# Flutter analysis with an Euler-based solver in OpenFOAM

Andrea Mannarino, presenter Sergio Ricci

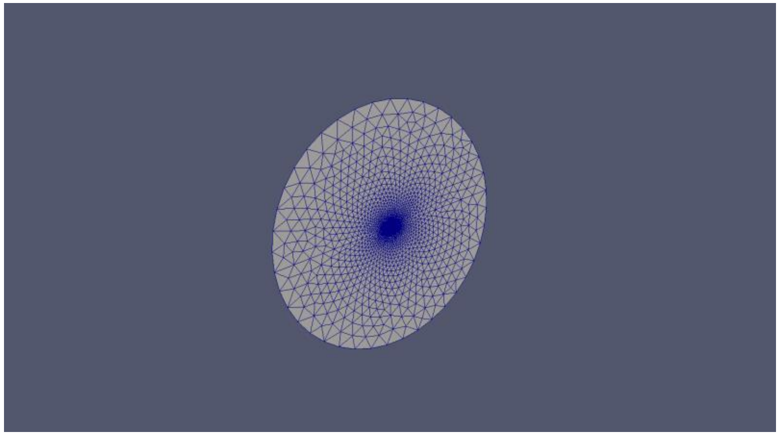
POLITECNICO
MILANO 1863

Politecnico di Milano
Department of Aerospace Science and Technology
Milano - Italy

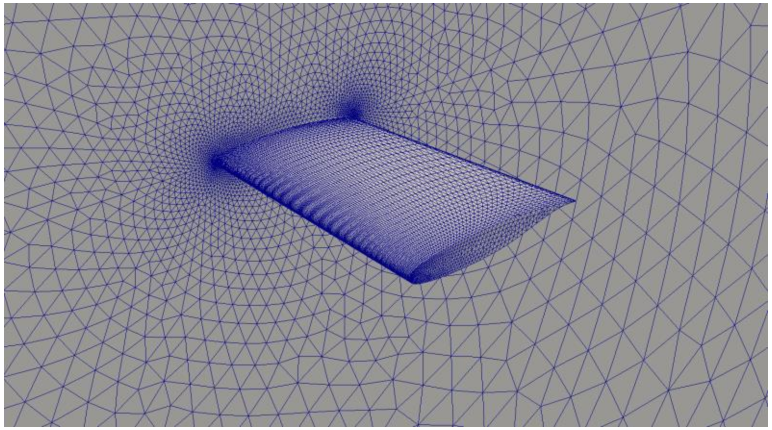
# Table of contents

- Mesh generation
- Aerodynamic solver
- Flutter point estimation
- Results analysis
- Concluding remarks

# Mesh generation



Semi-spheric domain

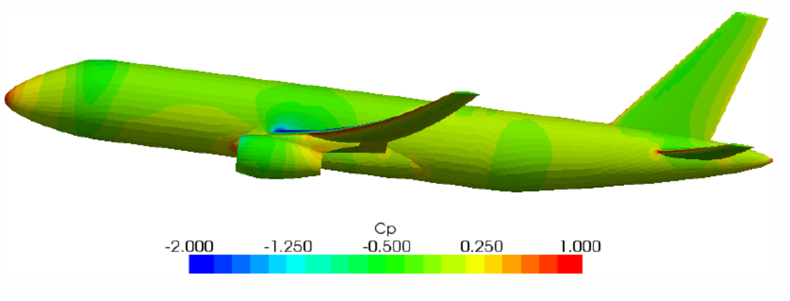


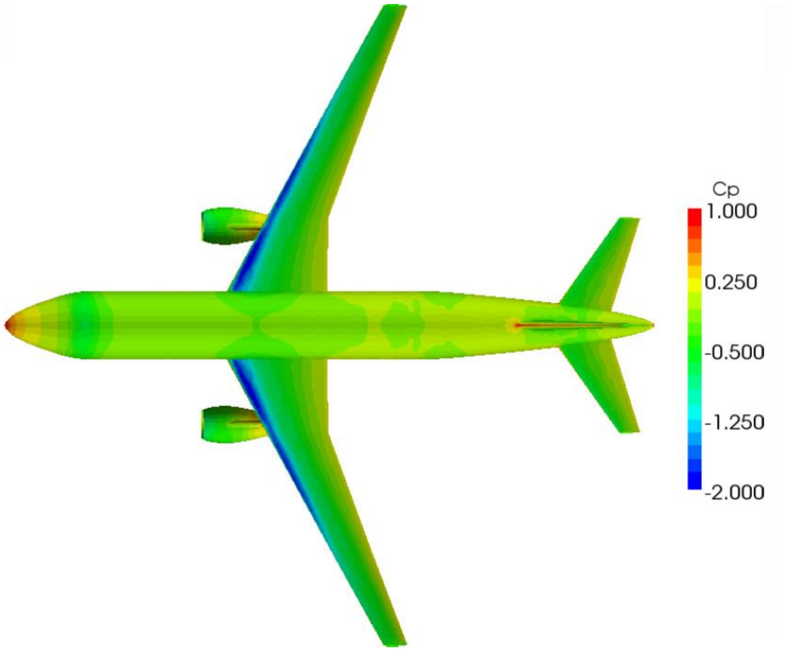
Wing close-up

- Problems in converting the provided meshes to OpenFOAM
- Mesh created with Pointwise through the IGES file available on the AEPW2 project web site
- Spatial discretization of the domain:
  - Coarse mesh with 320k cells
  - Medium mesh with 690k cells

# Aerodynamic solver: AeroFoam

- In-house solver developed at the DAST supported by OpenFOAM libraries
- RANS, cell-centered, density based solver for aero-servo-elastic applications
- First density-based RANS solver implemented in OpenFOAM to overcome the limits of the available pressure based solvers in transonic application



