Acceleration of Turbomachinery steady CFD Simulations on GPU

Mohamed H. Aissa[1,2]
Dr. Tom Verstraete[1]
Prof. Cornelis Vuik[2]

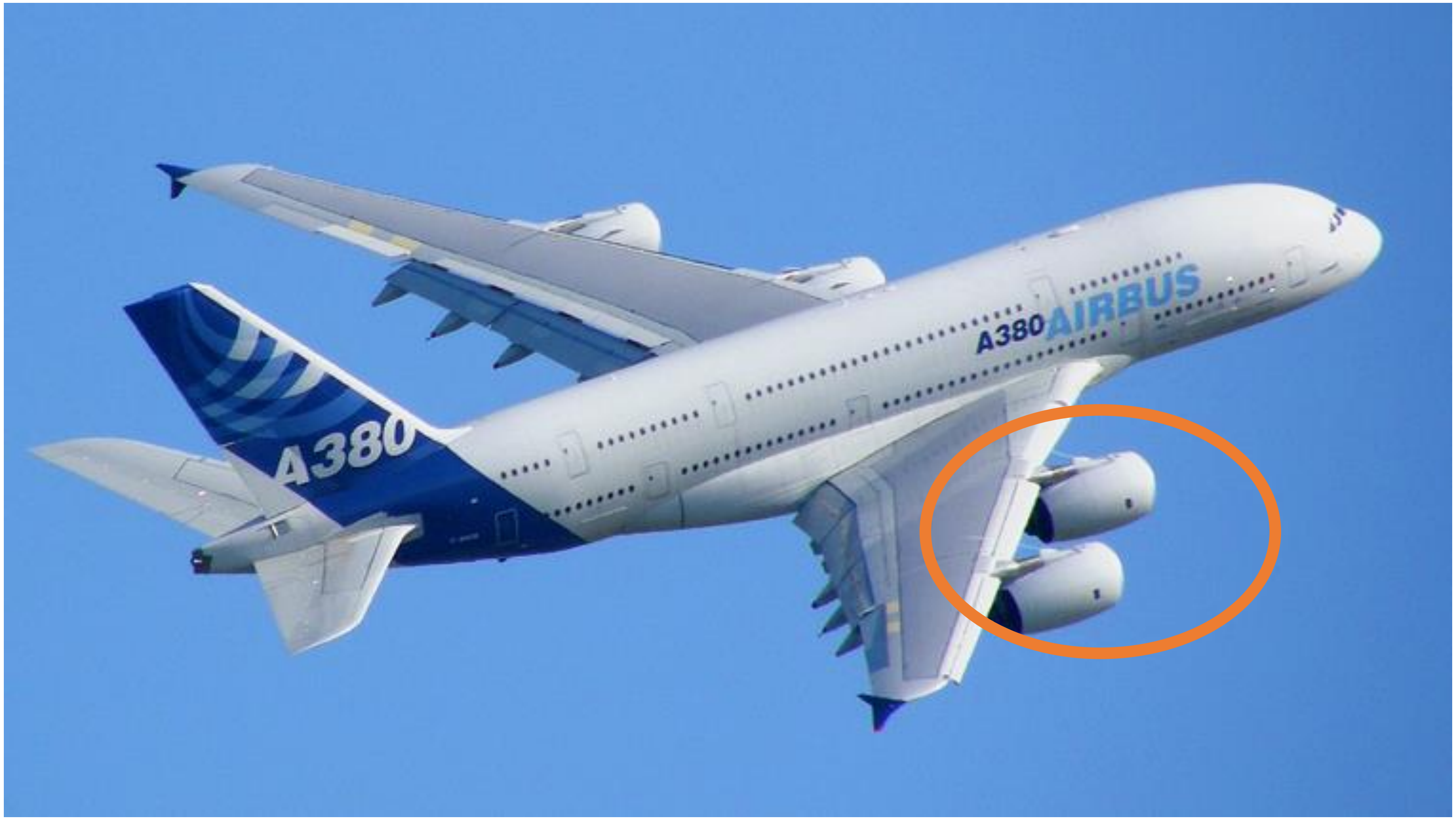
[1] Turbomachinery Department , Von Karman Institute, Belgium
[2] Delft Institute of Applied Mathematics, TU Delft, the Netherlands

VON KARMAN INSTITUTE FOR FLUID DYNAMICS

TU Delft Delft University of Technology

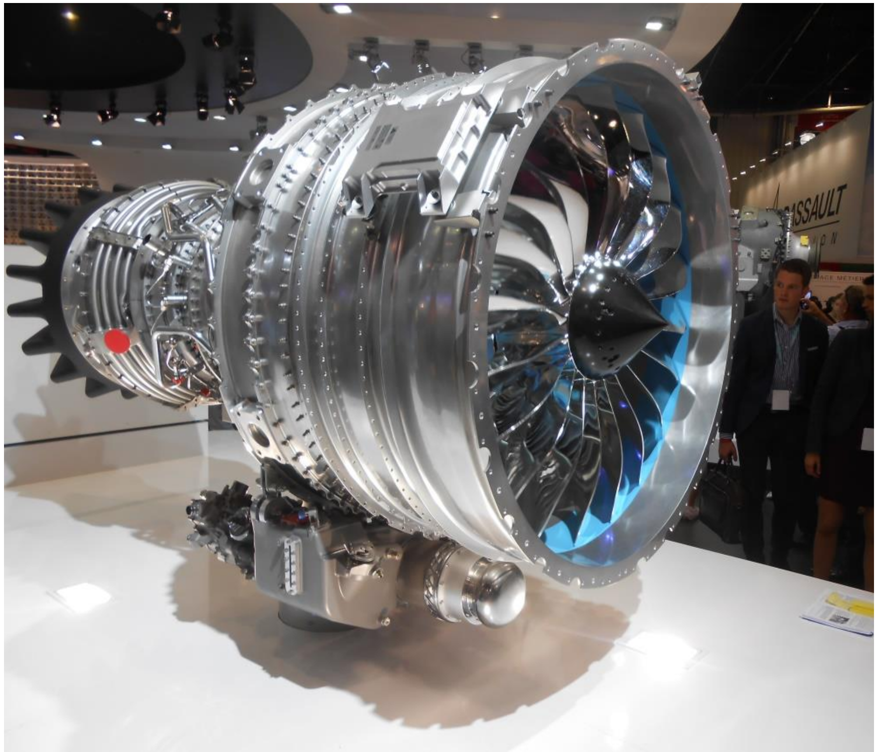
Unconventional HPC,
EuroPar 2016 WS
Grenoble, 23.08.2016

Topic of Interest:
Reduce Fuel Consumption and CO_2 Emission

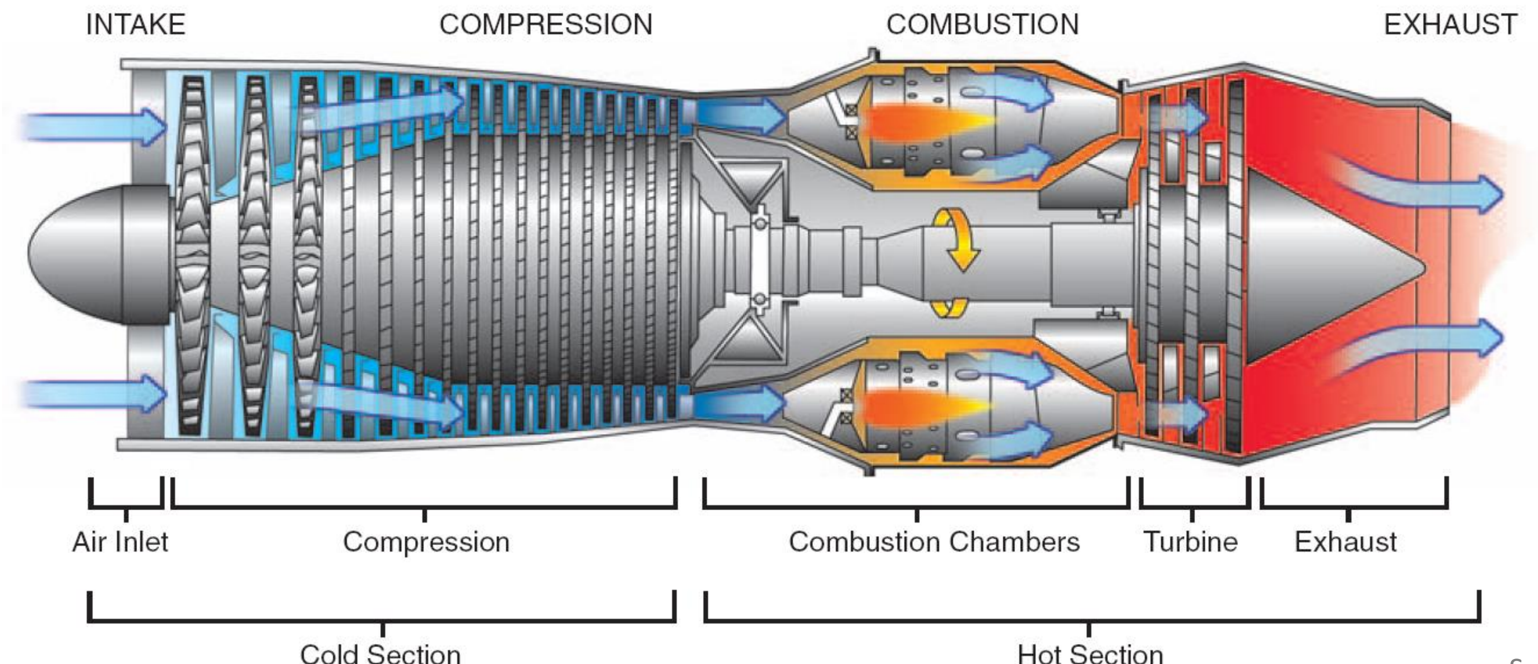


Wikipedia.org

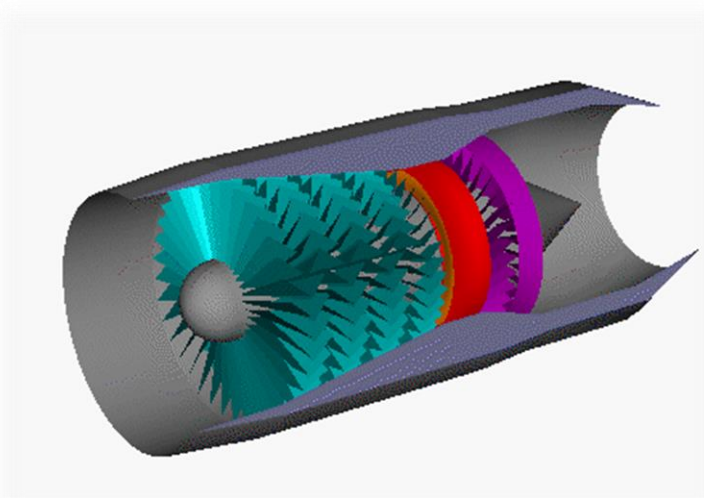
Turbomachinery is about Performance and Efficiency



