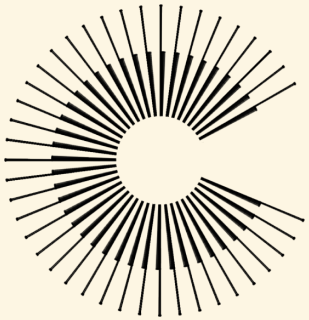
HPC: Accelerating Progress in CFD

Radouan Boukharfane

Mohammed VI Polytechnic University (UM6P), Benguerir, Morocco



College of Computing UM6P

UM6P University Mohammed VI Polytechnic

UM6P, Benguerir, 29th April 2024

HPC/AI Days

Overview

Introduction to HPC for CFD
- What is CFD?
- Why use CFD?
- Where is CFD used?
- Why use HPC for CFD?
- Where is CFD headed in light of NASA2030's vision?
- What are the main CFD methods?

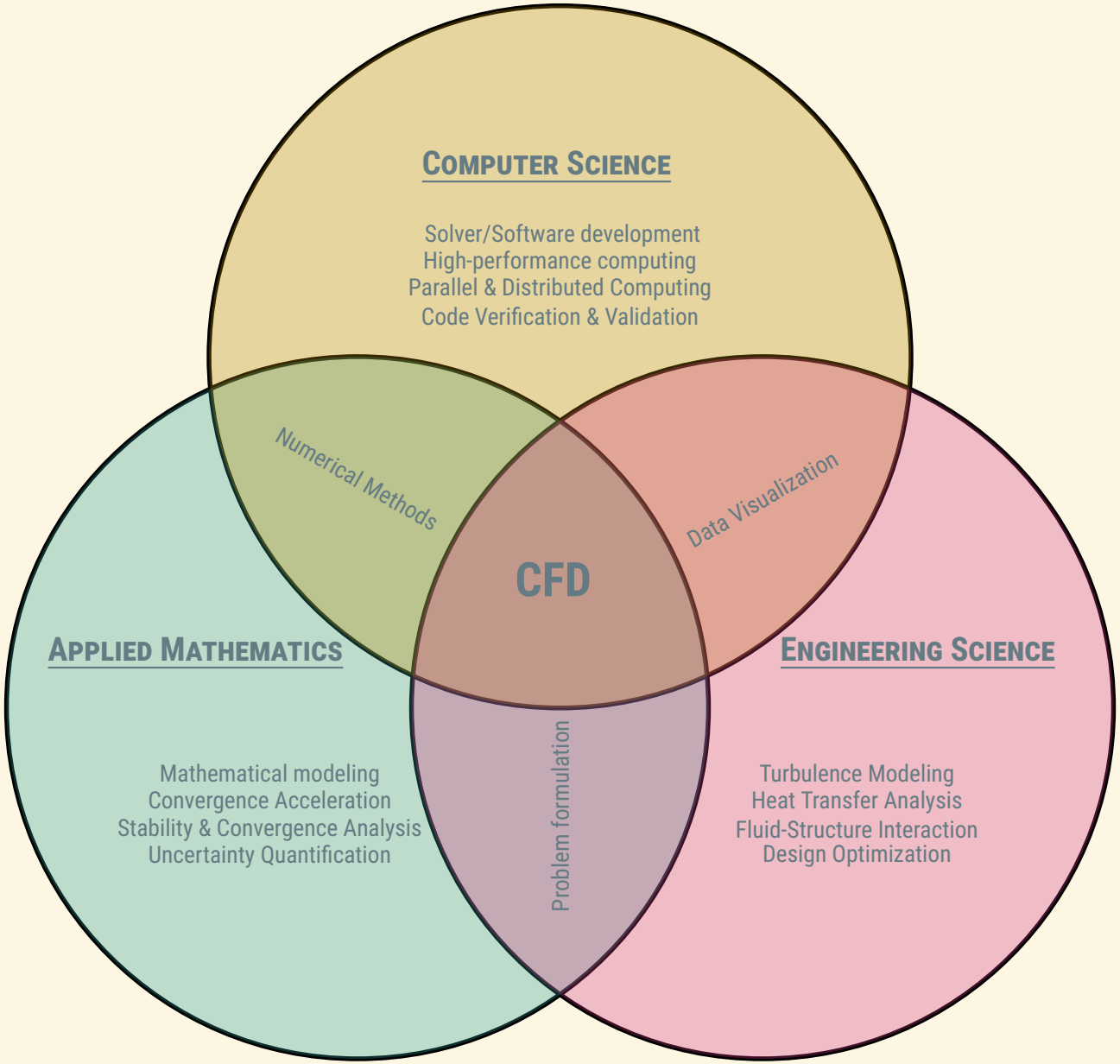
LES of Wakes of Wind Turbines Operating in Turbulent Inflow
- Presentation of the numerical framework
- Operating Optimization of ALM
- Application to small wind tunnel turbines
- Future work

Outline

What is CFD?

- Fluid dynamics
  - Fluid dynamics is a subdiscipline of fluid mechanics that describes the flow of fluids.
  - Fluid flow is commonly studied in one of three methods:
    - Analytical methods
    - Experimental methods
    - Numerical methods: Computational fluid dynamics (CFD)
- What is CFD?
  - Computational fluid dynamics (CFD) is a branch of fluid mechanics that uses numerical analysis and data structures to analyze and solve problems that involve fluid flows.
  - Key governing equations: Navier-Stokes equation, Continuity equation, energy equation.
  - Numerical methods: Finite Difference Method (FDM), Finite Element Method FEM), Finite Volume Method (FVM), ...

What is CFD?

