An Introduction to the Finite Volume Method with Turbulence Modeling

Michele Girfoglio

Course on
Models and Applications in Computational Fluid Mechanics
SISSA, Trieste, June 16 - July 3, 2025

Outline

What we will see today?

- Introduction to **the finite volume Method**
  - **Spatial** Discretization
  - **Temporal** Discretization
  - **Boundary** Conditions
- Resolution of linear **systems of Equations**
- The **NS-Equations**
- **Pressure-Velocity** coupling
- **Turbulence Modeling**
  - Introduction to **RANS** methods
  - Introduction to **LES**
  - **Filtering** Techniques

References

Here there is a **list of good references** to know more about the **finite volume method** and **computational fluid dynamics**:

- **The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction With OpenFOAM and Matlab**, F. Moukalled, L. Mangani, M. Darwish. 2015, Springer-Verlag.
- **Finite Volume Methods for Hyperbolic Problems** R. Leveque. 2002, Cambridge University Press.
- **Riemann Solvers and Numerical Methods for Fluid Dynamics**, Toro, E. F. (1999), Springer-Verlag.
- **Finite Volume Methods: Foundation and Analysis**, Timothy Barth Raphaèle Herbin Mario Ohlberger, 2017, In Encyclopedia of Computational Mechanics Second Edition (eds E. Stein, R. Borst and T. J. Hughes).
- **Computational Methods for Fluid Dynamics** J. H. Ferziger, M. Peric. 2001, Springer
- **Error analysis and estimation in the Finite Volume method with applications to fluid flows**, H. Jasak. PhD Thesis. 1996. Imperial College, London
- **An Introduction to Computational Fluid Dynamics**, H. K. Versteeg, W. Malalasekera. 2007, Prentice Hall.

What is the Finite Volume Method in a nutshell

General Features

- It is a **discretization method** to solve partial differential equations describing **conservation laws**.
- **Divergence terms** are converted as surface integrals using the **Gauss' theorem** and computed as a sum of fluxes over the surfaces of each element.
- It does not have special requirements concerning the mesh structure and can work also on unstructured meshes.
- Most of **commercial codes** for CFD are based on this discretization technique (Ansys Fluent, STARCC, Ansys CFX, SymFlow ...).
- There are well developed **opensource** numerical packages specifically designed for CFD (OpenFOAM).
- In this course we will use codes based on the **OpenFOAM** framework.

FVM or FEM?

What are the **advantages** and **disadvantages** of these two discretization techniques?

- The FVM can be seen as a **"Galerkin discontinuous finite element method"** where the shape functions are defined by $\phi_K(\boldsymbol{x})$:

$$\begin{cases} \phi_k(\boldsymbol{x}) = 1 & \text{if } k = \boldsymbol{x}, \\ \phi_k(\boldsymbol{x}) = 0 & \text{if } k \neq \boldsymbol{x}. \end{cases}$$

FEM

FEAT.: More used to deal with Structural and Elliptic problems, for CFD it usually relies on a **coupled solution strategy**, it uses polynomial shape functions to approximate the solution, mainly developed and used by community of **mathematicians** (especially in CFD).
PROS: Stronger **Mathematical Framework**, easier to reach **high order accuracy**.
CONS: Usually **only globally conservative**, **less** available material for **turbulent flows**, **no-physical invocation**.

FVM

FEAT.: Particularly used for **CFD problem** and **less used for elliptic problems**, it uses **constant shape functions** to approximate the solution, mainly developed and used by **community of Engineers**.
PROS: **Easier Implementation**, works on **generic polyhedral cells**, locally Conservative. Direct physical invocation.
CONS: Irregular geometries require far more effort, more difficulties to reach high order accuracy.

#CFD

Discretization of an advection-diffusion equation

Advection-Diffusion equations

$$\frac{\partial \phi(\mathbf{x},t)}{\partial t} - \nabla \cdot (\mu(\mathbf{x},t)\nabla\phi(\mathbf{x},t)) + \nabla \cdot (\mathbf{u}(\mathbf{x},t)\phi(\mathbf{x},t)) = f(\mathbf{x},t), \quad \forall \mathbf{x} \in \Omega, t > 0,$$

$$\begin{cases} \phi(\mathbf{x},t) = g_D(\mathbf{x},t) & \forall \mathbf{x} \in \Gamma_D, t > 0 \\ -\frac{\partial \phi(\mathbf{x},t)}{\partial \mathbf{n}} = \mathbf{g}_N(\mathbf{x},t) & \forall \mathbf{x} \in \Gamma_N, t > 0, \\ \phi(\mathbf{x},0) = \phi_0(\mathbf{x}) & \forall \mathbf{x} \in \Omega \end{cases}$$

Discretization is based on the **integral form** of the equation over each element of the cell, $V \in \mathcal{M}$

$$\int_V \frac{\partial \phi}{\partial t} dV - \int_V \nabla \cdot (\mu \nabla \phi) dV + \int_V \nabla \cdot (\mathbf{u}\phi) dV = \int_V f dV.$$

Discretization

- Cell and face centroid

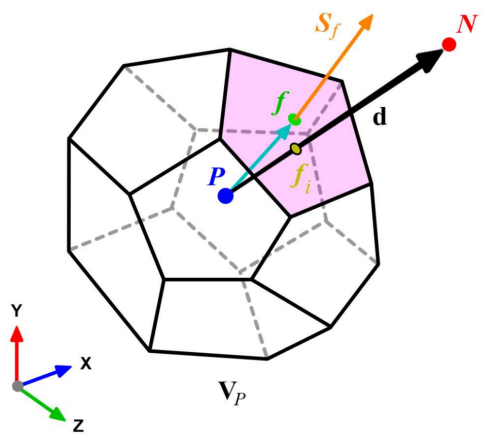
$$\mathbf{x}_P \in V_p \text{ s.t. } \int_{V_p} (\mathbf{x} - \mathbf{x}_P) dV = 0, \quad \mathbf{x}_f \in S_f \text{ s.t. } \int_{S_f} (\mathbf{x} - \mathbf{x}_f) d\gamma = 0$$

- Second order variation in space

$$\phi(\mathbf{x}, t) = \phi(\mathbf{x}_P, t) + \nabla\phi(\mathbf{x}_P, t) \cdot (\mathbf{x} - \mathbf{x}_P) + \mathrm{O}(\mathbf{x}^2)$$

- Second order variation in time

$$\phi(\mathbf{x}, t + \Delta t) = \phi(\mathbf{x}, t) + \frac{\partial\phi(\mathbf{x}, t)}{\partial t}\Delta t + \mathrm{O}(\Delta t^2)$$



Credit: *Joel Guerrero*

Volume and Surface Integrals

- Volume integral

$$\begin{aligned}\int_{V_P} \phi(\mathbf{x},t)dV &= \int_{V_P} \phi(\mathbf{x}_P,t) + \nabla\phi(\mathbf{x}_P,t)\cdot(\mathbf{x}-\mathbf{x}_P)dV \\ &= \phi(\mathbf{x}_P,t)V_P + \nabla\phi(\mathbf{x}_P,t)\cdot\int_{V_P}(\mathbf{x}-\mathbf{x}_P)dV \\ &= \phi_P(t)V_P\end{aligned}$$

