Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Computer Applications in Science & Engineering (CASE)
Barcelona Supercomputing Center - BSC
Barcelona, Spain

Generation of large-scale curved meshes for complex virtual geometries

Eloi Ruiz-Gironés with Xevi Roca

Tetrahedron VII Workshop

European Research Council
Established by the European Commission

Acknowledgements

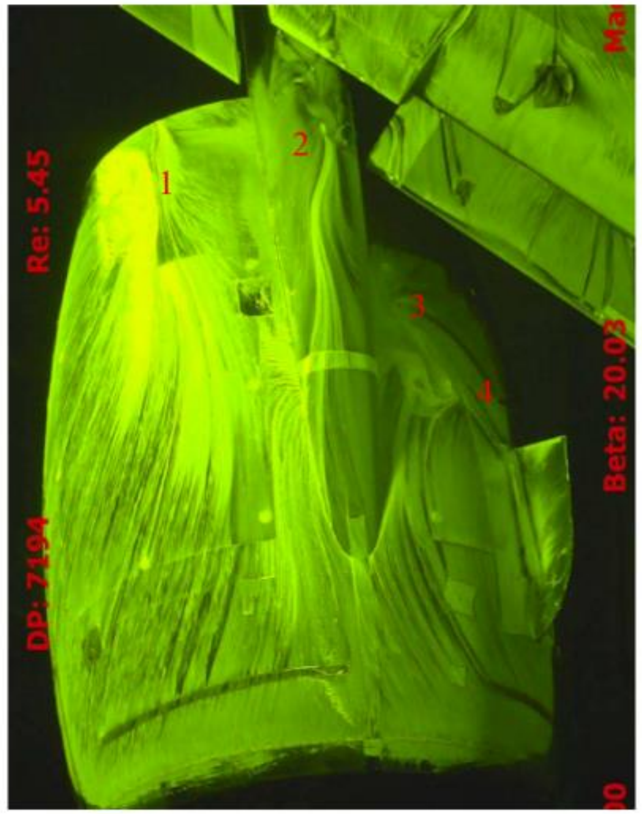
- **Co-authors:**
  - J. Sarrate (UPC), A. Gargallo (BSC) & X. Roca (BSC)

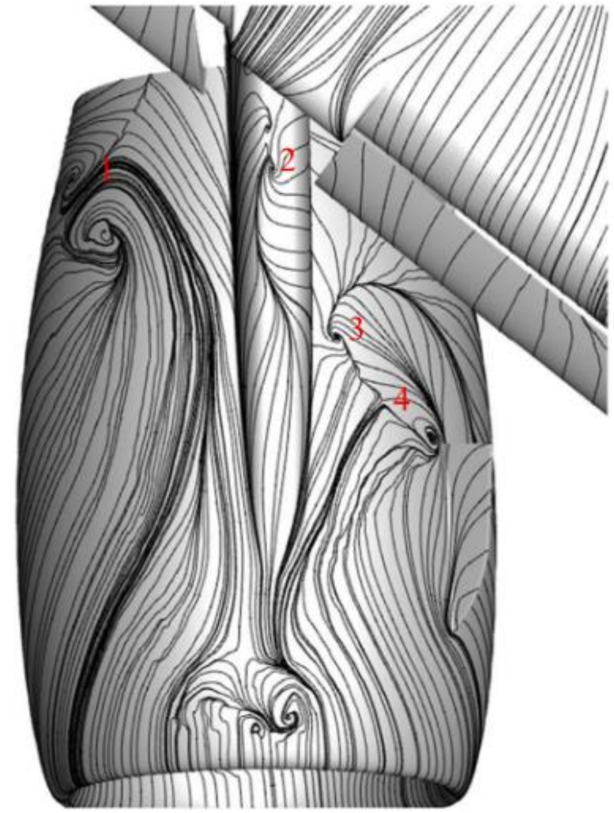
- **Simulations on our curved meshes:**
  - 4^{th} & 5^{th} HLPW high-order group
  - Z.J. Wang, University of Kansas, USA
  - Oriol Lehmkuhl, BSC, Spain

- **Computing hours:**
  - BSC
  - PRACE program

Motivation: Wind Tunnel vs Simulation

Wind tunnel data
(4th HLPW)
(NASA AIAA)





High-order simulation
(ZJ Wang AIAA'23)

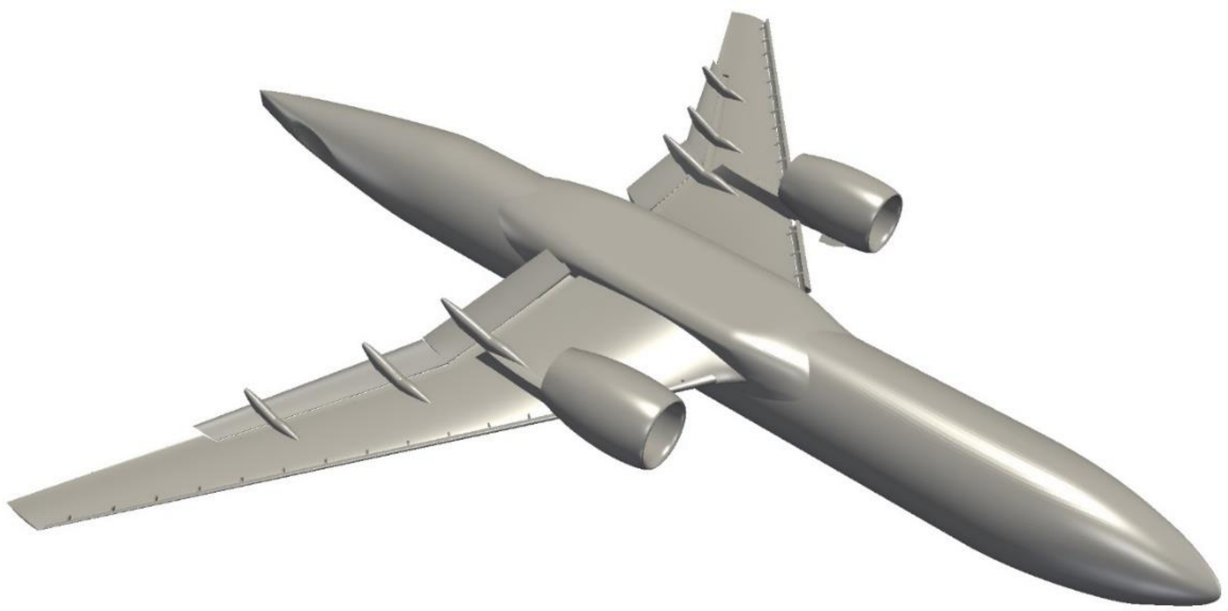
Unstructured high-order methods

- **Geometric flexibility:** using unstructured meshes
- **More accuracy:** with same #DOFs, less dissipation and dispersion

They require curved high-order meshes

- **Geometric error:** straight elements hamper simulation accuracy

Motivation: Curved Meshes for Flow Simulation



Large-scale curved meshes in complex virtual geometries

- **Approximate virtual CAD B-rep:** using curved elements
- **Mesh features:** small elements & size gradation, boundary layer...
- **Mesh quality:** facilitates solving the simulation problems

Mesh Curving Methods

Direct methods: Create curved mesh from scratch

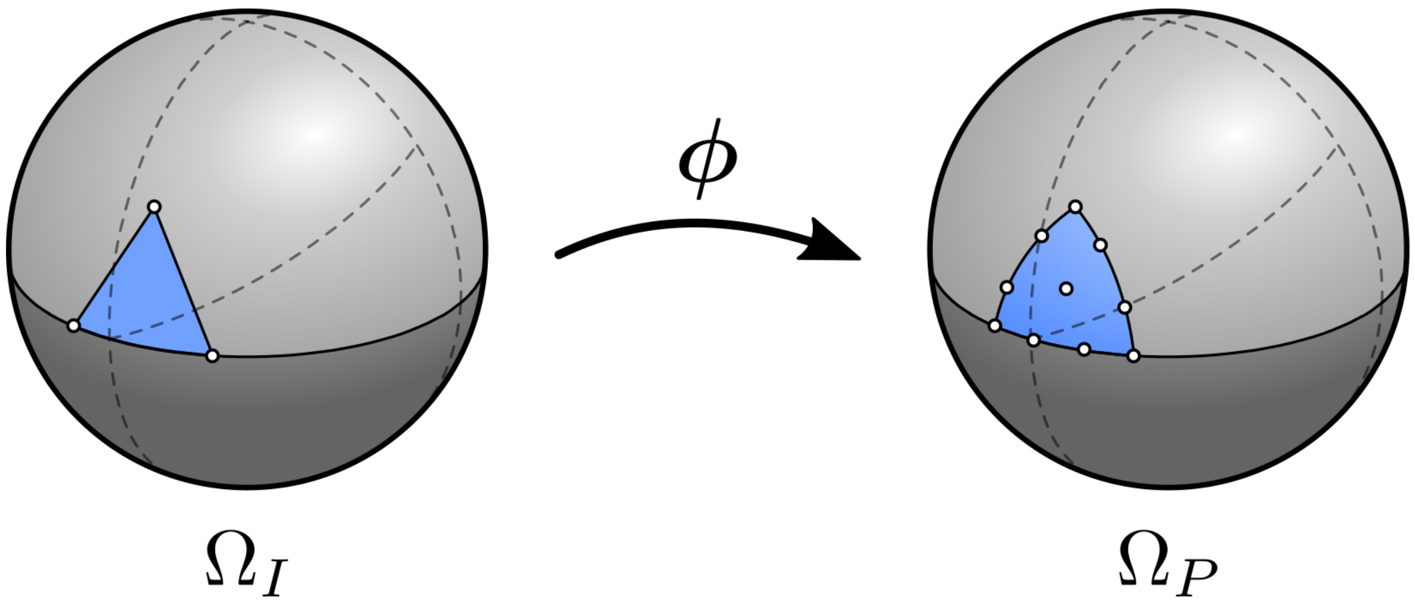
- Delaunay (Feng, Alliez, Busé, Delingette, Desbrun ToG'18)
- Advancing front (Mohammadi, Shontz IMR'21)

Indirect methods: Linear mesh generation + curving step

- **PDE-based**
  - Linear / non-linear elasticity
    (Persson, Peraire), (Xie, Poya, Sevilla, Hassan), (Turner, Moxey, Sherwin, Peiró)
  - Winslow (Fortunato, Persson), ...
- **Optimization-based**
  - Mesh distortion / quality (Tomov, Mittal, Kolev), (Karman),
    (Gargallo-Peiró, Ruiz-Gironés, Sarrate, Roca), (Feuillet, Loseille, Alauzet)...
  - Nodal displacement (Toulorge, Johnen, Lambrechts, Remacle), ...
  - Other quantities (Stees, Shontz), ...