Axial Jet Engine



Source:
Wikipedia.org



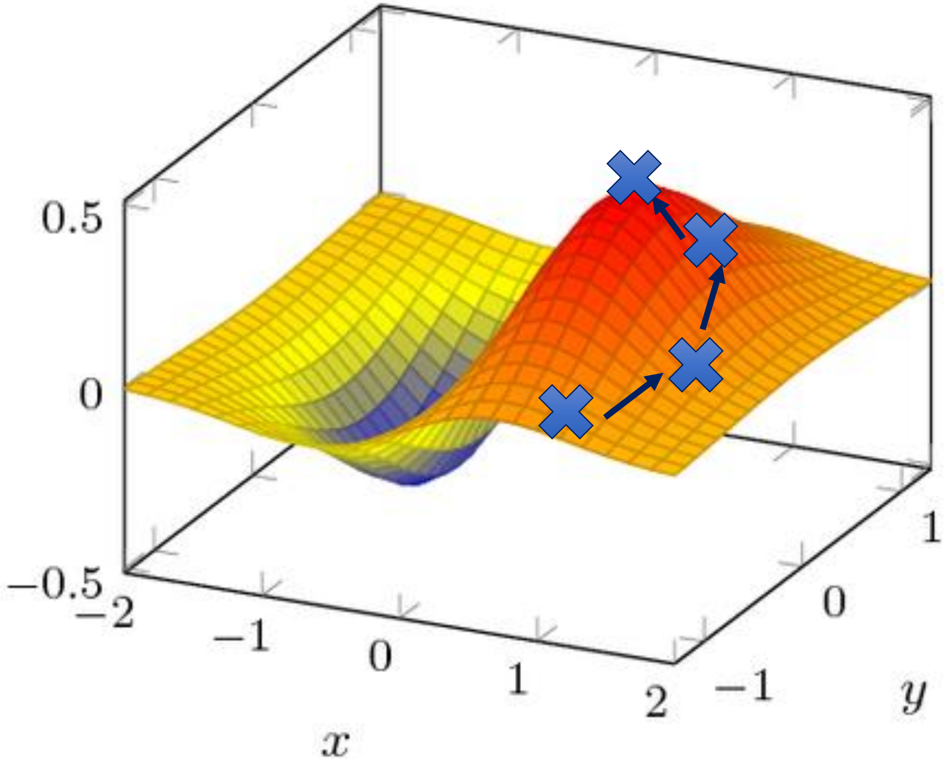
Content

- Multidisciplinary Optimization
- CFD simulations on GPU
  - Literature review
  - Implicit RANS Implementation
  - Benchmark
- Optimization Case

Optimization algorithm

Derivative-based optimization	Derivative free methods: e.g. Population based
• fast convergence but .. • derivative evaluation could be complicated and problem specific (adjoint, automatic differentiation)	• Simplicity • Black box approach of the evaluation but .. • Large number of evaluations

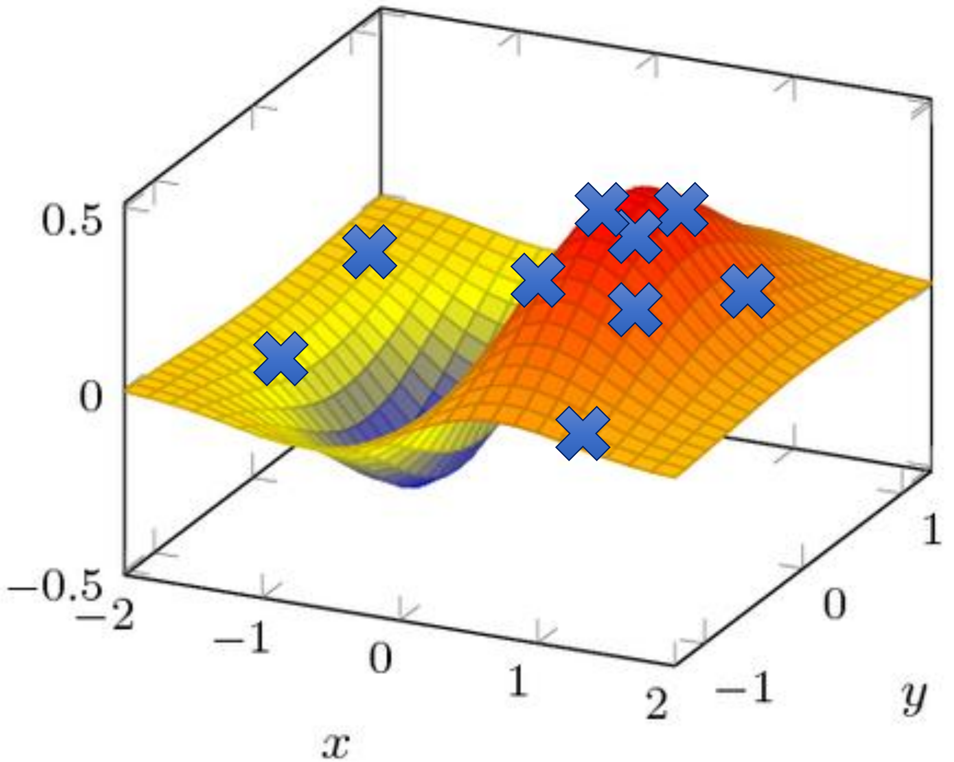
$$\min f(\boldsymbol{x})$$
$$\textit{subject to}$$
$$g(\boldsymbol{x}) \leq 0$$



Optimization algorithm

Derivative-based optimization	Derivative free methods: e.g. Population based
• fast convergence but .. • derivative evaluation could be complicated and problem specific (adjoint, automatic differentiation)	• Simplicity • Black box approach of the evaluation but .. • Large number of evaluations

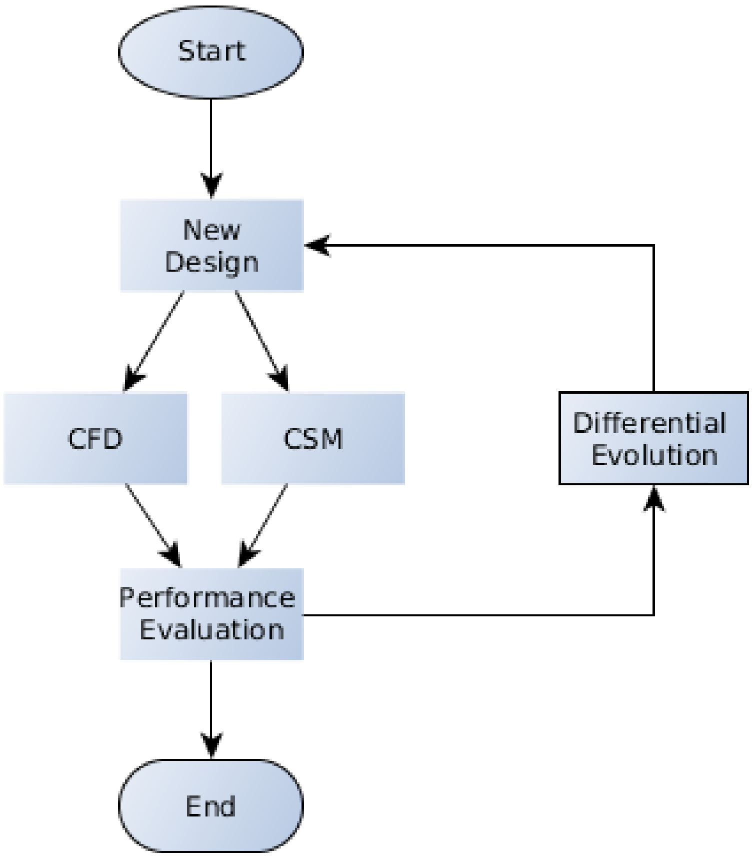
$$\min \mathrm{f}(\boldsymbol{x})$$
$$\textit{subject to}$$
$$g(\boldsymbol{x}) \leq 0$$