# Aerodynamic solver: AeroFoam

- Euler-option is selected for the following simulations: viscosity and thermal conductivity effects are not modeled
- Convective fluxes are discretized by the Roe's approximated Riemann solver, blended by the centered approximation of Lax-Wendroff
- Entropy fix of Harten and Hyman and van Leer flux limiter
- Time discretization performed by an explicit multi-step Runge-Kutta scheme of the 5$^{th}$ order
- Combined dual-time stepping and a full-approximation storage multi-grid technique to speed up the convergence between time steps

# Simulation settings

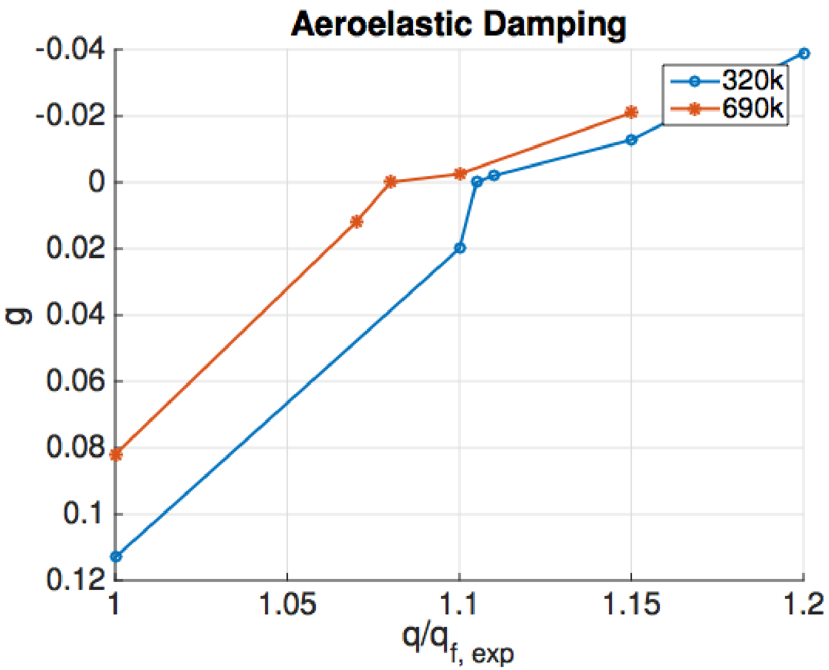
## Aeroelastic interface

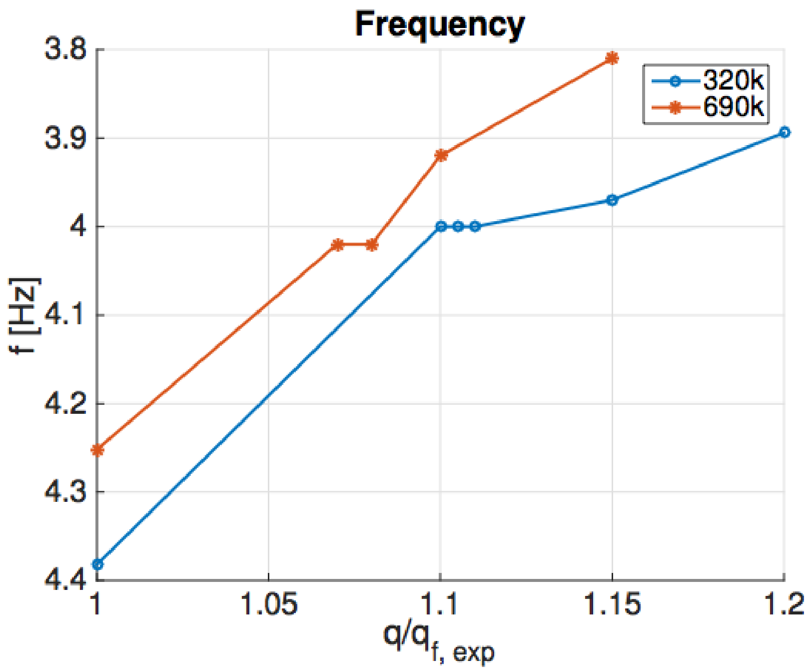
- Mode shapes downloaded from the AEPW2 project web site
- Because the wing is rigid, the mesh is translated and rotated rigidly during the simulation
- The coupling between structural and aerodynamic models is performed at each time step through a linear method

## Aerodynamic solver parameters

- Time step convergence analysis: 1e-3, 5e-4, 2.5e-4 s
- CFL set to 2.0
- 1000 iterations in pseudo-time are allowed between each time step
- Two multi-grid levels are used (V-cycle)