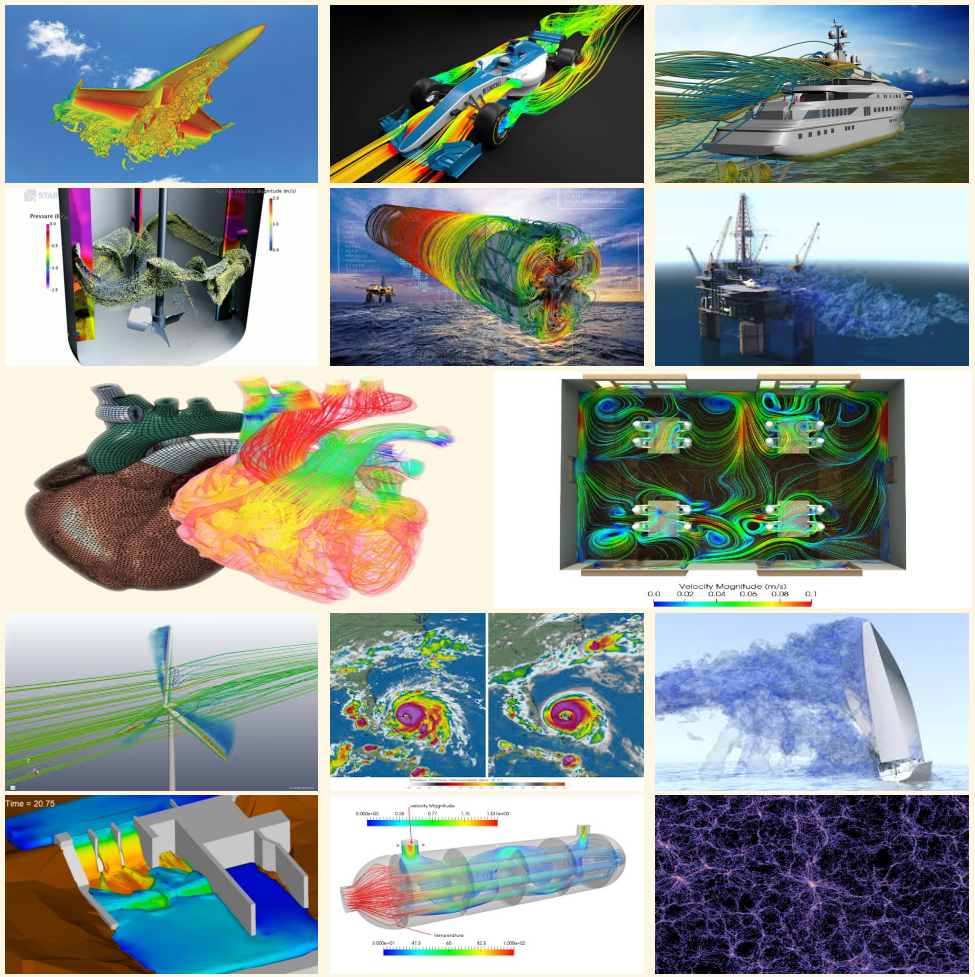

COMPUTER SCIENCE
Solver/Software development
High-performance computing
Parallel & Distributed Computing
Code Verification & Validation
Numerical Methods
Data Visualization
CFD
APPLIED MATHEMATICS
Mathematical modeling
Convergence Acceleration
Stability & Convergence Analysis
Uncertainty Quantification
Problem formulation
ENGINEERING SCIENCE
Turbulence Modeling
Heat Transfer Analysis
Fluid-Structure Interaction
Design Optimization

Why use CFD?

- Numerical solutions
  - Analytical solutions of some Patial Differential Equations (PDEs) cannot be obtained currently. So Analytical methods are limited to simplified cases such as solving 1D or 2D geometry, 1D flow, and steady flow.
- Relatively low cost
  - Experimental methods need a lot of resources such as electricity, expensive equipment, data monitoring, and data post-processing.
  - CFD simulations are relatively inexpensive, and costs are likely to decrease as computers become more powerful.
- Ability to simulate any conditions
  - CFD provides the ability to theoretically simulate any physical condition.
- Comprehensive information
  - CFD allows the analyst to examine a large number of locations in the region of interest, and yields a comprehensive set of flow parameters for examination.

Where is CFD used?

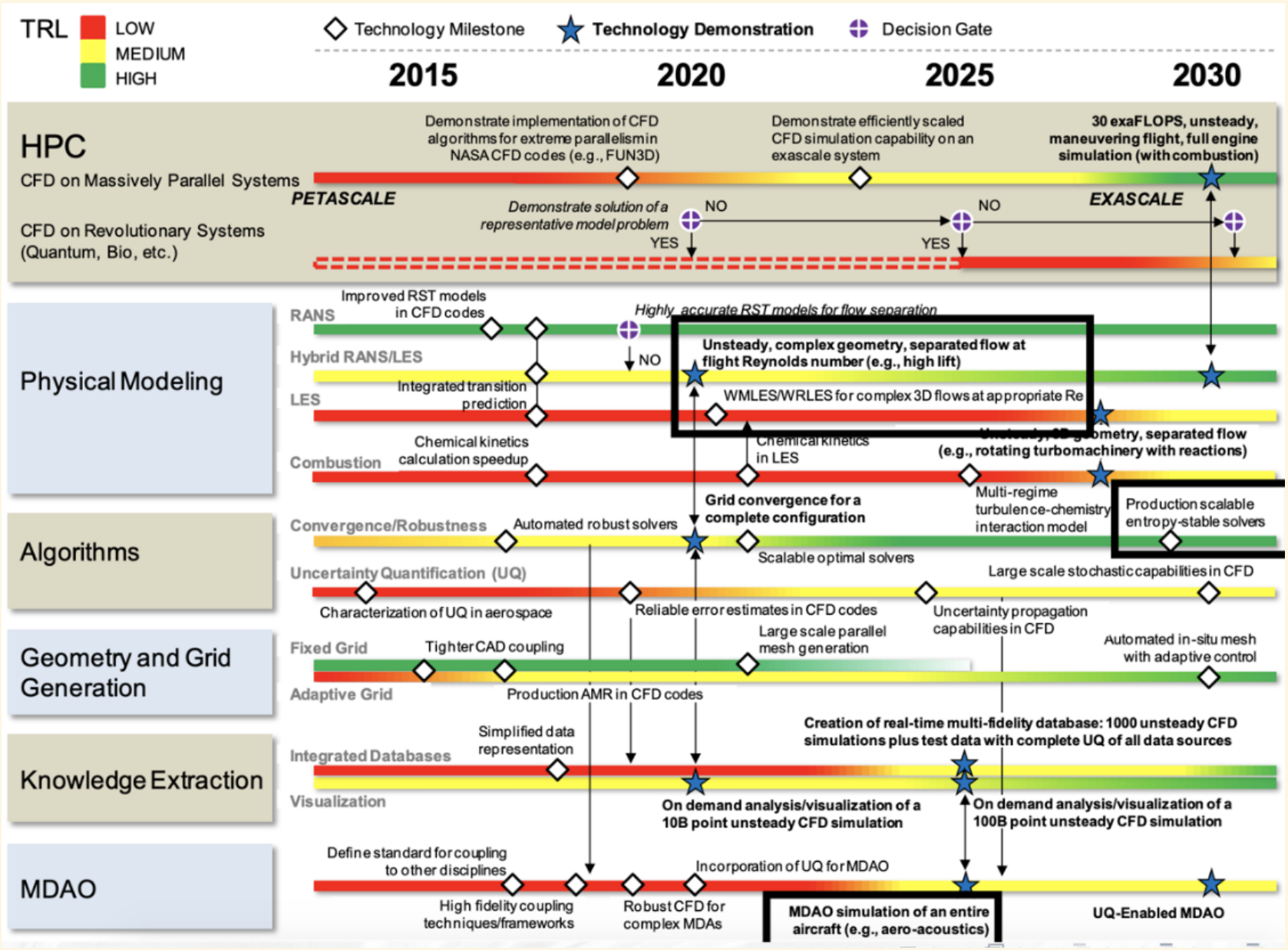
- **CFD is used in**:
  - Aerospace
  - Automotive
  - Chemical Processing
  - Hydraulics
  - Marine
  - Oil & Gas
  - Power Generation
  - Weather forecasting
  - Ocean
  - ...