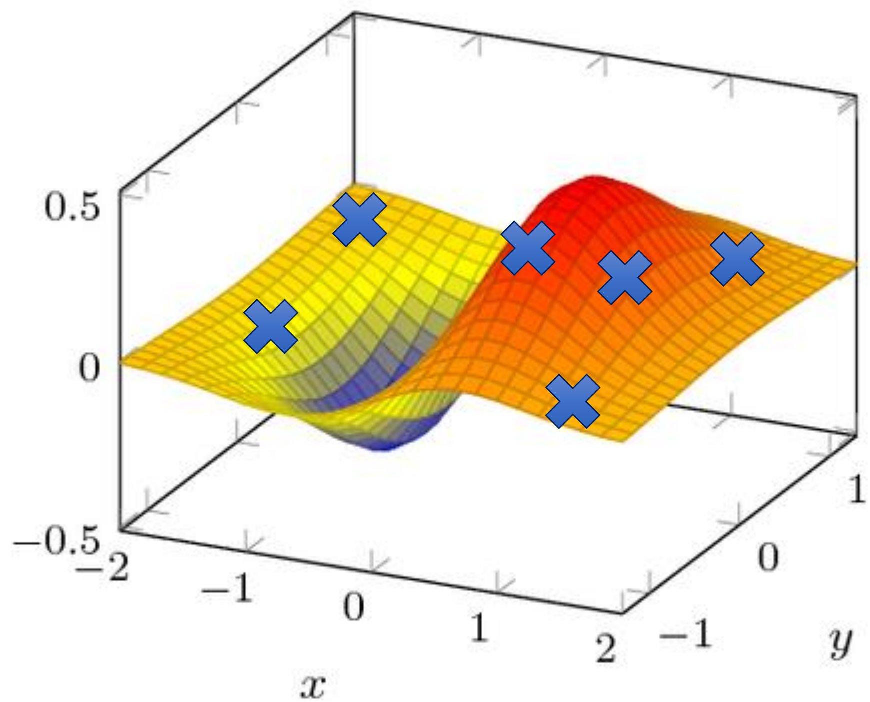
CFD: Core of the Optimization

CFD much slower than CSM

Need for acceleration -> GPU

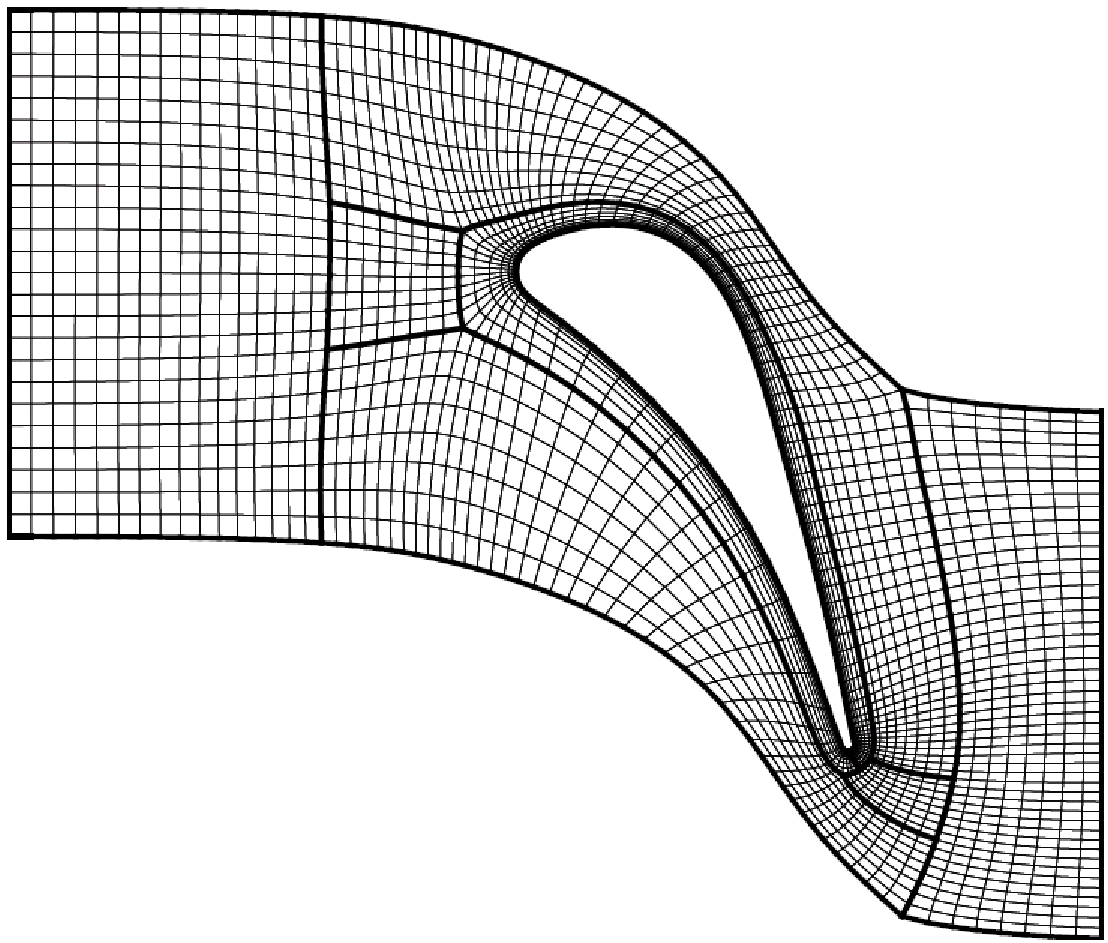


CADO: the VKI in-house optimizer



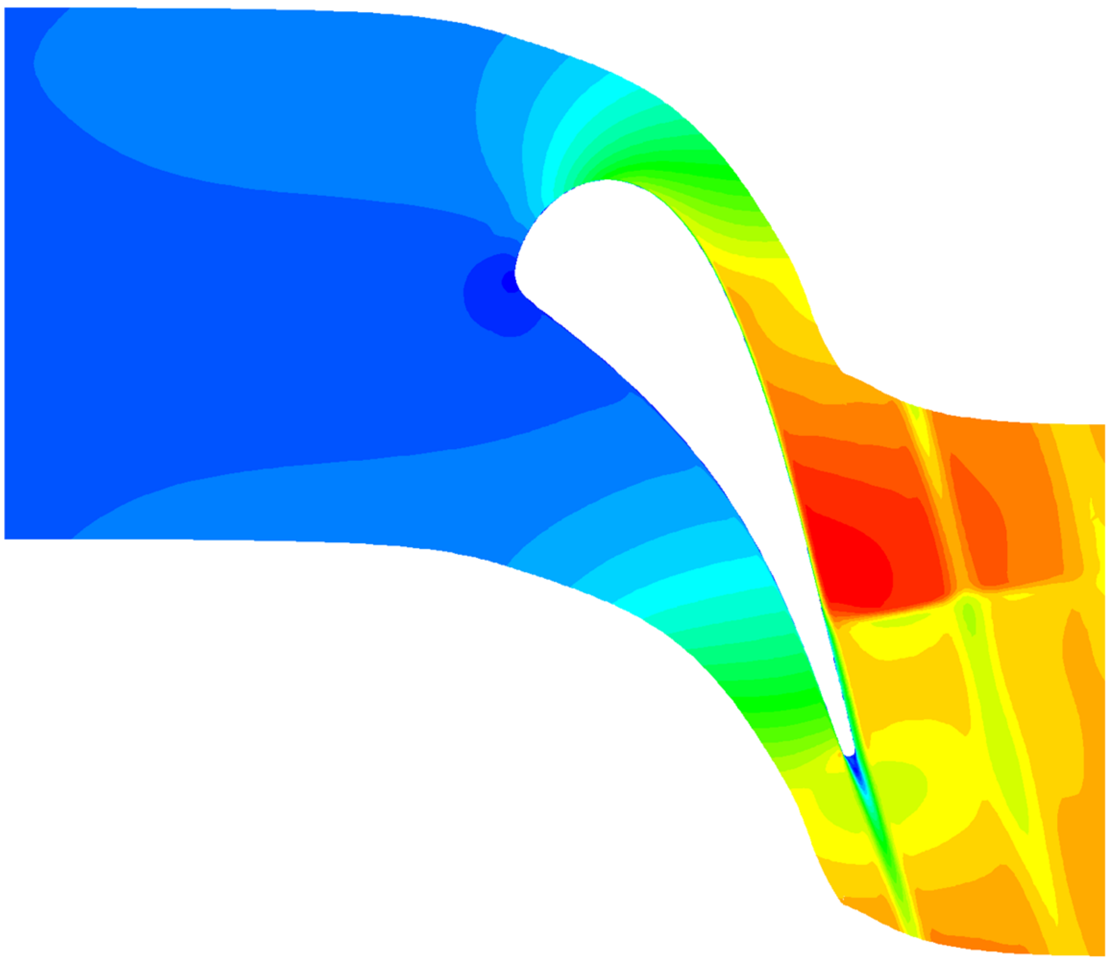
Steady CFD Simulations

- Simulation with a unique solution for given boundary Conditions.
- A start solution is advanced iteratively in time until convergence



Steady CFD Simulations

- Simulation with a unique solution for given boundary Conditions.
- A start solution is advanced iteratively in time until convergence



Numerical Scheme:

$$\frac{\partial}{\partial t}\int_{\Omega} \boldsymbol{W}\, d\Omega + \oint_{\partial\Omega} (\boldsymbol{F_c} - \boldsymbol{F_v})\, dS = \int_{\Omega} \boldsymbol{Q}\, d\Omega \qquad \boldsymbol{W} = \{\rho, \rho V_x, \rho V_y, \rho V_z, \rho E\}$$

$$\frac{(\Omega\bar{M})_I}{\Delta t_I}\Delta\vec{W}_I^n = -\beta\vec{R}_I^{(n+1)} - (1-\beta)\vec{R}_I^n$$

Explicit Time Stepping (β=0):

$$\Delta\boldsymbol{W}^n = -\frac{\Delta t}{\Omega}\boldsymbol{R}^n$$

Implicit Time Stepping (β=1):

$$\vec{R}_I^{n+1} \approx \vec{R}_I^n + \left(\frac{\delta\vec{R}}{\delta\vec{W}}\right)_I \Delta\vec{W}^n$$