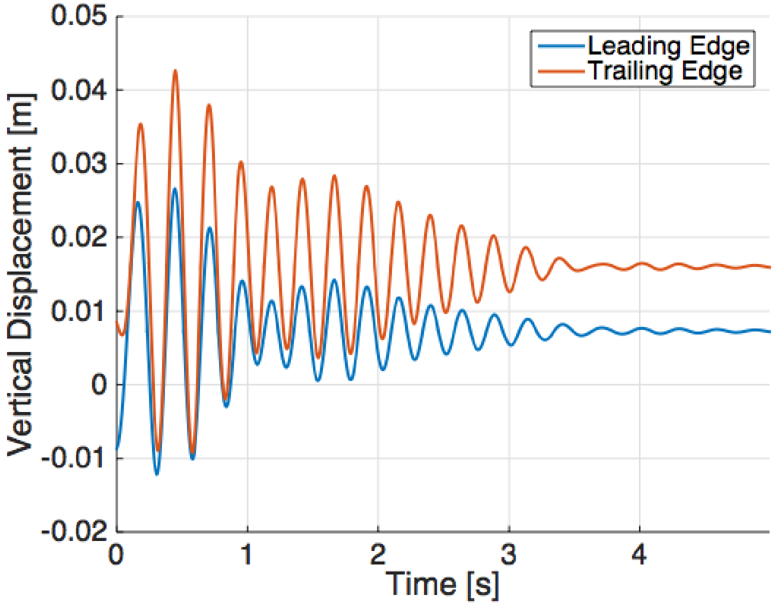
# Flutter point estimation



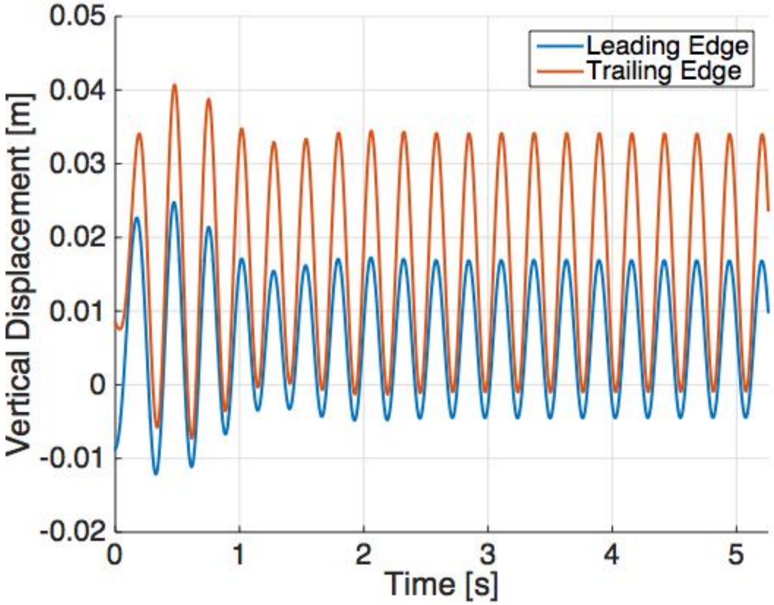




- Flutter point always overestimated vs experimental value
- Error on flutter frequency smaller than damping
- Flutter point for 320k mesh -> 1.105 experimental value
- Flutter point for 690k mesh -> 1.080 experimental value
- Flutter frequency always around 4 Hz