Outline

- Curving solution
- Complex geometry in parallel
- Large-scale distributed curving
- High-Lift Prediction Workshop

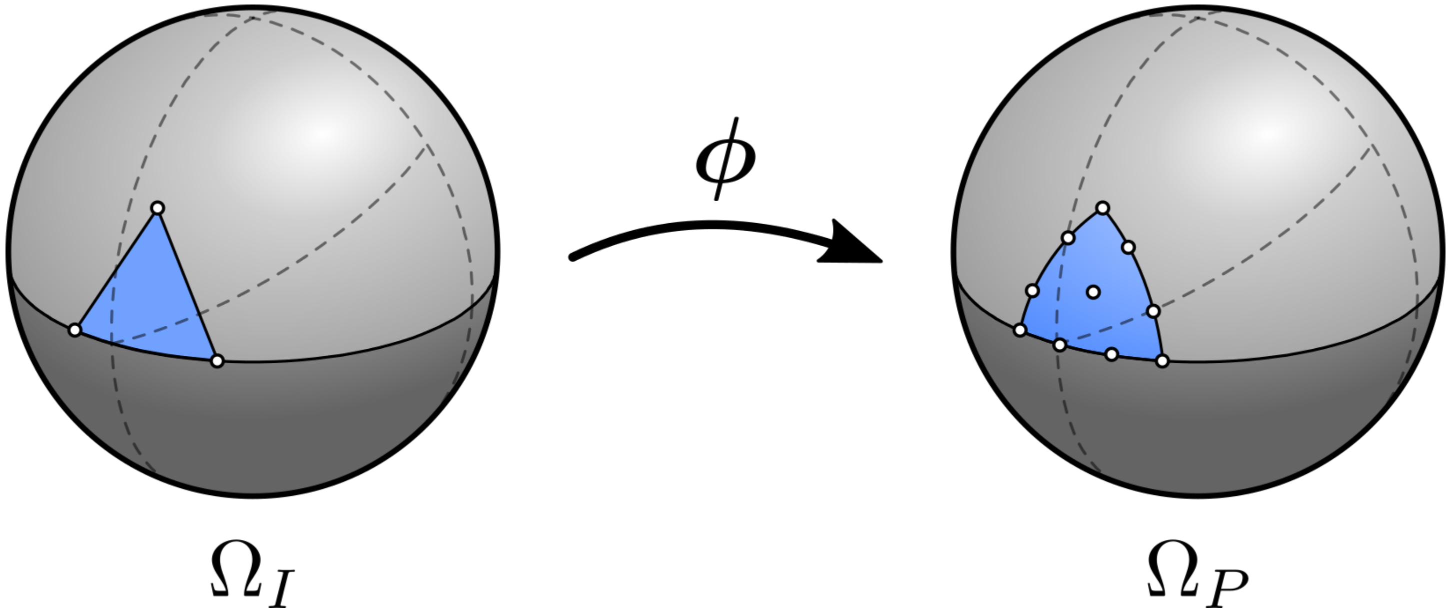


Curving Solution

(Ruiz-Gironés, Gargallo, Sarrate, Roca IMR'17 & CAD'19)
(Ruiz-Gironés, Roca IMR'18, IMR'19 & CAD'22)

- **Constrained optimization problem**
- **Continuous penalty**

Our Formulation : Constrained Optimization



$$M\boldsymbol{\phi} = \frac{|\boldsymbol{D\phi}|^2}{n\sigma_0(\boldsymbol{D\phi})^{2/n}}$$

(Knupp SIAM J. Sci. Comput.'01)
(Roca, Gargallo, Sarrate IMR'12)
(Gargallo, Roca, Peraire, Sarrate IJNME'15)

$$\min_{\boldsymbol{\phi}} ||M\boldsymbol{\phi}||^2_{\Omega_I}$$

constrained to:

$$\mathrm{T}\boldsymbol{\phi} = \boldsymbol{g}_D(\mathrm{T}\boldsymbol{\phi})$$

(Ruiz-Gironés, Roca, Sarrate CAD'16)
(Ruiz-Gironés, Gargallo, Sarrate, Roca IMR'17 & CAD'19)
(Ruiz-Gironés, Sarrate, Roca IMR'16)

(Ruiz-Gironés, Sarrate, Roca IMR'15 & JCP'21)

$$\boldsymbol{g}_D(\mathrm{T}\boldsymbol{\phi}) = \sum_{i=1}^{N} \pi_{\Omega_i}(\boldsymbol{x}_i)N_i$$

(Ruiz-Gironés, Roca IMR'18, IMR'19, CAD'22, AIAA'22)

Our Solution: Continuous Penalty Method

(Ruiz-Gironés, Roca IMR'18, IMR'19 & CAD'22)

Penalty method: Optimize several unconstrained problems

$$E_\mu(\boldsymbol{\phi}) = ||M\boldsymbol{\phi}||^2_{\Omega_I} + \mu||\mathrm{T}\boldsymbol{\phi} - \boldsymbol{g}_D(\mathrm{T}\boldsymbol{\phi})||^2_{\partial\Omega_I}$$

$$E_\mu : \mathcal{H}^1(\Omega) \longrightarrow \mathbb{R}$$

$$\boldsymbol{g}_D : \mathcal{L}^2(\partial\Omega) \longrightarrow \mathcal{L}^2(\partial\Omega)$$

Non-linear problem: volume & boundary

Fix-point iterative solver: Newton + backtracking line-search

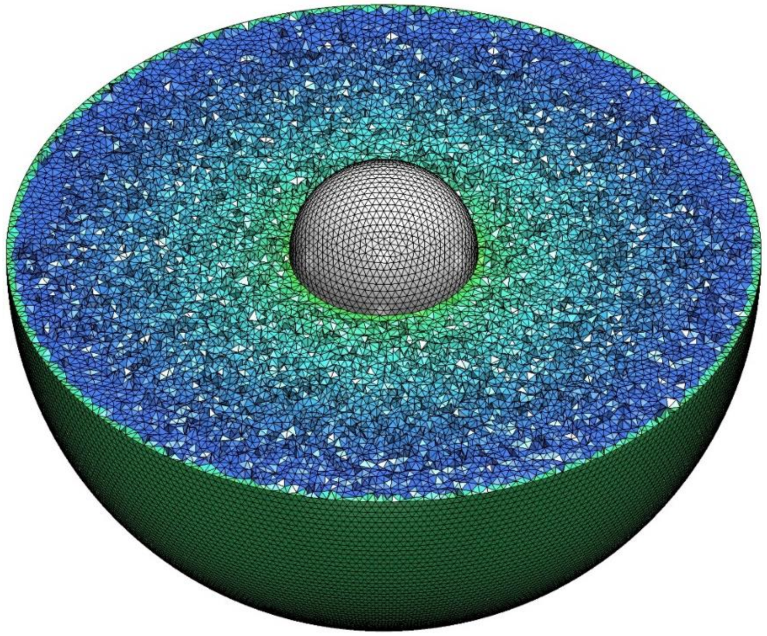
Compute target configuration

$$\boldsymbol{g}^k_D = \boldsymbol{g}_D(\mathrm{T}\boldsymbol{\phi}^k)$$

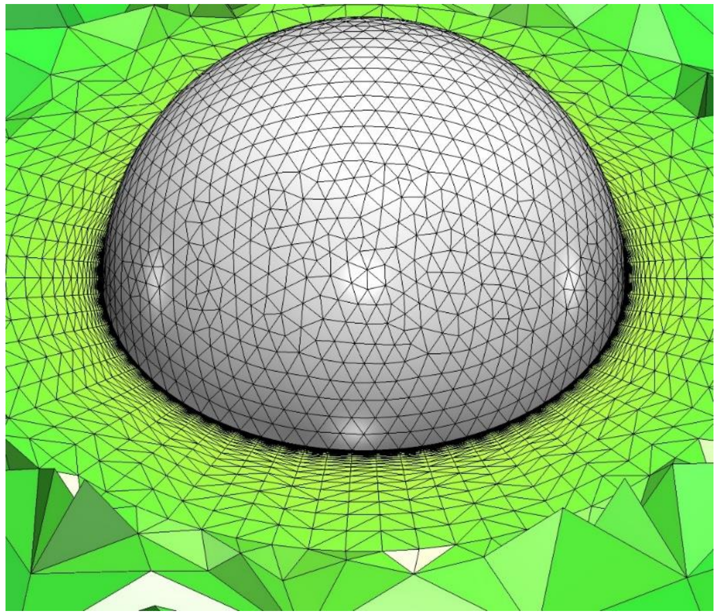
$$\boldsymbol{\phi}^{k+1} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\, E_\mu(\boldsymbol{\phi}, \boldsymbol{g}^k_D)$$