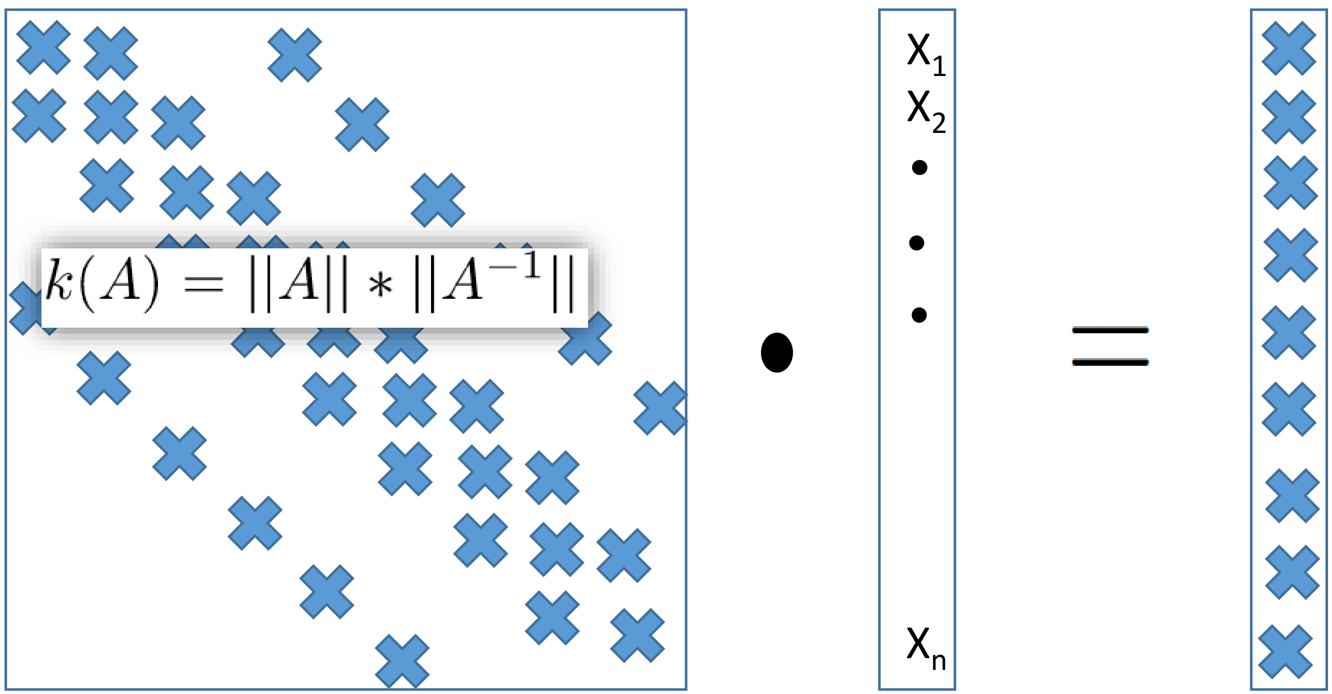
$$\left[\frac{(\Omega I)}{\Delta t} + \left(\frac{\delta\boldsymbol{R}}{\delta\boldsymbol{W}}\right)\right]\Delta\boldsymbol{W}^n = -\boldsymbol{R}^n$$

Aissa, M.H., Verstraete, T., Vuik, C. "Aerodynamic Optimization of Supersonic Compressor Cascade using Differential Evolution on GPU". 13th Int. Conf. of Numerical Analysis and Applied Mathematics (ICNAAM 2015)

Implicit Time Stepping is more Stable but …

$$\left[\frac{(\Omega I)}{\Delta t} + \left(\frac{\delta \boldsymbol{R}}{\delta \boldsymbol{W}}\right)\right] \Delta \boldsymbol{W}^n = -\boldsymbol{R}^n$$

$$A\,x = b$$



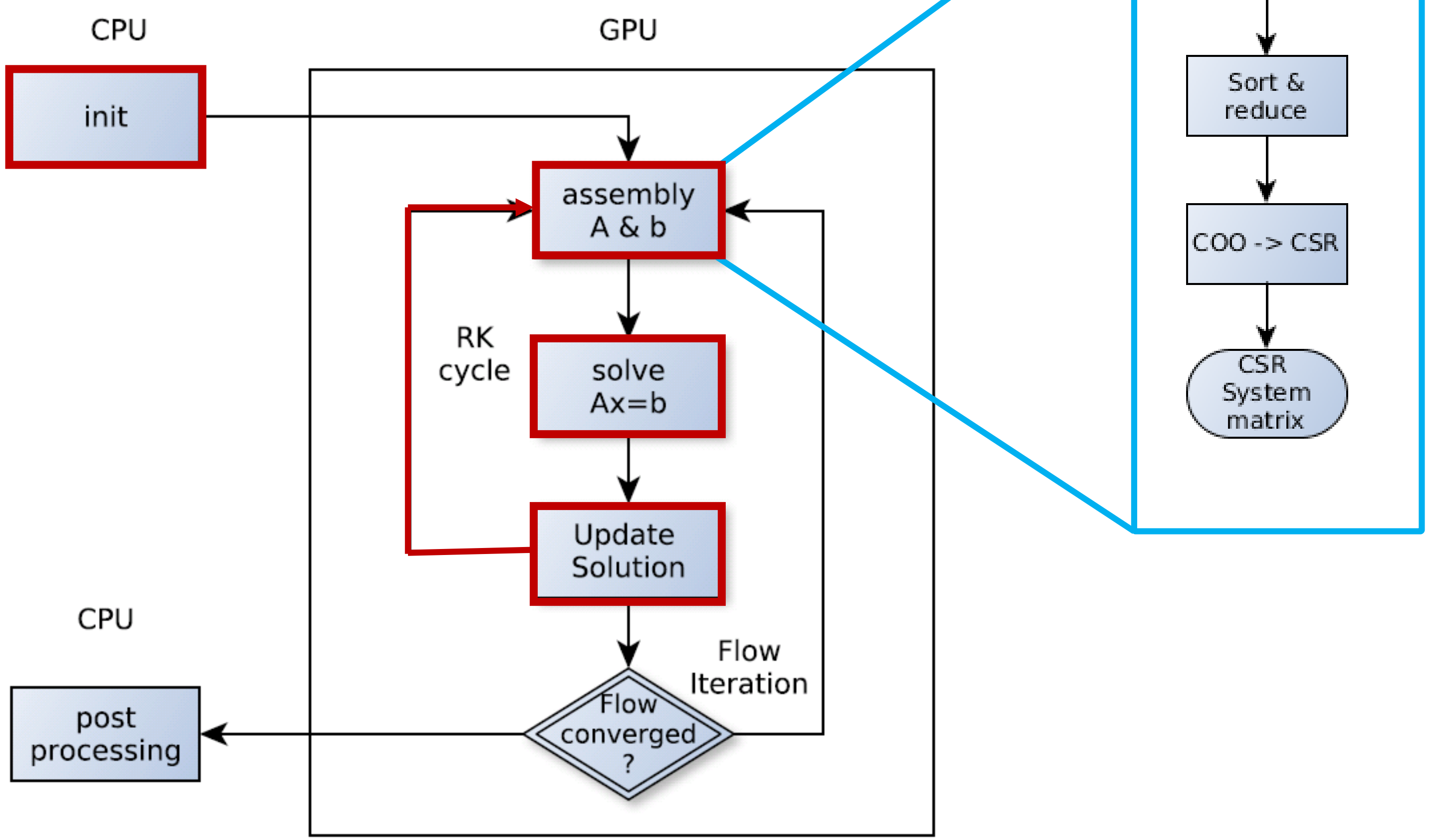
$$M^{-1}Ax = M^{-1}b$$

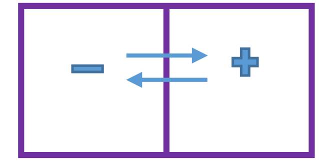
Literature Review

- What to Port
  - only linear solver when it is dominant
  - both assembly and solve is optimal (no communication)
- Linear solver
  - Library : code maturity but restrictive
    (petsc-dev, Paralution, AmgX, ViennaCL …)
  - Own code: flexibility
- Storage format
  - Standard (CSR,DIA …)
  - New (hybrid)

CFD Solver (Standard)

$$\left[\frac{(\Omega I)}{\Delta t} + \left(\frac{\delta \boldsymbol{R}}{\delta \boldsymbol{W}}\right)\right] \Delta \boldsymbol{W}^n = -\boldsymbol{R}^n$$