Why use HPC for CFD?

- CFD: A Leading consumer of High-Performance Computing
- Speed
  - With the help of HPC, CFD simulations can be executed in a short period of time, which means engineering data can be introduced early in the design process.
- *Memory*
  - *A large CFD simulation can't typically fit into the memory of a single machine.*

Where is CFD headed in light of NASA2030's vision?



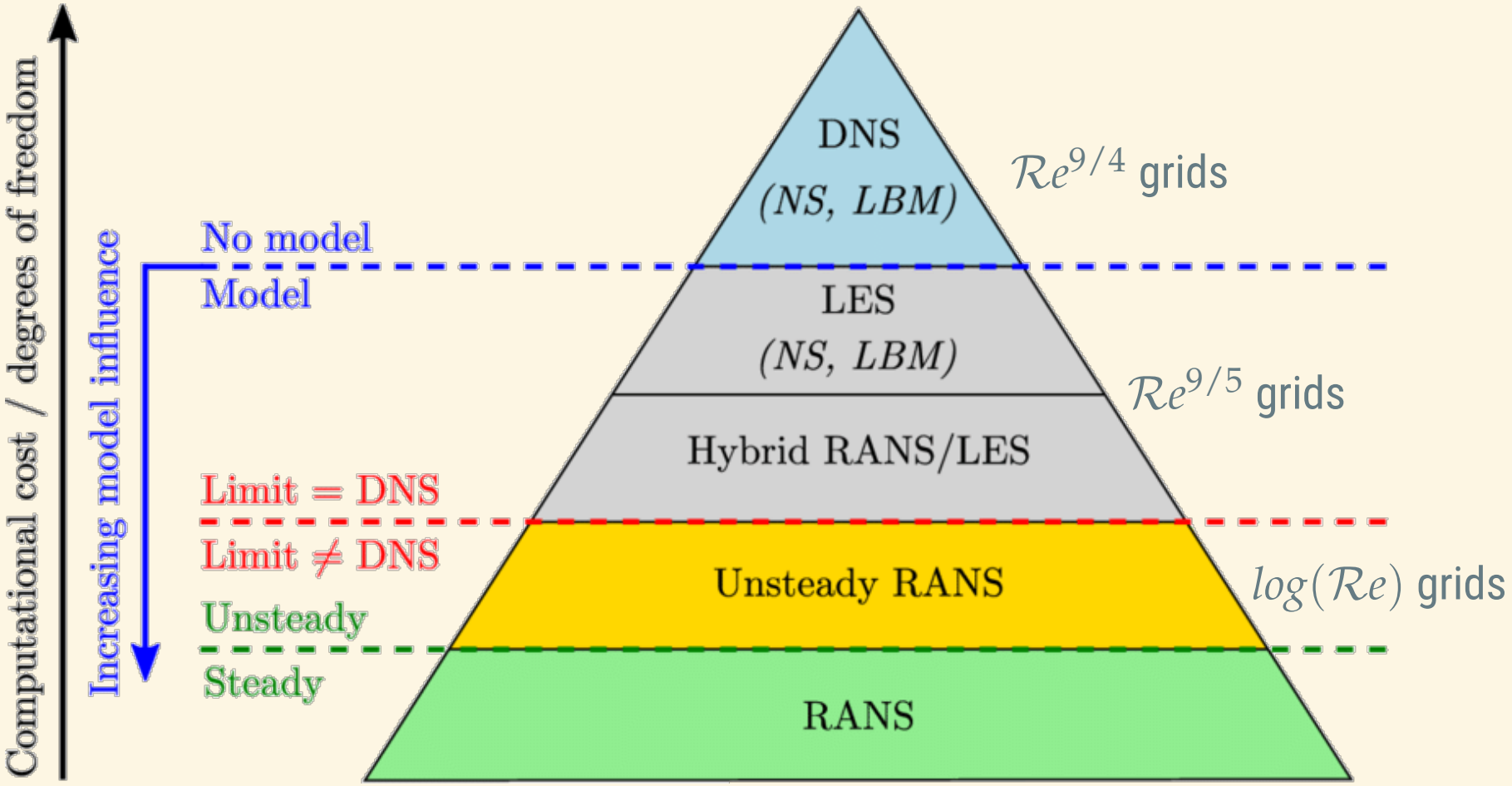
What are the main CFD methods?

$$
\begin{cases}
\dfrac{\partial \overline{\rho}}{\partial t} + \dfrac{\partial \overline{\rho}\widetilde{u}_i}{\partial x_i} = 0 \\
\dfrac{\partial \overline{\rho}\widetilde{u}_i}{\partial t} + \dfrac{\partial \overline{\rho}\widetilde{u}_i\widetilde{u}_j}{\partial x_j} = -\dfrac{\partial \overline{p}}{\partial x_i} + \dfrac{\partial \check{\tau}_{ij}}{\partial x_j} - \dfrac{\partial \tau_{ij}^{\text{sgs}}}{\partial x_j} + \dfrac{\partial}{\partial x_j}\left(\overline{\tau}_{ij} - \check{\tau}_{ij}\right) \\
\dfrac{\partial \overline{\rho}\widetilde{Y}_\alpha}{\partial t} + \dfrac{\partial \overline{\rho}\widetilde{Y}_\alpha\widetilde{u}_i}{\partial x_i} = -\dfrac{\partial \check{J}_{\alpha i}}{\partial x_i} - \dfrac{\partial J_{\alpha i}^{\text{sgs}}}{\partial x_i} - \dfrac{\partial}{\partial x_i}\left(\bar{J}_{\alpha i} - \check{J}_{\alpha i}\right) + \overline{\rho}\widetilde{\dot{\omega}}_\alpha \\
\dfrac{\partial \bar{\rho}\check{e}_t}{\partial t} + \dfrac{\partial \bar{\rho}\check{e}_t\tilde{u}_j}{\partial x_j} = -\dfrac{\partial \bar{p}\tilde{u}_j}{\partial x_j} + \dfrac{\partial \check{\tau}_{ij}\tilde{u}_i}{\partial x_j} - \dfrac{\partial \check{q}_j}{\partial x_j} - (B_1 + B_2 + B_3) + (B_4 + B_5 + B_6) - B_7
\end{cases}
\tag{1}
$$