Optimize mesh

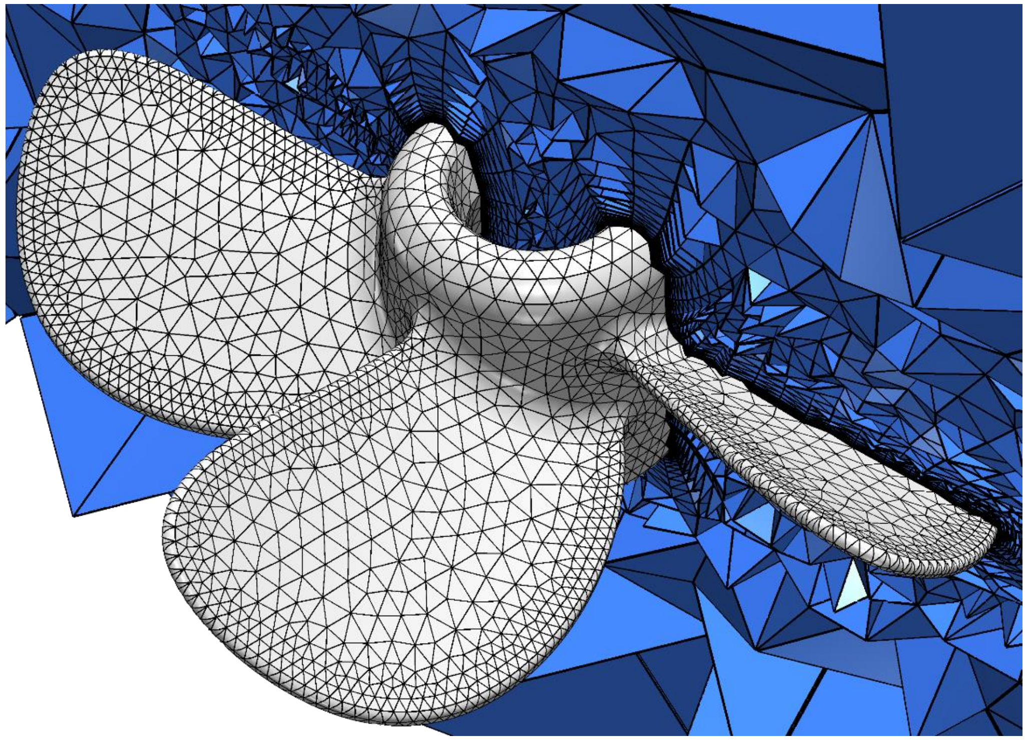
Example: Sample meshes



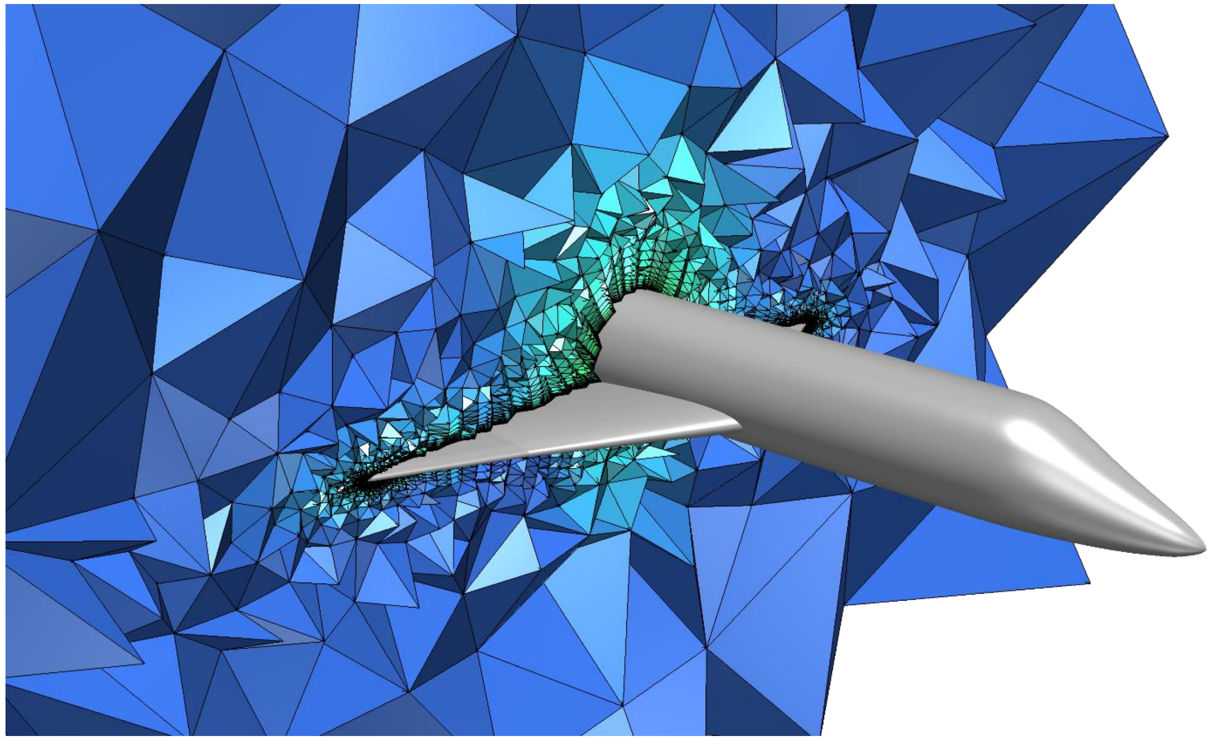
3.6M tets, p=4



0.7M tets, p=4
stretching $1:10^5$



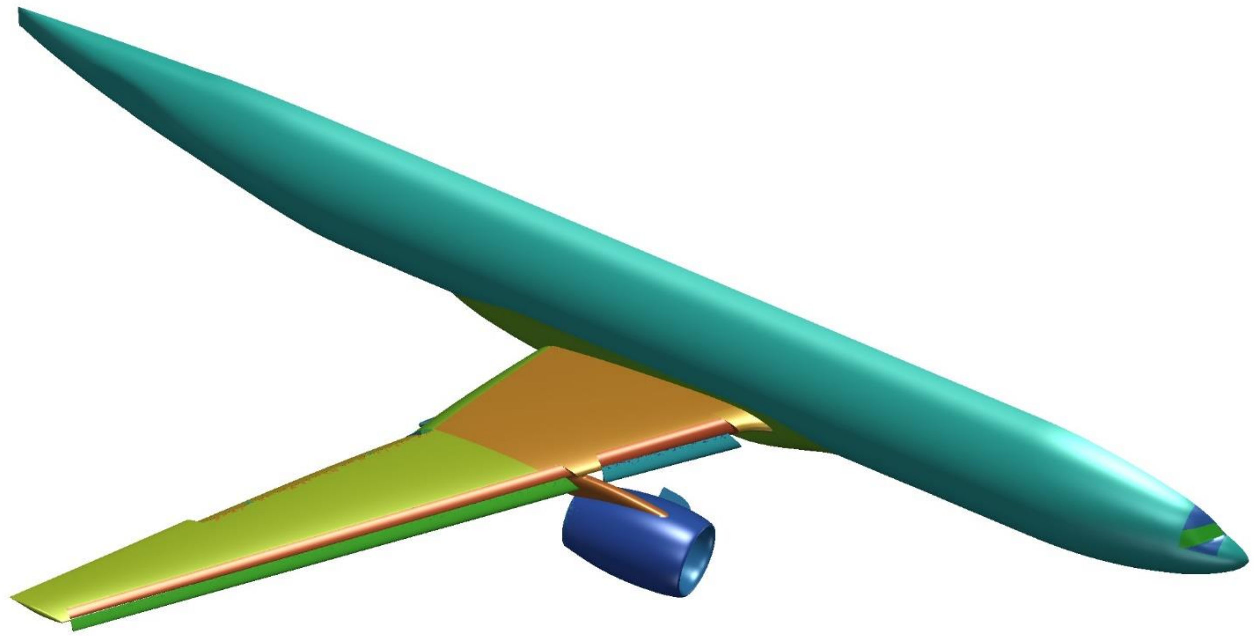
1.6M tets, p=3, stretching 1:750



4M tets, p=4, stretching 1:400

Features

- **Constrained optimization problem:**
  Minimize mesh distortion while approximating the virtual model
- **Mesh floats:**
  Mesh approximates the geometry
- **Mesh is always valid:**
  No need to introduce untangling
- **Virtual geometry aware:**
  Elements span several entities
- **Tight tolerances:**
  Fully converged meshes avoid element oscillations
- **Newton’s method with backtracking line-search:**
  Ensures quadratic convergence near solution



Complex geometry in parallel

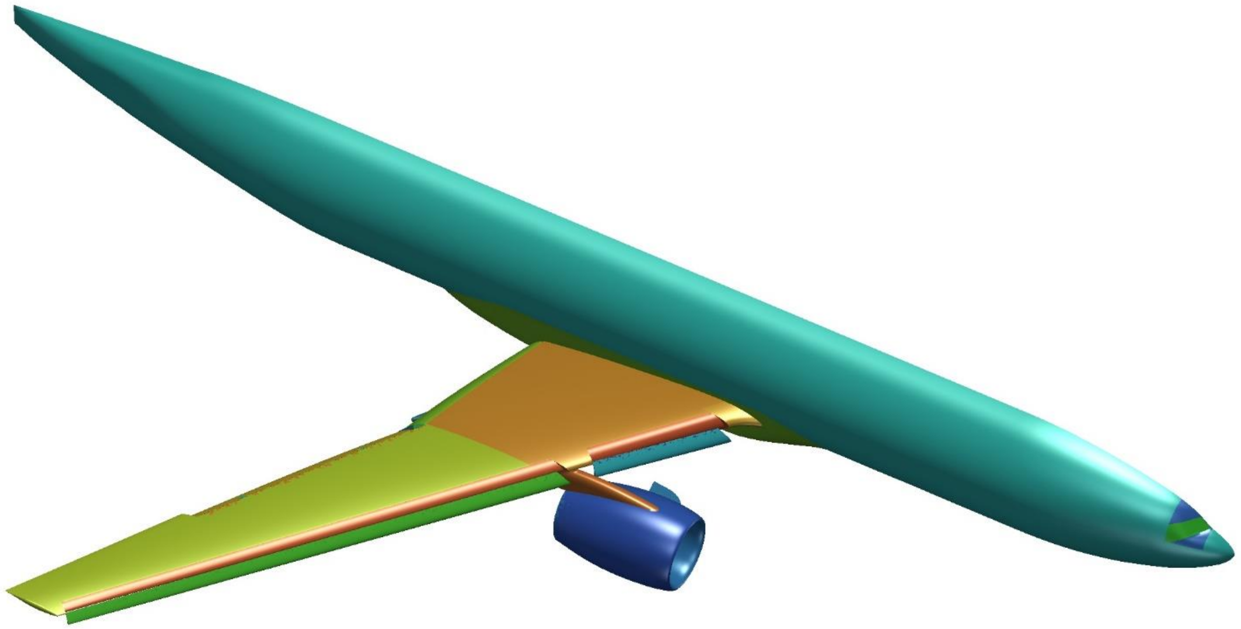
(Ruiz-Gironés, Roca IMR'18, IMR'19 & CAD'22)

(Ruiz-Gironés, Roca AIAA'22)