# Experimental vs numerical flutter




Experimental

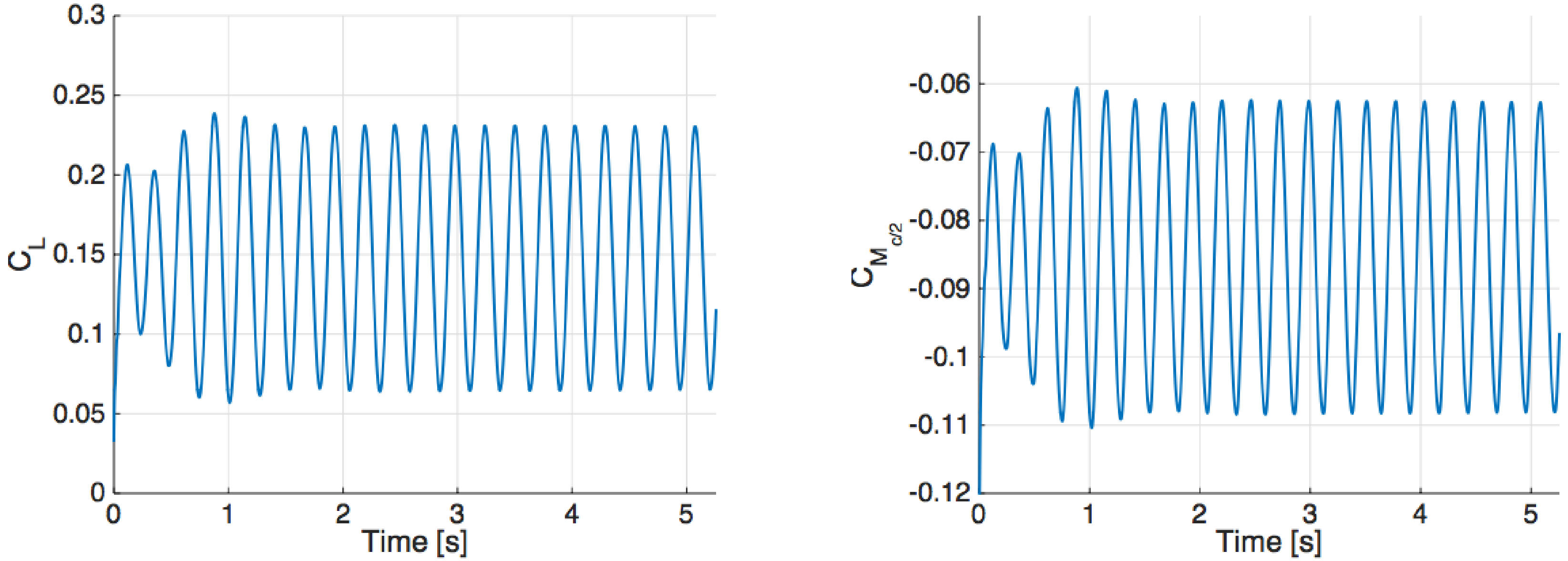



Numerical

## Temporal convergence at flutter speed

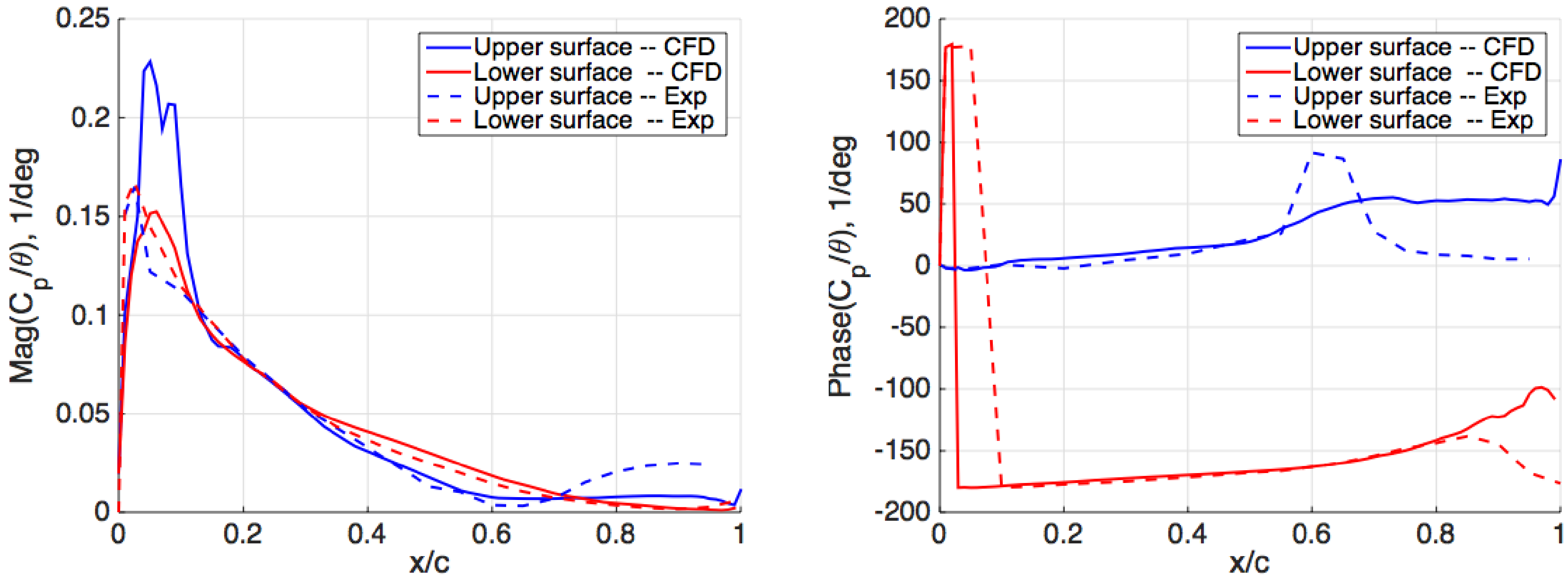
- Damping variation from dt = 1e-3 to 5e-4
- No variation of damping from dt = 5e-4 to 2.5e-4

# Analysis of the flutter solution




- The oscillations are not symmetric with respect to the origin
- The average angle of rotation is negative
- The average wing plunge is positive

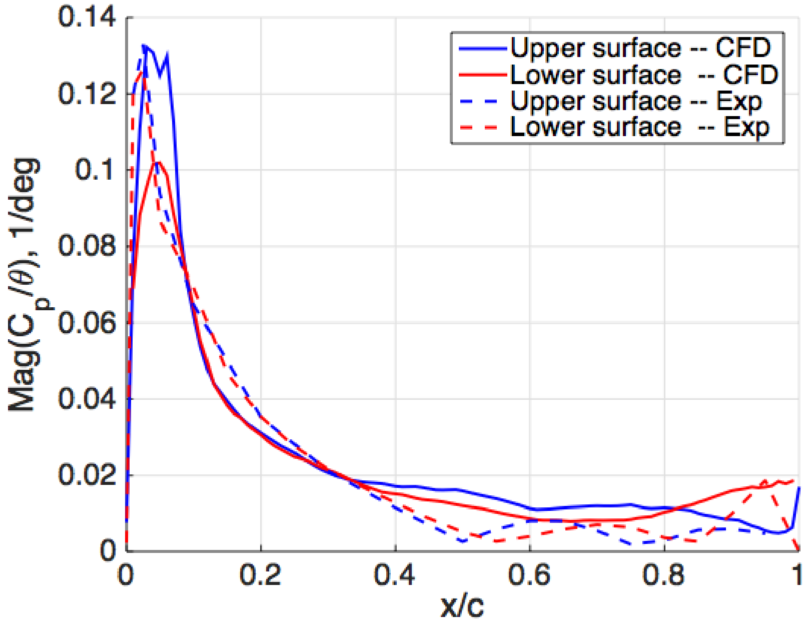
# Load distribution at computational flutter

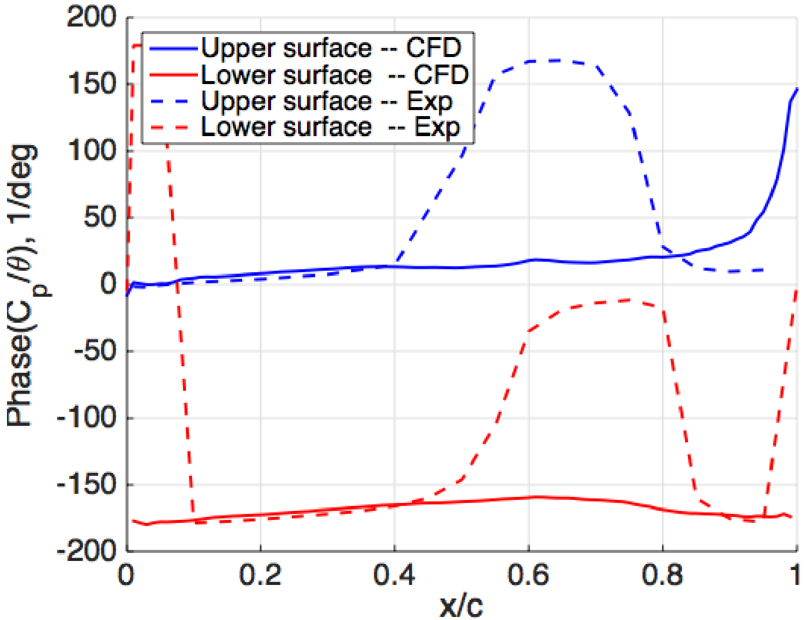



**FRF @ 60% span**

- The computational model presents a higher peak on the upper surface
- Phase predicted with good accuracy, missing effect at 60% chord, probably due to boundary layer transition