- Surface integral

$$\begin{aligned}\int_S \phi(\mathbf{x},t)\mathbf{n}(\mathbf{x})d\gamma &= \sum_{S_f\in S}\int_{S_f}\phi(\mathbf{x},t)\mathbf{n}(\mathbf{x})d\gamma \\ &= \sum_{S_f\in S}\int_{S_f}[\phi(\mathbf{x}_f,t) + \nabla\phi(\mathbf{x}_f,t)\cdot(\mathbf{x}-\mathbf{x}_f)]\mathbf{n}(\mathbf{x})d\gamma \\ &= \sum_{S_f\in S}\phi_f(t)\mathbf{s}_f\end{aligned}$$

Assuming linear variation of the variable leads to a second-order discretization

Gauss' Theorem

$$\int_V \nabla \cdot \mathbf{a}(\mathbf{x}) dV = \int_{\partial V} \mathbf{a}(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x}) d\gamma$$

We use this theorem to turn derivatives into flux integrals

$$\frac{\partial}{\partial t} \int_{V_P} \phi dV - \int_{\partial V_P} (\mu \nabla \phi) \cdot \mathbf{n} d\gamma + \int_{\partial V_P} \phi \mathbf{u} \cdot \mathbf{n} d\gamma = \int_{V_P} f dV.$$

- Time and space domain are discretized
- Assumptions on temporal and spatial variations of the solution

We now discuss the discretization of each of these terms

Temporal Derivative

Captures the dynamics of the variable ϕ
Defining the timestep Δt, we have $t^{n+1} = t^n + \Delta t$

$$\phi^n := \phi(t^n), \quad \phi^{n+1} := \phi(t^{n+1}) = \phi(t^n + \Delta t)$$

We use finite difference schemes for the discretization of the dime derivative

1st order	$\frac{\phi^{n+1} - \phi^n}{\Delta t}$
2nd order	$\frac{\frac{3}{2}\phi^{n+1} - 2\phi^n + \frac{1}{2}\phi^{n-1}}{\Delta t}$

As usual, increasing the accuracy makes the stencil dimension bigger

$$\int_{V_P} \frac{\partial \phi}{\partial t} dV = \frac{\phi^{n+1} - \phi^n}{\Delta t} V_P$$

Independently of the order of the time discretization, this term only requires values assumed by the variable ϕ at the cell V_P

Convective Term

Advection of ϕ due to the velocity field $\mathbf{u}$

$$\int_{\partial V_P} \phi \mathbf{u} \cdot \mathbf{n} dV = \sum_{f \in \partial V_P} \int_f \phi \mathbf{u} \cdot \mathbf{n} d\gamma = \sum_{f \in \partial V_P} \phi_f (\mathbf{u}_f \cdot \mathbf{s}_f) = \sum_{f \in \partial V_P} F_f \phi_f$$

- ϕ_f value of the variable ϕ at the cell face f
- $\mathbf{s}_f := \int_f \mathbf{n} d\gamma$
- $F_f := \mathbf{u}_f \cdot \mathbf{s}_f$ face flux

In a collocated grid we only have values at the cell centers, we need to **interpolate** these values to obtain ϕ_f

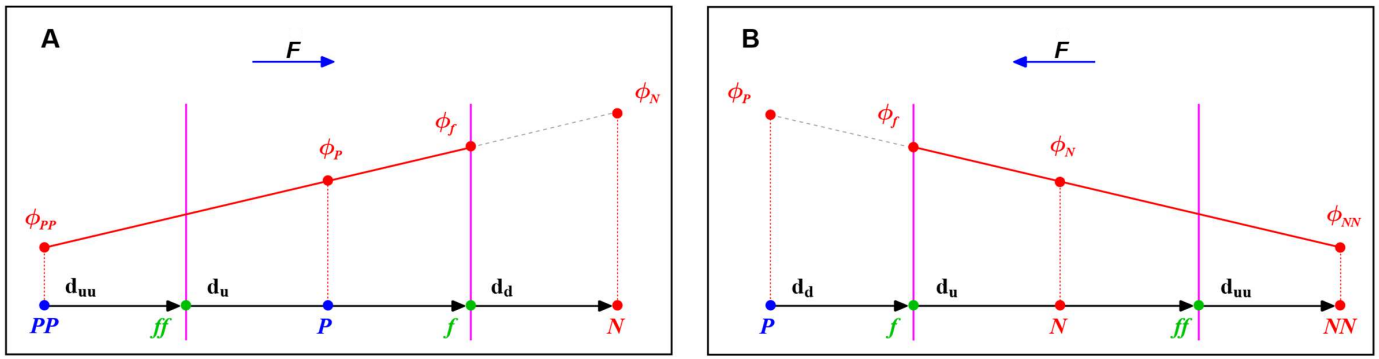
Convective Flux Interpolation

The face value could be computed by a central difference scheme

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

This is a second-order but **unstable** scheme

The convective flux interpolation must take into account the advection of the information



Credit: *Joel Guerrero*

Convective Flux Interpolation

- **Upwind**, 1st order, diffusive

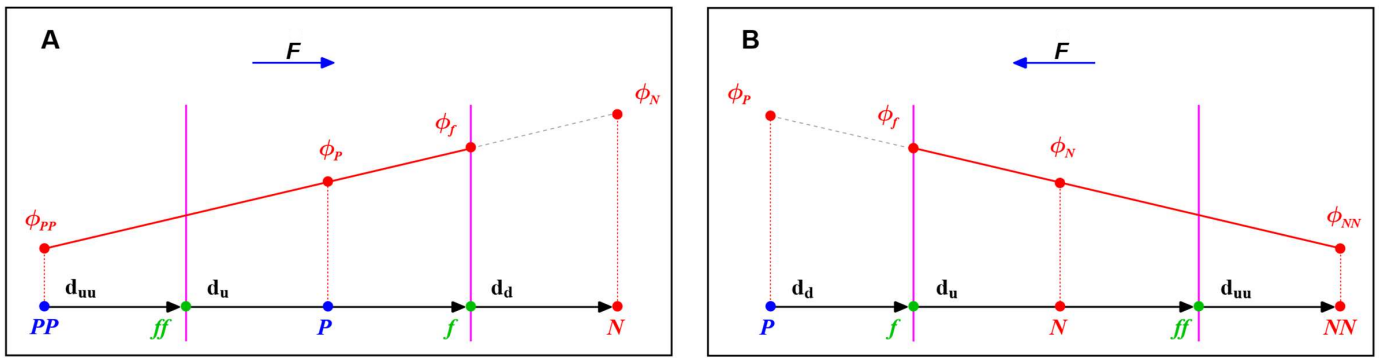
$$\phi_f = \begin{cases} \phi_P & \text{if } F \geq 0 \\ \phi_N & \text{if } F < 0 \end{cases}$$

- **Linear upwind**, 2nd order, oscillatory

$$\phi_f = \begin{cases} \phi_P + \frac{1}{2}(\phi_P - \phi_{PP}) & \text{if } F \geq 0 \\ \phi_N + \frac{1}{2}(\phi_{NN} - \phi_N) & \text{if } F < 0 \end{cases}$$

