓	✓	✓	✓						✓
buoyantSimpleFoam		✓	✓	✓	✓						✓
chemFoam	✓			✓		✓					
chtMultiRegionFoam	✓	✓	✓	✓	✓					✓	✓
coldEngineFoam	✓	✓	✓	✓		✓			✓		✓
engineFoam	✓	✓	✓	✓		✓			✓		✓
fireFoam	✓	✓	✓	✓	✓	✓				✓	✓
icoFoam	✓										
interFoam	✓		✓				✓		✓		✓
laplacianFoam	✓										✓
pimpleFoam	✓		✓						✓		✓
pisoFoam	✓		✓								✓

Solvers

basic	potential flow, Laplace and transport equations
combustion	combustion fluid flows
compressible	compressible fluid flows
discreteMethods	molecule movement
DNS	direct numerical simulations
electromagnetics	electrostatics, magnetohydrodynamics
financial	Black-Scholes equation to price commodities
heatTransfer	heat transfers flows
incompressible	incompressible flows
lagrangian	Lagrangian approach to CFD
multiphase	multiphase flows
stressAnalysis	stress analysis in a solid body

https://www.openfoam.com/documentation/guides/latest/doc/openfoam-guide-applications-solvers.html

Capability matrix

Solver	transient	compressible	turbulence	heat-transfer	buoyancy	combustion	multiphase	particles	dynamic mesh	multi-region	fvOptions
boundaryFoam											
buoyantPimpleFoam	✓	✓	✓	✓	✓						✓
buoyantSimpleFoam		✓	✓	✓	✓						✓
chemFoam	✓			✓		✓					
chtMultiRegionFoam	✓	✓	✓	✓	✓					✓	✓
coldEngineFoam	✓	✓	✓	✓		✓			✓		✓
engineFoam	✓	✓	✓	✓		✓			✓		✓
fireFoam	✓	✓	✓	✓	✓	✓				✓	✓
icoFoam	✓										
interFoam	✓		✓				✓		✓		✓
laplacianFoam	✓										✓
pimpleFoam	✓		✓						✓		✓
pisoFoam	✓		✓								✓

Solvers

OpenFOAM

scalarTransportFoam

Overview

- Category: **Basic**
- transient
- incompressible

Equations

Evolves a transport equation for the scalar T

$$\frac{\partial}{\partial t}(T) + \nabla \cdot (\mathbf{u}T) - \nabla \cdot (D_T \nabla T) = S_T$$

Input requirements

Mandatory fields:

- `U`: velocity [m/s]
- `T`: scalar [-]

Physical models

- Transport `$FOAM_CASE/constant/transportProperties` requirements:

```
// Diffusion coefficient [m2/s]
DT              4e-05;
```

Solvers

OpenFOAM

scalarTransportFoam

Overview

- Category: **Basic**
- transient
- incompressible

Equations

Evolves a transport equation for the scalar T

$$\frac{\partial}{\partial t}(T) + \nabla \cdot (\mathbf{u}T) - \nabla \cdot (D_T \nabla T) = S_T$$

Input requirements

Mandatory fields:

- U: velocity [m/s]
- T: scalar [-]

Physical models

- Transport `$FOAM_CASE/constant/transportProperties` requirements:

```
// Diffusion coefficient [m2/s]
DT              4e-05;
```

Exemplos

1. → $\frac{\partial T}{\partial t} = D_T \Delta T$

2. → $\frac{\partial T}{\partial t} + \nabla \cdot \boldsymbol{u}T = 0$

3. → $\frac{\partial T}{\partial t} + \nabla \cdot \boldsymbol{u}T = D_T \Delta T$

Solvers

OpenFOAM

scalarTransportFoam

Overview

- Category: **Basic**
- transient
- incompressible

Equations

Evolves a transport equation for the scalar T

$$\frac{\partial}{\partial t}(T) + \nabla \cdot (\mathbf{u}T) - \nabla \cdot (D_T \nabla T) = S_T$$

Input requirements

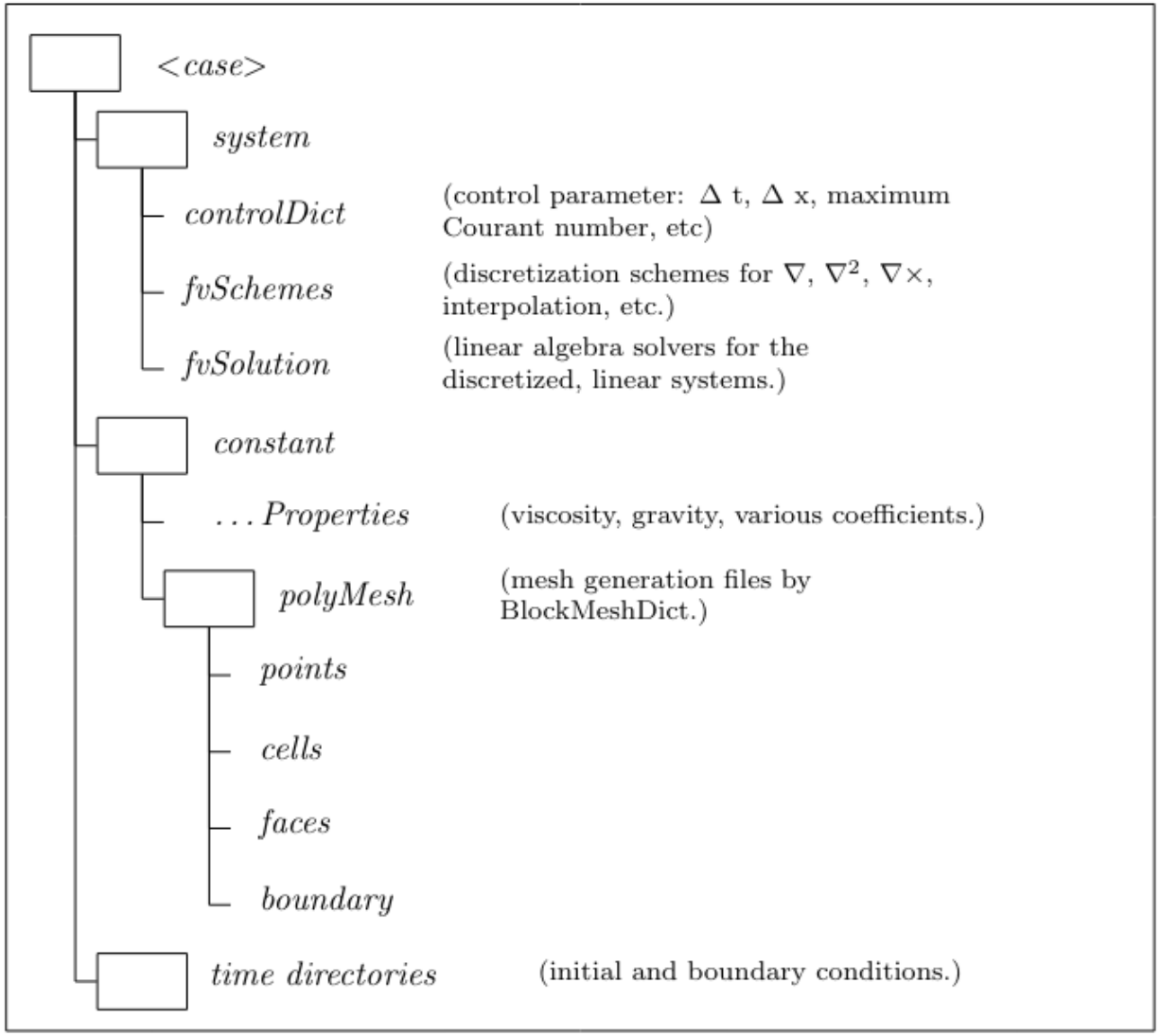
Mandatory fields:

- `U`: velocity [m/s]
- `T`: scalar [-]

Physical models

- Transport `$FOAM_CASE/constant/transportProperties` requirements:

```
// Diffusion coefficient [m2/s]
DT              4e-05;
```



Solvers

OpenFOAM

scalarTransportFoam

Overview

- Category: **Basic**
- transient
- incompressible

Equations

Evolves a transport equation for the scalar T

$$\frac{\partial}{\partial t}(T) + \nabla \cdot (\mathbf{u}T) - \nabla \cdot (D_T \nabla T) = S_T$$

Input requirements

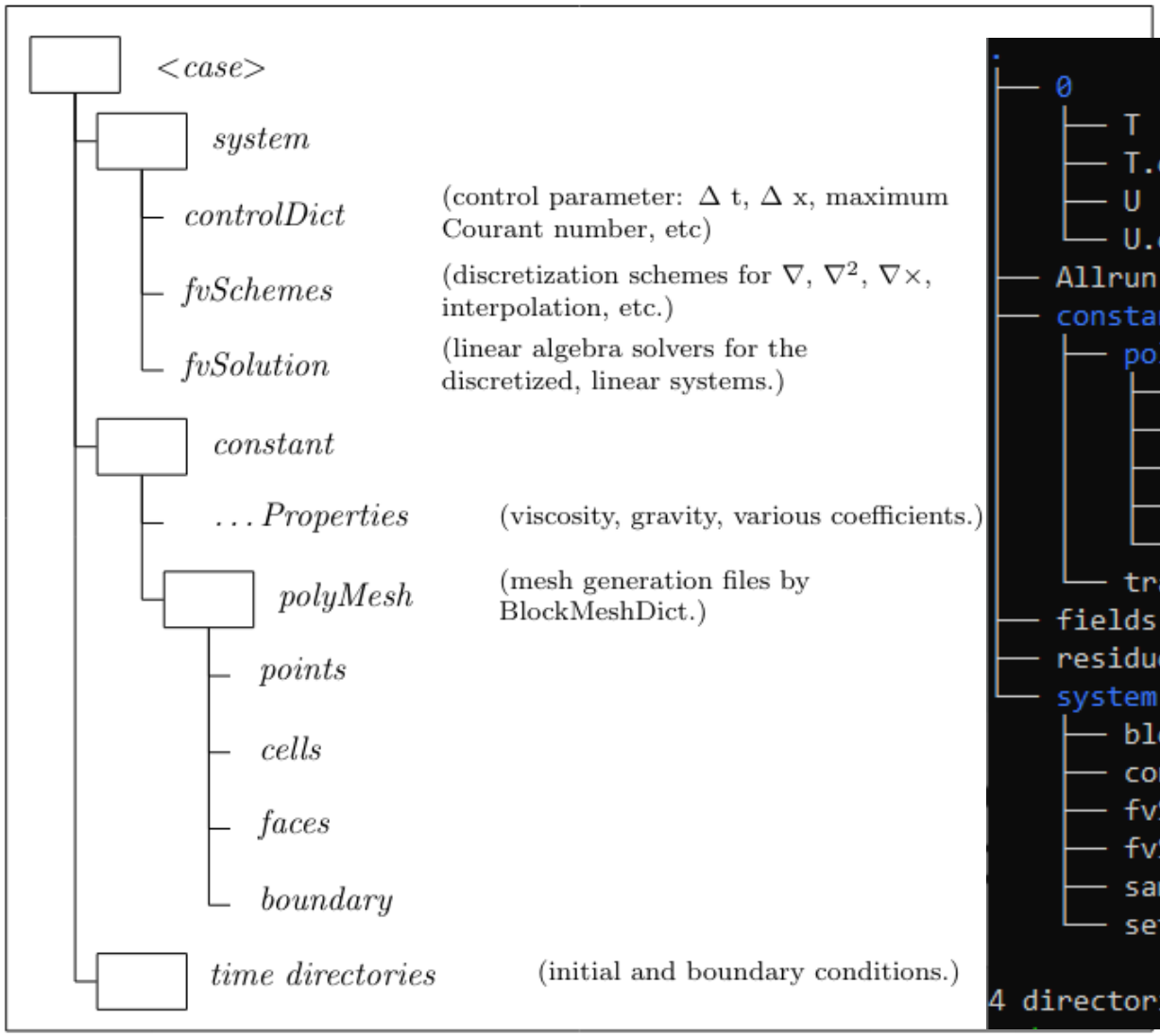
Mandatory fields:

- U: velocity [m/s]
- T: scalar [-]

Physical models

- Transport `$FOAM_CASE/constant/transportProperties` requirements:

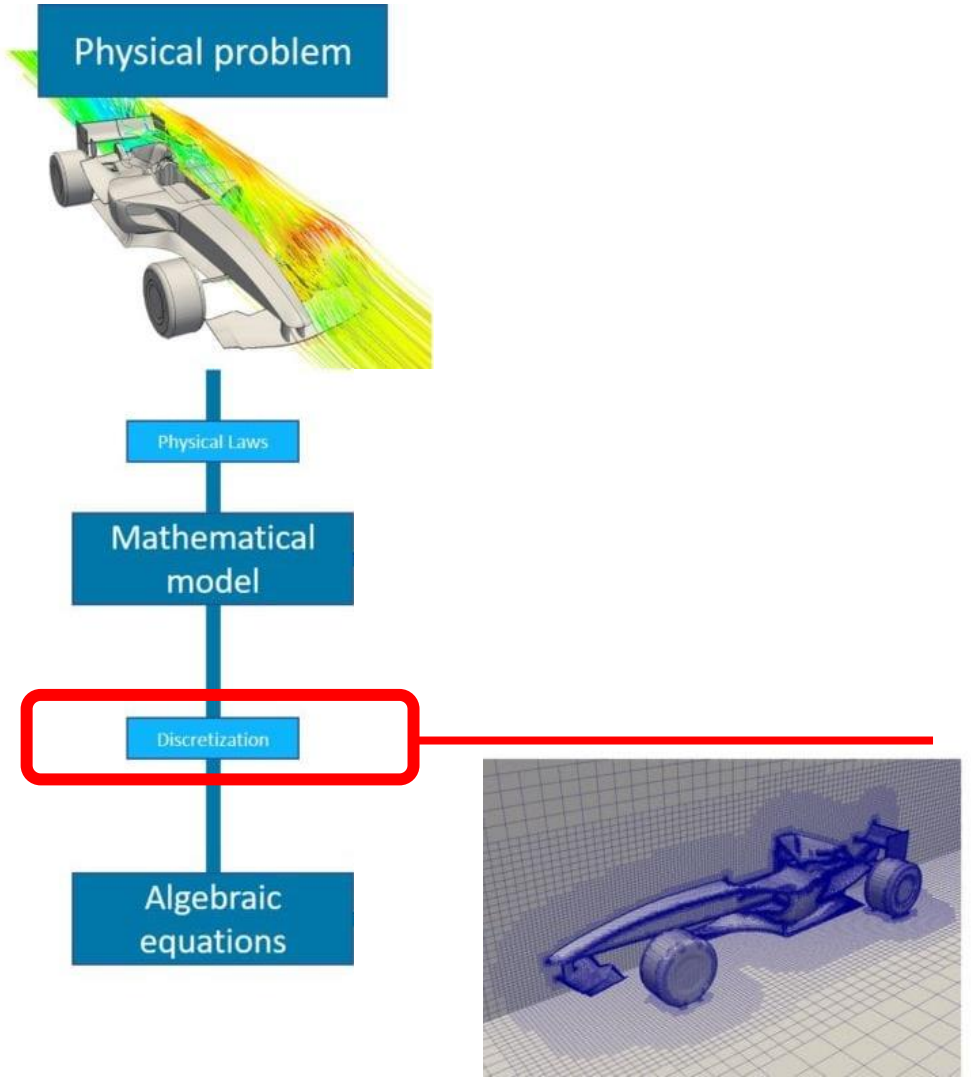
```
// Diffusion coefficient [m2/s]
DT              4e-05;
```

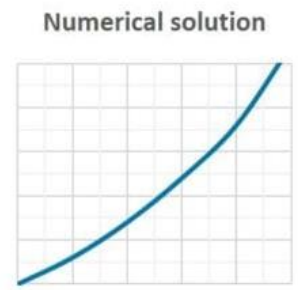



```
.
├── 0
│   ├── T
│   ├── T.orig
│   ├── U
│   └── U.orig
├── Allrun
├── constant
│   ├── polyMesh
│   │   ├── boundary
│   │   ├── faces
│   │   ├── neighbour
│   │   ├── owner
│   │   └── points
│   └── transportProperties
├── fields.foam
├── residuos.py
└── system
    ├── blockMeshDict
    ├── controlDict
    ├── fvSchemes
    ├── fvSolution
    ├── sample
    └── setFieldsDict

4 directories, 19 files
```

Fundamentos do Método dos volumes finitos (*Finite Volume Method – FVM*)





Métodos de discretização:

- Método dos volumes finitos
- Método dos elementos finitos
- Método de diferenças finitas

Método dos volumes finitos

- Let us use the general transport equation as the starting point to explain the FVM,

$$\int_{V_P} \underbrace{\frac{\partial \rho \phi}{\partial t} dV}_{\text{temporal derivative}} + \int_{V_P} \underbrace{\nabla \cdot (\rho \mathbf{u} \phi)\, dV}_{\text{convective term}} - \int_{V_P} \underbrace{\nabla \cdot (\rho \Gamma_\phi \nabla \phi)\, dV}_{\text{diffusion term}} = \int_{V_P} \underbrace{S_\phi(\phi)\, dV}_{\text{source term}}$$

- We want to solve the general transport equation for the transported quantity ϕ in a given domain, with given boundary conditions BC and initial conditions IC.

Método dos volumes finitos

- Hereafter we are going to assume that the discretization practice is at least second order accurate in space and time.
- As consequence of the previous requirement, all dependent variables are assumed to vary linearly around a point P in space and instant t in time,

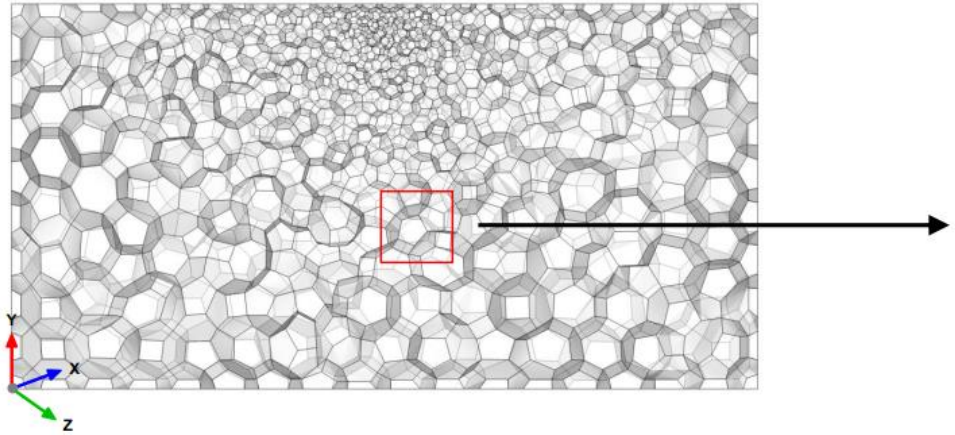
$$\phi(\mathbf{x}) = \phi_P + (\mathbf{x} - \mathbf{x}_P) \cdot (\nabla\phi)_P \qquad \text{where} \qquad \phi_P = \phi(\mathbf{x}_P)$$

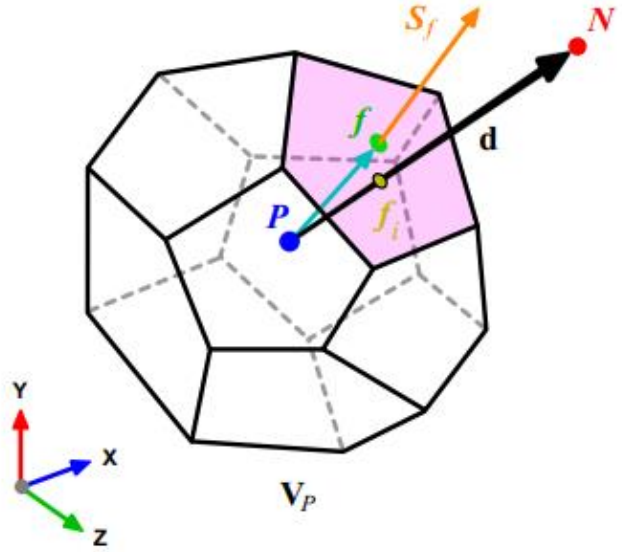
$$\phi(t + \delta t) = \phi^t + \delta t \left(\frac{\partial \phi}{\partial t}\right)^t \qquad \text{where} \qquad \phi^t = \phi(t)$$

Profile assumptions using Taylor expansions around point *P* (in space) and point *t* (in time)

Método dos volumes finitos

- In the FVM, a lot of overhead goes into the data book-keeping of the domain information.
- We know the following information of every control volume V_P in the domain:
  - The control volume V_P has a volume V and is constructed around point P, which is the centroid of the control volume. Therefore the notation V_P.
  - The vector from the centroid P of V_P to the centroid N of V_N is named $\mathbf{d}$.
  - We also know all neighbors V_N of the control volume V_P.
  - The control volume faces are labeled f, which also denotes the face center.
  - The location where the vector $\mathbf{d}$ intersects a face is f_i.
  - The face area vector S_f point outwards from the control volume, is located at the face centroid, is normal to the face and has a magnitude equal to the area of the face.
  - The vector from the centroid P to the face center f is named Pf.





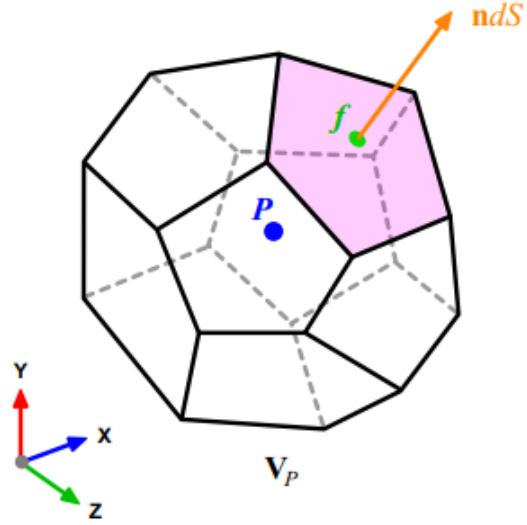
Método dos volumes finitos

- Let us recall the Gauss or Divergence theorem,

$$\int_V \nabla \cdot \mathbf{a} dV = \oint_{\partial V} d\mathbf{S} \cdot \mathbf{a}$$

where ∂V_P is a closed surface bounding the control volume V_P and dS represents an infinitesimal surface element with associated normal $\mathbf{n}$ pointing outwards of the surface ∂V_P, and $\mathbf{n}dS = d\mathbf{S}$

- The Gauss or Divergence theorem simply states that the outward flux of a vector field through a closed surface is equal to the volume integral of the divergence over the region inside the surface.
- This theorem is fundamental in the FVM, it is used to convert the volume integrals appearing in the governing equations into surface integrals.



Método dos volumes finitos

- Let us use the Gauss theorem to convert the volume integrals into surface integrals,

$$\int_{V_P} \underbrace{\frac{\partial \rho\phi}{\partial t} dV}_{\text{temporal derivative}} + \int_{V_P} \underbrace{\nabla \cdot (\rho \mathbf{u}\phi)\, dV}_{\text{convective term}} - \int_{V_P} \underbrace{\nabla \cdot (\rho \Gamma_\phi \nabla \phi)\, dV}_{\text{diffusion term}} = \int_{V_P} \underbrace{S_\phi(\phi)\, dV}_{\text{source term}}$$

$$\int_V \nabla \cdot \mathbf{a} dV = \oint_{\partial V} d\mathbf{S} \cdot \mathbf{a}$$

$$\frac{\partial}{\partial t} \int_{V_P} (\rho\phi)\, dV + \oint_{\partial V_P} \underbrace{d\mathbf{S} \cdot (\rho \mathbf{u}\phi)}_{\text{convective flux}} - \oint_{\partial V_P} \underbrace{d\mathbf{S} \cdot (\rho \Gamma_\phi \nabla \phi)}_{\text{diffusive flux}} = \int_{V_P} S_\phi(\phi)\, dV$$

- At this point the problem reduces to interpolating somehow the cell centered values (known quantities) to the face centers.

Método dos volumes finitos

- Integrating in space each term of the general transport equation and by using Gauss theorem, yields to the following discrete equations for each term

Convective term:

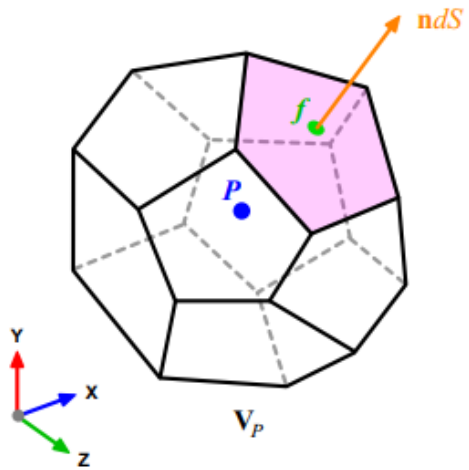
$$\int_{V_P} \underbrace{\nabla \cdot (\rho \mathbf{u} \phi)\, dV}_{\text{convective term}} = \oint_{\partial V_P} \underbrace{d\mathbf{S} \cdot (\rho \mathbf{u} \phi)}_{\text{convective flux}} = \sum_f \int_f d\mathbf{S} \cdot (\rho \mathbf{u} \phi)_f \approx \underbrace{\sum_f \mathbf{S}_f \cdot (\overline{\rho \mathbf{u} \phi})_f}_{} = \sum_f \mathbf{S}_f \cdot (\rho \mathbf{u} \phi)_f$$

By using Gauss theorem we convert volume integrals into surface integrals

where we have approximated the integrant by means of the mid point rule, which is second order accurate

Gauss theorem:

$$\int_V \nabla \cdot \mathbf{a} dV = \oint_{\partial V} d\mathbf{S} \cdot \mathbf{a}$$



Método dos volumes finitos

- Integrating in space each term of the general transport equation and by using Gauss theorem, yields to the following discrete equations for each term

Diffusive term:

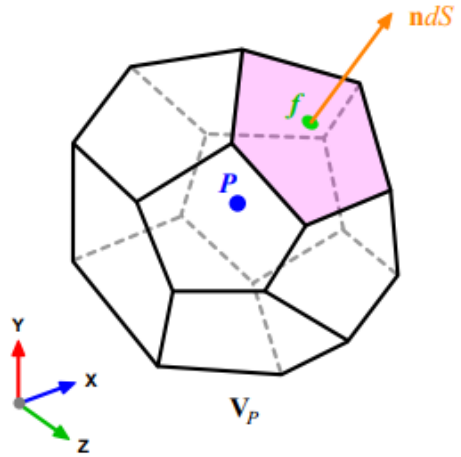
$$\int_{V_P} \underbrace{\nabla \cdot (\rho\Gamma_\phi \nabla\phi)\, dV}_{\text{diffusion term}} = \oint_{\partial V_P} \underbrace{d\mathbf{S} \cdot (\rho\Gamma_\phi \nabla\phi)}_{\text{diffusive flux}} = \sum_f \int_f d\mathbf{S} \cdot (\rho\Gamma_\phi \nabla\phi)_f \approx \underbrace{\sum_f \mathbf{S}_f \cdot \left(\overline{\rho\Gamma_\phi \nabla\phi}\right)_f}_{} = \sum_f \mathbf{S}_f \cdot (\rho\Gamma_\phi \nabla\phi)_f$$

By using Gauss theorem we convert volume integrals into surface integrals

where we have approximated the integrant by means of the mid point rule, which is second order accurate

Gauss theorem:

$$\int_V \nabla \cdot \mathbf{a} dV = \oint_{\partial V} d\mathbf{S} \cdot \mathbf{a}$$