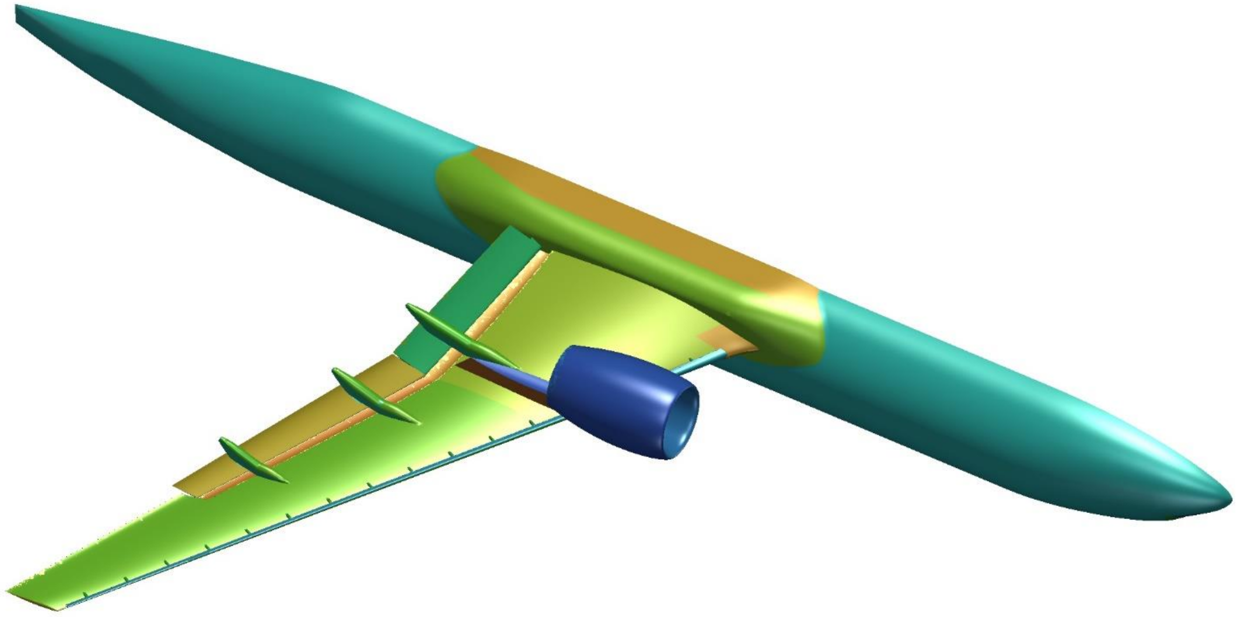
- **Virtual model**
- **Point projection**
- **Parallel distribution of input & output**

Virtual Model

- **Virtually join surfaces:** Simulation intent
- **Decouple CAD & mesh topologies:** one group for fuselage, wings, …
- **Surfaces:** From 415 original surfaces to 215 virtual surfaces



Virtual surfaces: top view

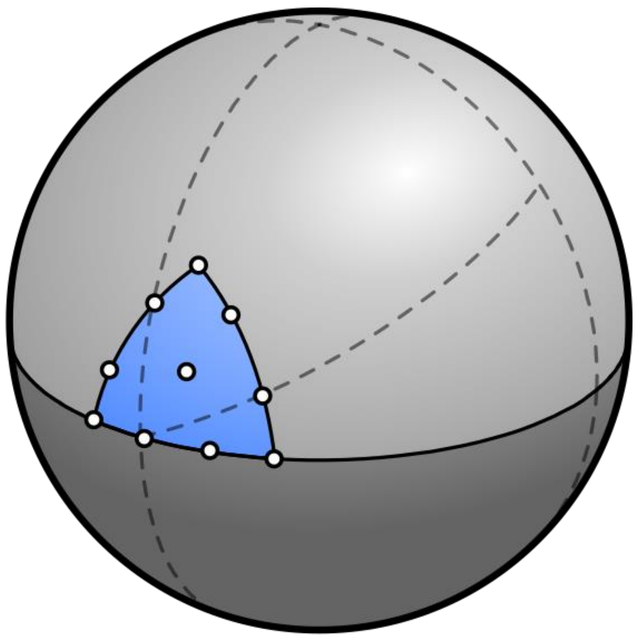


Virtual surfaces: bottom view

Point Projection onto Virtual Models

- **Projection onto virtual surface:**
  (Ruiz-Gironés, Roca IMR'18, IMR'19 & CAD'22)
  - Loop over the geometric surfaces

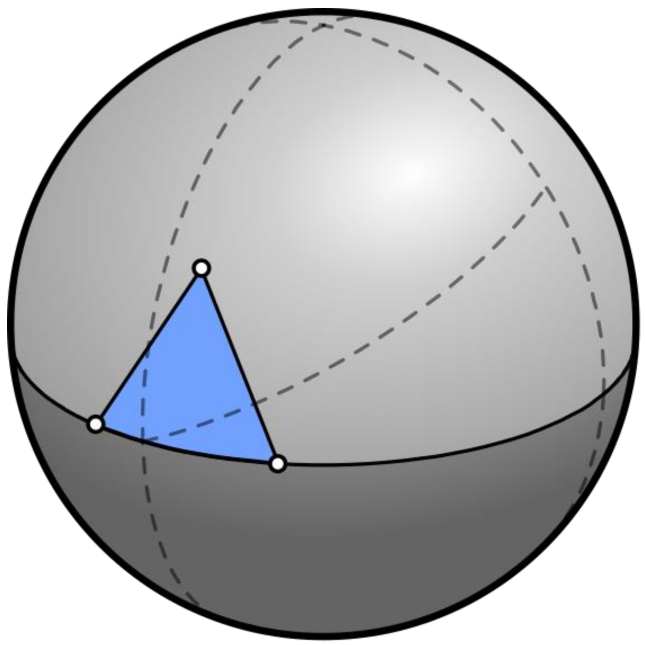
$$\Pi_{\mathcal{S}}(\mathbf{x}) = \arg\min_{\mathbf{y}\in\mathcal{S}} \|\mathbf{x}-\mathbf{y}\|$$



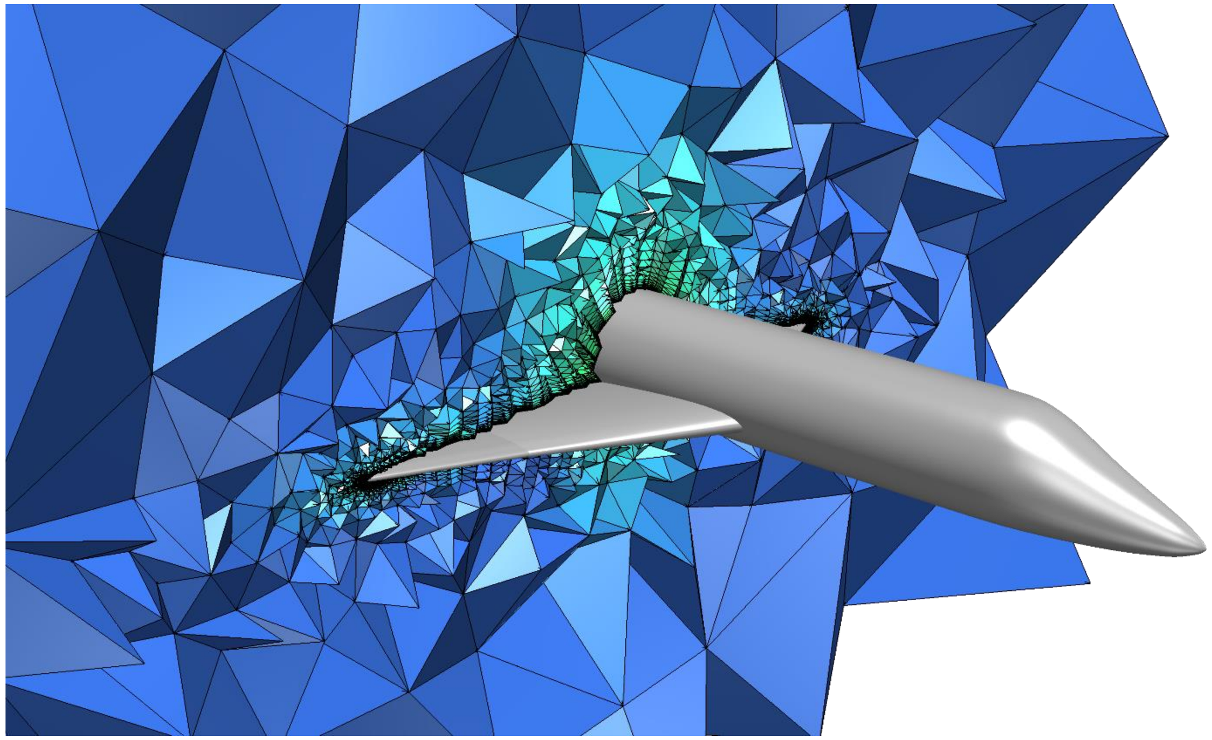
- **Projection onto virtual curve:** Dealing with surface gaps
  (Ruiz-Gironés, Roca AIAA'22)
  - Project the node in-between the surface gap

$$\Pi_{C}(\mathbf{x}) = \frac{1}{2}\left(\Pi_{\mathcal{S}_1}\left(\Pi_{\mathcal{S}_2}(\mathbf{x})\right) + \Pi_{\mathcal{S}_2}\left(\Pi_{\mathcal{S}_1}(\mathbf{x})\right)\right)$$

Parallel Distribution of input & output



- **Input data:**
  - Marked linear mesh
  - Virtual CAD model
- **Linear & high-order meshes:**
  - Each processor owns a set of elements and nodes
  - Each processor projects his boundary nodes
- **Virtual model:**
  - Each processor has a copy of the geometry
  - Easy to distribute, just read the CAD file
  - Processors have enough memory for this approach



Large-scale distributed curving

(Ruiz-Gironés, Roca IMR'19 & CAD'22)

- Reduce computational time
- Reduce memory footprint
- Reduce energy consumption

Lagrange Multiplier Approximation

- **In penalty method:** for μ large enough, Lagrange multiplier is like

$$\lambda \simeq -2\mu \left(\boldsymbol{T\phi} - \mathbf{g}_D(\boldsymbol{T\phi}) \right)$$

- **Function over the mesh boundary:**
  - When converging, doubling μ halves the constraint values