- **TVD**, 2nd order, the behavior depends on choice of ψ

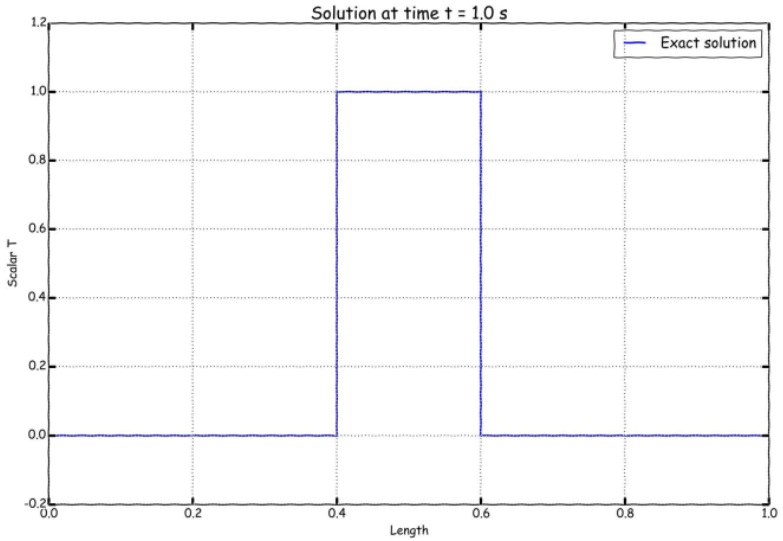
$$\phi_f = \begin{cases} \phi_P + \frac{1}{2}\psi_P^-(\phi_P - \phi_{PP}) & \text{if } F \geq 0 \\ \phi_N + \frac{1}{2}\psi_P^+(\phi_{NN} - \phi_N) & \text{if } F < 0 \end{cases}$$

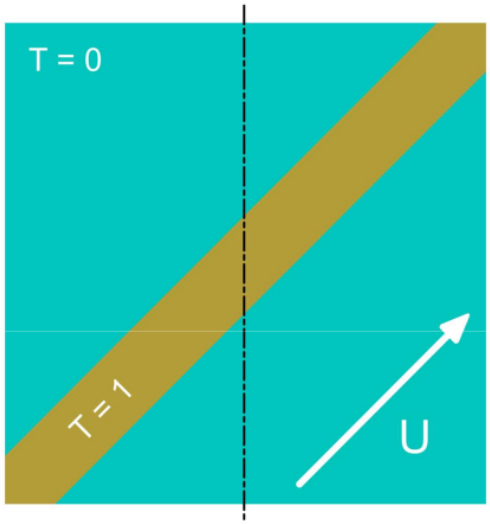


Credit: *Joel Guerrero*

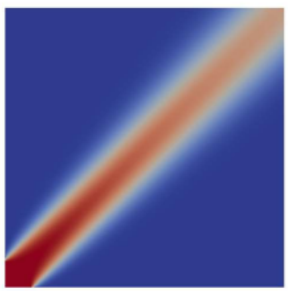
Convective Flux Interpolation

Test case: oblique double step profile, pure convection





Upwind – 1st order **Linear Upwind – 2nd order** **SuperBee – TVD**



Credit: *Joel Guerrero*

Diffusive Term

$$\int_{\partial V_P} (\mu \nabla \phi) \cdot \mathbf{n} d\gamma = \sum_{f \in \partial V_P} \int_f (\mu \nabla \phi) \cdot \mathbf{n} d\gamma = \sum_{f \in \partial V_P} \mu_f (\nabla \phi)_f \cdot \mathbf{s}_f$$

- μ_f computed by central difference of cell values
- In **orthogonal** grids, $\mathbf{s}_f \cdot \mathbf{d}_f = |\mathbf{s}_f||\mathbf{d}_f|$ with $\mathbf{d}_f := \mathbf{x}_P - \mathbf{x}_N$. Central difference approximation of the derivative, it is second order accurate.

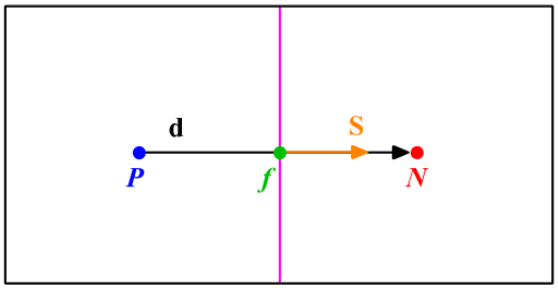
$$(\nabla \phi)_f \cdot \mathbf{s}_f = \frac{\phi_P - \phi_N}{|\mathbf{d}_f|} |\mathbf{s}_f|$$

- In **non-orthogonal** grids, a correction term is added

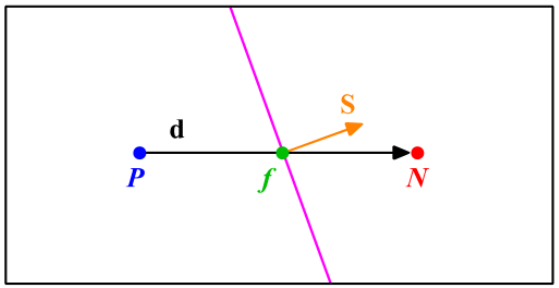
$$(\nabla \phi)_f \cdot \mathbf{s}_f = \underbrace{\boxed{\frac{\phi_P - \phi_N}{|\mathbf{d}_f|} \boldsymbol{\Delta}_\perp}}_{\text{Orthogonal term}} + \underbrace{\boxed{(\nabla \phi)_f \cdot \mathbf{k}}}_{\text{Correction}}, \quad \boldsymbol{\Delta}_\perp := \frac{\mathbf{d}_f}{\mathbf{d}_f \cdot \mathbf{s}_f} |\mathbf{s}_f|^2$$

Mesh Non-Orthogonality

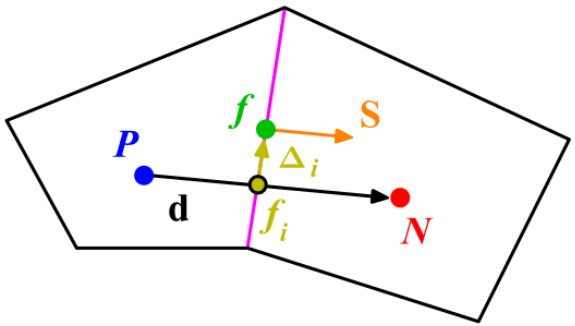
$$(\nabla\phi)_f \cdot \mathbf{s}_f = \underbrace{\boxed{\frac{\phi_P - \phi_N}{|\mathbf{d}_f|}\boldsymbol{\Delta}_\perp}}_{\text{Orthogonal term}} + \underbrace{\boxed{(\nabla\phi)_f \cdot \mathbf{k}}}_{\text{Correction}}$$



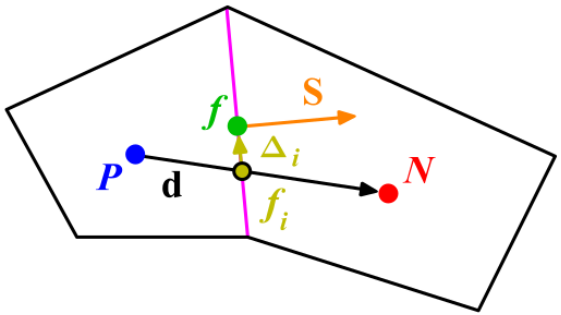
Orthogonal and non skew mesh



Non-orthogonal and non skew mesh



Orthogonal and skew mesh



Non-orthogonal and skew mesh

Credit: *Joel Guerrero*

Source Term and Boundary Conditions

Source Term

- $\int_{V_P} f dV = f_P V_p$
- If $f(\phi)$, it is generally linearized to improve stability and boundedness