# Load distribution at computational flutter

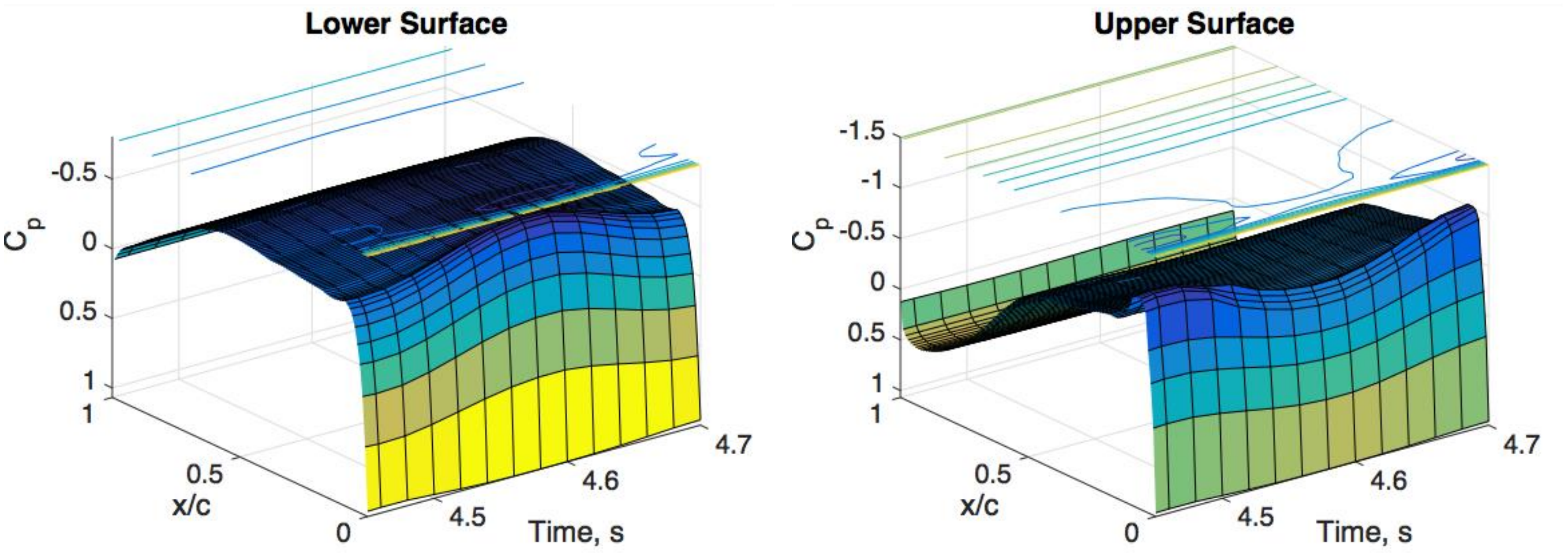



**FRF @ 95% span**

- The computational model presents a smaller peak on the lower surface
- Phase not well predicted, still missing effect at 60% chord, probably due to boundary layer transition