- **We use this to improve the solver:**
  - Given a target constraint norm, which μ we need?

p-Continuation: Less DOF's & Sparser Matrices

Idea: Use a lower degree solution as an initial approximation

Implementation:

- Increase polynomial degree when boundary constraint is *"good enough"*

$$\alpha \varepsilon^p < \varepsilon^{p+1}, \quad \varepsilon^p = \|\boldsymbol{T}\boldsymbol{\phi}^p - \mathbf{g}_D(\boldsymbol{T}\boldsymbol{\phi}^p)\|_{\partial \mathcal{M}_I}$$

- Approximate next penalty parameter using constraint norm

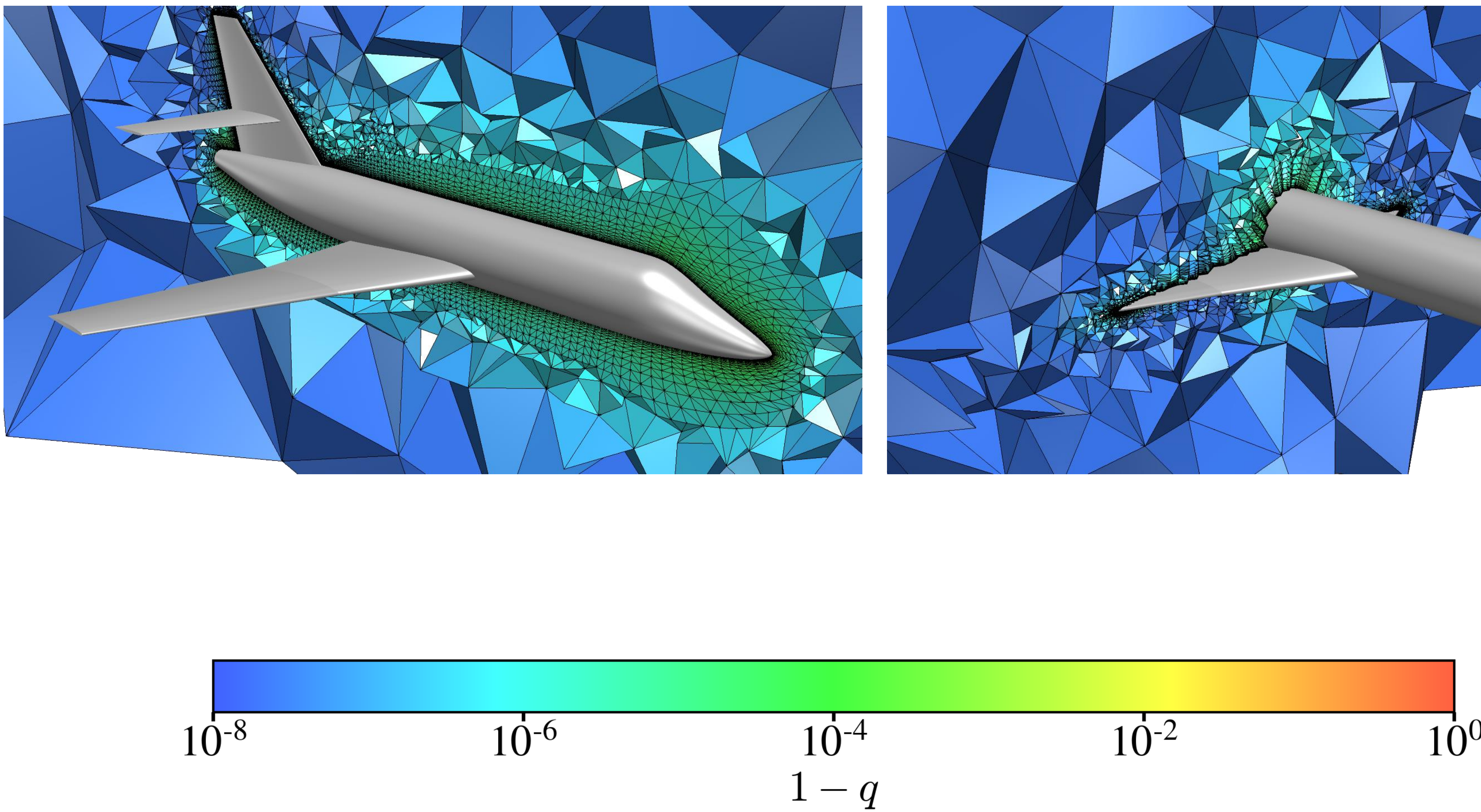
$$\mu^{p+1} = \mu^p \frac{\|\boldsymbol{T}\boldsymbol{\phi}^p - \mathbf{g}_D(\boldsymbol{T}\boldsymbol{\phi}^p)\|_{\partial \mathcal{M}_I}}{\|\boldsymbol{T}\boldsymbol{\phi}^{p+1} - \mathbf{g}_D(\boldsymbol{T}\boldsymbol{\phi}^{p+1})\|_{\partial \mathcal{M}_I}}$$

Examples

p-continuation: Falcon aircraft

Mesh: Degree 4, 4M elements, boundary layer stretching 1:400

Optimization: 2400 cores, RASDD(1) - SSOR(2)