Dirichlet Boundary Conditions

- $\phi_f = \phi_b$
- Convective term $\rightarrow$ $\mathbf{F}_f \phi_b$
- Diffusive term $\rightarrow$ $(\nabla\phi)_b \cdot \mathbf{s}_b = \frac{\phi_P - \phi_b}{|\mathbf{d}_f|}|\mathbf{s}_b|$

Neumann Boundary Conditions

- $\nabla\phi_b \cdot \mathbf{n}_b = g_b$
- Convective term $\rightarrow$ $\mathbf{F}_f \phi_b$, $\phi_b = \phi_P + \mathbf{d}_b \cdot (\nabla\phi)_b = \phi_P + |\mathbf{d}_b| g_b$
- Diffusive term $\rightarrow$ $(\nabla\phi)_b \cdot \mathbf{s}_b = g_b |\mathbf{s}_b|$

#CFD

Linear System of Equations for the Finite Volume Method

Matrix Assembly

- Writing the discrete equation for **each cell** inside the domain one obtains an equation of this type:

$$a_p x_p + \sum_N a_N x_N = b$$

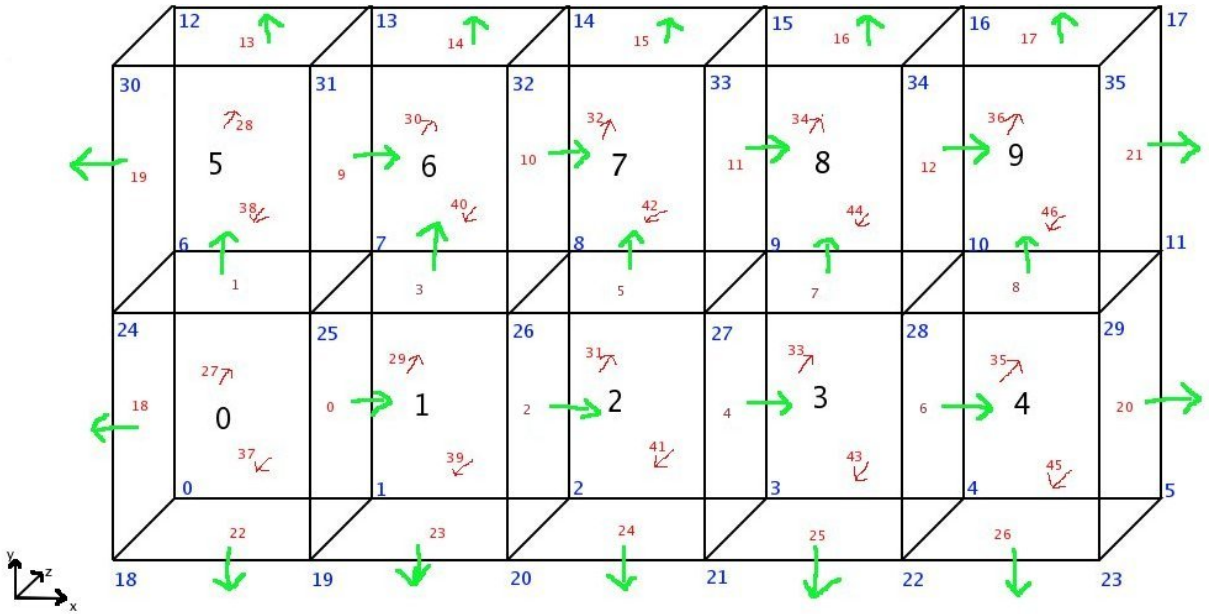
  - x_p and a_p are the unknown value relative to the cell for which we have written the conservation equation and the multiplying coefficients, respectively.
  - x_n and a_n assume the same meaning but related to all the neighbouring cells.
  - b includes the source term and other contributions.

- The equations for all the cells can be recast in **matrix form**:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

  - $\boldsymbol{A}$ is a **square matrix** of coefficients which is big ($N_{\text{cells}} \times N_{\text{cells}}$) and **sparse**.
  - $\boldsymbol{x}$ is the vector of unknowns.
  - $\boldsymbol{b}$ is the vector containing all the source terms and other contributions.

Matrix Assembly - Mesh Example



Now we see an example with only 10 cells:

- There are 36 **nodes**.
- There are 47 **faces**.
- In order to write the system of equations we have to write an equilibrium equation **for all the cells**.
- For a scalar quantity, the resulting system of equations will be of dimension $10 * 10$

Matrix Assembly - Mesh Example

The $\boldsymbol{A}$ matrix looks like this:

$$\begin{pmatrix}
a_{0,0} & a_{0,1} & 0 & 0 & 0 & a_{0,5} & 0 & 0 & 0 & 0 \\
a_{1,0} & a_{1,1} & a_{1,2} & 0 & 0 & 0 & a_{1,6} & 0 & 0 & 0 \\
0 & a_{2,1} & a_{2,2} & a_{2,3} & 0 & 0 & 0 & a_{2,7} & 0 & 0 \\
0 & 0 & a_{3,2} & a_{3,3} & a_{3,4} & 0 & 0 & 0 & a_{3,8} & 0 \\
0 & 0 & 0 & a_{4,3} & a_{4,4} & 0 & 0 & 0 & 0 & a_{4,9} \\
a_{5,0} & 0 & 0 & 0 & a_{5,5} & a_{5,6} & 0 & 0 & 0 & 0 \\
0 & a_{6,1} & 0 & 0 & a_{6,5} & a_{6,6} & a_{6,7} & 0 & 0 & 0 \\
0 & 0 & a_{7,2} & 0 & 0 & a_{7,6} & a_{7,7} & a_{7,8} & 0 & 0 \\
0 & 0 & 0 & a_{8,3} & 0 & 0 & a_{8,7} & a_{8,8} & a_{8,9} & 0 \\
0 & 0 & 0 & 0 & a_{9,4} & 0 & 0 & 0 & a_{9,8} & a_{9,9}
\end{pmatrix}$$

- The number of **diagonal elements** is equal to the **number of cells**.
- The number of **non-zero** upper and lower elements is equal to the **number of internal faces**.
- Also for such a simple case **most of the elements** are identically equal to **zero**.
- For this reason only non-zero elements are stored.