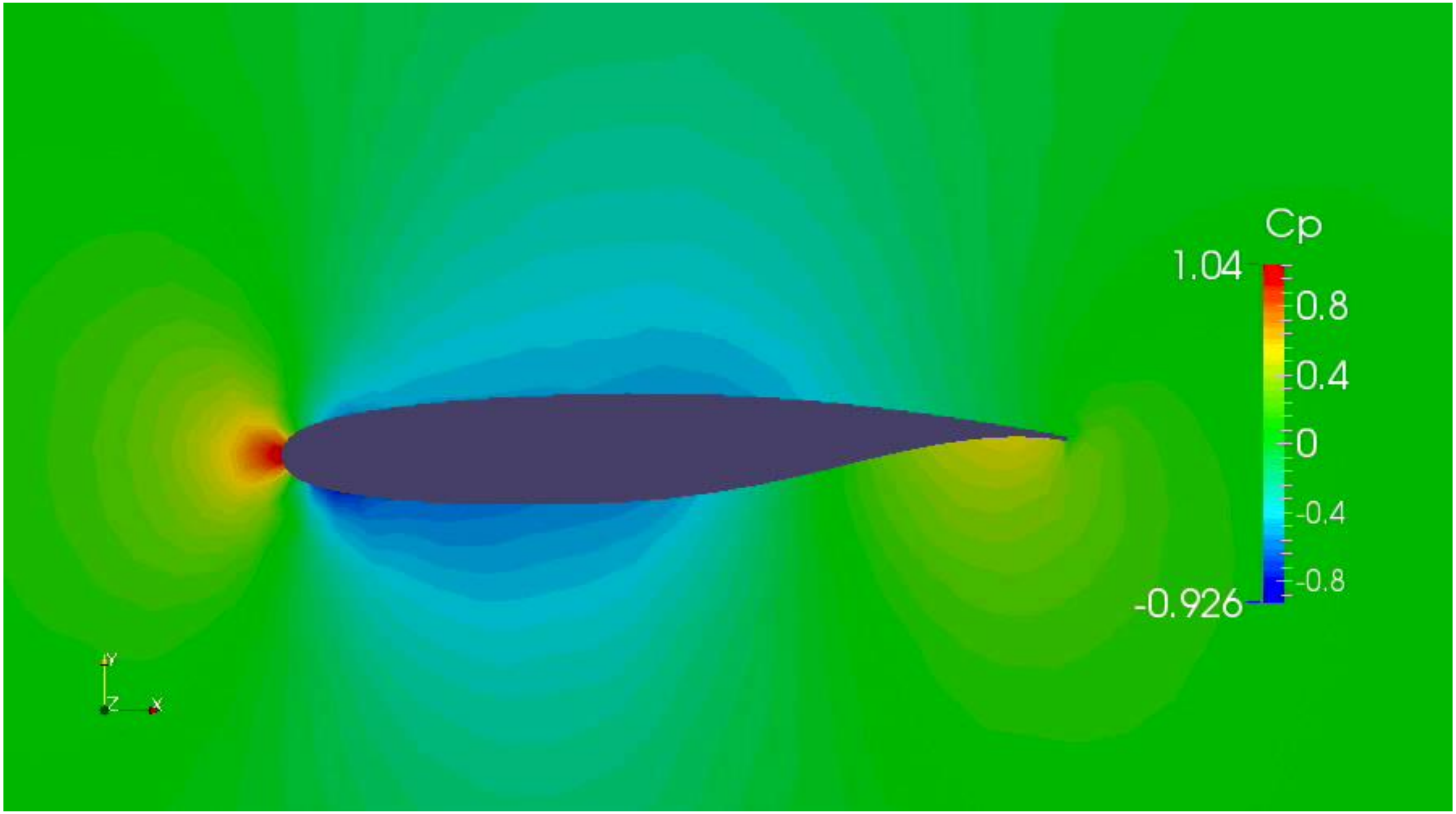
# Load distribution vs time



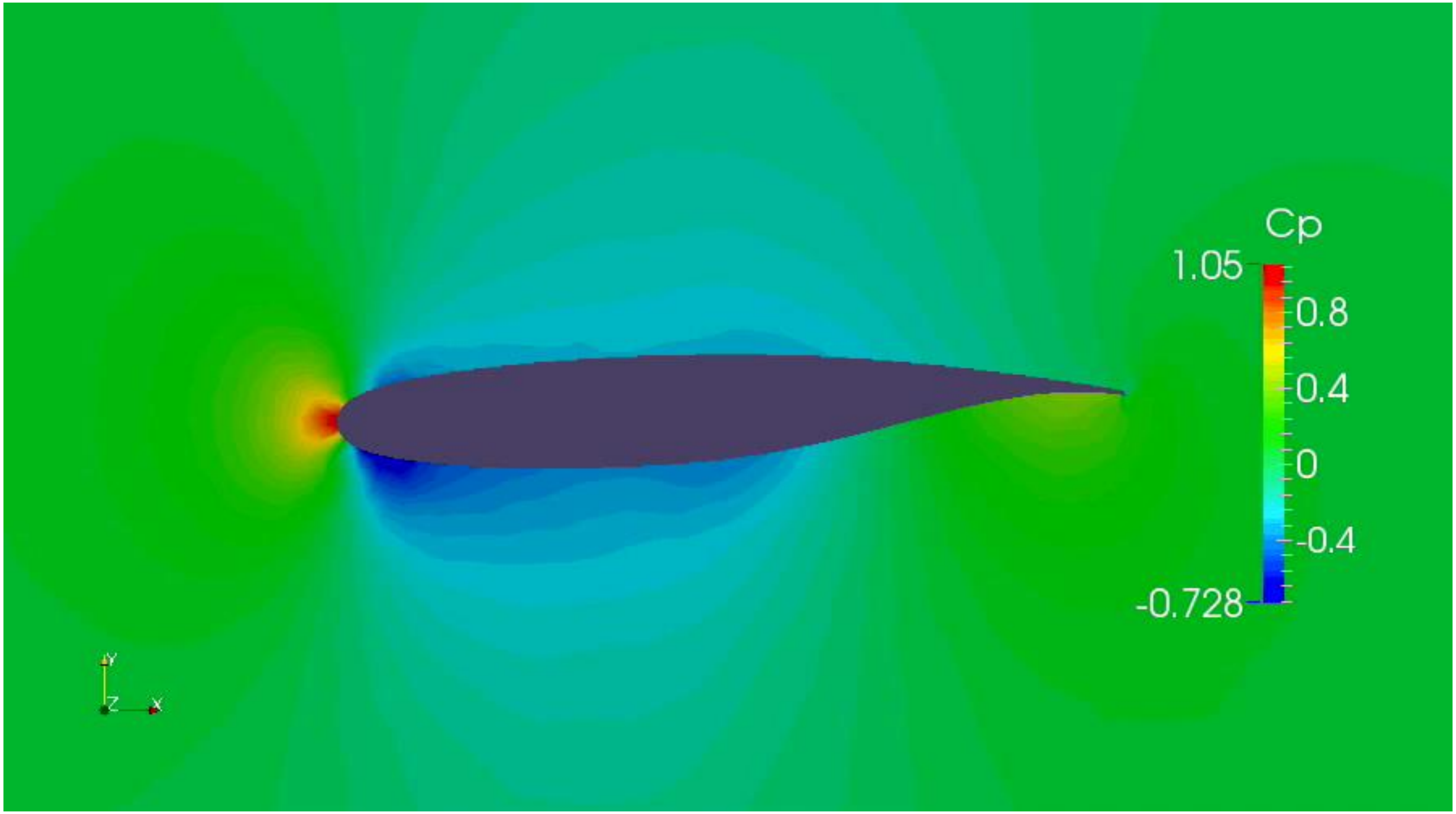

**Cp @ 60% span**

- A weak shock moves back and forth the chord on both surfaces
- A narrow peak is present on the leading edge of the upper surface

# Pressure field @ 60% span



# Pressure field @ 95% span



# Concluding remarks

- Euler-based flutter simulations have been presented
- Good accuracy in flutter estimation (error smaller than 10%) has been found
- The flutter point is always overpredicted with respect to the experimental value
- Not so accurate in load distribution predictions, probably due to non-modeled effects
- Additional analyses with refined meshes should be carried out to confirm the convergence toward the flutter predicted by the experiments

Thank you!!!

Any question?