Thrust

CUSP

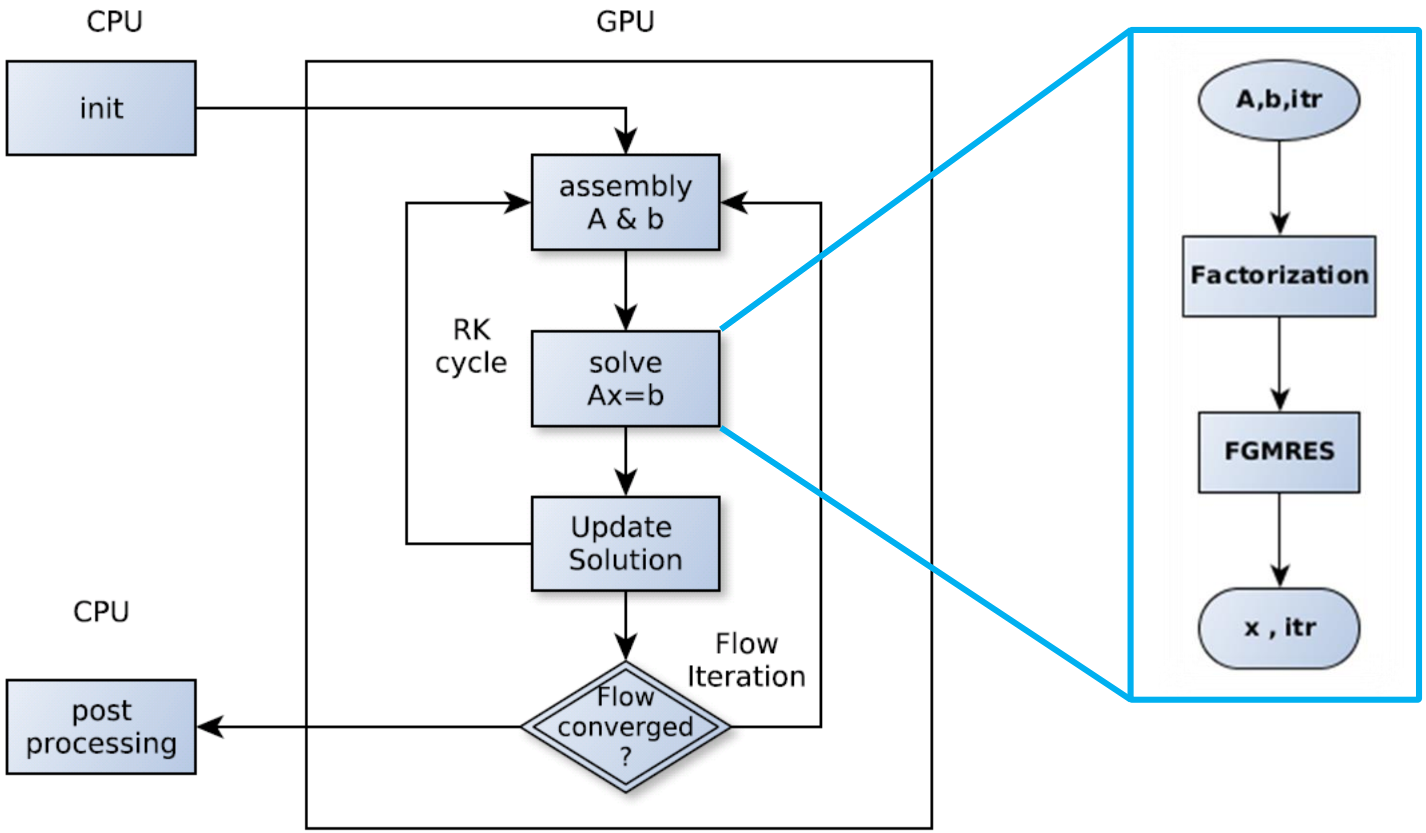
PARALUTION

$$\begin{aligned}
\boldsymbol{W}^{(0)} &= \boldsymbol{W}^n \\
A^{(0)}[\boldsymbol{W}^{(1)} - \boldsymbol{W}^{(0)}] &= -\alpha_1 \boldsymbol{R}(\boldsymbol{W}^{(0)}) \\
A^{(0)}[\boldsymbol{W}^{(2)} - \boldsymbol{W}^{(0)}] &= -\alpha_2 \boldsymbol{R}(\boldsymbol{W}^{(1)}) \\
&\vdots \\
A^{(0)}[\boldsymbol{W}^{(m)} - \boldsymbol{W}^{(0)}] &= -\alpha_m \boldsymbol{R}(\boldsymbol{W}^{(m-1)}) \\
\boldsymbol{W}^{n+1} &= \boldsymbol{W}^{(0)} + [\boldsymbol{W}^{(m)} - \boldsymbol{W}^{(0)}]
\end{aligned}$$

Implicit Runge-Kutta scheme

Xu et Al. JCP 2015

CFD Solver (Standard)

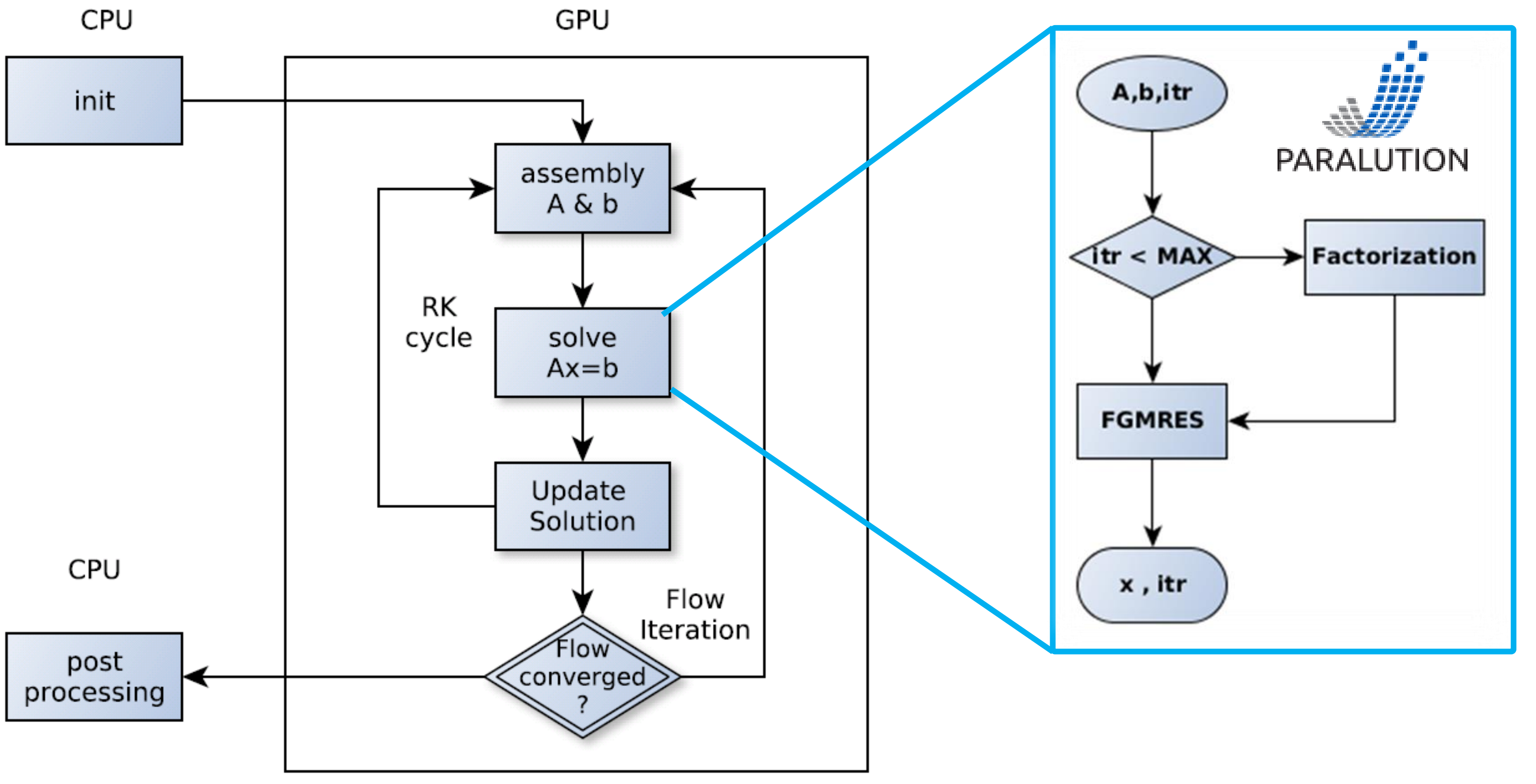


PARALUTION

Implicit Runge-Kutta scheme

$$
\begin{aligned}
\boldsymbol{W}^{(0)} &= \boldsymbol{W}^{n} \\
A^{(0)}[\boldsymbol{W}^{(1)} - \boldsymbol{W}^{(0)}] &= -\alpha_1 \boldsymbol{R}(\boldsymbol{W}^{(0)}) \\
A^{(0)}[\boldsymbol{W}^{(2)} - \boldsymbol{W}^{(0)}] &= -\alpha_2 \boldsymbol{R}(\boldsymbol{W}^{(1)}) \\
&\vdots \\
A^{(0)}[\boldsymbol{W}^{(m)} - \boldsymbol{W}^{(0)}] &= -\alpha_m \boldsymbol{R}(\boldsymbol{W}^{(m-1)}) \\
\boldsymbol{W}^{n+1} &= \boldsymbol{W}^{(0)} + [\boldsymbol{W}^{(m)} - \boldsymbol{W}^{(0)}]
\end{aligned}
$$

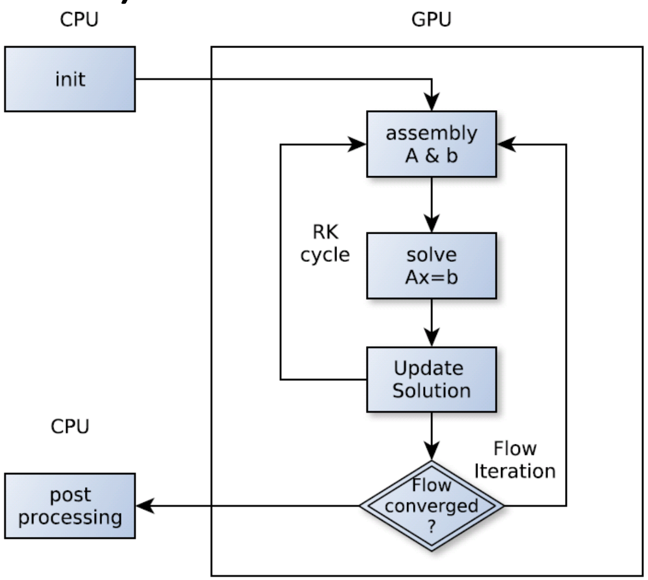
CFD Solver (*On-demand* Factorization)



Implicit Runge-Kutta scheme

$$
\begin{aligned}
\boldsymbol{W}^{(0)} &= \boldsymbol{W}^{n} \\
A^{(0)}[\boldsymbol{W}^{(1)} - \boldsymbol{W}^{(0)}] &= -\alpha_1 \boldsymbol{R}(\boldsymbol{W}^{(0)}) \\
A^{(0)}[\boldsymbol{W}^{(2)} - \boldsymbol{W}^{(0)}] &= -\alpha_2 \boldsymbol{R}(\boldsymbol{W}^{(1)}) \\
&\vdots \\
A^{(0)}[\boldsymbol{W}^{(m)} - \boldsymbol{W}^{(0)}] &= -\alpha_m \boldsymbol{R}(\boldsymbol{W}^{(m-1)}) \\
\boldsymbol{W}^{n+1} &= \boldsymbol{W}^{(0)} + [\boldsymbol{W}^{(m)} - \boldsymbol{W}^{(0)}]
\end{aligned}
$$