Linear Systems

- Once the linear system is assembled it is necessary to find a good **solution strategy** taking into account that the system matrix is potentially **big and sparse**.
- A **direct approach** is usually **not feasible**. We have to rely on **iterative solvers**:
  - We start from an initial guess x^0 and the iteration rule is:

$$x^{k+1} = Ax^k$$

  - We stop when the residual is smaller then a certain tolerance:

$$||Ax^{k+1} - b|| < tol$$

- In order to accelerate the solution procedure we usually precondition the system matrix in such a way that $\text{cond}(A) \approx 1$.
- We look for a matrix $\boldsymbol{M}$ that is easy to invert:

$$\boldsymbol{M}^{-1}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{M}^{-1}\boldsymbol{b}$$

- Easiest choice is to take $\boldsymbol{M} = \boldsymbol{I}$ (nothing changes)
- Best choice would be to take $\boldsymbol{M} = \boldsymbol{A}$ (not a good choice, $\boldsymbol{A}^{-1}$ is costly to obtain)

Different options are possible to obtain a preconditioner:

- Diagonal preconditioner $\rightarrow \boldsymbol{M} = \text{diag}(\boldsymbol{A})$
- Incomplete LU preconditioner $\boldsymbol{A} = \boldsymbol{LU} + \boldsymbol{R}$, $\boldsymbol{M} = \boldsymbol{LU}$
- Incomplete Cholesky preconditioner $\boldsymbol{A} = \boldsymbol{GG}^T + \boldsymbol{R}$, $\boldsymbol{M} = \boldsymbol{GG}^T$

Many options are also possible for the resolution of the linear system of equations, we list here some of them:

- Fixed-Point method
- Krilov Space solvers
  - The Conjugated Gradient Method
  - Bi-Conjugated Gradient method
  - Generalized Minimal Residual (GMRES) method
- Algebraic Multigrid
- ....

#CFD

Navier-Stokes Equations

Continuum Navier-Stokes equations

- **Continuity equation**

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

where ρ is the density of the fluid and $\mathbf{u}$ its punctual velocity;

- **Momentum equation**

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u}\mathbf{u}) - \nabla \cdot [\mu(\nabla \mathbf{u} + (\nabla \mathbf{u})^T)] = \rho \mathbf{g} - \nabla \left(P + \frac{2}{3}\mu \nabla \cdot \mathbf{u} \right)$$

where μ is the dynamic viscosity and $\mathbf{g}$ is the gravity force;

- **Energy equation**

$$\frac{\partial \rho e}{\partial t} + \nabla \cdot (\rho e \mathbf{u}\mathbf{u}) - \nabla \cdot (k \nabla T) = \rho \mathbf{g} \cdot \mathbf{u} - \nabla \cdot \left(\frac{2}{3}\mu (\nabla \cdot \mathbf{u}) \mathbf{u} \right) +$$

$$+ \nabla \cdot [\mu(\nabla \mathbf{u} + (\nabla \mathbf{u})^T) \cdot \mathbf{u}] + \rho Q$$

where k is the material conductivity while Q is the term related to possible sources;

- **State equation**

$$\rho = \rho(P, T)$$

Incompressible case

When the velocity of the fluid is sufficiently slow (**Mach number** < 0.3) the flow regime is low subsonic and pressure changes are not affecting the density anymore $\implies$ the flow is meant to be **incompressible** $\implies$ **constant density**.

Neglect of pressure-density link implies the neglect of thermodynamics $\implies$ the energy equation is now decoupled from the system.

Incompressible Navier-Stokes equations:

$$\begin{cases} \dfrac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} + \dfrac{\nabla P}{\rho} = \nu \nabla^2 \mathbf{u} \\ \nabla \cdot \mathbf{u} = 0 \end{cases}$$

where $\nu = \dfrac{\mu}{\rho}$ is the kinematic viscosity.

Pay attention:

- $\rho = const$ does not imply $\nabla \cdot (\rho \mathbf{u}) = 0 \implies$ is not the same as $\nabla \cdot \mathbf{u} = 0$;

Pressure-velocity coupling

From now on let us assume the density field to be not only constant but also **uniform** $\implies \nabla\rho = 0 \implies \nabla \cdot \mathbf{u} = 0$.

If we look at the momentum equation, it could easily be noticed that the **non-linearity** in the velocity terms would be handled by the use of an **iterative algorithm**. Unfortunately it is not that straightforward because of the presence of **pressure gradient**.

The continuity equation can be seen as a **scalar constraint** for the velocity field $\implies$ the pressure gradient is forcing term which has the role of **enforcing the divergence-free condition** on velocity field (it is much easier to be seen in Stokes equations where pressure can be considered as the Lagrange multiplier for the continuity constraint).

Why a segregated method is better?

Because of the coupling between pressure and velocity the two Navier-Stokes equations can not be solved separately.

We may think about them as a system. There exist a bunch of different methods for linear systems but in this case the convection term is **non-linear**. Let us analyze two different solution to the problem:

1. **Iterative block solvers**;
2. **Iterative segregated solvers**.

In the first case a **very big matrix** has to be stored. The second one is then to be preferred from a performance point of view: it requires 1/4 of the storage needed by block solvers. The difficulty about segregated methods is that we need to find out **two decoupled equations**, one for velocity and the other one for pressure, to be solved iteratively and an algorithm to link them between each others.

Derivation of an equation for pressure

The idea is to rearrange continuity equation into an equation for pressure while using the momentum one for velocity.

Let us take the divergence of the incompressible momentum equation:

$$\frac{\partial}{\partial x}\frac{\partial u}{\partial t}+\frac{\partial u}{\partial x}\frac{\partial u}{\partial x}+u\frac{\partial^2 u}{\partial x^2}+\frac{\partial v}{\partial x}\frac{\partial u}{\partial y}+v\frac{\partial^2 u}{\partial x \partial y}+\frac{\partial}{\partial y}\frac{\partial v}{\partial t}+\frac{\partial u}{\partial y}\frac{\partial v}{\partial x}+u\frac{\partial^2 v}{\partial x \partial y}+\frac{\partial v}{\partial y}\frac{\partial v}{\partial y}+$$
$$+v\frac{\partial^2 v}{\partial y^2}=-\frac{1}{\rho}\frac{\partial^2 P}{\partial x^2}+\nu\left(\frac{\partial^3 u}{\partial x^3}+\frac{\partial^3 u}{\partial x \partial y^2}\right)-\frac{1}{\rho}\frac{\partial^2 P}{\partial y^2}+\nu\left(\frac{\partial^3 v}{\partial y^3}+\frac{\partial^3 v}{\partial x^2 \partial y}\right)$$

where $\mathbf{u}=(u,v)$. Summing everything up and applying the continuity constrain we get:

$$\left(\frac{\partial u}{\partial x}\right)^2+2\frac{\partial u}{\partial y}\frac{\partial v}{\partial x}+\left(\frac{\partial v}{\partial y}\right)^2=-\frac{1}{\rho}\left(\frac{\partial^2 P}{\partial x^2}+\frac{\partial^2 P}{\partial y^2}\right)$$

The final result is:

$$\nabla^2 P=-\rho\left[\left(\frac{\partial u}{\partial x}\right)^2+2\frac{\partial u}{\partial y}\frac{\partial v}{\partial x}+\left(\frac{\partial v}{\partial y}\right)^2\right]$$

that is a Poisson Equation for pressure, which ensures that continuity is satisfied.

Semi-Implicit Algorithm for Pressure-Linked Equations

It is possible to write down the momentum equation in a more comfortable way:

$$\mathbf{A}\mathbf{u}'' = \mathbf{H}(\mathbf{u}') - \nabla P'$$

where $\mathbf{A}\mathbf{u}$ is the diagonal part of the momentum operator related to velocity, $\mathbf{H}(\mathbf{u})$ is its extra diagonal part, $(\cdot)''$ terms are the unknowns while $(\cdot)'$ are the variables evaluated at the previous iteration step.