with

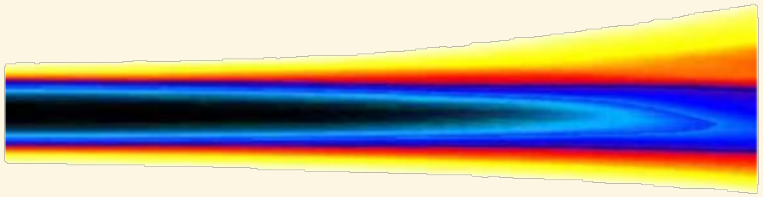
$$
\begin{cases}
B_1 = \dfrac{\partial}{\partial x_j}(e\bar{u}_j - \tilde{e}\tilde{u}_j) \approx \dfrac{\partial \tilde{c}_v \theta_j^{\text{sgs}}}{\partial x_j} \\
B_2 = p\dfrac{\bar{\partial} u_j}{\partial x_j} - \bar{p}\dfrac{\partial \tilde{u}_j}{\partial x_j} = \Pi_{\text{dil}}^{\text{sgs}} \\
B_3 = \dfrac{\partial}{\partial x_j}(\tau_{ij}^{\text{sgs}}\tilde{u}_i) \\
B_4 = \tau_{ij}^{\text{sgs}}\dfrac{\partial}{\partial x_j}\tilde{u}_i \\
B_5 = \tau_{ij}\dfrac{\bar{\partial}}{\partial x_j}u_i - \bar{\tau}_{ij}\dfrac{\partial}{\partial x_j}\tilde{u}_i = \epsilon_v^{\text{sgs}}
\end{cases}
\tag{2}
$$

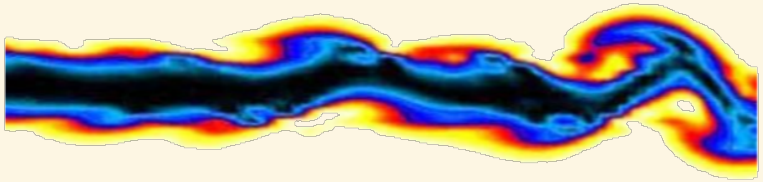
What are the main CFD methods?

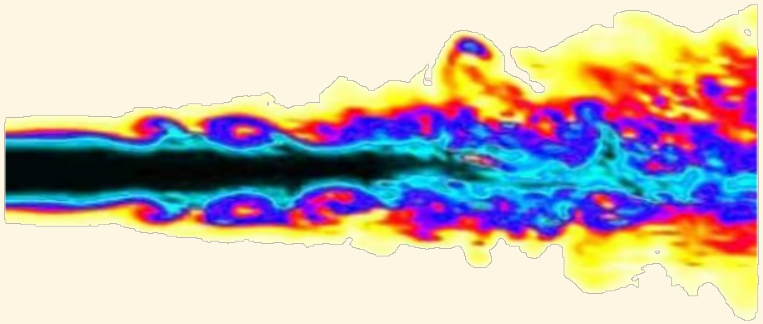


*$\mathcal{R}e$ is a measure of turbulence intensity.

What are the main CFD methods?

RANS  $\mathcal{O}\left(\mathcal{R}e\right)$

LES  $\mathcal{O}\left(\mathcal{R}e^2 - \mathcal{R}e^{2.5}\right)$

DNS  $\mathcal{O}\left(\mathcal{R}e^3\right)$

Need for more computing power in turbulence simulations relevant to high $\mathcal{R}e$ continues...

Outline

Introduction to HPC for CFD
- What is CFD?
- Why use CFD?
- Where is CFD used?
- Why use HPC for CFD?
- Where is CFD headed in light of NASA2030's vision?
- What are the main CFD methods?

LES of Wakes of Wind Turbines Operating in Turbulent Inflow
- Presentation of the numerical framework
- Operating Optimization of ALM
- Application to small wind tunnel turbines
- Future work

Context

- The global energy challenge:
  - Energy supply stability
  - Climate change
- **Increasing demand** for renewable energies: there is a need to increase electrical output and reduce the LCOE[1].
- Morocco has **favorable climatic and geographic conditions** for installing wind turbines.
  - Wind generation potential of 5000 TWh per year
  - Useful capacity of 25,000 MW

- **New challenges**:
  - Larger rotors with flexible blades
  - Complexity of servo-hydro-aero-elasticity
  - Use of high-fidelity simulations to capture complexe phenomena (i.e. dynamic wake meandering, wake steering, blade loading cycles to include wake turbulence,...).

Offshore turbines. Photo: Xlinks

[1] Veers, P. et al. (2017). Science

Context

- Large-Eddy Simulation is a promising tool for predicting wind turbine performances as it provides access to unsteady flow features.
- Simulating the aerodynamics of an operating wind turbine at a high fidelity level is a challenging task due to the extremely high Reynolds number involved.
  - Develop a computational in-house parallel solver at UM6P.
  - Implement a hybrid approach combining Large Eddy Simulation (LES) and Actuator Line Modeling (ALM) techniques.
  - Enhance accuracy in predicting the flow field and aerodynamic loading on turbine blades.
  - Achieve advantages over a pure LES approach, such as optimal workload balacing and return time.