Método dos volumes finitos

- Integrating in space each term of the general transport equation and by using Gauss theorem, yields to the following discrete equations for each term

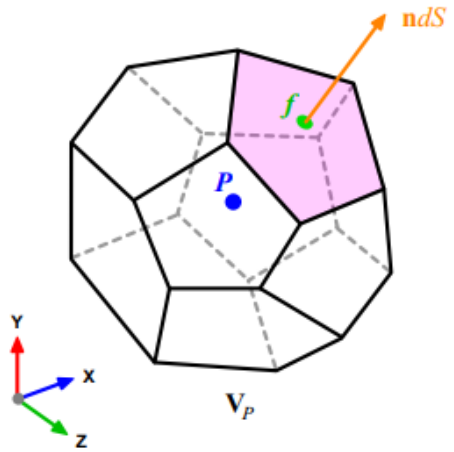
Gradient term:

$$(\nabla\phi)_P = \frac{1}{V_P}\sum_f (\mathbf{S}_f\phi_f)$$

where we have approximated the centroid gradients by using the Gauss theorem.
This method is second order accurate

Gauss theorem:

$$\int_V \nabla\cdot\mathbf{a}dV = \oint_{\partial V} d\mathbf{S}\cdot\mathbf{a}$$



Método dos volumes finitos

- Using the previous equations to evaluate the general transport equation over all the control volumes, we obtain the following semi-discrete equation

$$\int_{V_P} \underbrace{\frac{\partial \rho \phi}{\partial t} dV}_{\text{temporal derivative}} + \sum_f \underbrace{\mathbf{S}_f \cdot (\rho \mathbf{u} \phi)_f}_{\text{convective flux}} - \sum_f \underbrace{\mathbf{S}_f \cdot (\rho \Gamma_\phi \nabla \phi)_f}_{\text{difussive flux}} = \underbrace{(S_c V_P + S_p V_P \phi_P)}_{\text{source term}}$$

where $\mathbf{S} \cdot (\rho \mathbf{u} \phi) = F^C$ is the convective flux and $\mathbf{S} \cdot (\rho \Gamma_\phi \nabla \phi) = F^D$ is the diffusive flux.

- And recall that all variables are computed and stored at the centroid of the control volumes.

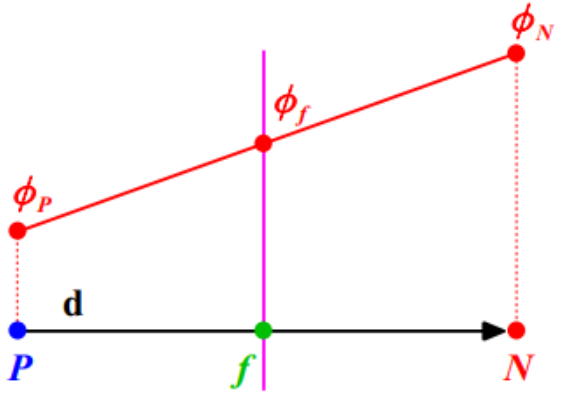
- The face values appearing in the convective and diffusive fluxes have to be computed by some form of interpolation from the centroid values of the control volumes at both sides of face f.

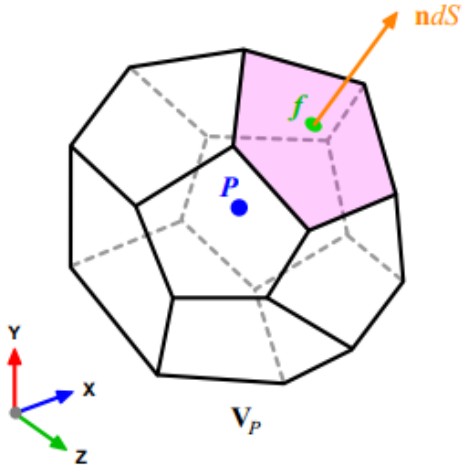
Método dos volumes finitos

Interpolation of the convective fluxes

- By looking the figure below, the face values appearing in the convective flux can be computed as follows,

$$\sum_f \mathbf{S}_f \cdot (\rho \mathbf{u} \phi)_f$$





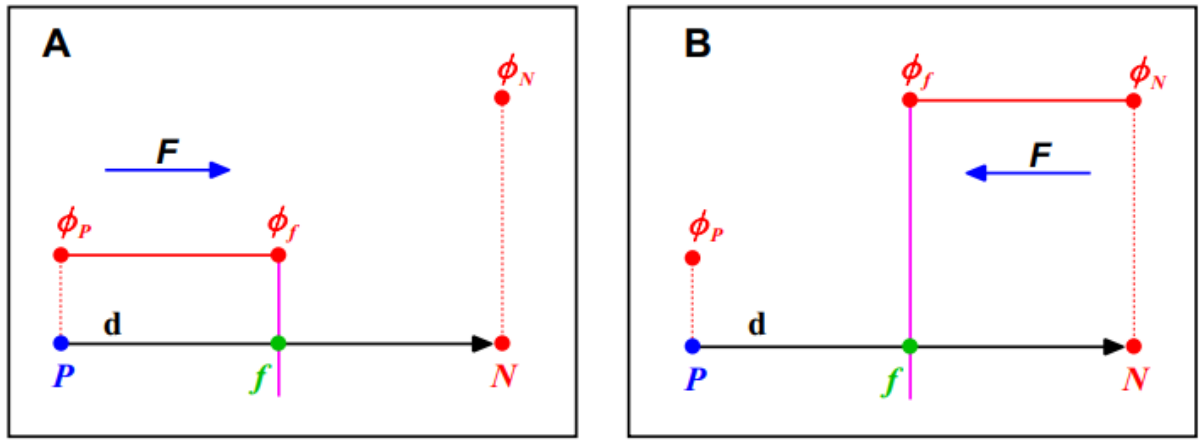
$$\phi_f = f_x \phi_P + (1 - f_x)\, \phi_N \qquad f_x = \frac{fN}{PN} = \frac{|\, \mathbf{x}_f - \mathbf{x}_N \,|}{|\, \mathbf{d} \,|}$$

- This type of interpolation scheme is known as linear interpolation or central differencing and it is second order accurate.
- However, it may generate oscillatory solutions (unbounded solutions).

Método dos volumes finitos

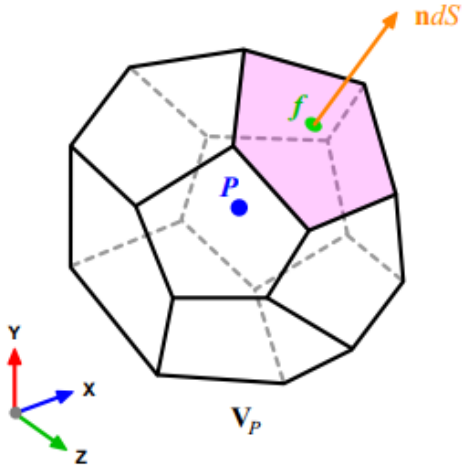
Interpolation of the convective fluxes

- By looking the figure below, the face values appearing in the convective flux can be computed as follows,



$$\phi_f = \begin{cases} \phi_f = \phi_P & \text{for} \quad \mathring{F} \geq 0, \\ \phi_f = \phi_N & \text{for} \quad \mathring{F} < 0. \end{cases}$$

$$\sum_f \mathbf{S}_f \cdot (\rho \mathbf{u} \phi)_f$$

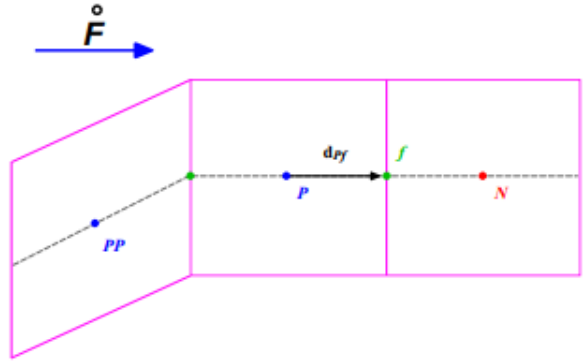


- This type of interpolation scheme is known as upwind differencing and it is first order accurate.
- This scheme is bounded (non-oscillatory) and diffusive.

Método dos volumes finitos

Interpolation of the convective fluxes – Unstructured meshes

- A simple way around this problem is to redefine higher-order schemes in terms of gradients at the control volume P.
- For example, using the gradient of the cells, we can compute the face values as follows,



Upwind $\rightarrow \quad \phi_f = \phi_P$

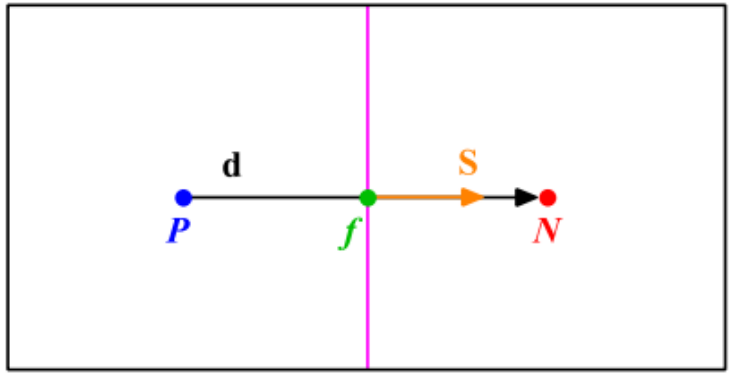
Central difference $\rightarrow \quad \phi_f = \phi_P + \nabla\phi_f \cdot \mathbf{d}_{Pf}$

Second order upwind differencing $\rightarrow \quad \phi_f = \phi_P + (2\nabla\phi_P - \nabla\phi_f) \cdot \mathbf{d}_{Pf}$

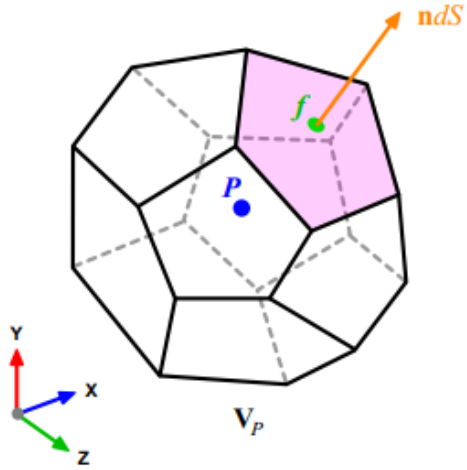
Método dos volumes finitos

Interpolation of diffusive fluxes in a orthogonal mesh

- By looking the figures below, the face values appearing in the diffusive flux in an orthogonal mesh can be computed as follows,



$$\sum_f \mathbf{S}_f \cdot (\rho \Gamma_\phi \nabla \phi)_f$$



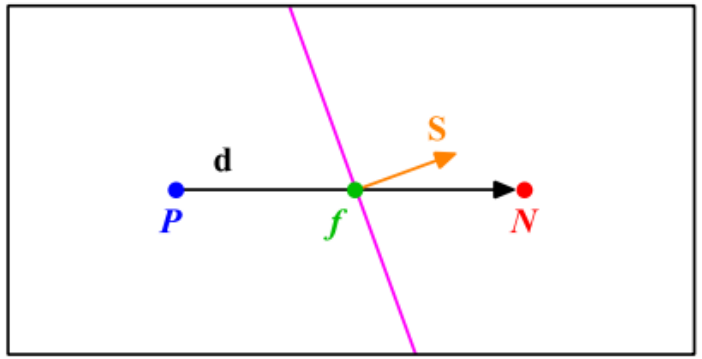
$$\mathbf{S} \cdot (\nabla \phi)_f = |\mathbf{S}| \frac{\phi_N - \phi_P}{|\mathbf{d}|}.$$

- This is a central difference approximation of the first order derivative. This type of approximation is second order accurate.

Método dos volumes finitos

Interpolation of diffusive fluxes in a non-orthogonal mesh

- By looking the figures below, the face values appearing in the diffusive flux in a non-orthogonal mesh ($20°$) can be computed as follows,



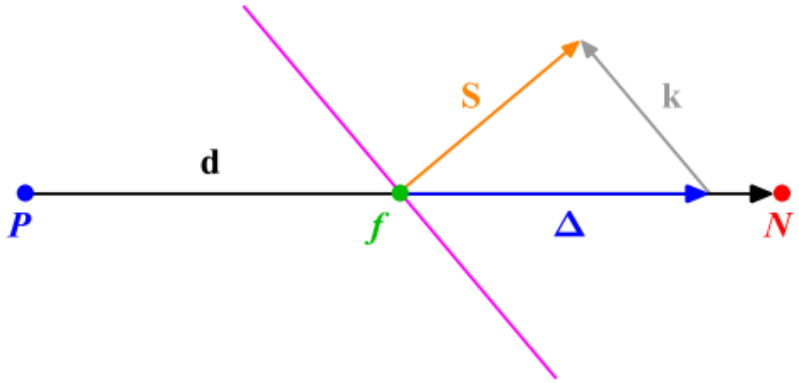
$$\mathbf{S}\cdot(\nabla\phi)_f = \underbrace{|\Delta_\perp|\frac{\phi_N-\phi_P}{|\mathbf{d}|}}_{\text{orthogonal contribution}} + \underbrace{\mathbf{k}\cdot(\nabla\phi)_f}_{\text{non-orthogonal contribution}} .$$

- This type of approximation is second order accurate but involves a larger truncation error. It also uses a larger numerical stencil, which make it less stable.