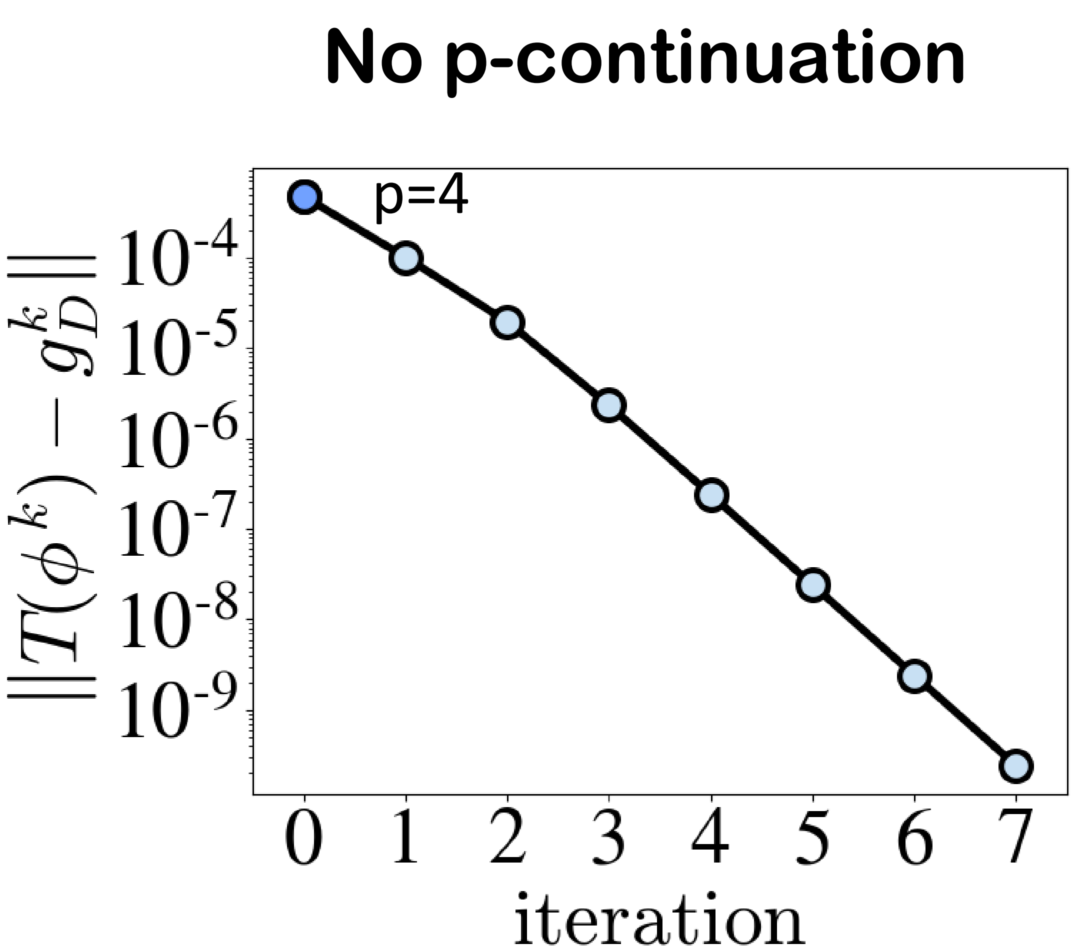


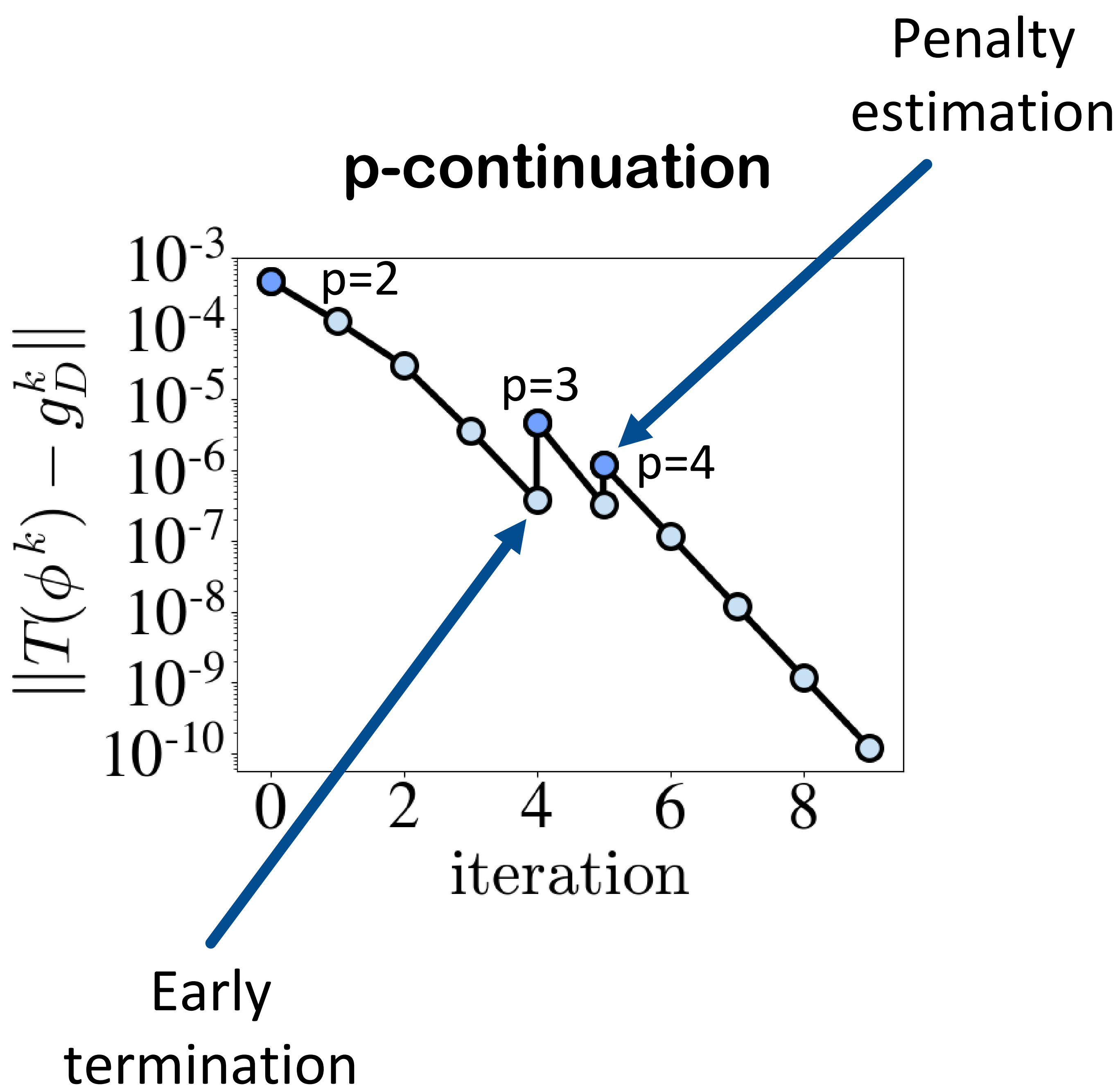
Examples

p-continuation: Falcon aircraft

Mesh: Degree 4, 4M elements, boundary layer stretching 1:400

Optimization: 2400 cores, RASDD(1) - SSOR(2)



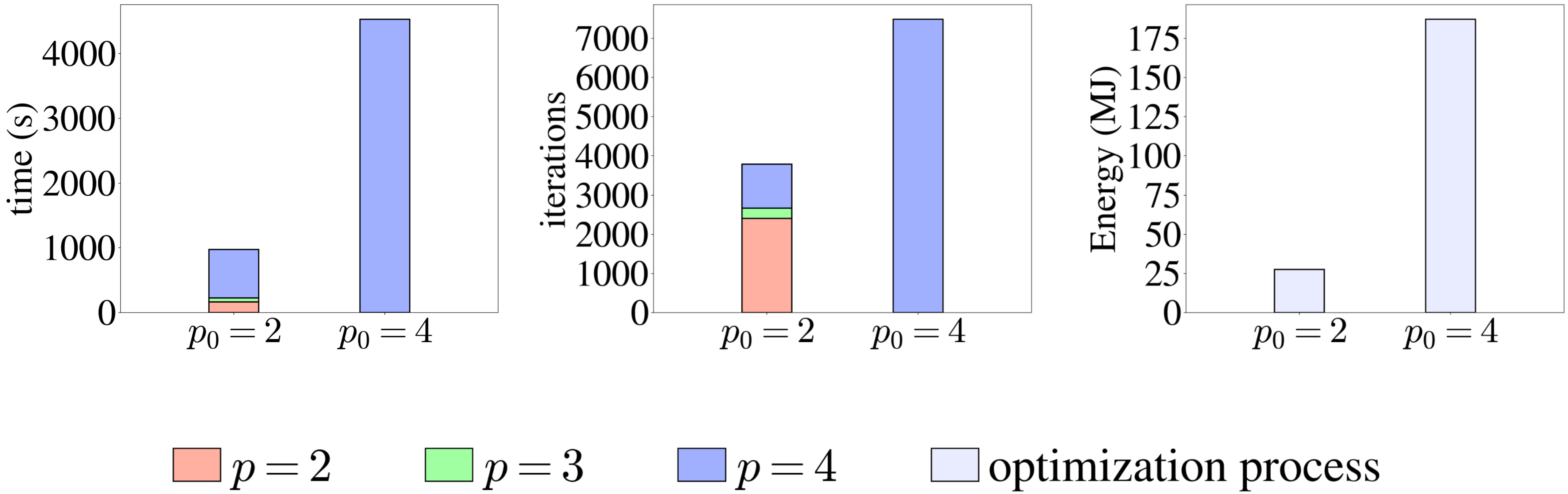


Examples

p-continuation: Falcon aircraft

Mesh: Degree 4, 4M elements, boundary layer stretching 1:400

Optimization: 2400 cores, RASDD(1) - SSOR(2)



p-continuation technique allows:

- 4 times reduction in time
- 8 times reduction in energy

Penalty Estimation: Less non-linear problems

Optimal penalty parameter

$$\mu^* = \mu_k m^* = \mu_k 1.01 \frac{\varepsilon^*}{\varepsilon_k}$$

Convergence indicator

$$s_k = \frac{\mu_{k-1}}{\mu_k} \frac{\varepsilon_{k-1}}{\varepsilon_k} \qquad e_k = \max\{s_k, 1/s_k\} - 1$$