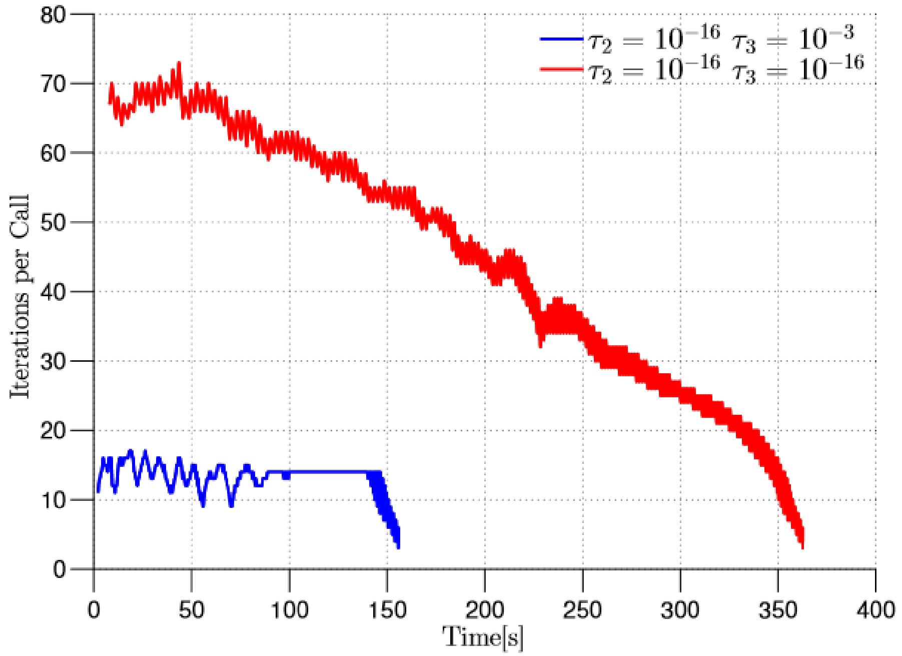
CFD Solver (*On-demand* Factorization)



$$r = Ax - b$$

Stop condition relative, absolute or a combination

$$\frac{||r_k||}{||b||} < \tau_3 \qquad ||r_k|| < \tau_2$$

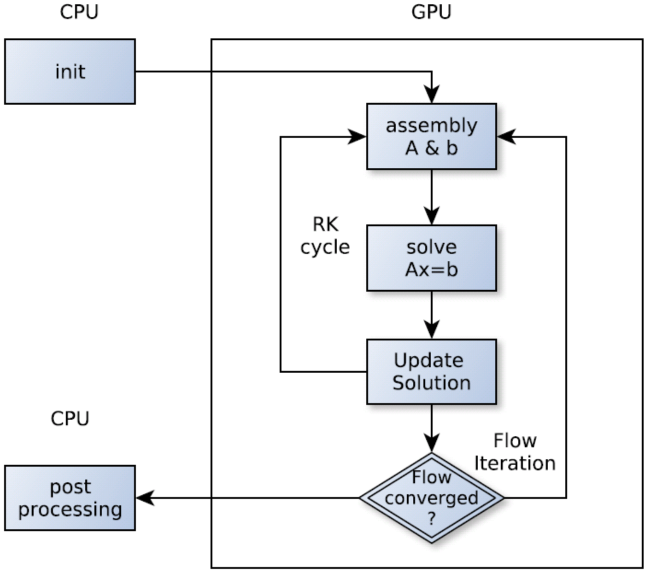


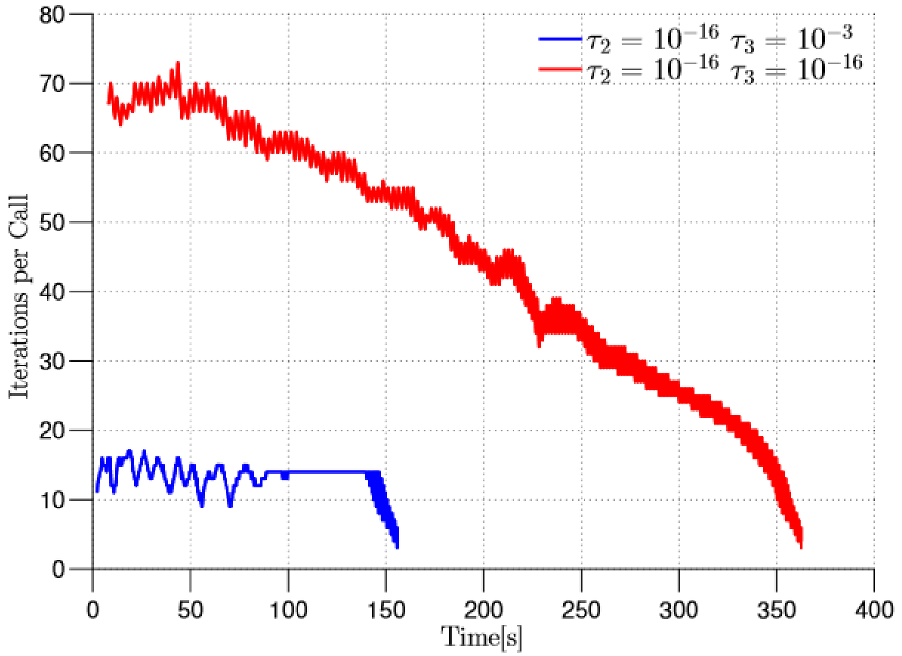
CFD Solver (*On-demand* Factorization)

$$r = Ax - b$$

Stop condition relative, absolute or a combination

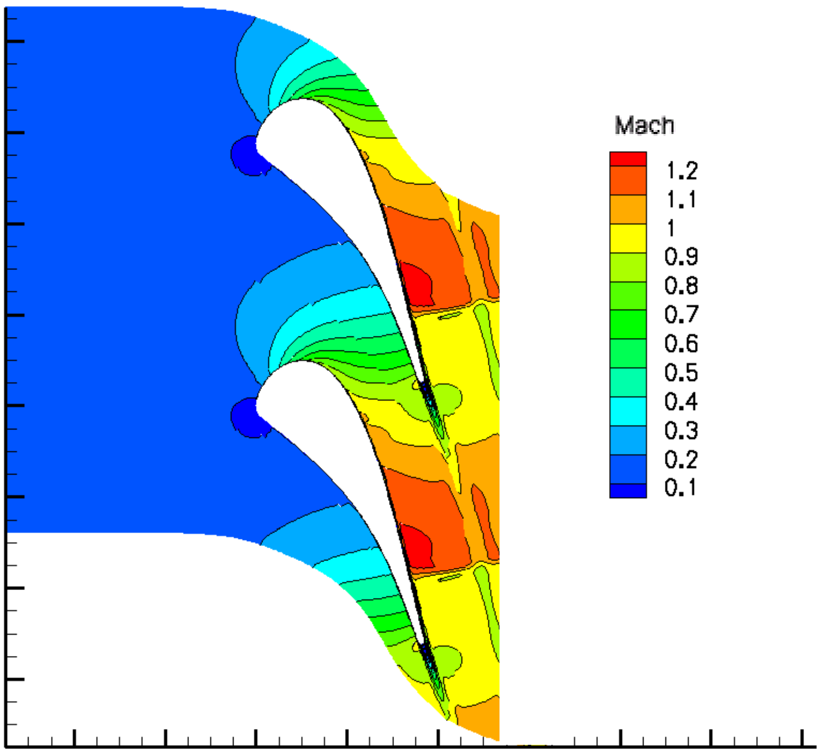
$$\frac{||r_k||}{||b||} < \tau_3 \qquad ||r_k|| < \tau_2$$

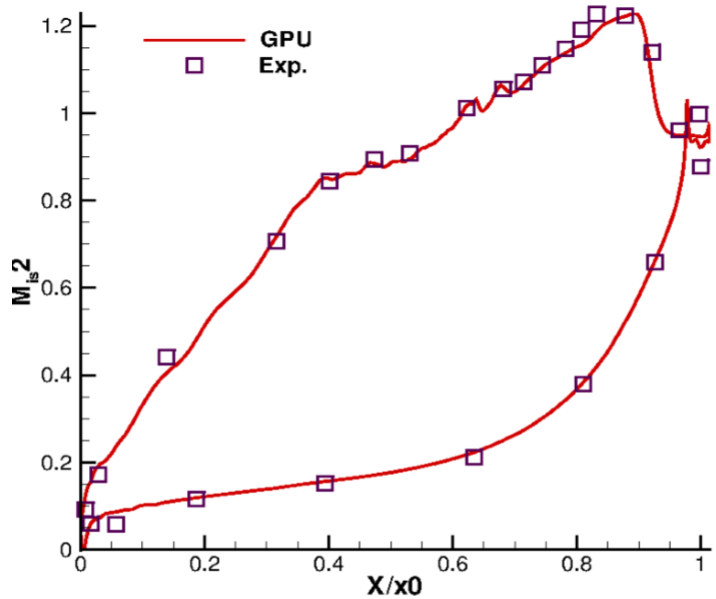




$$MAX\_ITR = \alpha * \frac{1}{m}\sum_{1}^{m} N_{ITR_i}$$

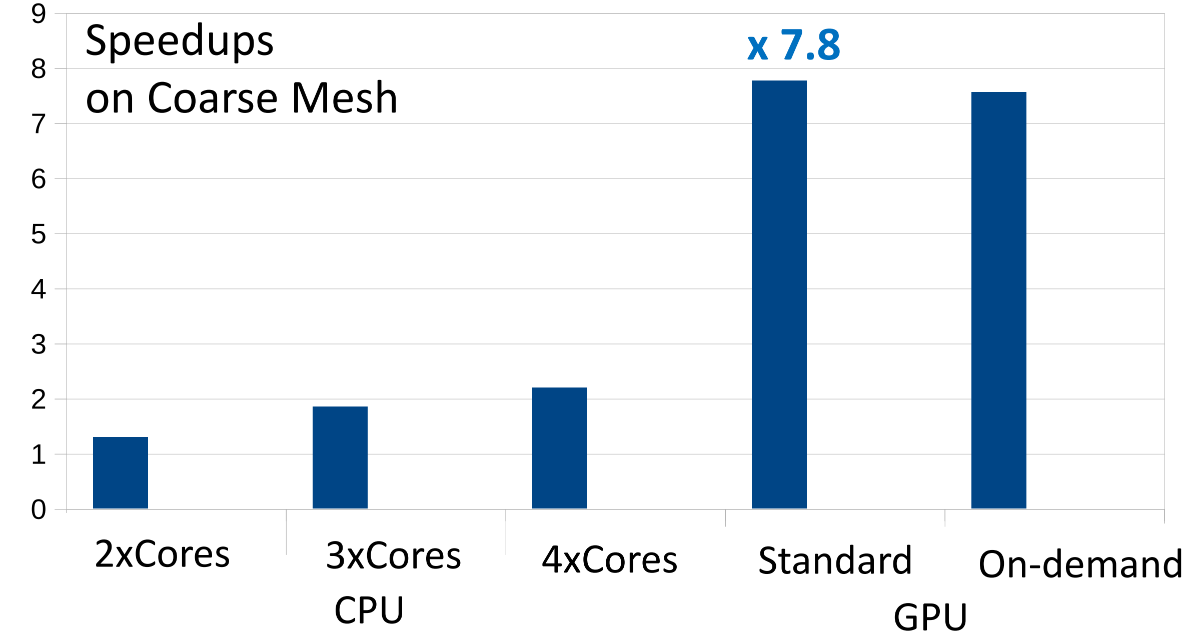
Benchmark: Flow around LS89
2-Stages Runge-Kutta