Método dos volumes finitos

Correction of diffusive fluxes in a non-orthogonal mesh

- By looking the figures below, the face values appearing in the diffusive flux in a non-orthogonal mesh (40°) can be computed as follows.
- Using the over-relaxed approach, the diffusive fluxes can be corrected as follow,



Over-relaxed approach

$$\Delta_\perp = \frac{\mathbf{d}}{\mathbf{d}\cdot\mathbf{S}}|\mathbf{S}|^2.$$

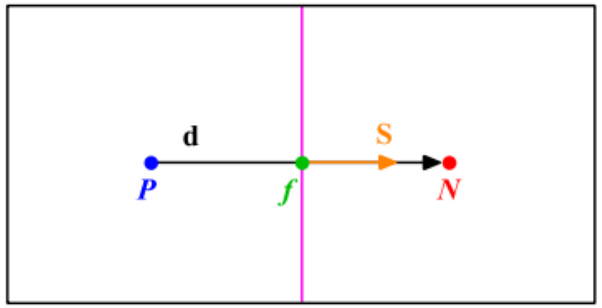
$$\mathbf{S} = \Delta_\perp + \mathbf{k}.$$

$$\mathbf{S}\cdot(\nabla\phi)_f = \underbrace{|\Delta_\perp|\frac{\phi_N - \phi_P}{|\mathbf{d}|}}_{\text{orthogonal contribution}} + \underbrace{\mathbf{k}\cdot(\nabla\phi)_f}_{\text{non-orthogonal contribution}}.$$

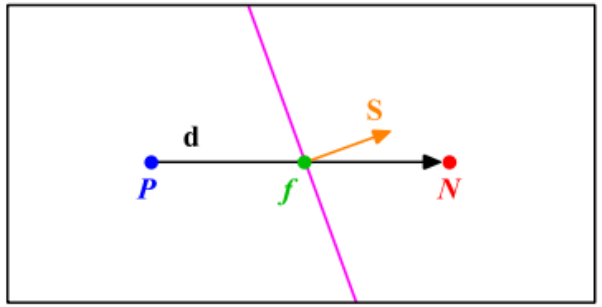
Método dos volumes finitos

Mesh induced errors

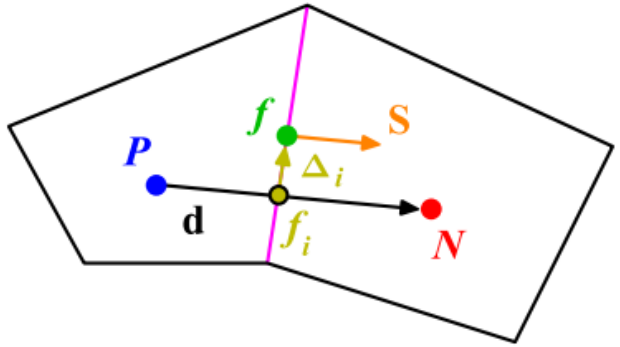
- In order to maintain second order accuracy, and to avoid unboundedness, we need to correct non-orthogonality and skewness errors.
- The ideal case is to have an orthogonal and non skew mesh, but this is the exception rather than the rule.



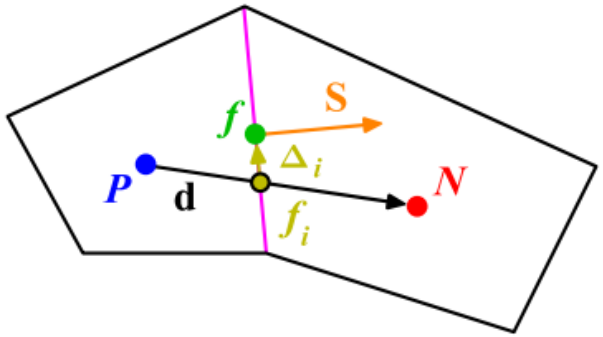
Orthogonal and non skew mesh



Non-orthogonal and non skew mesh



Orthogonal and skew mesh



Non-orthogonal and skew mesh

Método dos volumes finitos

Temporal discretization

- Now, we evaluate in time the semi-discrete general transport equation

$$\int_t^{t+\Delta t} \left[\left(\frac{\partial \rho \phi}{\partial t} \right)_P V_P + \sum_f \mathbf{S}_f \cdot (\rho \mathbf{u} \phi)_f - \sum_f \mathbf{S}_f \cdot (\rho \Gamma_\phi \nabla \phi)_f \right] dt$$
$$= \int_t^{t+\Delta t} (S_c V_P + S_p V_P \phi_P)\, dt.$$

- At this stage, we can use any time discretization scheme, *e.g.*, Crank-Nicolson, euler implicit, forward euler, backward differencing, adams-bashforth, adams-moulton.

- It should be noted that the order of the temporal discretization of the transient term does not need to be the same as the order of the discretization of the spatial terms. Each term can be treated differently to yield different accuracies. As long as the individual terms are at least second order accurate, the overall accuracy will also be second order.

Método dos volumes finitos

Where do we set all the discretization schemes in OpenFOAM®?

```
ddtSchemes                                  <-- ∂φ/∂t
{
    default         backward;
}

gradSchemes                                 <-- ∇φ_P
{
    default         Gauss linear;
    grad(p)         Gauss linear;
}

divSchemes                                  <-- ∇ · (Uφ)
{
    default         none;
    div(phi,U)      Gauss linear;
}

laplacianSchemes                            <-- ∇ · Γ∇φ
{
    default         Gauss linear orthogonal;
}

interpolationSchemes                        <-- φ_f = f_x φ_P + (1 − f_x) φ_N
{                                               f_x = fN/PN = |x_f − x_N| / |d|
    default         linear;
}

snGradSchemes
{
    default         orthogonal;             <-- n_f · ∇φ_f
}
```

- The `fvSchemes` dictionary contains the information related to the discretization schemes for the different terms appearing in the governing equations.
- The discretization schemes can be chosen in a term-by-term basis.
- The keyword **ddtSchemes** refers to the time discretization.
- The keyword **gradSchemes** refers to the gradient term discretization.
- The keyword **divSchemes** refers to the convective terms discretization.
- The keyword **laplacianSchemes** refers to the Laplacian terms discretization.
- The keyword **interpolationSchemes** refers to the method used to interpolate values from cell centers to face centers. It is unlikely that you will need to use something different from linear.
- The keyword **snGradSchemes** refers to the discretization of the surface normal gradients evaluated at the faces.
- Remember, if you want to know the options available for each keyword you can use the banana method.

Método dos volumes finitos

Where do we set all the discretization schemes in OpenFOAM®?

```
ddtSchemes                                   ∂φ/∂t
{
    default         backward;
}

gradSchemes                                  ∇φ_P
{
    default         Gauss linear;
    grad(p)         Gauss linear;
}

divSchemes                                   ∇ · (Uφ)
{
    default         none;
    div(phi,U)      Gauss linear;
}

laplacianSchemes                             ∇ · Γ∇φ
{
    default         Gauss linear orthogonal;
}

interpolationSchemes                         φ_f = f_x φ_P + (1 − f_x) φ_N
{                                            f_x = fN/PN = |x_f − x_N| / |d|
    default         linear;
}

snGradSchemes
{
    default         orthogonal;              n_f · ∇φ_f
}
```

- The *fvSchemes* dictionary contains the information related to the discretization schemes for the different terms appearing in the governing equations.
- The discretization schemes can be chosen in a term-by-term basis.
- The keyword **ddtSchemes** refers to the time discretization.
- The keyword **gradSchemes** refers to the gradient term discretization.
- The keyword **divSchemes** refers to the convective terms discretization.
- The keyword **laplacianSchemes** refers to the Laplacian terms discretization.
- The keyword **interpolationSchemes** refers to the method used to interpolate values from cell centers to face centers. It is unlikely that you will need to use something different from linear.
- The keyword **snGradSchemes** refers to the discretization of the surface normal gradients evaluated at the faces.
- Remember, if you want to know the options available for each keyword you can use the banana method.

OpenFOAM

scalarTransportFoam

```
.
├── 0
│   ├── T
│   ├── T.orig
│   ├── U
│   └── U.orig
├── Allrun
├── constant
│   ├── polyMesh
│   │   ├── boundary
│   │   ├── faces
│   │   ├── neighbour
│   │   ├── owner
│   │   └── points
│   └── transportProperties
├── fields.foam
├── residuos.py
└── system
    ├── blockMeshDict
    ├── controlDict
    ├── fvSchemes
    ├── fvSolution
    ├── sample
    └── setFieldsDict

4 directories, 19 files
```

Método dos volumes finitos

Linear system solution

- After spatial and temporal discretization and by using equation

$$\int_t^{t+\Delta t}\left[\left(\frac{\partial \rho\phi}{\partial t}\right)_P V_P + \sum_f \mathbf{S}_f\cdot(\rho\mathbf{u}\phi)_f - \sum_f \mathbf{S}_f\cdot(\rho\Gamma_\phi\nabla\phi)_f\right]dt = \int_t^{t+\Delta t}(S_c V_P + S_p V_P \phi_P)dt$$

in every control volume V_P of the domain, a system of linear algebraic equations for the transported quantity ϕ is assembled,

$$\begin{pmatrix} a_{11} & a_{12} & & \ddots & & & & \\ a_{21} & a_{22} & a_{23} & & \ddots & & & \\ & \ddots & \ddots & \ddots & & \ddots & & \\ \ddots & & \ddots & \ddots & \ddots & & \ddots & \\ & a_S & & a_W & a_P & a_E & & a_N \\ & & \ddots & & \ddots & \ddots & \ddots & \\ & & & \ddots & & \ddots & \ddots & \ddots \\ & & & & \ddots & & \ddots & a_{PP} \end{pmatrix} \times \begin{pmatrix} \\ \phi_S \\ \\ \phi_W \\ \phi_P \\ \phi_E \\ \\ \phi_N \\ \end{pmatrix} = \begin{pmatrix} \\ b_S \\ \\ b_W \\ b_P \\ b_E \\ \\ b_N \\ \end{pmatrix}$$

- This system can be solved by using any iterative or direct method.

Método dos volumes finitos

Linear system solution

- After spatial and temporal discretization and by using equation

$$\int_t^{t+\Delta t}\left[\left(\frac{\partial \rho\phi}{\partial t}\right)_P V_P + \sum_f \mathbf{S}_f\cdot(\rho\mathbf{u}\phi)_f - \sum_f \mathbf{S}_f\cdot(\rho\Gamma_\phi\nabla\phi)_f\right]dt = \int_t^{t+\Delta t}(S_c V_P + S_p V_P \phi_P)dt$$

in every control volume V_P of the domain, a system of linear algebraic equations for the transported quantity ϕ is assembled,

$$\begin{pmatrix} a_{11} & a_{12} & & \ddots & & & & \\ a_{21} & a_{22} & a_{23} & & \ddots & & & \\ & \ddots & \ddots & \ddots & & \ddots & & \\ \ddots & & \ddots & \ddots & \ddots & & \ddots & \\ & a_S & & a_W & a_P & a_E & & a_N \\ & & \ddots & & \ddots & \ddots & \ddots & \\ & & & \ddots & & \ddots & \ddots & \ddots \\ & & & & \ddots & & \ddots & a_{PP} \end{pmatrix} \times \begin{pmatrix} \\ \\ \phi_S \\ \\ \phi_W \\ \phi_P \\ \phi_E \\ \\ \phi_N \\ \\ \end{pmatrix} = \begin{pmatrix} \\ \\ b_S \\ \\ b_W \\ b_P \\ b_E \\ \\ b_N \\ \\ \end{pmatrix}$$

- This system can be solved by using any iterative or direct method.

OpenFOAM

scalarTransportFoam

```
.
├── 0
│   ├── T
│   ├── T.orig
│   ├── U
│   └── U.orig
├── Allrun
├── constant
│   ├── polyMesh
│   │   ├── boundary
│   │   ├── faces
│   │   ├── neighbour
│   │   ├── owner
│   │   └── points
│   └── transportProperties
├── fields.foam
├── residuos.py
└── system
    ├── blockMeshDict
    ├── controlDict
    ├── fvSchemes
    ├── fvSolution
    ├── sample
    └── setFieldsDict

4 directories, 19 files
```

Método dos volumes finitos

Linear solvers

```
solvers
{
    p
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-06;
        relTol          0;
    }
    pFinal
    {
        $p;
        relTol  0;
    }
    U
    {
        solver          PBiCGStab;
        preconditioner  DILU;
        tolerance       1e-08;
        relTol          0;
    }
}

PISO
{
    nCorrectors  2;
    nNonOrthogonalCorrectors  1;
}
```

- The equation solvers, tolerances, and algorithms are controlled from the sub-dictionary **solvers** located in the *fvSolution* dictionary file.
- In the dictionary file *fvSolution*, and depending on the solver you are using you will find the additional sub-dictionaries **PISO, PIMPLE,** and **SIMPLE**, which will be described later.
- In this dictionary is where we are telling OpenFOAM® how to crunch numbers.
- The **solvers** sub-dictionary specifies each linear-solver that is used for each equation being solved.
- If you forget to define a linear-solver, OpenFOAM® will let you know what are you missing.
- The syntax for each entry within the **solvers** sub-dictionary uses a keyword that is the word relating to the variable being solved in the particular equation and the options related to the linear-solver.

https://www.openfoam.com/documentation/guides/latest/doc/guide-solvers.html

Método dos volumes finitos

Linear solvers

```
solvers
{
    p
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-06;
        relTol          0;
    }
    pFinal
    {
        $p;
        relTol  0;
    }
    U
    {
        solver          PBiCGStab;
        preconditioner  DILU;
        tolerance       1e-08;
        relTol          0;
    }
}

PISO
{
    nCorrectors  2;
    nNonOrthogonalCorrectors  1;
}
```

- The equation solvers, tolerances, and algorithms are controlled from the sub-dictionary **solvers** located in the *fvSolution* dictionary file.
- In the dictionary file *fvSolution*, and depending on the solver you are using you will find the additional sub-dictionaries **PISO, PIMPLE,** and **SIMPLE**, which will be described later.
- In this dictionary is where we are telling OpenFOAM® how to crunch numbers.
- The **solvers** sub-dictionary specifies each linear-solver that is used for each equation being solved.
- If you forget to define a linear-solver, OpenFOAM® will let you know what are you missing.
- The syntax for each entry within the **solvers** sub-dictionary uses a keyword that is the word relating to the variable being solved in the particular equation and the options related to the linear-solver.