Overview of the Numerical Solver

- The numerical framework is an in-house finite volume solver designed for incompressible flows and massively parallel computations.
  - ✔ IO formats: Fluent, Ensight, prepartionned HDF5 (XDMF) with compression
  - ✔ Partitioned mesh support for HDF5 independent of the number of processors
  - ✔ Parallel interpolator for partitioned HDF5 meshes
- Utilizes a Projection Method Based on Fractional Time Steps, developed by Chorin[2], and Improved by Kim & Moin[3].
  - ✔ Highly efficient Deflated Preconditioned Conjugate Gradient: Use of a residual recycling method and an adaption of convergence criteria.
- Turbulence model implemented: Constant Smagorinsky, Localized dynamic Smagorinsky, Vreman, WALE, σ-model.
- It Implements two levels of parallelism to enable massively parallel computations:
  - ✔ Single Domain Decomposition.
  - ✔ Double Domain Decomposition.

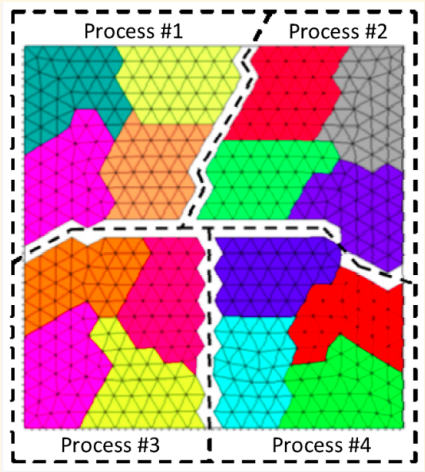
[2] Chorin, A. J. (1968). Mathematics of Computation.

[3] Kim, J., & Moin, P. (1985). Journal of Computational Physics.

Mesh Management on Processors

- First Level: Single-Level Domain Decomposition
- Utilization of Metis and Scotch for Mesh Partitioning: Ensuring optimal workload distribution is essential.
- Implementation of MPI library for Processor Information Exchange at Interfaces

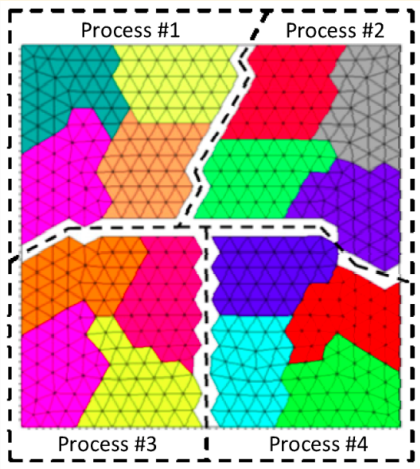
Dashed lines represent the interfaces between processes, necessitating communication.

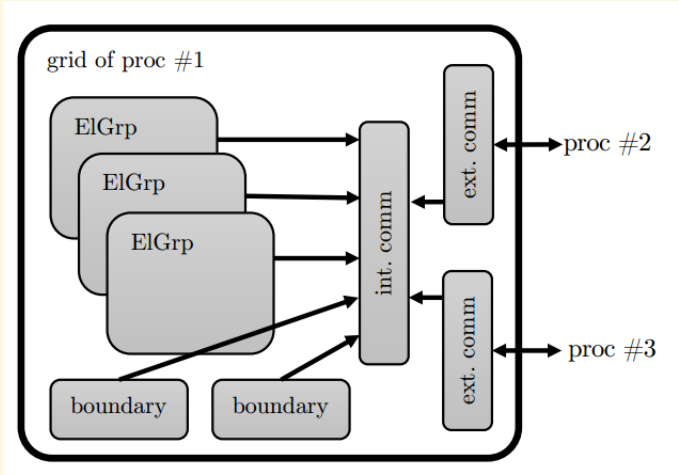


- An initially well-partitioned mesh can become imbalanced with local refinement or coarsening.
- Accessing Data from RAM demands significant processor cycles.
  - ✔ Cache blocking technique implemented through double domain decomposition.

Mesh Management

- Second Level: Double Domain Decomposition.
- Each process divides its subdomain into several groups of elements:
  - ✔ Small enough to fit into L_3 or even L_2 cache.
  - ✔ Groups of elements are constructed similarly to the subdomains in the SDD.
  - ✔ Two communicators are defined: Internal communicator handles data exchange between groups, and External communicator (via MPI) manages inter-processor data exchange.

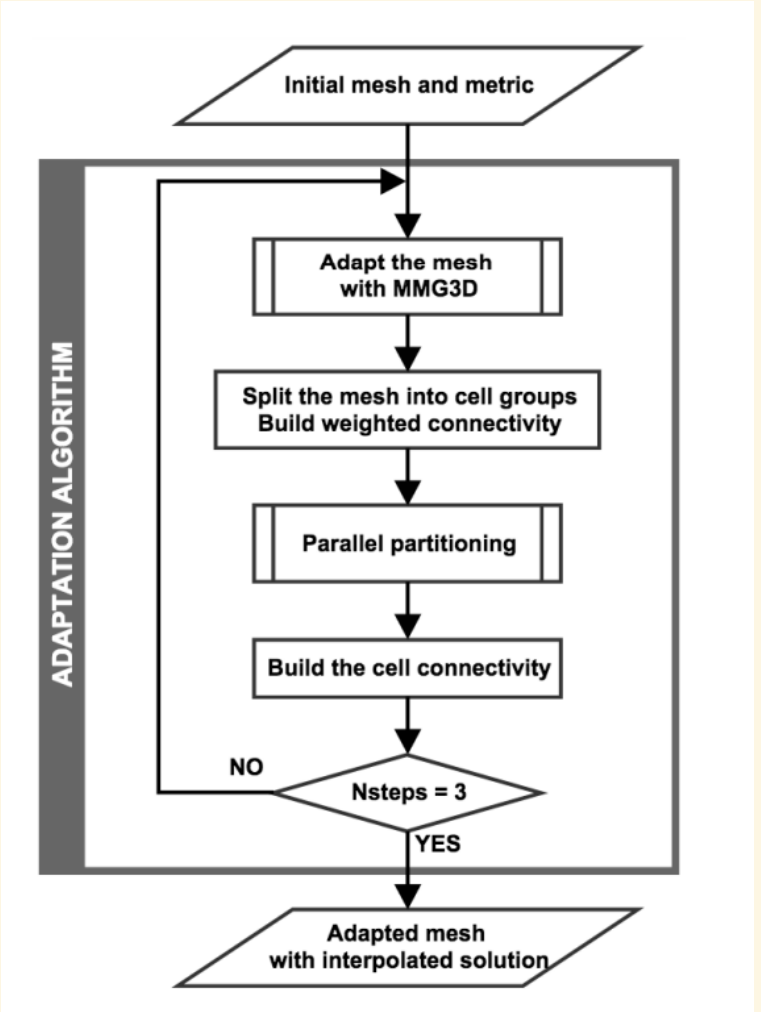




- Beneficial in cases requiring dynamic load balancing.
- Since groups of elements serve as nodes of the deflation grid, solving Poisson's equation with the DPCG algorithm is significantly faster.