Next penalty parameter

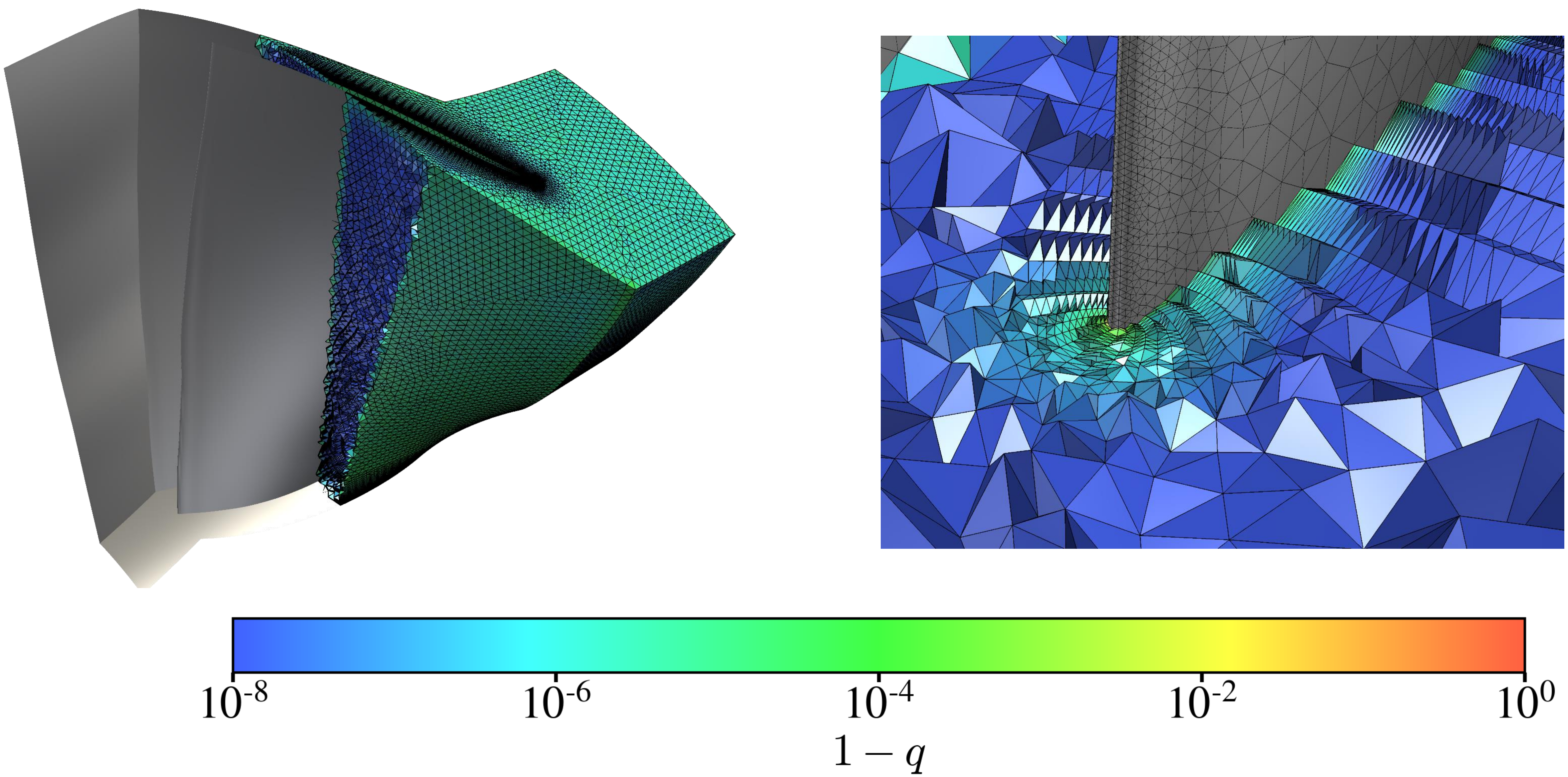
$$m_k = \max\{10, 1/e_k\}$$

$$\mu_{k+1} = \mu_k \min\{m^*, m_k\}$$

Example: Periodic Mesh for the Rotor 67

Set periodic condition in target boundary

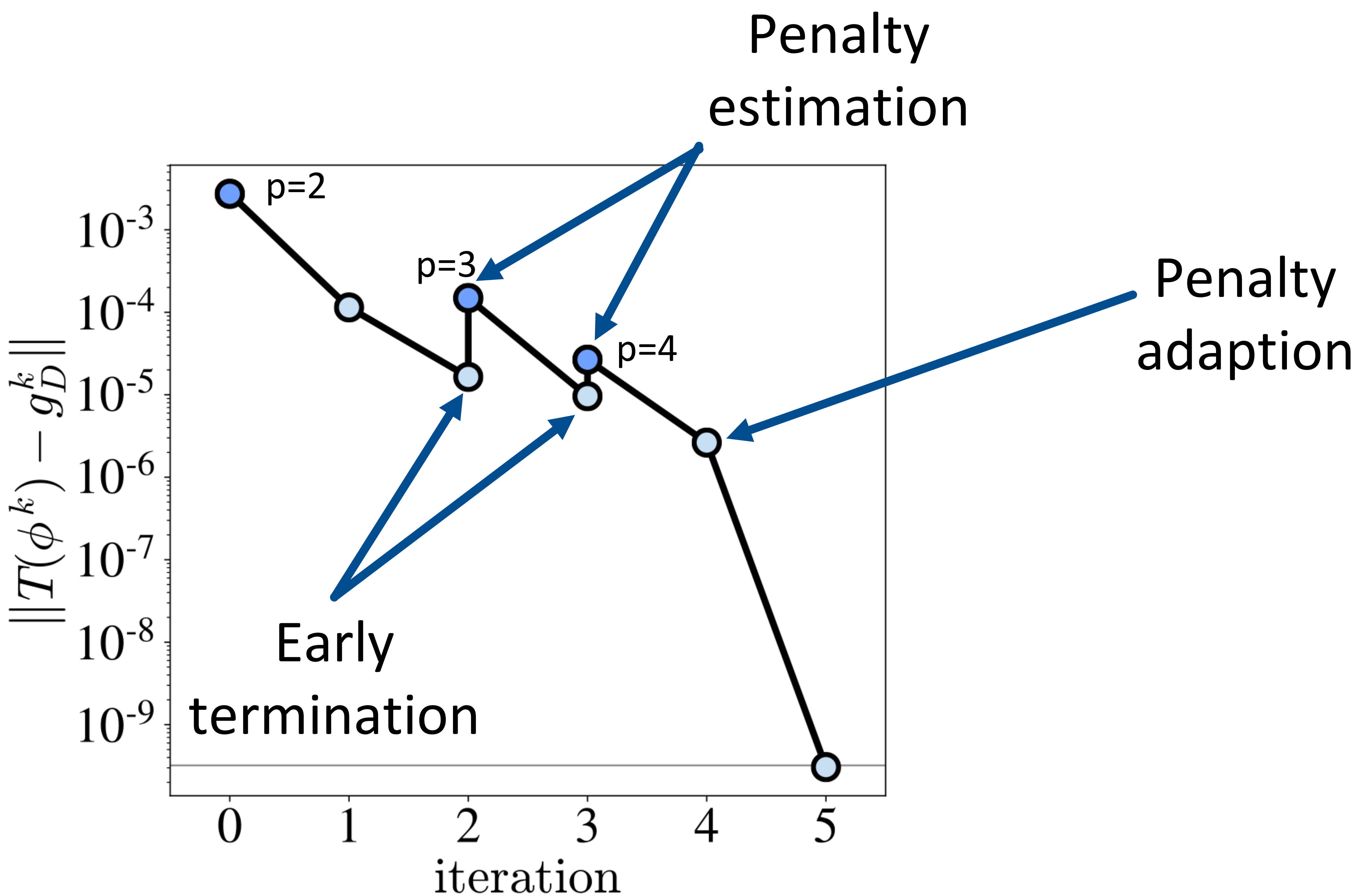
- **Mesh:** p=4, 3.6M elements, boundary layer stretching 1:25
- **Optimization:** 768 cores, 52 minutes, 31.5MJ
- **Metrics:** minQ = 0.987, avg_dist = $1.5 \cdot 10^{-8}$, max_dist = $2.91 \cdot 10^{-5}$



Example: Periodic Mesh for the Rotor 67

Periodic mesh: Impose periodic boundary condition

Mesh: p=4, 3.6M elements, boundary layer stretching 1:25



Forcing Term: Less Linear Iterations

Adapt linear solver tolerance: From loose to tight tolerance

Track the progress of the optimization: use boundary constraint

$$t_k = \frac{\log\left(\frac{\varepsilon_0}{\varepsilon_k / m_k}\right)}{\log\left(\frac{\varepsilon_0}{\varepsilon^*}\right)}$$

Linear solver tolerance

$$\delta = {\delta_{\text{loose}}}^{1-t_k} \cdot {\delta_{\text{tight}}}^{t_k}$$

Pre-conditioned GMRES: 3 Times Less Memory

- Separate the linear problem in smaller blocks: x, y, z coordinates

$$\mathbf{H} = \begin{pmatrix} \mathbf{H}_{1,1} & \mathbf{H}_{1,2} & \mathbf{H}_{1,3} \\ \mathbf{H}_{2,1} & \mathbf{H}_{2,2} & \mathbf{H}_{2,3} \\ \mathbf{H}_{3,1} & \mathbf{H}_{3,2} & \mathbf{H}_{3,3} \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{pmatrix}$$

- Block-based SOR and forward substitution

$$\begin{pmatrix} \mathbf{H}_{1,1} & \mathbf{0} & \mathbf{0} \\ \mathbf{H}_{2,1} & \mathbf{H}_{2,2} & \mathbf{0} \\ \mathbf{H}_{3,1} & \mathbf{H}_{3,2} & \mathbf{H}_{3,3} \end{pmatrix} \begin{pmatrix} \mathbf{x}_1^{l+1} \\ \mathbf{x}_2^{l+1} \\ \mathbf{x}_3^{l+1} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_3 \end{pmatrix} - \begin{pmatrix} \mathbf{0} & \mathbf{H}_{1,2} & \mathbf{H}_{1,3} \\ \mathbf{0} & \mathbf{0} & \mathbf{H}_{2,3} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{x}_1^{l} \\ \mathbf{x}_2^{l} \\ \mathbf{x}_3^{l} \end{pmatrix}$$

- Each block: GMRES preconditioned with RASDD(1)+SSOR(2)
- Store the 3 diagonal blocks and use matrix-free products for the rest
- 3 iteration of block-SOR, $\delta_{pre} = \delta^{1/2}$

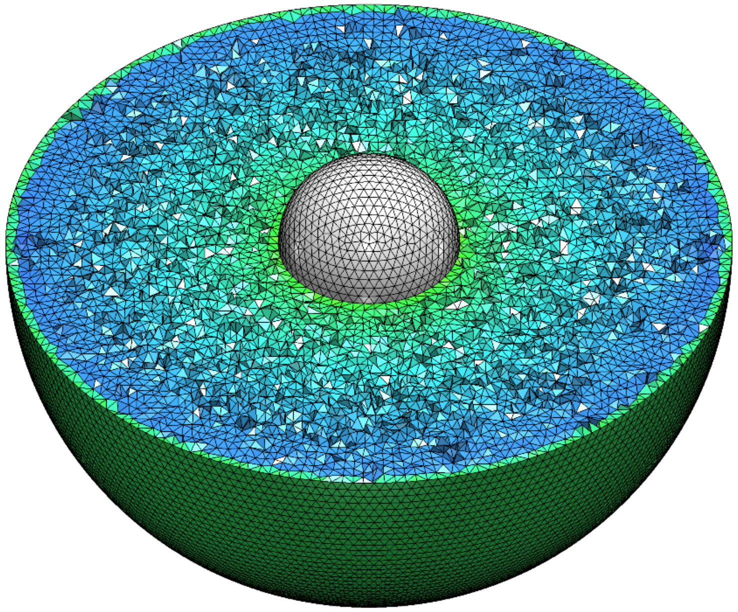
Example: Influence of the improvements

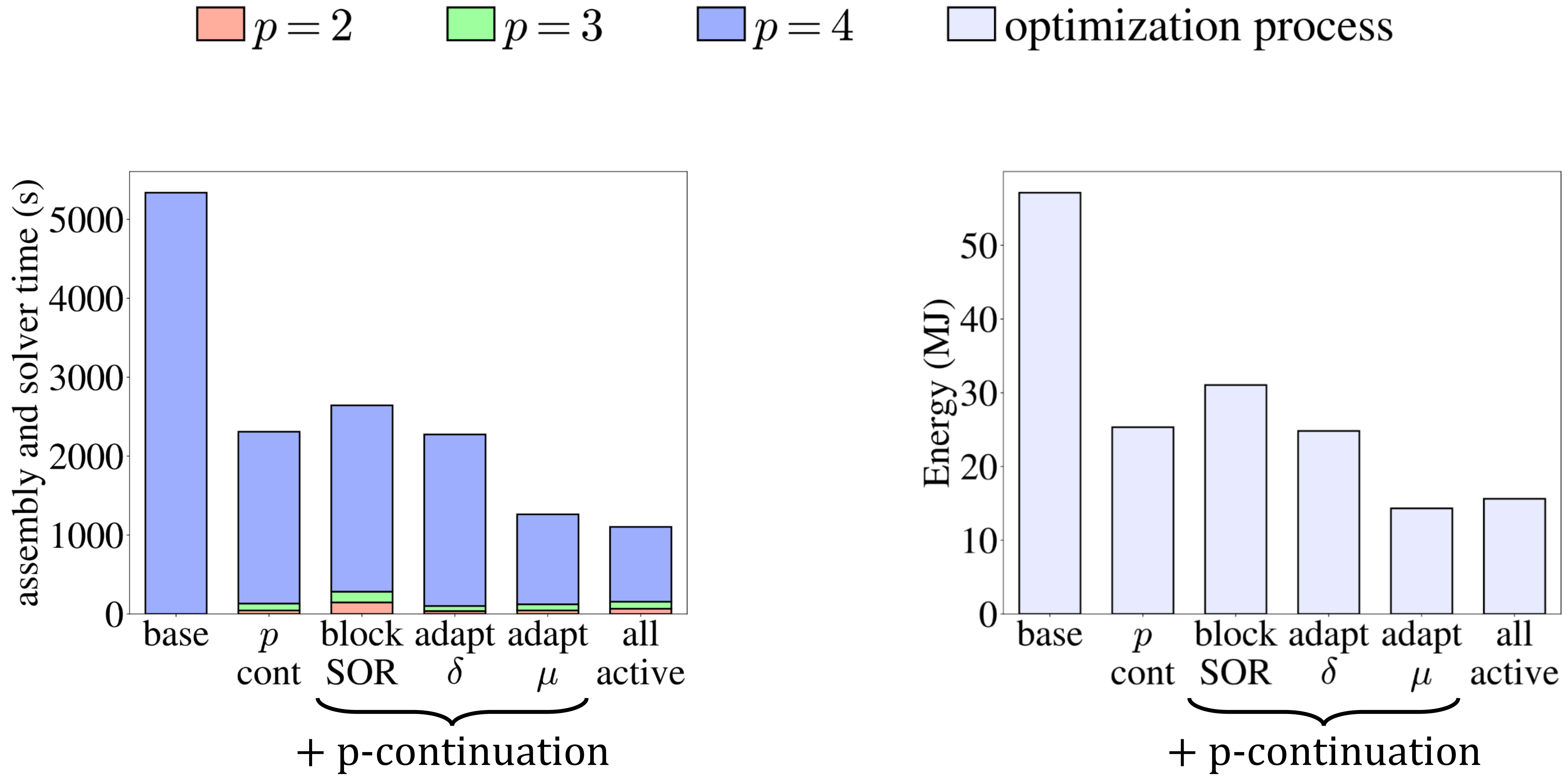
Uniform Mesh for a sphere

Isotropic mesh

1.44M elements, p=4

768 processors





Distributed Solver: Concluding Remarks

- **Key Improvements:**
  - p-continuation
  - Penalty parameter adaption
  - Block-SOR pre-conditioner
  - Forcing term (only for p=2)