OpenFOAM

scalarTransportFoam

```
.
├── 0
│   ├── T
│   ├── T.orig
│   ├── U
│   └── U.orig
├── Allrun
├── constant
│   ├── polyMesh
│   │   ├── boundary
│   │   ├── faces
│   │   ├── neighbour
│   │   ├── owner
│   │   └── points
│   └── transportProperties
├── fields.foam
├── residuos.py
└── system
    ├── blockMeshDict
    ├── controlDict
    ├── fvSchemes
    ├── fvSolution
    ├── sample
    └── setFieldsDict

4 directories, 19 files
```

https://www.openfoam.com/documentation/guides/latest/doc/guide-solvers.html

Método dos volumes finitos

Linear solvers

```
solvers
{
    p
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-06;
        relTol          0;
    }
    pFinal
    {
        $p;
        relTol   0;
    }
    U
    {
        solver          PBiCGStab;
        preconditioner  DILU;
        tolerance       1e-08;
        relTol          0;
    }
}

PISO
{
    nCorrectors     2;
    nNonOrthogonalCorrectors    1;
}
```

- In this generic case, to solve the pressure (**p**) we are using the **PCG** method with the **DIC** preconditioner, an absolute **tolerance** equal to 1e-06 and a relative tolerance **relTol** equal to 0.
- The entry **pFinal** refers to the final pressure correction (notice that we are using macro syntax), and we are using a relative tolerance **relTol** equal to 0.
- To solve the velocity field (**U**) we are using the **PBiCGStab** method with the **DILU** preconditoner, an absolute **tolerance** equal to 1e-08 and a relative tolerance **relTol** equal to 0.
- The linear solvers will iterative until reaching any of the tolerance values set by the user or reaching a maximum value of iterations (optional entry).
- FYI, solving for the velocity is relative inexpensive, whereas solving for the pressure is expensive.
- The pressure equation is particularly important as it governs mass conservation.
- If you do not solve the equations accurately enough (tolerance), the physics might be wrong.
- Selection of the tolerance is of paramount importance.

Método dos volumes finitos

Linear solvers

```
solvers
{
    p
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-06;
        relTol          0;
    }
    pFinal
    {
        $p;
        relTol   0;
    }
    U
    {
        solver          PCG;
        preconditioner  DILU;
        tolerance       1e-08;
        relTol          0;
    }
}

PISO
{
    nCorrectors    2;
    nNonOrthogonalCcorrectors    1;
}
```

- The linear solvers distinguish between symmetric matrices and asymmetric matrices.
- The symmetry of the matrix depends on the structure of the equation being solved.
- Pressure is a symmetric matrix and velocity is an asymmetric matrix.
- If you use the wrong linear solver, OpenFOAM® will complain and will let you know what options are valid.
- In the following error screen, we are using a symmetric solver for an asymmetric matrix,

```
-> FOAM FATAL IO ERROR :
Unknown asymmetric matrix solver PCG

Valid asymmetric matrix solvers are :

4
(
BICCG
GAMG
P
smoothSolver
)
```

Método dos volumes finitos

Linear solvers

```
solvers
{
    p
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-06;
        relTol          0;
    }
    pFinal
    {
        $p;
        relTol  0;
    }
    U
    {
        solver          PBiCGStab;
        preconditioner  DILU;
        tolerance       1e-08;
        relTol          0;
    }
}

PISO
{
    nCorrectors  2;
    nNonOrthogonalCorrectors  1;
}
```

- The linear solvers are iterative, *i.e.*, they are based on reducing the equation residual over a succession of solutions.
- The residual is a measure of the error in the solution so that the smaller it is, the more accurate the solution.
- More precisely, the residual is evaluated by substituting the current solution into the equation and taking the magnitude of the difference between the left and right hand sides (L2-norm).

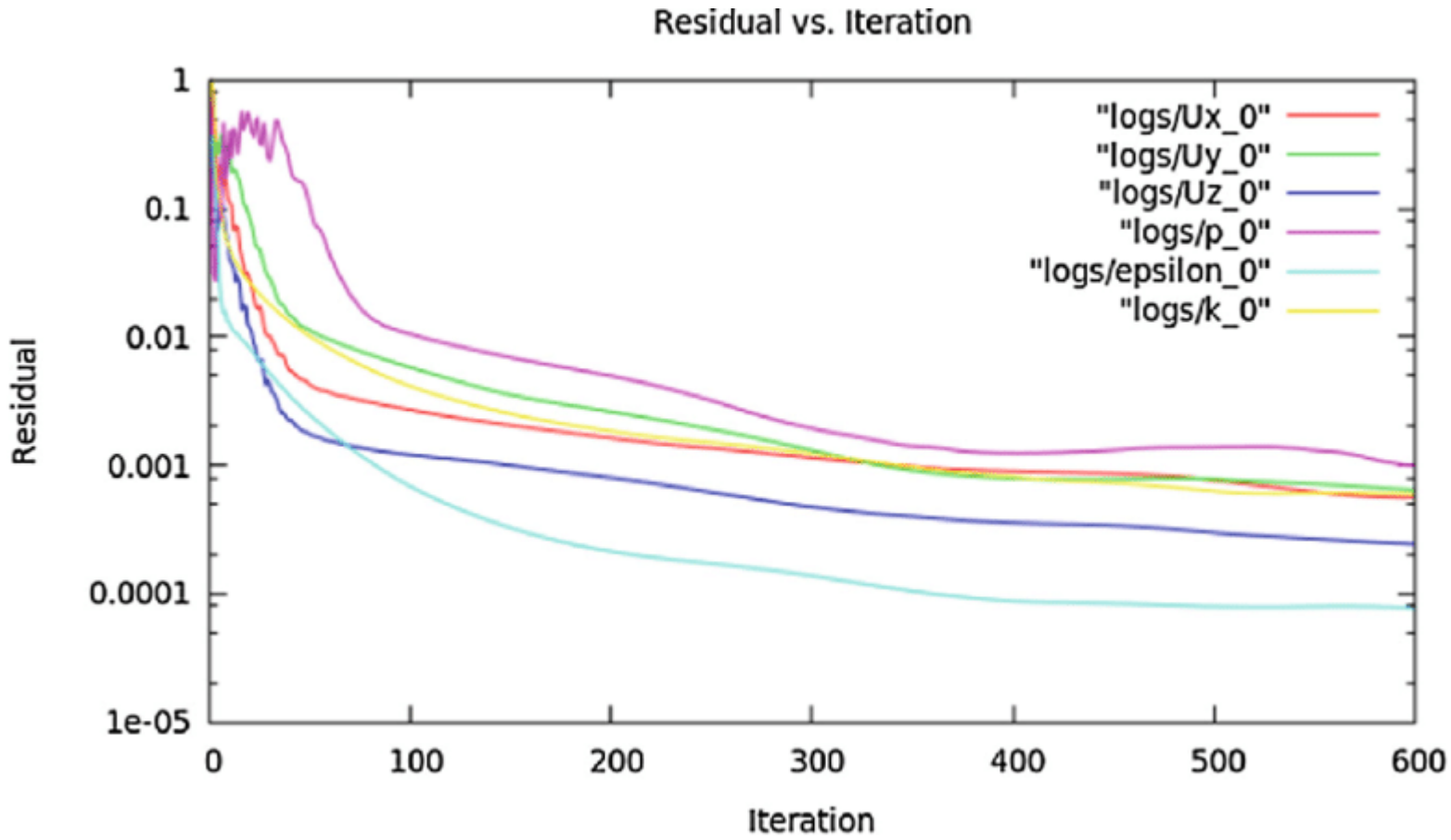
$$\left|\mathbf{A}\phi^{\mathrm{k}} - \mathbf{b}\right| = \left|\mathbf{r}^{\mathrm{k}}\right|$$

- It is also normalized to make it independent of the scale of the problem being analyzed.

$$\text{Residual} = \frac{|\mathbf{r}|}{\text{Normalization factor}} < \text{Tolerance}$$

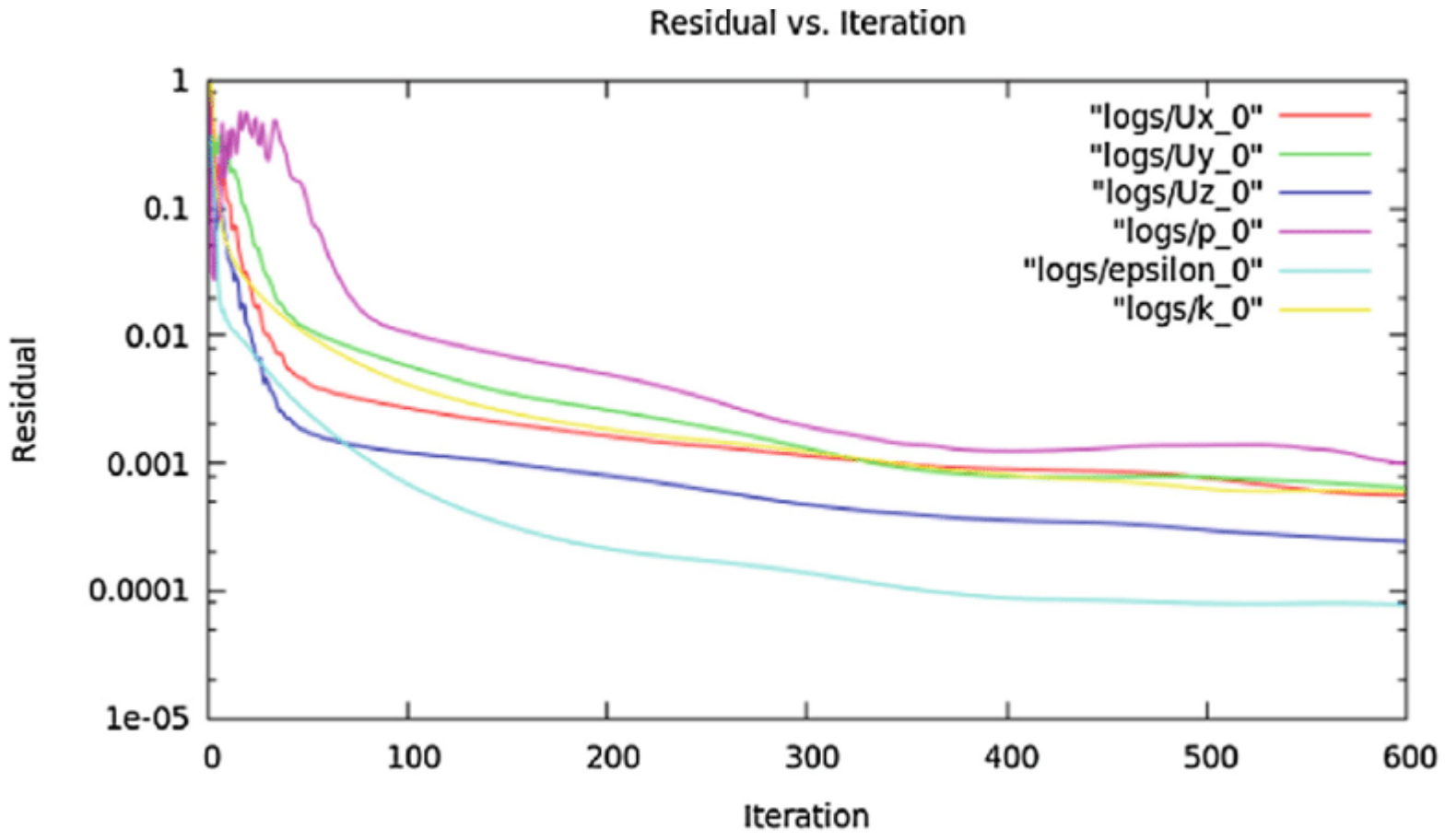
Método dos volumes finitos

PyFoam



Método dos volumes finitos

PyFoam



OpenFOAM

scalarTransportFoam

```
.
├── 0
│   ├── T
│   ├── T.orig
│   ├── U
│   └── U.orig
├── Allrun
├── constant
│   ├── polyMesh
│   │   ├── boundary
│   │   ├── faces
│   │   ├── neighbour
│   │   ├── owner
│   │   └── points
│   └── transportProperties
├── fields.foam
├── residuos.py
└── system
    ├── blockMeshDict
    ├── controlDict
    ├── fvSchemes
    ├── fvSolution
    ├── sample
    └── setFieldsDict

4 directories, 19 files
```

Método dos volumes finitos

Linear solvers

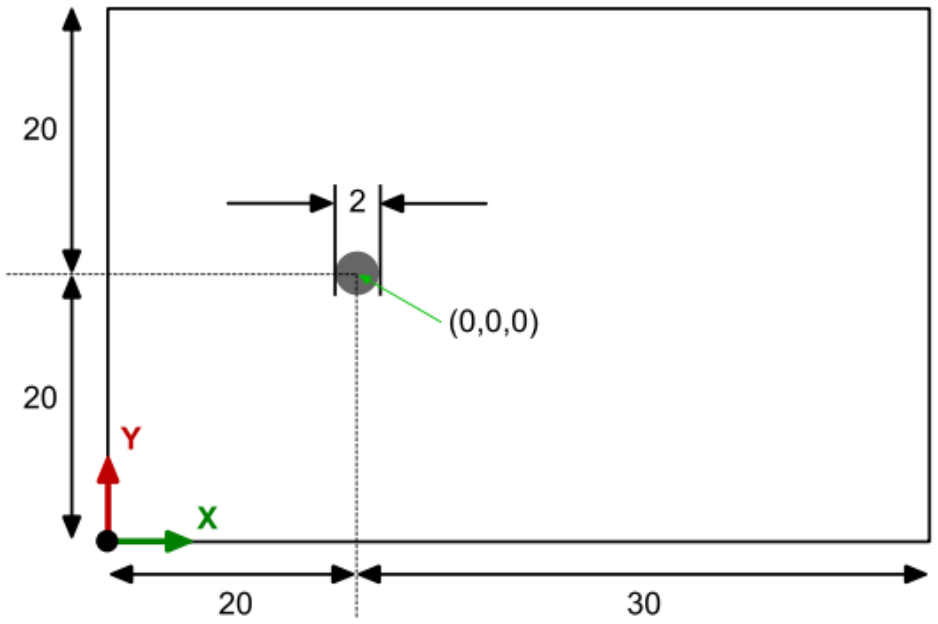
```
solvers
{
    p
    {
        solver          PCG;
        preconditioner  DIC;
        tolerance       1e-06;
        relTol          0;
    }
    pFinal
    {
        $p;
        relTol  0;
    }
    U
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-08;
        relTol          0;
        minIter         3;
        maxIter         100;
    }
}

PISO
{
    nCorrectors  2;
    nNonOrthogonalCorrectors  1;
}
```

- Before solving an equation for a particular field, the initial residual is evaluated based on the current values of the field.
- After each solver iteration the residual is re-evaluated. The solver stops if either of the following conditions are reached:
  - The residual falls below the solver tolerance, **tolerance**.
  - The ratio of current to initial residuals falls below the solver relative tolerance, **relTol**.
  - The number of iterations exceeds a maximum number of iterations, **maxIter**.
- The solver tolerance should represent the level at which the residual is small enough that the solution can be deemed sufficiently accurate.
- The keyword **maxIter** is optional and the default value is 1000.
- The user can also define the minimum number of iterations using the keyword **minIter**. This keyword is optional and the default value is 0.