Parallel Mesh Adaptation

- MMG3D is a sequential anisotropic mesh adaptation library for tetrahedral elements based on local mesh modifications such as edge flips, edge collapsing, node relocations and vertex insertions driven by isotropic or anisotropic metric specifications.
  - ➔ Iterative process based on sequential calls to MMG3D library on each processor.
  - ➔ The parallel graph partitioning is conducted at the cell group level.

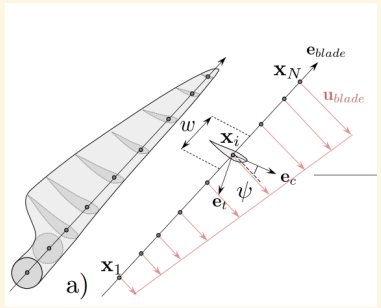


- H-adaptation refinement criterion based on local vorticity perfectly works in 2D simulations
- Enables to reach unprecedented precision for a given cell count

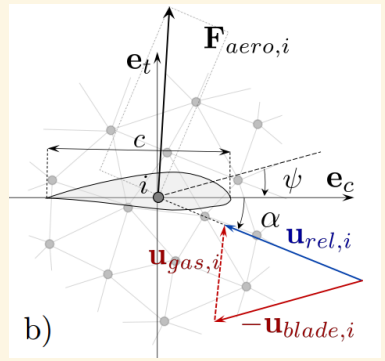
Implementation of ALM

The ALM methodology is implemented as follows:

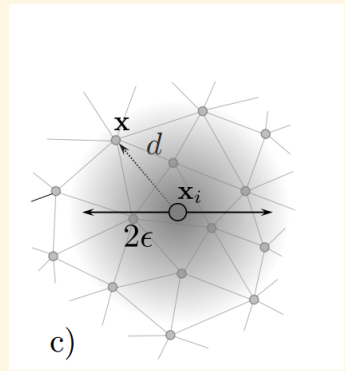
- **Blade Discretization**: The blade geometry is discretized into a line of N elements with a width w and chord c.
  - ➔ This airfoil is based on the actual blade geometry at $\mathbf{x}_i$.
  - ➔ The chord and thickness axes are rotated to consider ψ.



- **Local Velocity Evaluations**: The velocity relative to each blade element is evaluated as $\mathbf{u}_{\text{blade},i}$ and $u_{\text{gas},i}$.
  - ➔ $u_{\text{gas},i}$ is obtained by integrating the near velocity field with a weight function (mollification function) and then $\mathbf{u}_{\text{rel},i}$ is evaluated as $\mathbf{u}_{\text{rel},i} = \mathbf{u}_{\text{gas},i} - \mathbf{u}_{\text{blade},i}$



- **Force Computation and Mollification**: $\mathbf{u}_{\text{rel},i}$ is used to compute α from which we calculate:
  - ➔ Local forces $\mathcal{F}_i = \int_w \frac{1}{2}\rho\|\mathbf{u}_{\text{rel},i}\| c \left(C_L(\alpha_i)\mathbf{e}_L + C_D(\alpha_i)\mathbf{e}_D\right) \mathrm{d}w$
  - ➔ Body force source term $\mathbf{f}(x) = -\frac{1}{\rho}\sum_{i=1}^{N} \mathcal{F}_i \eta_\varepsilon\left(\|x - x_i\|\right)$ with $\eta_\varepsilon(d) = \frac{1}{\varepsilon^3 \pi^{3/2}} \exp\left[-(d/\varepsilon)^2\right]$
  - ➔ Various corrections have been implemented (3D stall delay, tip/hub losses, Dynamic stall, and Filtered lifting line).

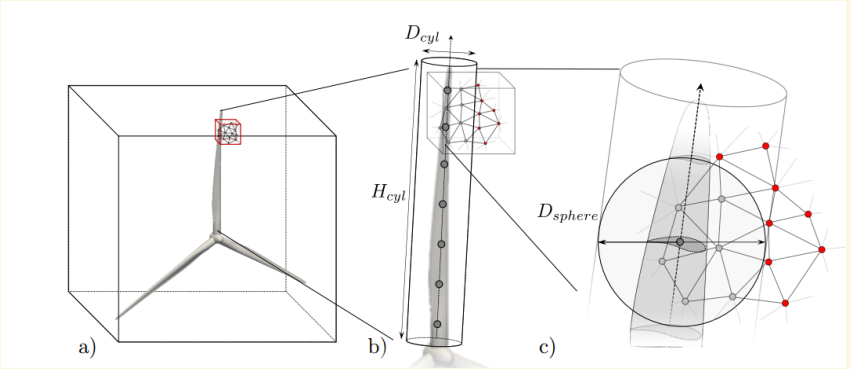


Operating Optimization of ALM

Implementing ALM in a massively parallel framework necessitates careful consideration of its impact on computational costs.

- Mollification Process: Applying local volume source terms on a sizable unstructured grid can result in high CPU costs.

(a) Coarse Level: Comparison of bounding boxes for rotor and element groups. (b) Medium Level: Proximity of nodes to the blade (Cylinder). (c) Fine Level: Proximity of nodes to the blade element (Sphere).



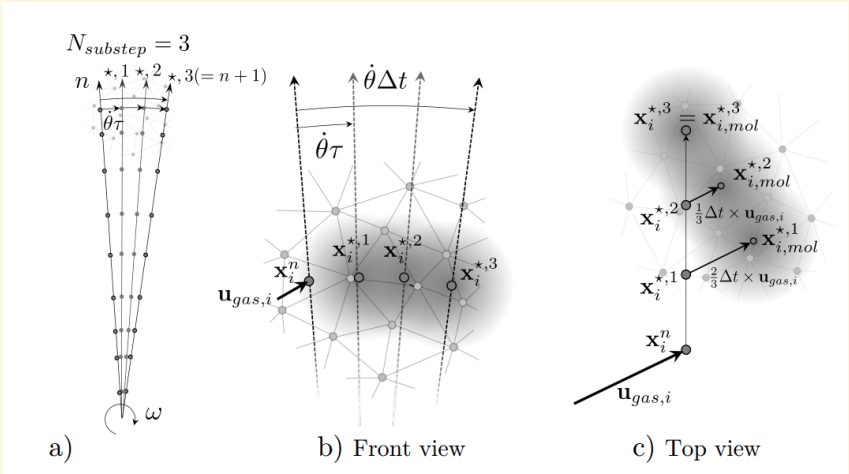
- Each turbine is known by each processor participating in the computation.
  - ➔ Duplication of information and the serial calculation of the actuator line methodology result in increased computational costs.
  - ➔ This cost increase remains around two percent with the addition of two turbines in the computational domain.