- **Improvements**:
  - Decrease time and energy: 4 times
  - Decrease memory footprint: 3 times

3 times larger meshes with the same resources

High-Lift Prediction Workshop

(Ruiz-Gironés, Roca AIAA'22)

- Pre-process
- Post-process
- Software, libraries, and languages

HLPW: Pre-processing for curving-friendly inputs

- **Setup the simulation intent:** repair geometry & virtual model

- **Linear mesh generation:**
  - **Element size:** Simulation and geometry accuracy
  - **Curving:** curving-friendly mesh leads to easy curving process

- **Convert sequential inputs to parallel inputs**
  - Sequential inputs are bottlenecks
  - Create a hdf5 parallel input

HLPW: Check curved mesh and create output file

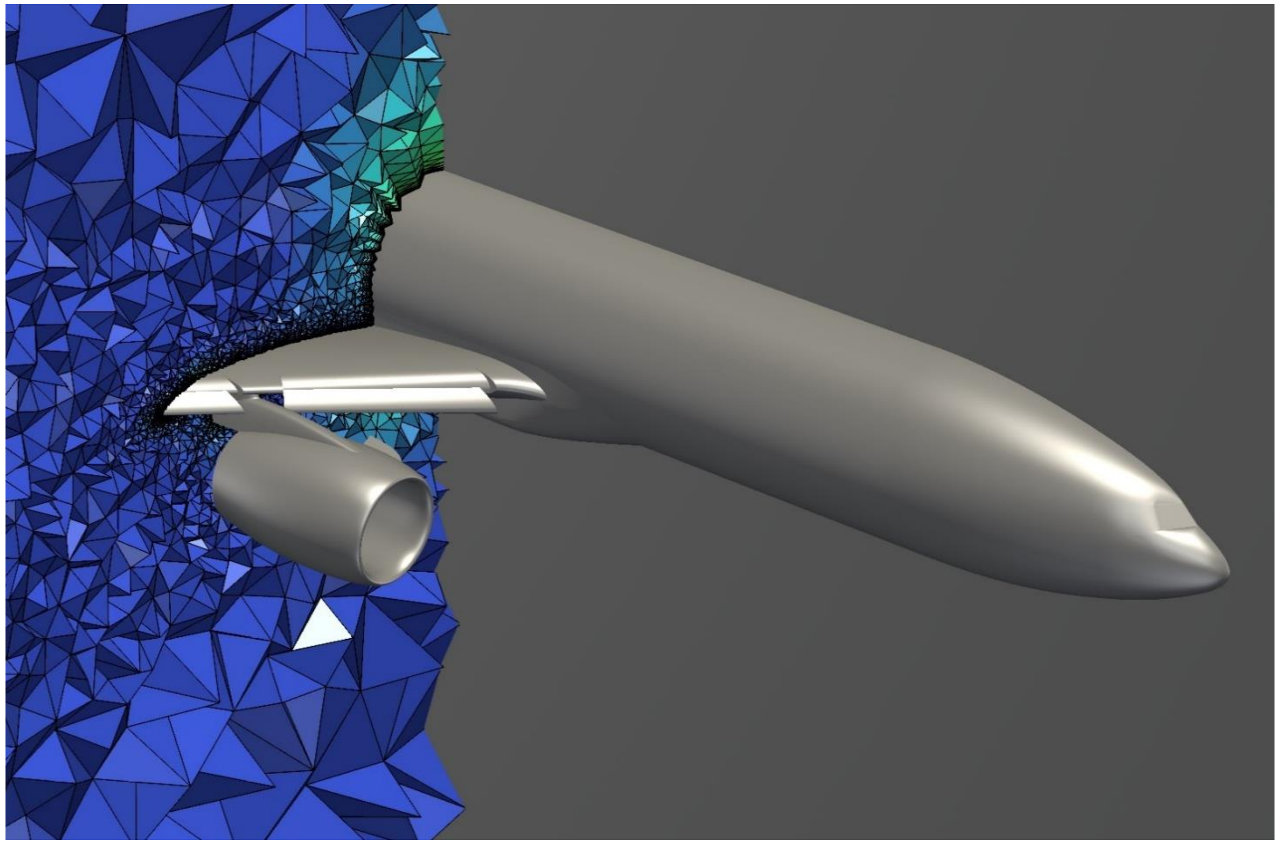
- **Mesh validity and quality:** Numeric validation
- **Visual inspection:** Paraview in distributed parallel
  Locate low-quality and low-accuracy elements
- **Curving iterative process:**
  Remesh low-quality and low-accuracy elements
- **Create output file:** python wrapper of cgns library

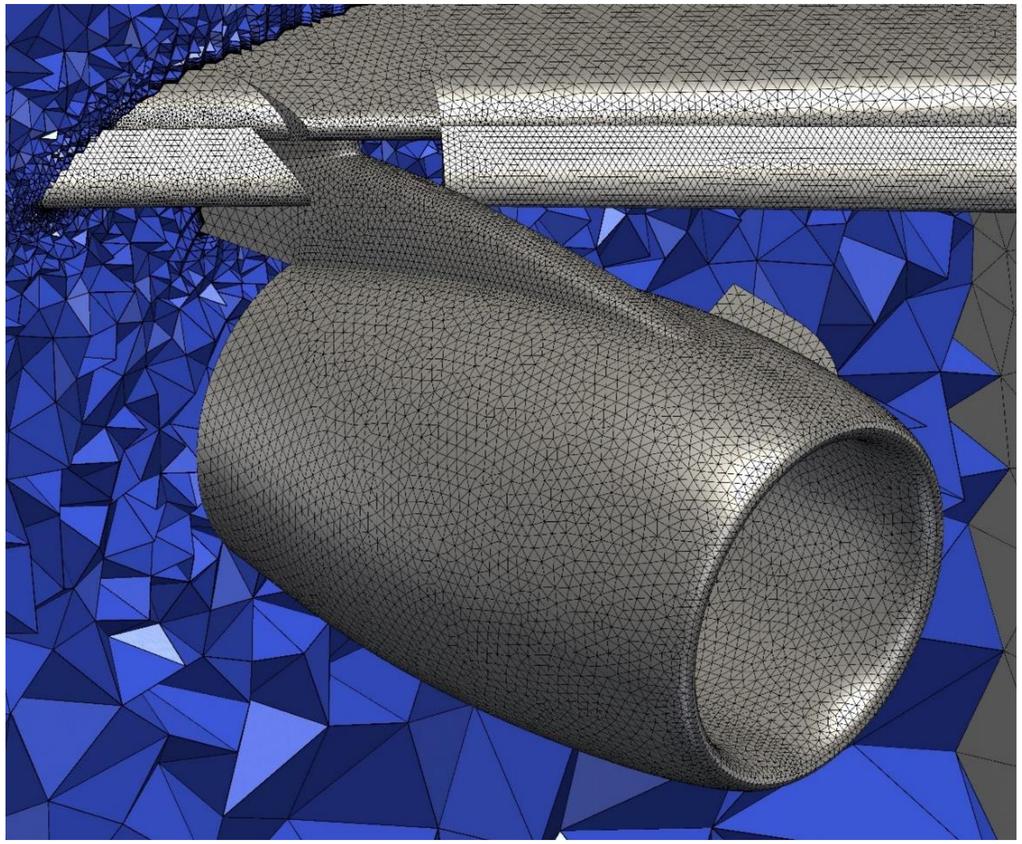
HLPW: Software, Libraries and Languages

- **Virtual model & linear mesh:** Pointwise
- **Distributed solver:** our python implementation with FEniCS library
- **CAD engine:** our python wrapper of Project Geode / OpenCASCADE
- **Linear solver library:** petsc4py interface to PETSc
- **Distributed parallel solver:** running on MareNostrum 4
- **Visualization:** distributed parallel Paraview
- **cgns output:** our python wrapper of cgns library

Example: CRM-HL of the 4^{th} HLPW

- **Mesh:** p=2 & p=3, 8M elements, boundary layer stretching 1:250
- **Accuracy:** relative to aircraft length $\sim 10^{-7} - 10^{-6}$
- **Computational resources:** 768 processors
  - p = 2 → 12 minutes
  - p = 3 → 48 minutes





Our meshes provided the best match with experimental results

(ZJ Wang AIAA'22)

Our Participation in HLPW: Concluding Remarks

- **Preparing curving-friendly inputs takes days** (human labor)
  - Tune the virtual model & linear mesh → Iterative process
  - Curving-friendly inputs → High-quality mesh in a short time

- **Mesh curving for the CRM-HL takes minutes** (computing wall time)
  - We generate larger meshes than the CFD community wants to run
  - Curving is a steady-state problem with less unknowns than CFD

- **You can try our meshes!**
  - Free to download in the 4^{th} & 5^{th} HLPW websites

Our meshes provided the best match with experimental results

Summary & Conclusions

Summary & conclusions: Large-Scale Curving

- **Mesh curving constrained formulation:**
  - Always numerically valid
  - Optimal quality
  - Approximates target geometry
  - Tightly converged
- **Complex geometry in parallel:** mesh approximates virtual geometry
- **Large-scale curving:** 3 times larger meshes on thousands of cores
- **High-lift prediction:** Our meshes lead to best match with experiments

Our curving enables high-fidelity simulations on complex geometries

Thank you for your attention!

European Research Council
Established by the European Commission

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement No 715546.