Aplicação – simulação Vórtice de von Kármán

Flow around a cylinder – 10 < Re < 2 000 000
Incompressible and compressible flow



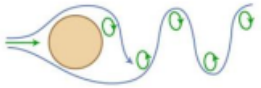
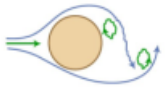
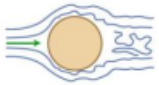
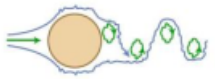
All the dimensions are in meters

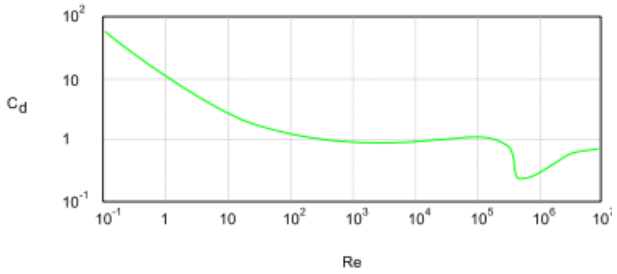
Physical and numerical side of the problem:

- In this case we are going to solve the flow around a cylinder. We are going to use incompressible and compressible solvers, in laminar and turbulent regime.
- Therefore, the governing equations of the problem are the incompressible/compressible laminar/turbulent Navier-Stokes equations.
- We are going to work in a 2D domain.
- Depending on the Reynolds number, the flow can be steady or unsteady.
- This problem has a lot of validation data.

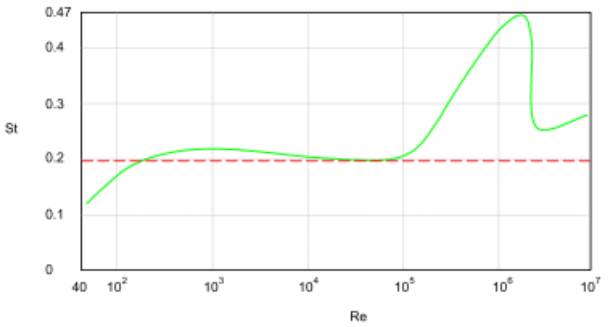
Aplicação – simulação Vórtice de von Kármán

Vortex shedding behind a cylinder

 Creeping flow (no separation) Steady flow	$Re < 5$
 A pair of stable vortices in the wake Steady flow	$5 < Re < 40 - 46$
 Laminar vortex street (Von Karman street) Unsteady flow	$40 - 46 < Re < 150$
 Laminar boundary layer up to the separation point, turbulent wake Unsteady flow	$150 < Re < 300$ Transition to turbulence $300 < Re < 3 \times 10^5$
 Boundary layer transition to turbulent Unsteady flow	$3 \times 10^5 < Re < 3 \times 10^6$
 Turbulent vortex street, but the wake is narrower than in the laminar case Unsteady flow	$3 \times 10^6 > Re$

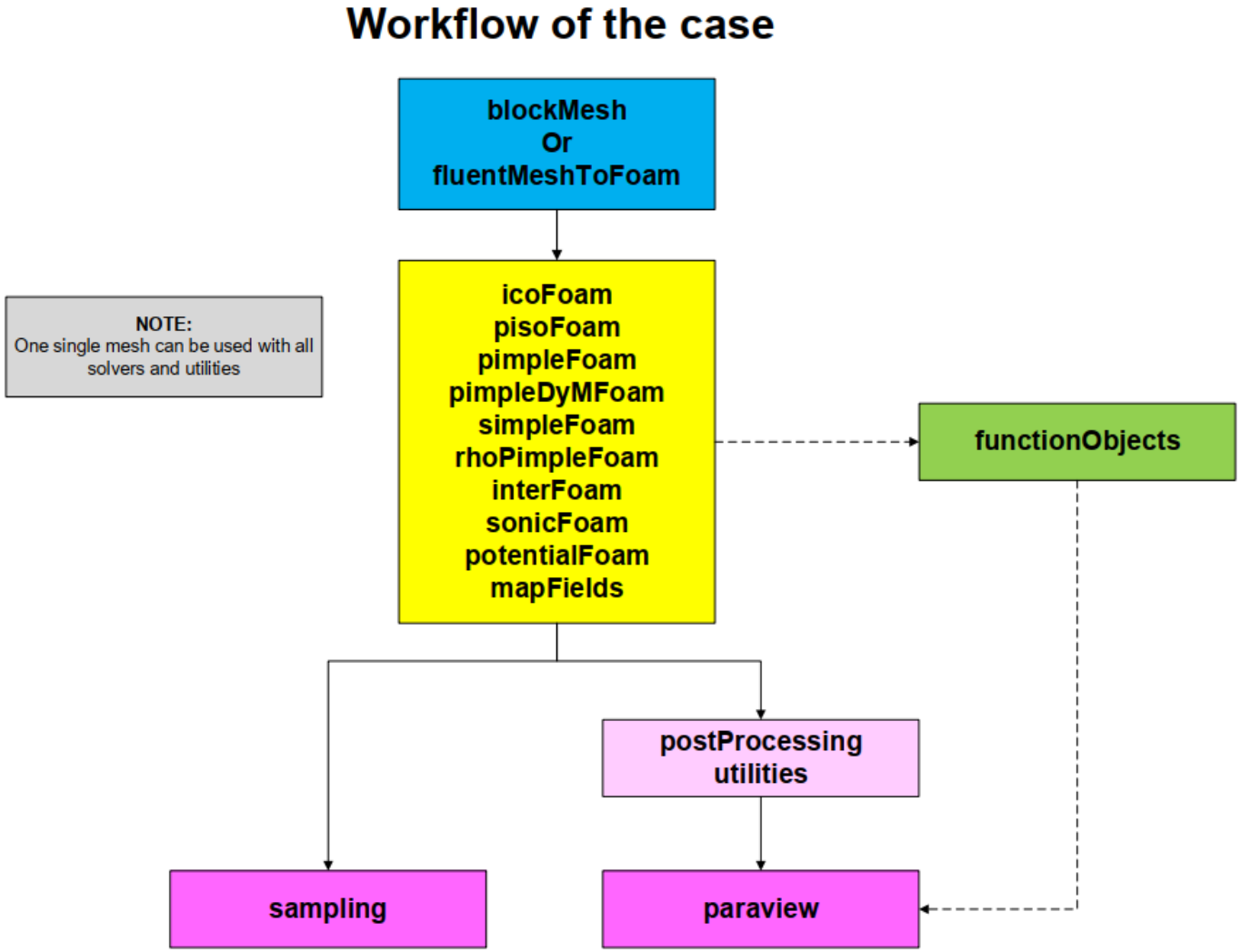


Drag coefficient



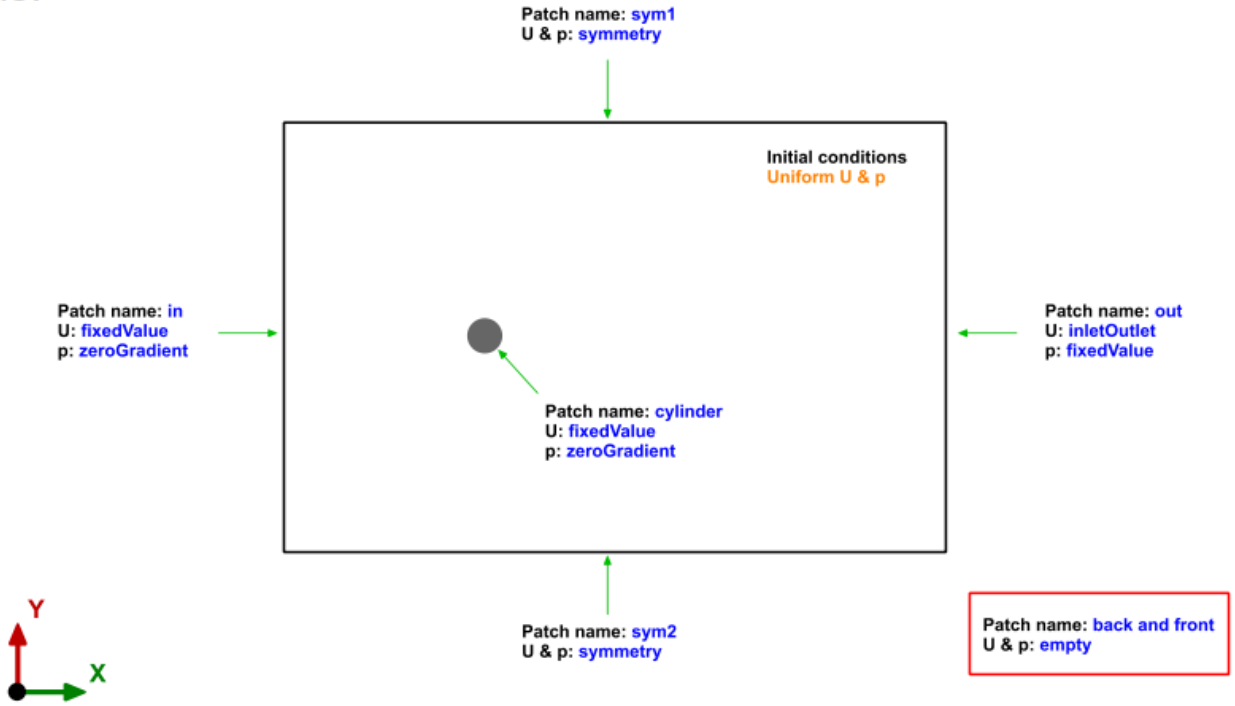
Strouhal number

Aplicação – simulação Vórtice de von Kármán



Aplicação – simulação Vórtice de von Kármán

- Remember, when converting meshes the **name** and **type** of the patches are not always set as you would like, so it is always a good idea to take a look at the file `boundary` and modify it according to your needs.
- Let us modify the `boundary` dictionary file.
- In this case, we would like to setup the following **numerical type** boundary conditions.



Aplicação – simulação Vórtice de von Kármán

inletOutlet Boundary Condition

The *inletOutlet* boundary condition is normally the same as *zeroGradient*, but it switches to *fixedValue* if the velocity vector next to the boundary aims inside the domain (backward flow). The value of that *fixedValue* is *inletValue*. See figure (□).

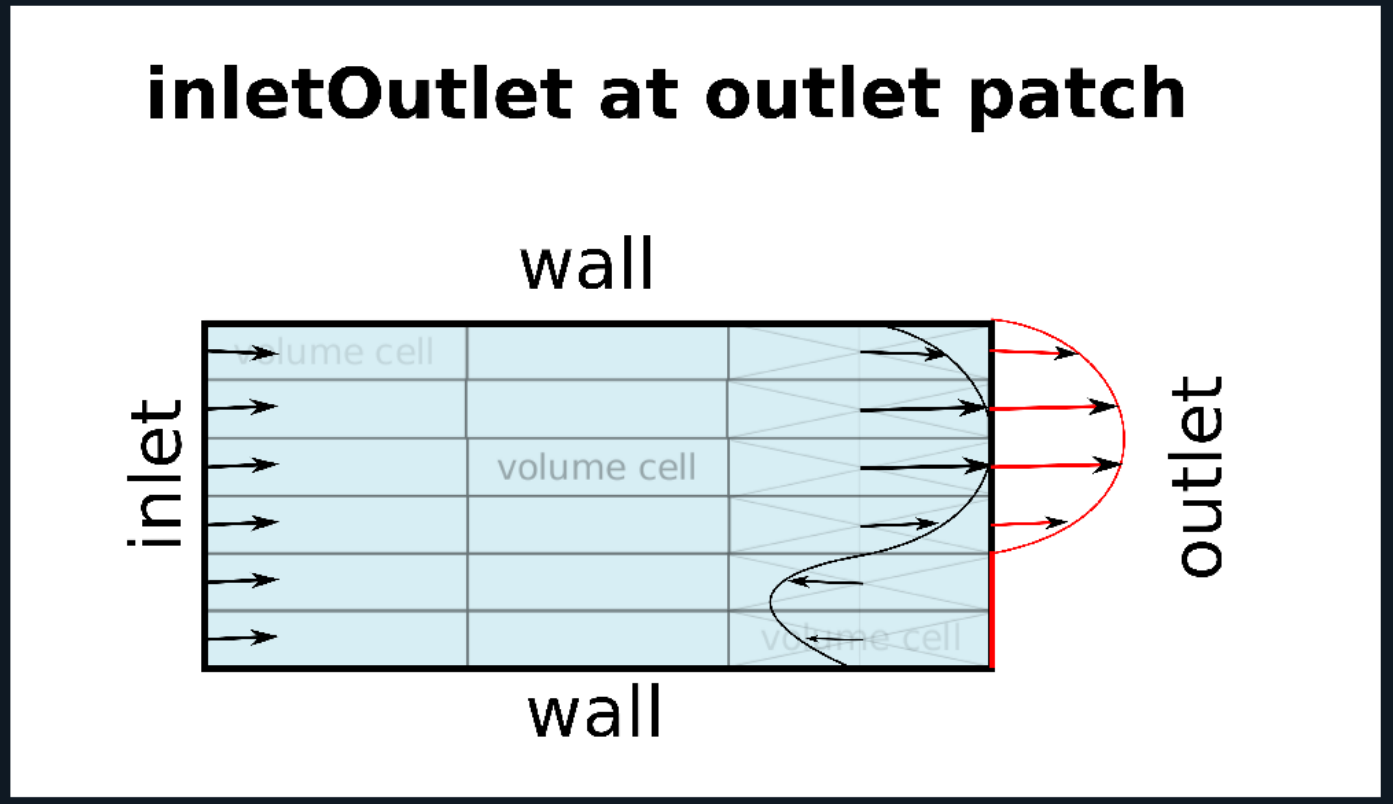


Figure: The meaning of inletOutlet boundary condition

U:

```
boundaryField
{
".*_outlet"
   {
        type            inletOutlet;
        value           $internalField;
        inletValue      uniform (0 0 0);
   }
}
```