The algorithm itself is composed by few very clear steps:

1. guess beginning pressure and velocity fields P^* and $\mathbf{u}^*$ respectively and set $P' = P^*$ and $\mathbf{u}' = \mathbf{u}^*$;
2. solve **momentum predictor** step: $\mathbf{u}'' = \mathbf{A}^{-1}(\mathbf{H}(\mathbf{u}') - \nabla P')$;
3. solve **pressure correction** step: $\nabla^2 P'' = \mathbf{F}(\mathbf{u}'')$;
4. if convergence check is not satisfied put $\mathbf{u}' = \mathbf{u}''$, $P' = P''$ and repeat from point 2 on, otherwise stop.

Pay attention: usually this algorithm is unstable and it does not converge.

An easy way to overtake the problem is to apply **under-relaxation**:

$$P'' = P' + \alpha_P(P'' - P') \qquad \mathbf{u}'' = \mathbf{u}' + \alpha_u(\mathbf{u}'' - \mathbf{u}')$$

where $0 < \alpha_P < 1$, $0 < \alpha_u < 1$ and $\alpha_P + \alpha_u \approx 1$.

What about fluxes?

As seen above, continuity equation has been turned into a pressure equation. So what guarantees divergence-free property of the flow is now pressure itself. How can we evaluate **face fluxes** so that this property is still valid?

$$\nabla \cdot \mathbf{u} = 0 \implies \int_{\Omega} \nabla \cdot \mathbf{u} = 0 \implies \sum_i \int_{\Omega_i} \nabla \cdot \mathbf{u} = 0 \implies \sum_i \int_{\delta\Omega_i} \mathbf{n} \cdot \mathbf{u} = 0$$

$$\implies \sum_i \sum_f (\mathbf{s}_i)_f \cdot \mathbf{u} = 0 \implies \sum_i \sum_f F = 0$$

where Ω is the global domain, Ω_i is one of the sub-domains used to discretize Ω, $\delta\Omega_i$ is its surface, $(\mathbf{s}_i)_f$ is one of the oriented faces of $\delta\Omega_i$ and F is the out-going flux related to $(\mathbf{s}_i)_f$.

It turn out:

$$F = -\mathbf{A}^{-1}(\mathbf{s}_i)_f \cdot \nabla P + \mathbf{A}^{-1}(\mathbf{s}_i)_f \cdot \mathbf{H}(\mathbf{u})$$

by using the expression for $\mathbf{u}$ coming from the momentum equation. So in order to have conservative face fluxes, they have to be assembled in a consistent way with respect to pressure equation.

Is the same algorithm useful in compressible case?

As we have seen before, a segregated algorithm is a good choice for incompressible flows solvers. Anyway the SIMPLE algorithm can be modified in order to be used in the compressible case.
What is different?

1. **density** is no more a constant;
2. the **energy** equation has to be included in the algorithm;
3. a **state** equation is needed in order to close the problem.

To overcome these points first of all a **gas model** has to be chosen. By the use of the consequent equation of state it is possible to write down a new equation for pressure. Moreover, once the gas model has been fixed, it is possible to find out the relation between energy and temperature.
Through all these steps the SIMPLE algorithm can be turned into its compressible formulation.

Some examples

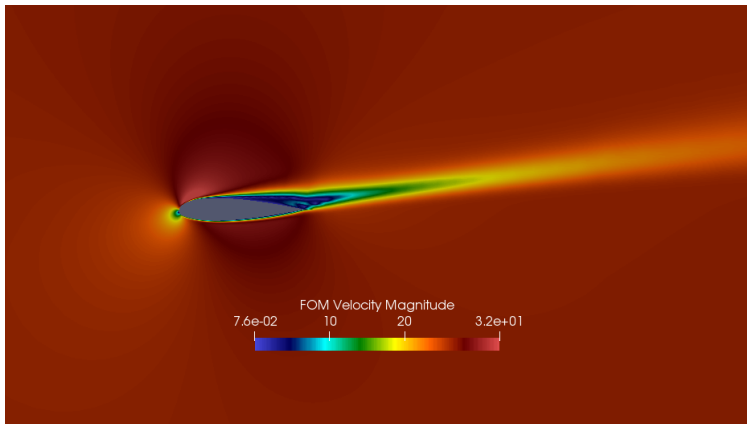
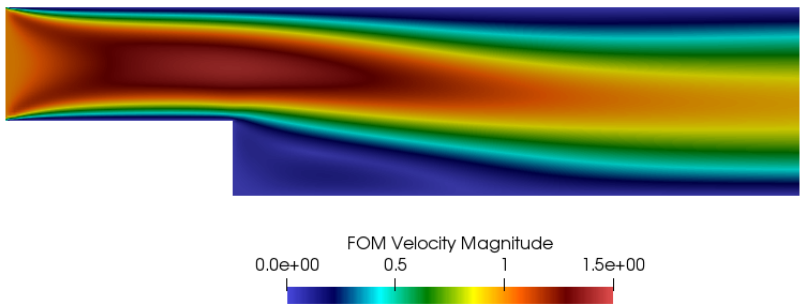


Figure: Full order velocity field for an air foil moving into an incompressible flow.

Figure: Backstep problem velocity field for an incompressible fluid.



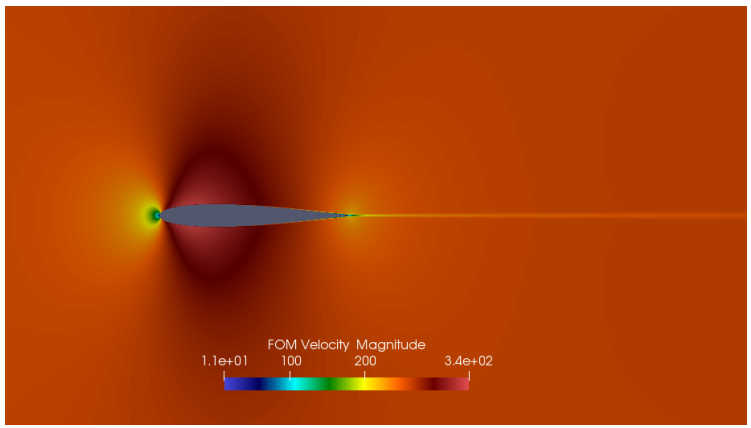


Figure: Full order velocity field for an air foil moving into a compressible flow.

#CFD

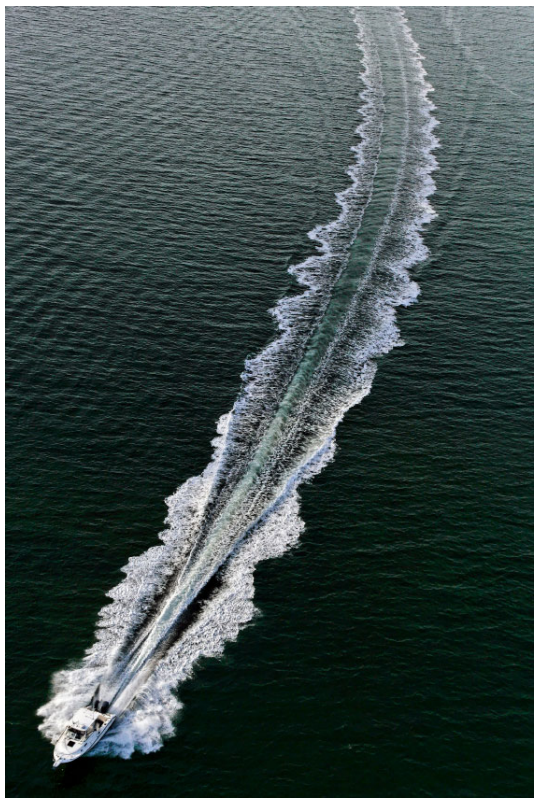
Turbulence Modeling

What is turbulence?