# Cases under investigation

case 1) Mach = 0.7

AoA = 3°

Dynamic data type = Forced oscillation, f = 10Hz, |theta| = 1°

notes: attached flow, OTT exp data, R-134°

case 2) Mach = 0.74

AoA = 0°

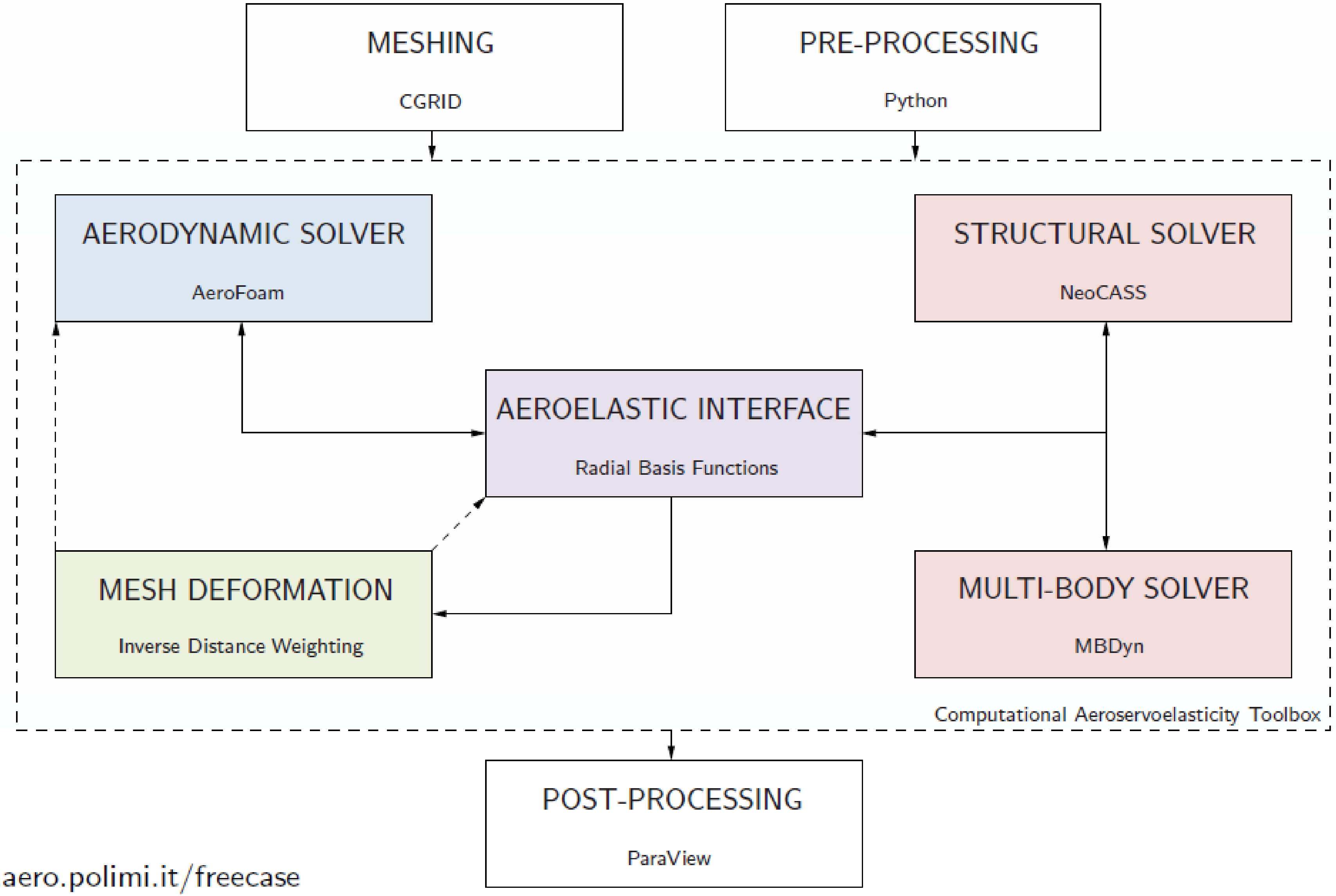
Dynamic data type = Flutter

notes = flow state unknown, PAPA exp. data, R-12

Solver:

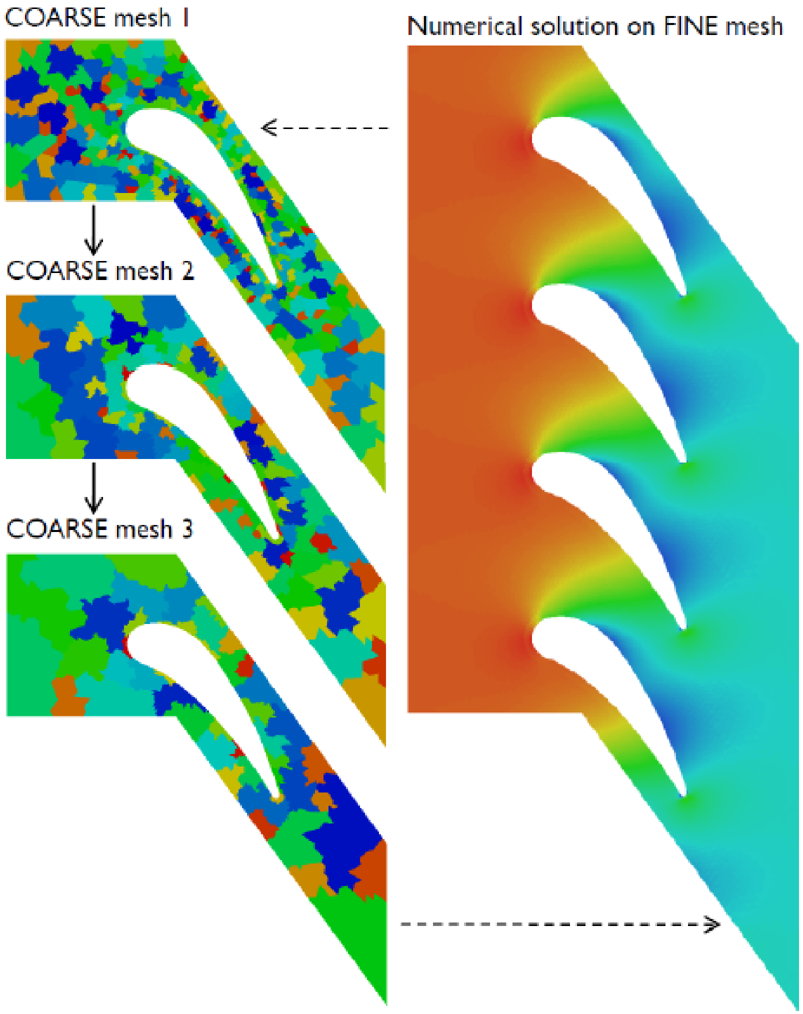
- Explicit - dual time stepping
- density based
- euler / rans (spalart allmaras / SST)
- grid deformation / traspiration
- GPU