Aplicação – simulação Vórtice de von Kármán

- At this point, check that the **name** and **type** of the **base type** boundary conditions and **numerical type** boundary conditions are consistent. If everything is ok, we are ready to go.
- Do not forget to explore the rest of the dictionary files, namely:
  - *0/p* (**p** is defined as relative pressure)
  - *0/U*
  - constant/*transportProperties*
  - *system/controlDict*
  - *system/fvSchemes*
  - *system/fvSolution*
- Reminder:
  - The diameter of the cylinder is 2.0 m.
  - And we are targeting for a Re = 200.

$$\nu = \frac{\mu}{\rho} \qquad Re = \frac{\rho \times U \times D}{\mu} = \frac{U \times D}{\nu}$$

Aplicação – simulação Vórtice de von Kármán

basic	potential flow, Laplace and transport equations
combustion	combustion fluid flows
compressible	compressible fluid flows
discreteMethods	molecule movement
DNS	direct numerical simulations
electromagnetics	electrostatics, magnetohydrodynamics
financial	Black-Scholes equation to price commodities
heatTransfer	heat transfers flows
incompressible	incompressible flows
lagrangian	Lagrangian approach to CFD
multiphase	multiphase flows
stressAnalysis	stress analysis in a solid body

Capability matrix

Solver	transient	compressible	turbulence	heat-transfer	buoyancy	combustion	multiphase	particles	dynamic mesh	multi-region	fvOptions
boundaryFoam											
buoyantPimpleFoam	✓	✓	✓	✓	✓						✓
buoyantSimpleFoam		✓	✓	✓	✓						✓
chemFoam	✓			✓		✓					
chtMultiRegionFoam	✓	✓	✓	✓	✓					✓	✓
coldEngineFoam	✓	✓	✓	✓		✓			✓		✓
engineFoam	✓	✓	✓	✓		✓			✓		✓
fireFoam	✓	✓	✓	✓	✓	✓				✓	✓
icoFoam	✓										
interFoam	✓		✓				✓		✓		✓
laplacianFoam	✓										✓
pimpleFoam	✓		✓						✓		✓
pisoFoam	✓		✓								✓

Aplicação – simulação Vórtice de von Kármán

Overview

- Category: **Incompressible**
- transient
- incompressible

Solver: icoFOAM OpenFOAM

Equations

The solver uses the **PISO** algorithm to solve the continuity equation:

$$\nabla \cdot \mathbf{u} = 0$$

Where:

$\mathbf{u}$ = Velocity

p = Kinematic pressure

and momentum equation:

$$\frac{\partial}{\partial t}(\mathbf{u}) + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) - \nabla \cdot (\nu \nabla \mathbf{u}) = -\nabla p$$

Aplicação – simulação Vórtice de von Kármán

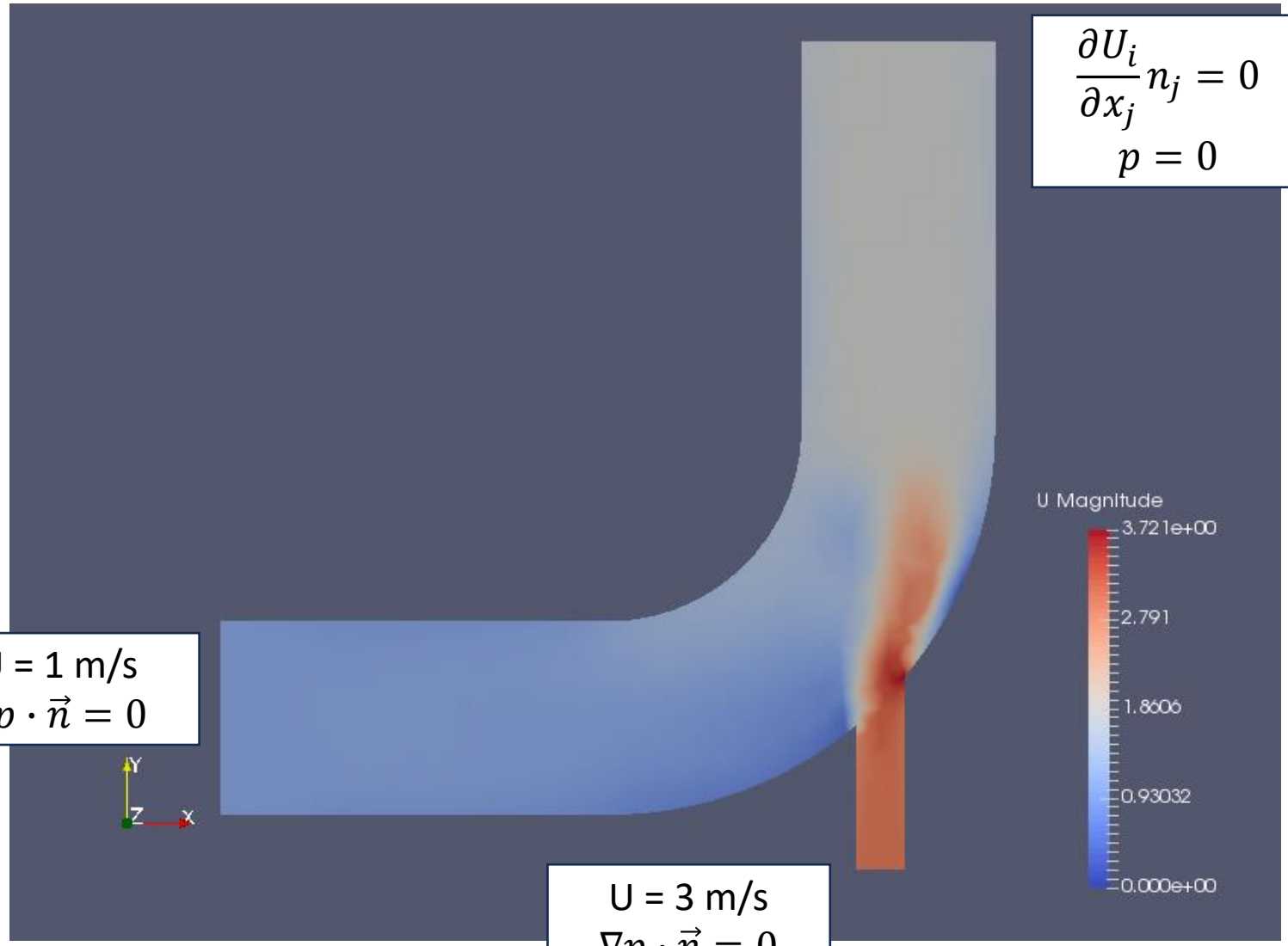
- Creating mesh in blockMesh - part of OpenFOAM
- Creating mesh in snappyHexMesh - part of OpenFOAM
- Creating mesh in foamyHexMesh - part of OpenFOAM
- Creating mesh with third party tools and importing it.
- OpenFOAM supports following mesh conversions:

ansysToFoam
cfx4ToFoam
datToFoam
fluent3DMeshToFoam
fluentMeshToFoam
foamMeshToFluent
foamToStarMesh
foamToSurface
gambitToFoam
gmshToFoam
ideasUnvToFoam
kivaToFoam
mshToFoam
netgenNeutralToFoam
plot3dToFoam
sammToFoam
star3ToFoam
star4ToFoam
tetgenToFoam
vtkUnstructuredToFoam

Aplicação – simulação Vórtice de von Kármán

- At this point try to use the following utilities. In the terminal type:
  - `$> postProcess -func vorticity -noZero`
    This utility will compute and write the vorticity field. The `–noZero` option means do not compute the vorticity field for the solution in the directory **0**. If you do not add the `-noZero` option, it will compute and write the vorticity field for all the saved solutions, including **0**
  - `$> postprocess -func 'grad(U)' -latestTime`
    This utility will compute and write the velocity gradient or `grad(U)` in the whole domain (including at the walls). The `-latestTime` option means compute the velocity gradient only for the last saved solution.
  - `$> postprocess -func 'grad(p)'`
    This utility will compute and write the pressure gradient or `grad(U)` in the whole domain (including at the walls).
  - `$> postProcess -func 'div(U)'`
    This utility will compute and write the divergence of the velocity field or `grad(U)` in the whole domain (including at the walls). You will need to add the keyword **div(U) Gauss linear**; in the dictionary `fvSchemes`.
  - `$> foamToVTK -time 50:300`
    This utility will convert the saved solution from OpenFOAM® format to VTK format. The `-time 50:300` option means convert the solution to VTK format only for the time directories **50** to **300**
  - `$> pisoFoam -postProcess -func CourantNo`
    This utility will compute and write the Courant number. This utility needs to access the solver database for the physical properties and additional quantities, therefore we need to tell what solver we are using. As the solver `icoFoam` does not accept the option `-postProcess`, we can use the solver `pisoFoam` instead. Remember, `icoFoam` is a fully laminar solver and `pisoFoam` is a laminar/turbulent solver.
  - `$> pisoFoam -postProcess -func wallShearStress`
    This utility will compute and write the wall shear stresses at the walls. As no arguments are given, it will save the wall shear stresses for all time steps.

Exercício – simulação tubulação



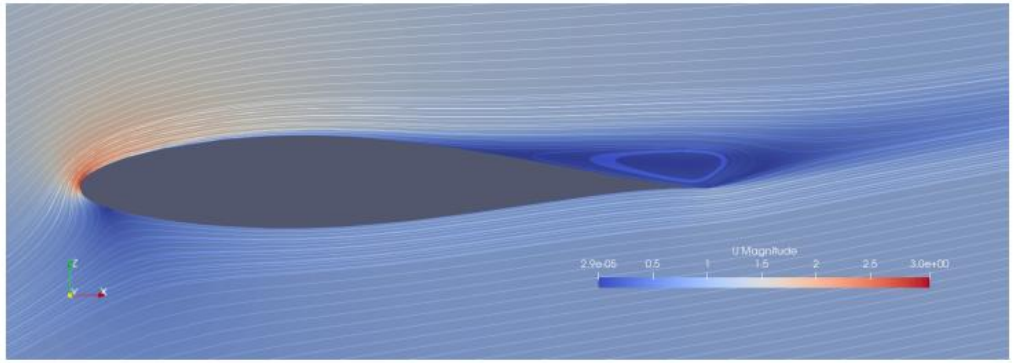
Na parede:

- U = 0
- $\nabla p \cdot \vec{n} = 0$

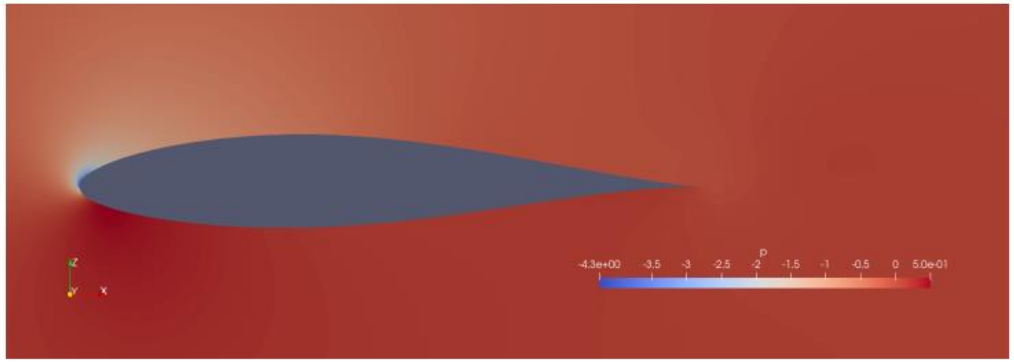
Propriedade do fluido:

$\nu = 0.01\ m^2/s$

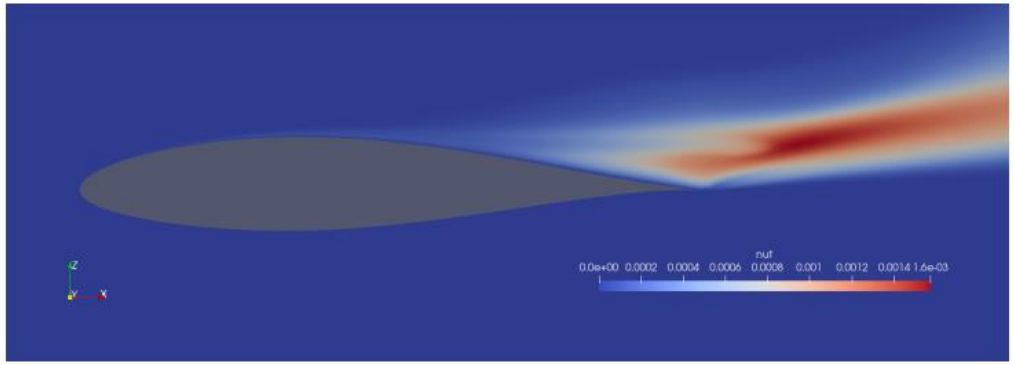
Exercício – simulação aerofólio



(a)