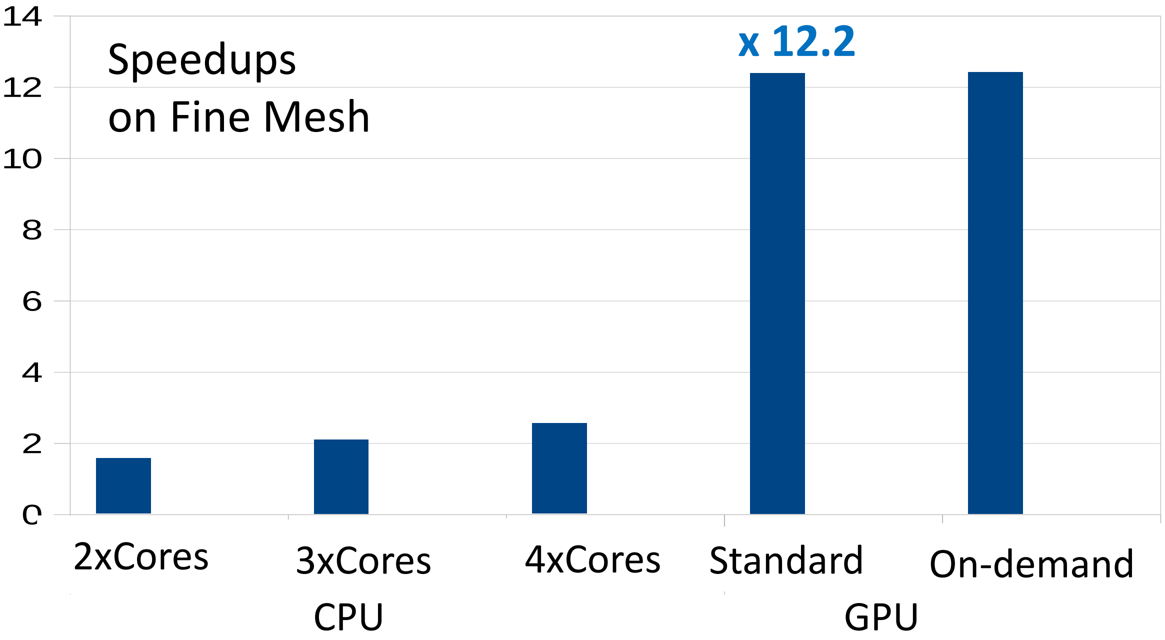


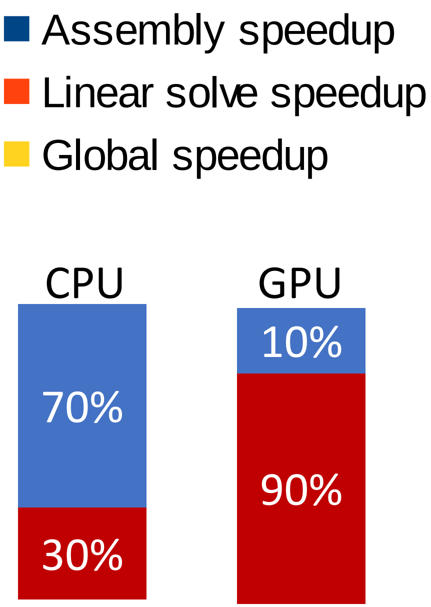


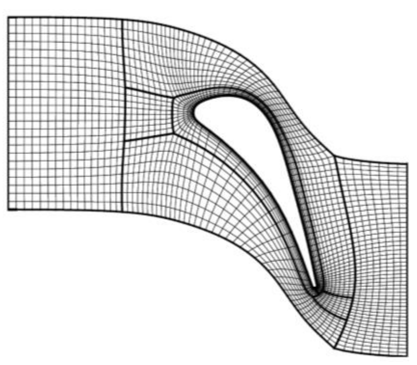
Mesh	N_{Cells}	N_{Rows}	nnz	nnz/row
Coarse	40k	200k	5.7M	[20 .. 30]
Fine	300k	1500k	52.6M	[20 .. 35]