Operating Optimization of ALM

- Substepping of ALMs: Adjustment of the limiting time step in ALMs, where $\mathrm{CFL}_{\mathrm{Rotor}} = \max_{j=1}^{N_{\mathrm{blade}}} \left(\max_{i=1}^{N} \left[\mathrm{CFL}_{j;i} \right] \right)$, with $\mathrm{CFL}_{j;i} = \frac{\|u_{j,i}\| \Delta t}{h_i}$. For flow around wind turbines, $\mathrm{CFL}_{\mathrm{Rotor}} \sim \mathrm{TSR} \times \mathrm{CFL} \times \min_{\mathrm{nodes}} \frac{h}{\|\mathbf{u}_{\mathrm{flow}}\|}$.

To prevent numerical fluctuations:

- Reduce the simulation time step.
- Preserve the time step and mollify the forces through a time mollification process with substepping.



At the end of the substeps loop, the body force source term on the Eulerian grid is calculated as:

$$\mathbf{f}(x) = -\frac{1}{\rho} \sum_{s=1}^{N_{\mathrm{substeps}}} \left(\sum_{i=1}^{N} \frac{\mathcal{F}_i}{N_{\mathrm{substeps}}} \eta_\varepsilon \left(\|x - x_{i,\mathrm{mol}}^{\star,s}\| \right) \right)$$

Operating Optimization of ALM

- The present implementation of ALM-LES takes into account any geometric detail
  - For the yaw and the tilt:

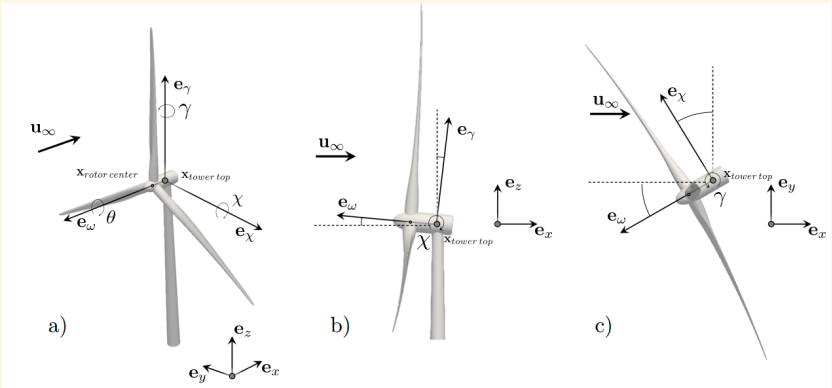
$$\mathcal{M}_\gamma = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(d\gamma) & -\sin(d\gamma) \\ 0 & \sin(d\gamma) & \cos(d\gamma) \end{bmatrix}, \quad \mathcal{M}_\chi = \begin{bmatrix} \cos(d\chi) & 0 & \sin(d\chi) \\ 0 & 1 & 0 \\ -\sin(d\chi) & 0 & \cos(d\chi) \end{bmatrix}$$

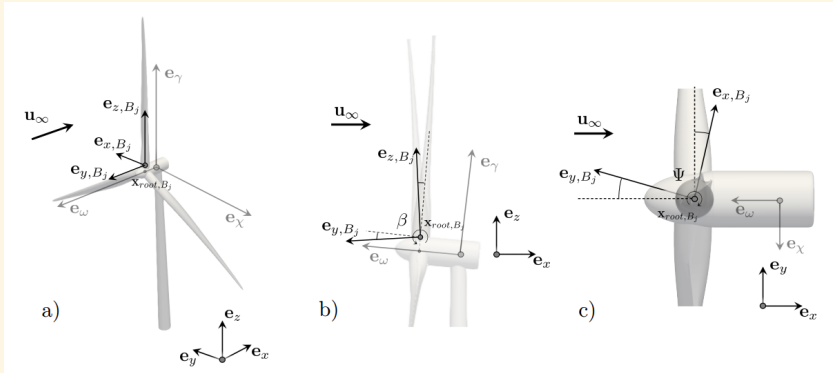
where $d\gamma$ and $d\chi$ are the advancement angles during the time step and are computed as $d\gamma = \Delta t\,\dot{\gamma} + \frac{1}{2}\Delta t^2\,\ddot{\gamma}$ and $d\chi = \Delta t\,\dot{\chi} + \frac{1}{2}\Delta t^2\,\ddot{\chi}$