(b)



(c)

COBEM 2023 27TH INTERNATIONAL CONGRESS OF MECHANICAL ENGENEERING

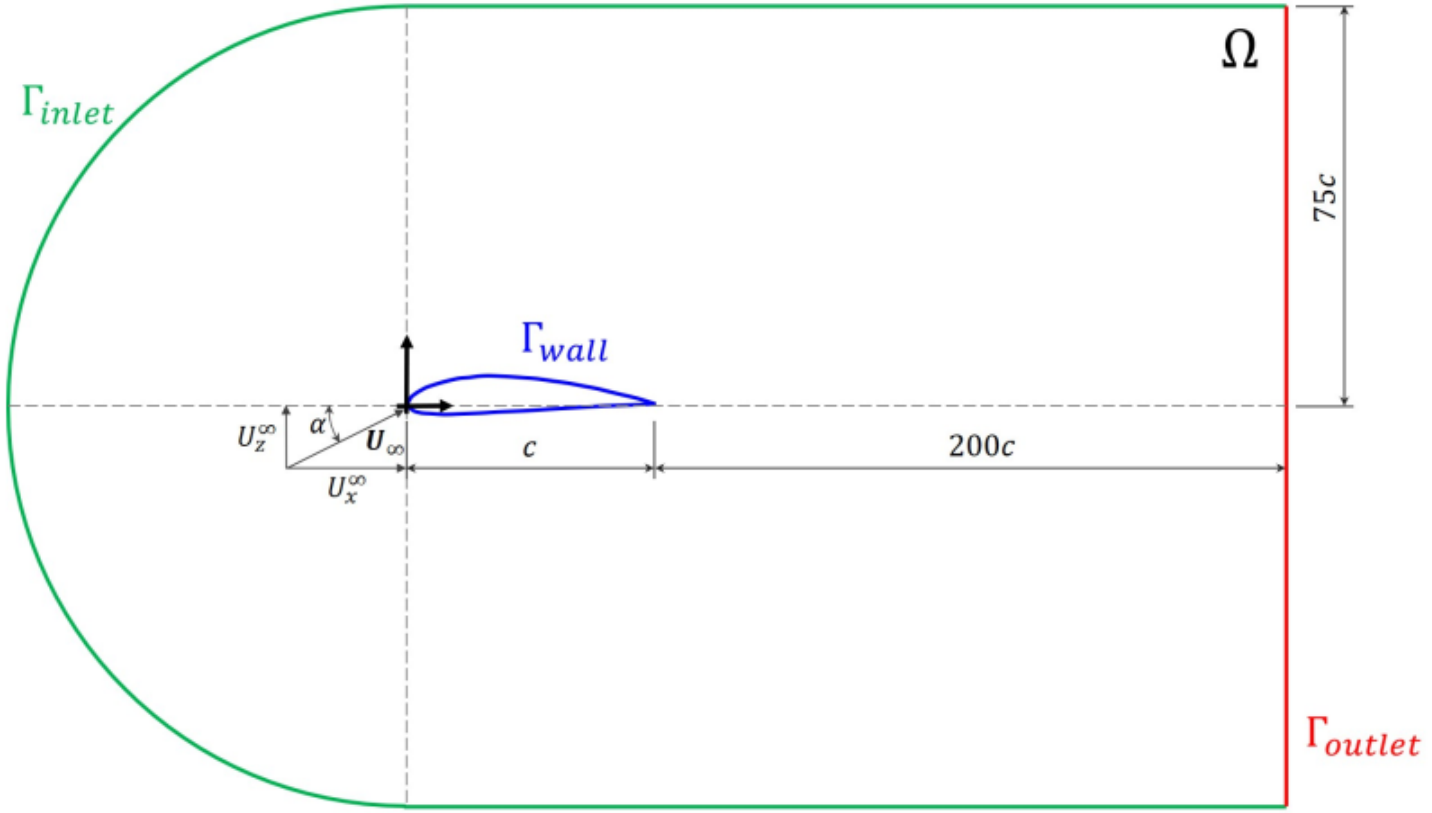
ABCM Brazilian Society of Mechanical Sciences and Engineering
27th ABCM International Congress of Mechanical Engineering
December 04-08, 2023, Florianópolis-SC, Brazil

COB-2023-0751

A COMPARATIVE STUDY OF A TRANSITION MODEL WITH A FULL TURBULENCE MODEL ON THE AERODYNAMIC ANALYSIS OF A WIND TURBINE AIRFOIL

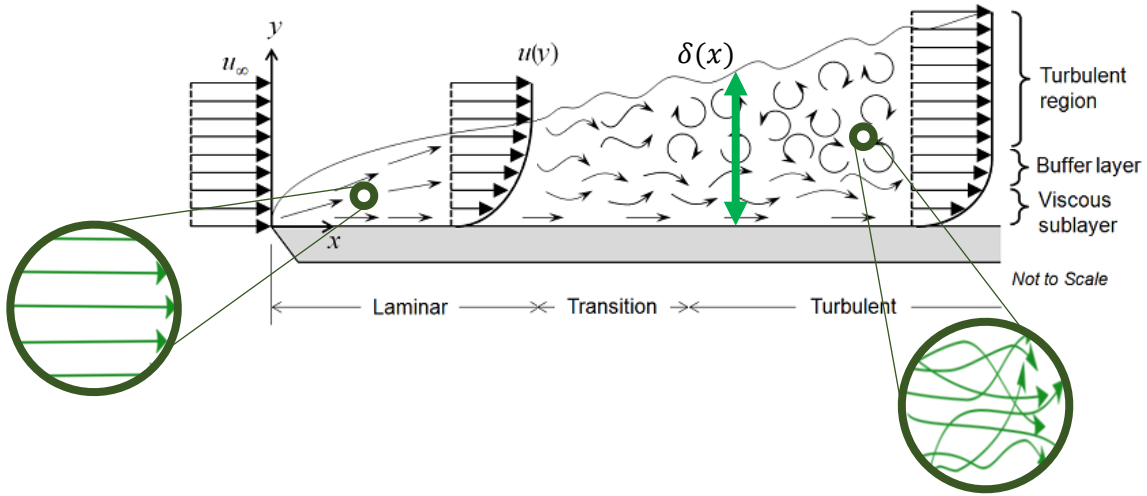
Anderson M. Ribeiro
Graduate Program in Computational Modeling, Federal University of Juiz de Fora. José Lourenço Kelmer St., São Pedro, Juiz de Fora, MG 36036-900, Brazil.
andersonmr254@gmail.com

Exercício – simulação aerofólio



$$
\begin{cases}
\frac{\partial(\rho U_j U_i)}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(2\mu S_{ij} - \rho\overline{U_i' U_j'}\right) & \text{in } \Omega \\
\frac{\partial U_i}{\partial x_i} = 0 & \text{in } \Omega \\
U_i n_i < 0 \implies U_i = U_i^\infty \vee U_i n_i \geq 0 \implies \frac{\partial U_i}{\partial x_j} n_j = 0 & \text{on } \Gamma_{inlet} \\
U_i = 0 & \text{on } \Gamma_{wall} \\
\frac{\partial U_i}{\partial x_j} n_j = 0 & \text{on } \Gamma_{outlet}
\end{cases}
$$

Conceitos de turbulência



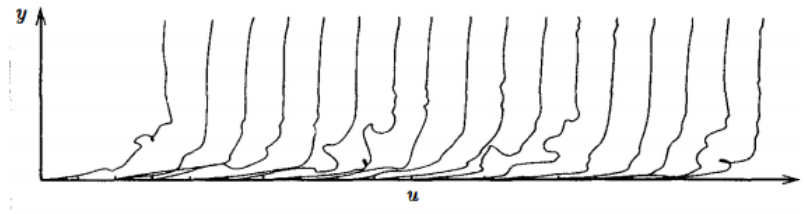
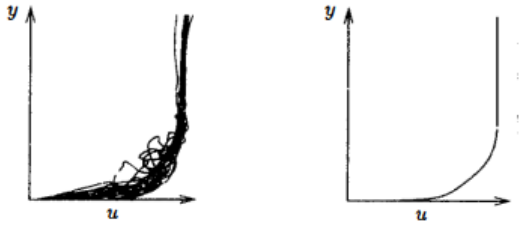


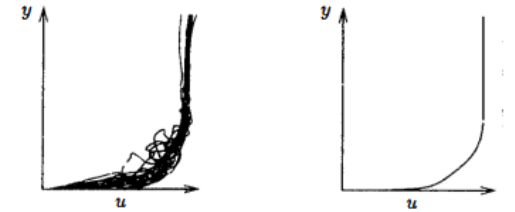
Figure 2.1: *Instantaneous boundary-layer velocity profiles at the same distance from the leading edge of a flat plate at 17 different instants. The profiles are shown with a series of staggered origins. [From Cebeci and Smith (1974) — Copyright © Academic Press 1974 — Used with permission.]*



(a) All profiles – one origin (b) Average profile

Figure 2.2: *Instantaneous and average boundary-layer velocity profiles at the same distance from the leading edge of a flat plate. [From Cebeci and Smith (1974) — Copyright © Academic Press 1974 — Used with permission.]*

Conceitos de turbulência



(a) All profiles – one origin (b) Average profile

Figure 2.2: *Instantaneous and average boundary-layer velocity profiles at the same distance from the leading edge of a flat plate. [From Cebeci and Smith (1974) — Copyright © Academic Press 1974 — Used with permission.]*

$$u_i(\mathbf{x},t) = U_i(\mathbf{x}) + u_i'(\mathbf{x},t)$$

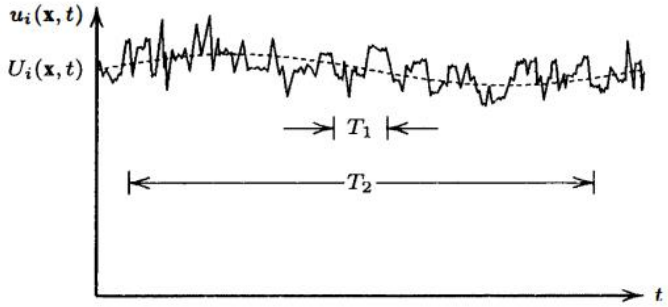


Figure 2.5: *Time averaging for nonstationary turbulence. Although obscured by the scale of the graph, the instantaneous velocity, $u_i(\mathbf{x},t)$, has continuous derivatives of all order.*

Conceitos de turbulência

- Fluidos Newtonianos
- Escoamento incompressível

Equação Cons. Massa

$$\frac{\partial u_i}{\partial x_i} = 0$$

Equação Cons. Qtde. Movimento
(Forma ñ conservativa)

$$\rho\frac{\partial u_i}{\partial t} + \rho u_j\frac{\partial u_i}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial t_{ji}}{\partial x_j}$$

$$t_{ij} = 2\mu s_{ij} \qquad s_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$$

Lai, M., Rubin, E., Krempl, E., Introduction to Continuum Mechanics, 3rd. Edition, ButterworthHeinemann, 1999, ISBN 0-7506-2894-4 .

$$\rho\frac{\partial u_i}{\partial t} + \rho\frac{\partial}{\partial x_j}(u_j u_i) = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}(2\mu s_{ji})$$

Reynolds Averaged Navier-Stokes (RANS)

$$\frac{\partial U_i}{\partial x_i} = 0$$

$$\rho\frac{\partial U_i}{\partial t} + \text{Termo Convectivo} = -\frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_j}(2\mu S_{ji})$$

Conceitos de turbulência

Reynolds Averaged Navier-Stokes (RANS)

$$\frac{\partial U_i}{\partial x_i} = 0$$

$$\rho\frac{\partial U_i}{\partial t} + \text{Termo Convectivo} = -\frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_j}(2\mu S_{ji})$$

$$\rho\frac{\partial U_i}{\partial t} + \rho U_j\frac{\partial U_i}{\partial x_j} = -\frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_j}\left(2\mu S_{ji} - \rho\overline{u_j' u_i'}\right)$$

$-\rho\overline{u_j' u_i'}$.: Tensor de Tensão de Reynolds

Escoamentos tridimensionais:

- 4 propriedades médias do escoamento → pressão e 3 componentes de velocidade
- 6 componentes do tensor de tensões de Reynolds

} 10 variáveis

$$\frac{\partial U_i}{\partial x_i} = 0$$

$$\rho\frac{\partial U_i}{\partial t} + \rho U_j\frac{\partial U_i}{\partial x_j} = -\frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_j}\left(2\mu S_{ji} - \rho\overline{u_j' u_i'}\right)$$

} 4 Equações

Conceitos de turbulência

Escoamentos tridimensionais:

- 4 propriedades médias do escoamento → pressão e 3 componentes de velocidade
- 6 componentes do tensor de tensões de Reynolds

10 variáveis

$$\frac{\partial U_i}{\partial x_i} = 0$$

$$\rho \frac{\partial U_i}{\partial t} + \rho U_j \frac{\partial U_i}{\partial x_j} = -\frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_j}\left(2\mu S_{ji} - \rho\overline{u_j' u_i'}\right)$$

4 Equações

$$-\rho\overline{u_i' u_j'} = \mu_t \left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} - \frac{1}{3}\frac{\partial U_k}{\partial x_k}\delta_{ij}\right) - \frac{2}{3}\rho k \delta_{ij}$$

Boussinesq, J. (19877), "Essai sur la théorie des eaux courantes", Mémoires présentés par divers savants à l'Académie des Sciences 23 (1): 1-680

https://turbmodels.larc.nasa.gov/