- Turbulence is a physical phenomenon which occurs in many fields in life.

Examples of turbulent flows

- flow around aircraft wing tips.
- flow in boat wakes.
- atmosphere and ocean currents.



Characteristics of turbulent flows

- Turbulence is very chaotic phenomenon which is irregular, random, disorder with no-repeatability.
- Turbulent flows are highly non-linear, non-stationary and three-dimensional flows.
- Turbulent flows are characterized with the existence of an energy cascade.
- In turbulent flows momentum and energy are of high diffusivity.
- Turbulence may occur in some parts of the domain and this is referred as intermittency.
- High concentration and intensity of vorticity.
- In turbulence a situation of laminar-turbulent transition may occur such as in free shear flows.

- Let's introduce incompressible Navier–Stokes equations for the vorticity $\boldsymbol{\omega} = \boldsymbol{\nabla} \times \boldsymbol{u}$:

$$\begin{cases} \textbf{2D}, & \partial_t \boldsymbol{\omega} + \boldsymbol{u} \cdot \boldsymbol{\nabla} \boldsymbol{\omega} = \nu \boldsymbol{\nabla}^2 \boldsymbol{\omega}, \\ \textbf{3D}, & \partial_t \boldsymbol{\omega} + \boldsymbol{u} \cdot \boldsymbol{\nabla} \boldsymbol{\omega} = \boxed{\boldsymbol{\omega} \cdot \boldsymbol{\nabla} \boldsymbol{u}} + \nu \boldsymbol{\nabla}^2 \boldsymbol{\omega}. \end{cases}$$

- In 2D, the energy, $\frac{1}{2}|\boldsymbol{u}^2|$, is transferred toward large scales and the enstrophy, $\frac{1}{2}|\boldsymbol{\omega}^2|$, is transferred toward small scales.

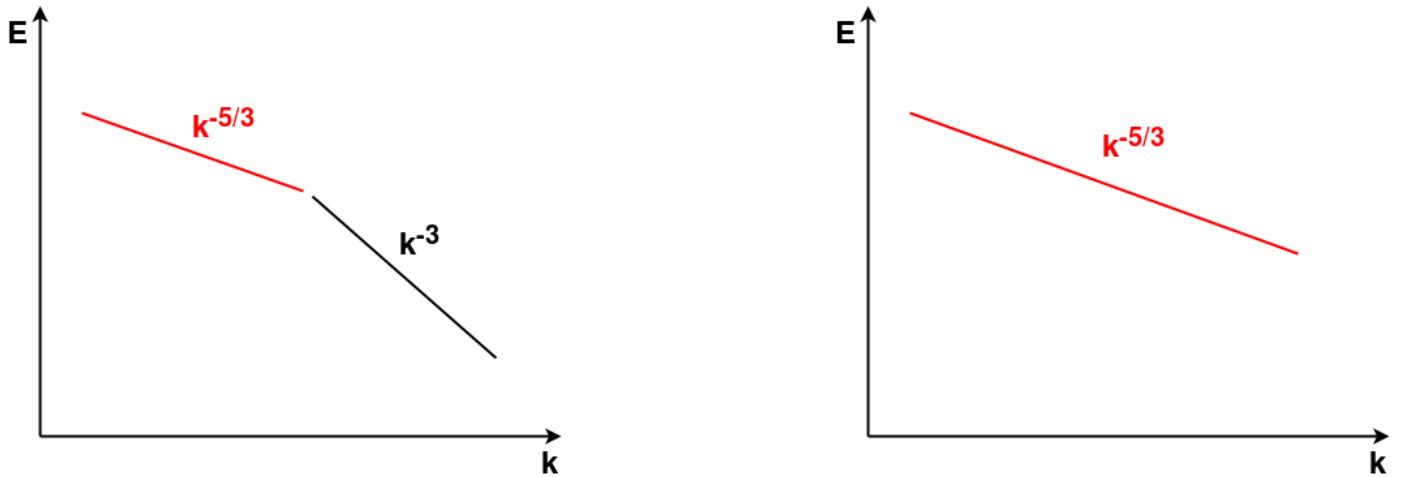


Figure: 2D (on the left) and 3D (on the right) energy spectrum in the inertial range.

How to model turbulence?

- There are three main methodologies to deal with turbulence
- **Direct Numerical Simulation (DNS)**
- **Reynolds-Averaged Navier-Stokes Equations (RANS)**
- **Large Eddy Simulation (LES)**

Direct Numerical Simulation (DNS)

- **DNS** is a method of solving the turbulent flows in which one performs the simulation for all the scales. In light of that, DNS is NOT a turbulence model.
- In order to do so, sufficient spatial and temporal resolutions are needed. This implies that we need high computer capabilities for performing the simulations.
- All the scales from the smallest dissipative scales (Kolmogorov microscales), up to the integral scale L are resolved. The Kolmogorov scale η is related to integral scale and the Reynolds number through this relation :

$$\frac{L}{\eta} \sim (Re)^{\frac{3}{4}}.$$

- The last requirement restricts the problems DNS can deal with to be moderate Re problems with simple geometries.

Reynolds-Averaged Navier-Stokes Equations (RANS)

- **RANS** is a different approach which is based on solving the fluid dynamics equations for the mean properties.
- This is motivated by the assumption that "turbulent fluctuations" are of no interest in many circumstances. Therefore one solves for the **mean** part of each fluid dynamics variable.
- The last assumption translates into the following decomposition of pressure and velocity fields into a **mean** part and another term which represents the **fluctuating** part that has zero mean:

$$\boldsymbol{u}_i = \overline{\boldsymbol{u}_i} + \boldsymbol{u}_i', \quad p = \overline{p} + p'.$$

This is called the Reynolds decomposition of the flow fields.

Turbulence Modelling

- After substituting the previous equations in the Navier-Stokes equations and time averaging them, one will be able to eliminate the fluctuating terms thanks to the assumption that their average or mean is zero:

$$\begin{cases} \frac{\partial \overline{\boldsymbol{u}}}{\partial t} + \boldsymbol{\nabla} \cdot (\overline{\boldsymbol{u}} \otimes \overline{\boldsymbol{u}}) - \boldsymbol{\nabla} \cdot (\nu \boldsymbol{\nabla} \overline{\boldsymbol{u}}) = -\boldsymbol{\nabla} \overline{p} + \boldsymbol{\nabla} \cdot \left(\overline{\boldsymbol{u}' \boldsymbol{u}'} \right), \\ \boldsymbol{\nabla} \cdot \overline{\boldsymbol{u}} = 0. \end{cases}$$

- The new term $\boldsymbol{R} = \overline{\boldsymbol{u}' \boldsymbol{u}'}$ is called the **Reynolds stress tensor** which is a second rank symmetric tensor.
- In order to close the system, there are two ideas:
- The usage of **Eddy Viscosity Models (EVMs)**, in which we try to model $\boldsymbol{R}$ by an algebraic equation as follows:

$$\boldsymbol{R} = \boldsymbol{f}(\overline{\boldsymbol{u}}, \overline{p})$$

- The second idea is to add more equations, for example a transport equation for $\boldsymbol{R}$.