  - The tilt and yaw transformation matrix are associated as

$$\Theta_G^R(d\gamma, d\chi) = \Theta_G^R \mathcal{M}_\gamma \mathcal{M}_\chi$$

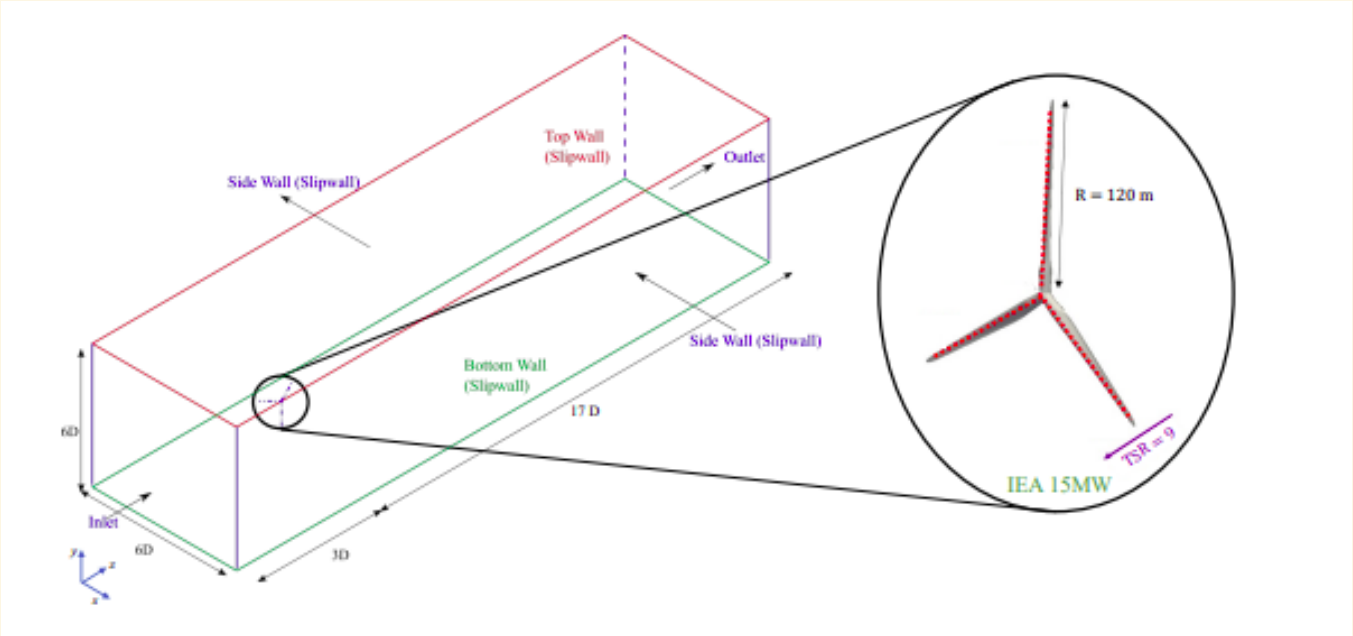
  - Same for the azimuth angle of the blades and the pitch angle.





Application to small wind tunnel turbines

- The numerical domain has dimensions $L_x \times L_y \times L_z = 300 \times 300 \times 1300$ [m] with 623.2 million cells (hexa)—an ASCC supercomputer record.
- Three TSR values: 6, 8, and 13 were used in the simulation.
- Use of 4080 processors for computation.
- Visualization includes front, side, and top views of the computational domain in the Norwegian University of Science and Technology wind tunnel, featuring uniform inflow conditions.

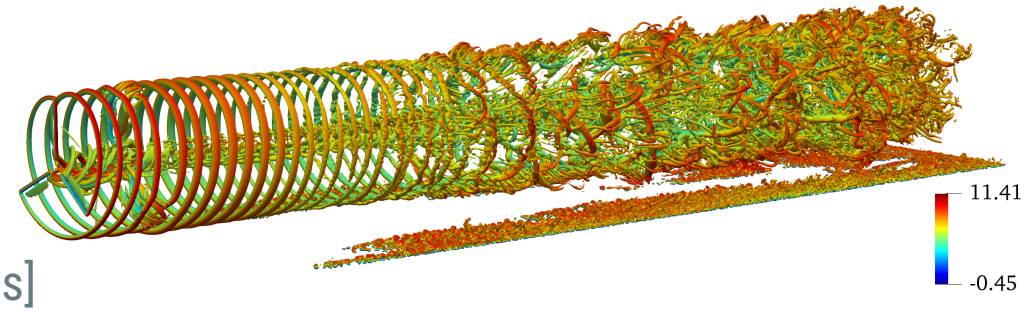


Application to small wind tunnel turbines

Helicoidal wake structure through instantaneous contours of $Q_{\text{criterion}}$

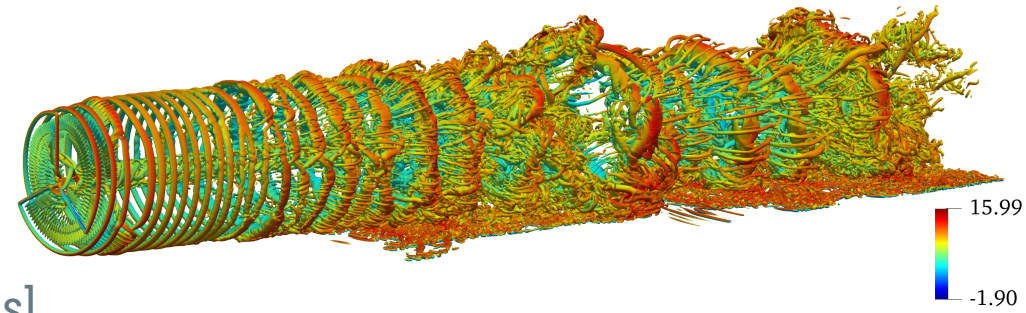
TSR = 06

$\mathcal{U}_\infty = 08$ [m/s] & $\omega = 0.762$ [rad/s]



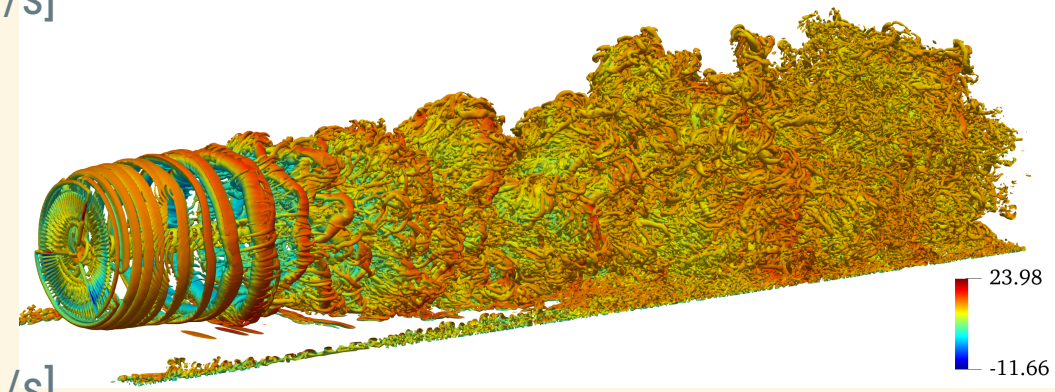
TSR = 08

$\mathcal{U}_\infty = 10$ [m/s] & $\omega = 1.270$ [rad/s]



TSR = 13

$\mathcal{U}_\infty = 12$ [m/s] & $\omega = 2.476$ [rad/s]



Future work

This simulation framework can be used for more complex configurations. It would be great if we could make progress on the following tasks:

- Improve coupling implementation (optimization of MPI communications and memory usage) of ALM data structure.
- Multiple rotors: analysis of the interaction of the wake on blade deformation turbulence injection.
- Simulate the flow around an entire wind farm and assess the inflow's impact.

Thanks for listening!