# FreeCASE toolbox




www.aero.polimi.it/freecase

# AeroFoam




Multi-Grid (MG) agglomeration and Full-Approximation-Storage (FAS)

## Motivation and objective:

a) First density-based ALE RANS solver in OpenFOAM

b) To overcome the limits of built-in pressure-based solvers in the transonic regime (e.g. sonicFoam)

c) Benchmarking vs. EDGE and FLUENT

## Features:

a) Coupled formulation in conservative variables

b) Space discretization ($1^{st}$, $2^{nd}$ order accuracy)

- Roe's Approximate Riemann Solver (ARS)
- Lax-Wendroff (LW) scheme with flux limiters
- Directional Residual Smoothing (RS)

c) Time discretization ($1^{st}$, $2^{nd}$ order accuracy)

- Explicit multi-step Runge-Kutta (RK) scheme
- Local Time-Stepping (LTS)
- Double Time-Stepping (DTS)
- Multi-Grid (MG) acceleration (FMG and FAS options)

d) Automatic handling of parallel communication, cyclic boundaries, Generic Grid Interface (GGI)

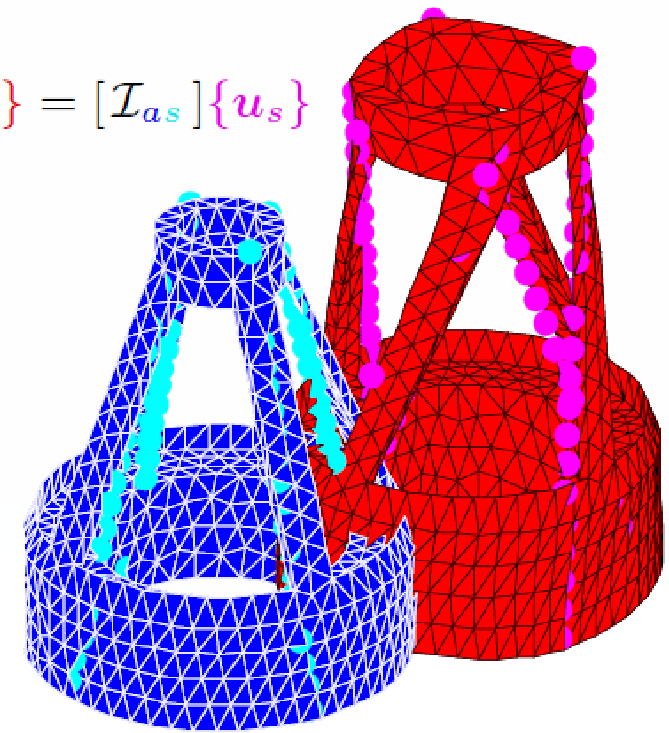
# FSI

**Target:** closed loop connection between structural and aerodynamic sub-systems

a) projection (in PVW sense) of aerodynamic forces onto structural displacements

b) translation of structural displacements into aerodynamic boundary conditions

$$\{\boldsymbol{u}_a\} = [\mathcal{I}_{as}]\{\boldsymbol{u}_s\}$$



**Moving Least Squares (MLS):**

- connect topologically different domains
- exact treatment of rigid motions
- accuracy, smoothness & efficiency trade-off

  Minimize $\displaystyle\int_\Gamma \phi(\,\mathrm{Tr}(\boldsymbol{u}_a)|_\Gamma - \mathrm{Tr}(\boldsymbol{u}_s)|_\Gamma\,)^2 \mathrm{d}\mathcal{S}$

- weighting via Radial Basis Functions (RBF)

# A hierarchy of mesh deformation tools

R Least-squares identification of translation vector **s** and linear map tensor **T**

$$\Delta \boldsymbol{x}_j = \mathbf{s} + \mathbf{T}\,\boldsymbol{x}_j + \boldsymbol{\varepsilon}_j = \mathbf{s} + (\mathbf{R} - \mathbf{I})\,\boldsymbol{x}_j + \mathbf{D}\,\boldsymbol{x}_j + \boldsymbol{\varepsilon}_j \quad \forall \quad j \in [1,\, N_b]$$

easier to implement, rotation tensor follows: $(\mathbf{R} - \mathbf{I}) = \mathsf{s}_\phi \mathbf{K}_\times + (1 - \mathsf{c}_\phi)\mathbf{K}_\times \mathbf{K}_\times$

E Elastic contribution by means of Sparse Inverse Distance Weighting (SIDW)

$$\Delta \boldsymbol{x}_k = \sum_{j=0}^{N_b} \frac{\mathbf{IDW}_{(k,\,j)}}{|\mathbf{IDW}_{(k,\,:)}|}\,\boldsymbol{\varepsilon}_j \quad \forall \quad k = [1,\, N_v] \quad \text{with} \quad \mathbf{IDW}_{(k,\,j)} = \frac{1}{\|\boldsymbol{x}_k - \boldsymbol{x}_j\|^p}$$

memory/efficiency trade-off sparse fix: $\mathbf{SIDW}_{(k,\,j)} = \mathbf{IDW}_{(k,\,j)}$ if $\mathbf{IDW}_{(k,\,j)} > \xi$

T Residuals (if any) simulated by means of Transpiration boundary conditions