Turbulence Modelling

- $\boldsymbol{R}$ must be expressed in terms of the mean part of the flow variables so as to obtain a closed problem.
- In the **EVMs** approach, **Boussinesq** assumption is used which assumes that $\boldsymbol{R}$ is a function of the symmetric velocity gradient $\overline{\boldsymbol{S}}$, i.e. $\boldsymbol{R} = \boldsymbol{f}(\overline{\boldsymbol{S}})$, where $\overline{\boldsymbol{S}} = \frac{1}{2}[\boldsymbol{\nabla}\overline{\boldsymbol{u}} + (\boldsymbol{\nabla}\overline{\boldsymbol{u}})^T]$
- Using dimensionality analysis, one can see the need for a pre-factor multiplication which has the same unit of the physical viscosity ν.
- Conveniently this term will be called the **turbulent viscosity** ν_t
- The **Boussinesq** assumption reads as follows:

$$\mathcal{R} = \frac{\nu_t}{2}[\boldsymbol{\nabla}\overline{\boldsymbol{u}} + (\boldsymbol{\nabla}\overline{\boldsymbol{u}})^T].$$

- The turbulent viscosity ν_t has to depend on the solution, it is reasonable to think that it depends on a length and time-scale (since it is measured in m^2/s).
- The velocity scale U is chosen as the time-scale, here it is the size of $\boldsymbol{u}'$, while the length scale Δ is set as the size of the energy-containing vortices.
- Therefore we will have

$$\nu_t = AU\Delta,$$

where A is a scaling constant to be tuned.

Turbulence Modelling

- The **EVMs** have been created in order to approximate the turbulent viscosity ν_t.
- They differ in the way of doing that, there are zero, one and two equations models available for the treatment of the additional turbulent/eddy viscosity.
- **Smagorinsky** model as Zero equation model : involves the approximation of the ν_t by the following:

$$\nu_t \approx (C_s \Delta)^2 |\overline{\boldsymbol{S}}|,$$

where C_S is the Smagorinsky "constant".

- $k - \epsilon$ as two equations model: in this model we have k the turbulent kinetic energy and ϵ the turbulent dissipation being found by solving two additional transport diffusion equations.
- k and ϵ are defined as follows

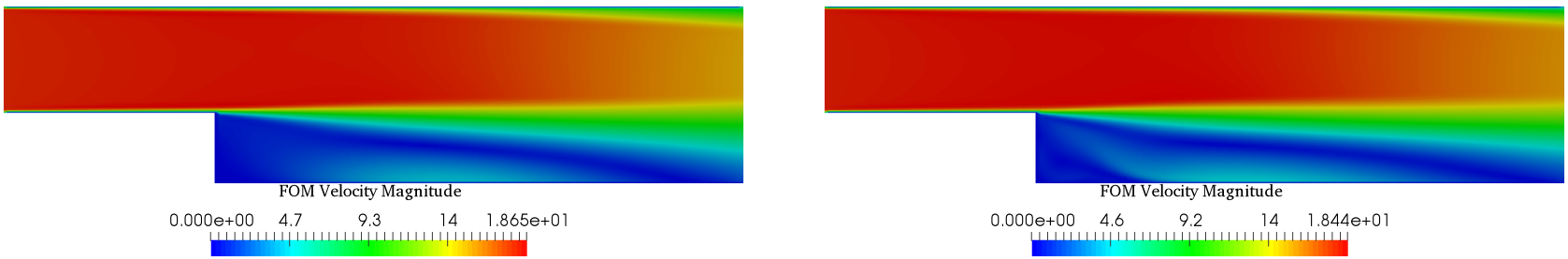
$$k = \frac{3}{2}\boldsymbol{u}^{2'}, \quad \epsilon = C_\epsilon \frac{k^{\frac{3}{2}}}{\Delta}.$$

- The additional equations can be derived by the momentum equation and then ν_t is approximated as follows

$$\nu_t \approx C_\mu \frac{k^2}{\epsilon}$$

Turbulence Modelling

- **Spalart–Allmaras** turbulence model is an example of one equation model, where it solves a transport equation for a viscosity-like variable called $\tilde{\nu}$.
- There is another common two equations model called the $k-\omega$ where ω is the specific turbulence dissipation rate.
- An example for the numerical results obtained by both $k-\epsilon$ and $k-\omega$ models can be seen below for the backstep case.
- In this case, the Reynolds number is about 17848.



(a) (b)

Figure: Different turbulence models results for the case of the backstep: (a) shows the velocity in the case of $k-\epsilon$ turbulence model while in (b) one can see the velocity in the case of $k-\omega$ turbulence model.

Near-wall treatment

- We must take into consideration the effect of having walls on modeling turbulence.
- Velocity and turbulence model variables have very steep gradients near the wall.
- In this case one has to either resolve for the cells in the boundary layer and in this case the mesh will be fine enough to get small values of the dimensionless wall distance $y^+ = \frac{u_* y}{\nu}$ (less than 1), with u_* being the friction velocity near the wall and y is the distance to the nearest wall.
- Or to use a coarse mesh near the wall and to model the attached flows by the so-called **wall functions**. This is typically done when we are not interested in the behavior near the wall and in this case the value of y^+ will be around $30 - 50$.

Large Eddy Simulations (LES)

- **LES** is a third approach which is based on filtering and solving the Navier–Stokes equations just for specific scales which are the large scales.
- The approach of time averaging used in **RANS** could cause the loss of important part of the physics of the problem if the fluctuation magnitude is relatively high with respect to the mean.
- Here comes the idea of **LES** where it models the small scales and simulates the larger ones.
- Filtered Navier-Stokes equations are obtained by carrying out the filtering as follows:

$$\overline{\boldsymbol{u}} = \int \boldsymbol{G}(\boldsymbol{x}, \boldsymbol{x}')\boldsymbol{u}(\boldsymbol{x}')d\boldsymbol{x}'$$

  where $\boldsymbol{G}(\boldsymbol{x}, \boldsymbol{x}')$ is the localized filter function.
- The filtered Navier–Stokes equations read as follows:

$$\frac{\partial \overline{\boldsymbol{u}}}{\partial t} + \boldsymbol{\nabla} \bullet (\overline{\boldsymbol{u}}\,\overline{\boldsymbol{u}}) - \boldsymbol{\nabla} \bullet (\nu \boldsymbol{\nabla} \overline{\boldsymbol{u}}) = -\boldsymbol{\nabla} \overline{p} + \boldsymbol{\nabla} \bullet \boldsymbol{\tau},$$

  with

$$\boldsymbol{\tau} = (\overline{\overline{\boldsymbol{u}}\,\overline{\boldsymbol{u}}} - \overline{\boldsymbol{u}}\,\overline{\boldsymbol{u}}) + (\overline{\overline{\boldsymbol{u}}\,\boldsymbol{u}'} + \overline{\boldsymbol{u}'\,\overline{\boldsymbol{u}}}) + \overline{\boldsymbol{u}'\,\boldsymbol{u}'}$$