Assembly Acceleration

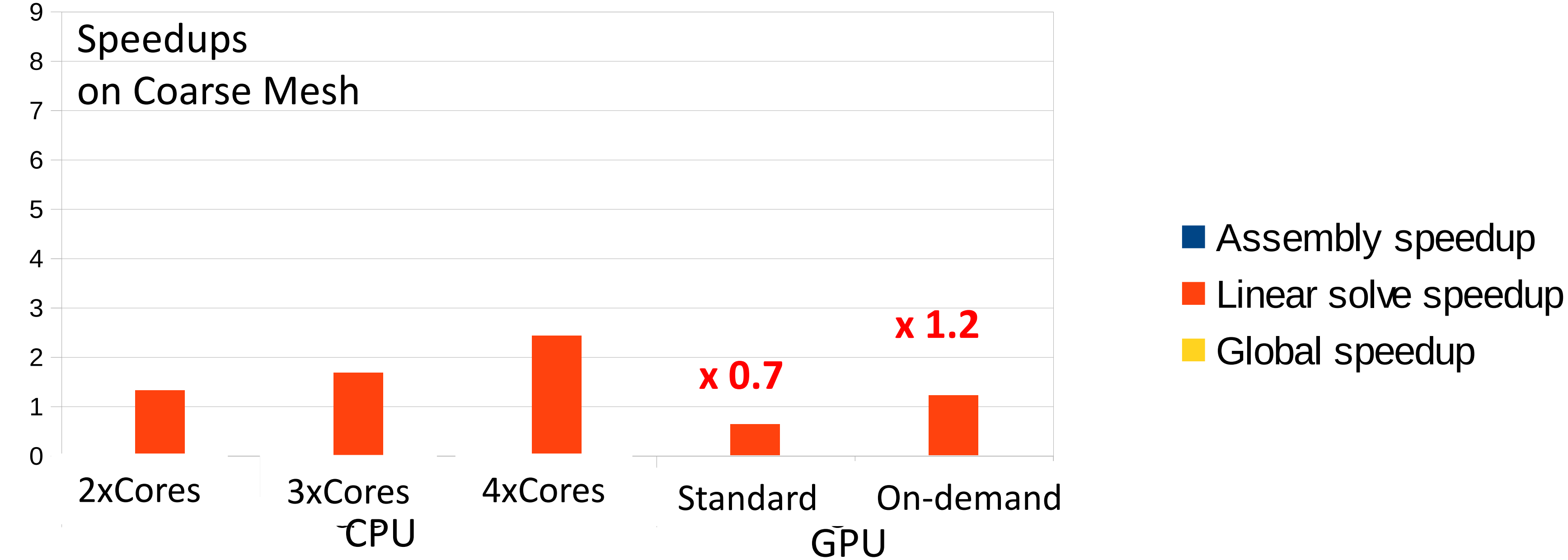


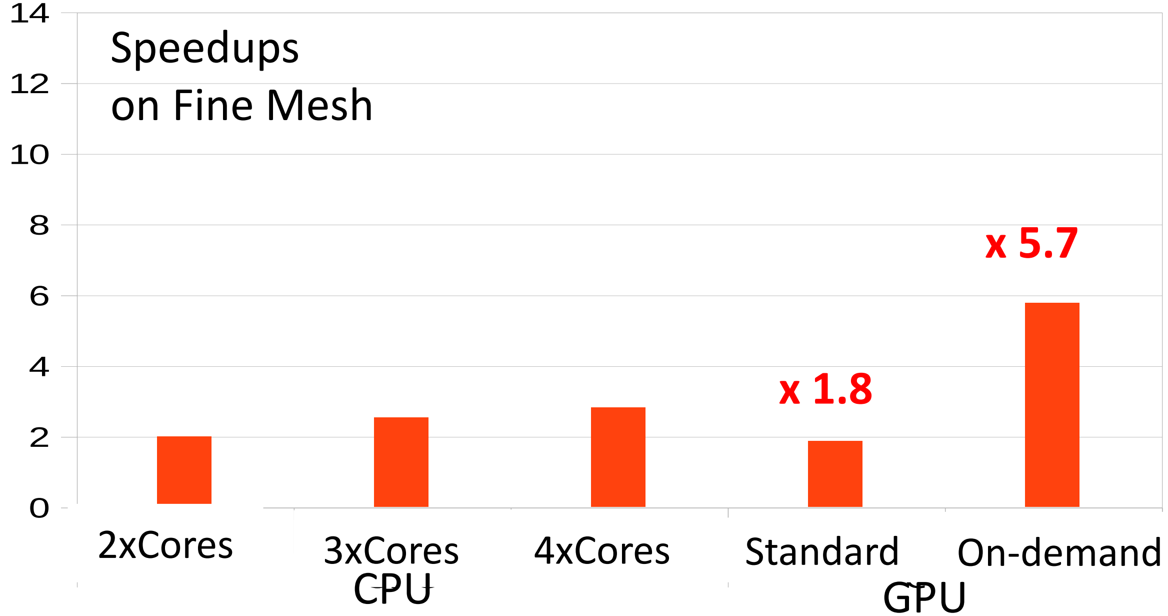


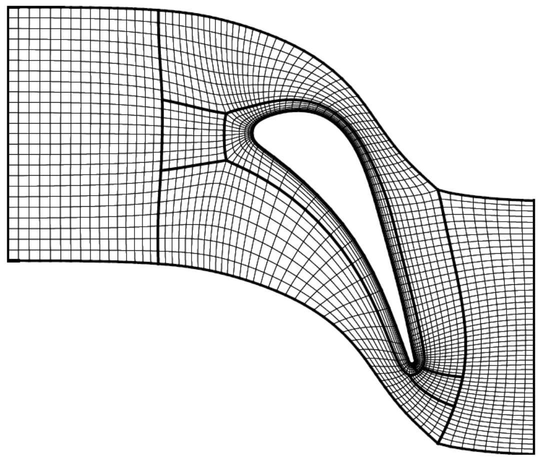




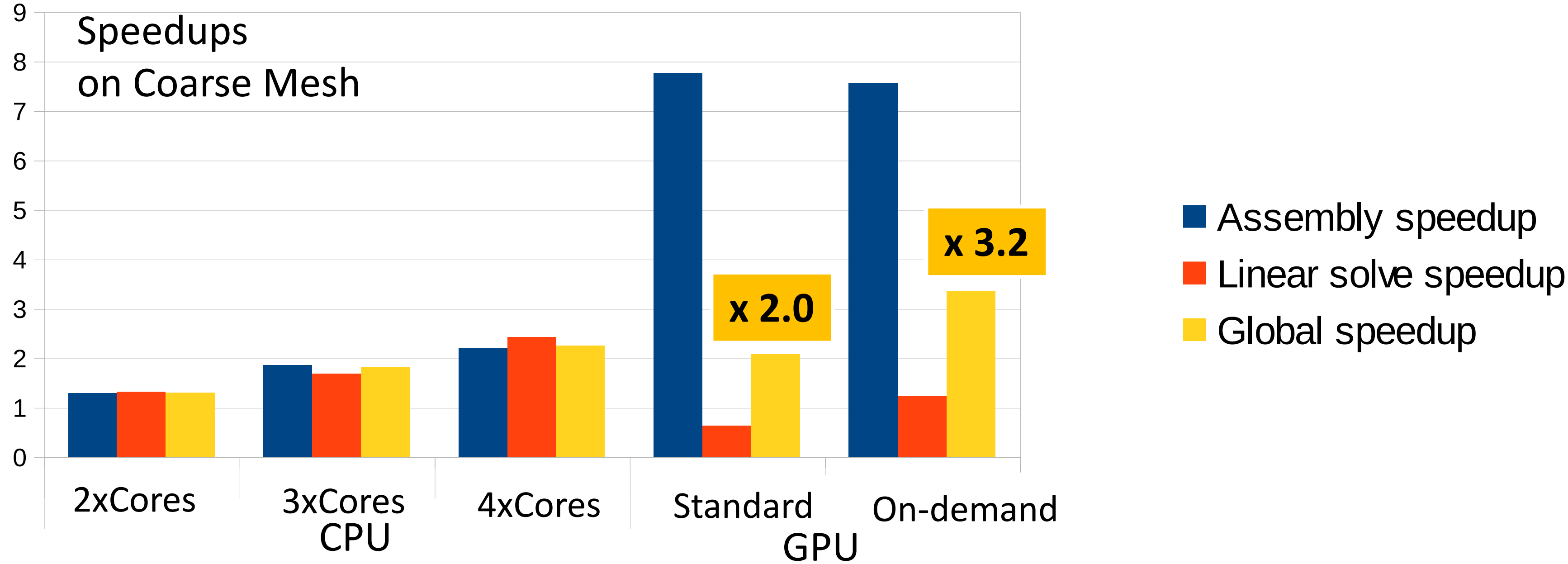
Linear Solver Acceleration

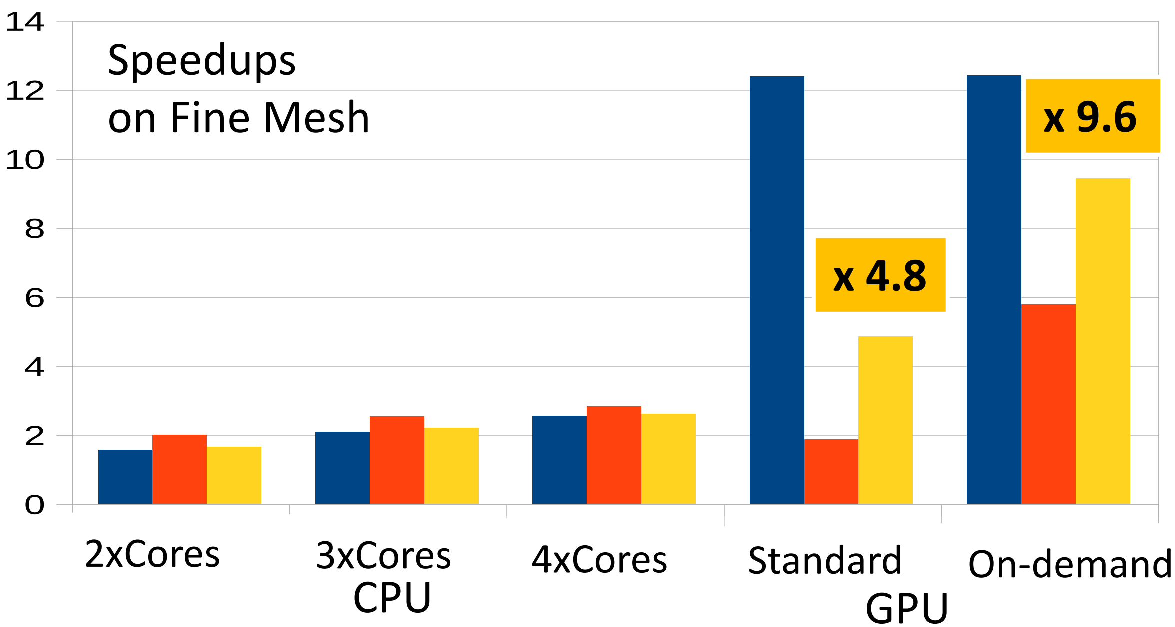






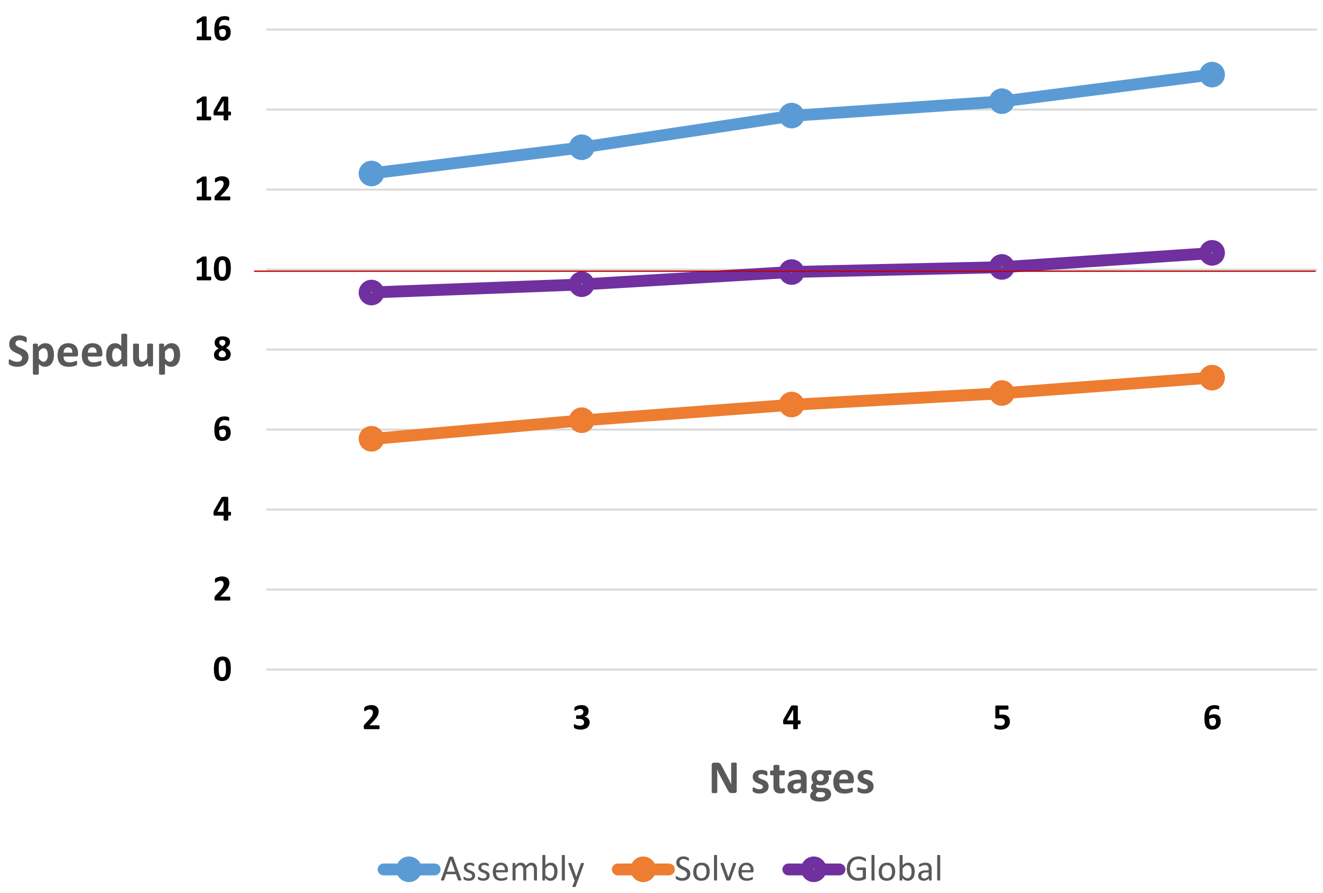
Global Acceleration





Suggestion for better Performance assessment are very welcome!

Increase of the Speedup for higher Numbers of Runge-Kutta Stages on Fine Mesh



Content

- Multidisciplinary Optimization
- CFD simulations on GPU
  - Literature review
  - Implicit RANS Implementation
  - Benchmark
- Optimization Case

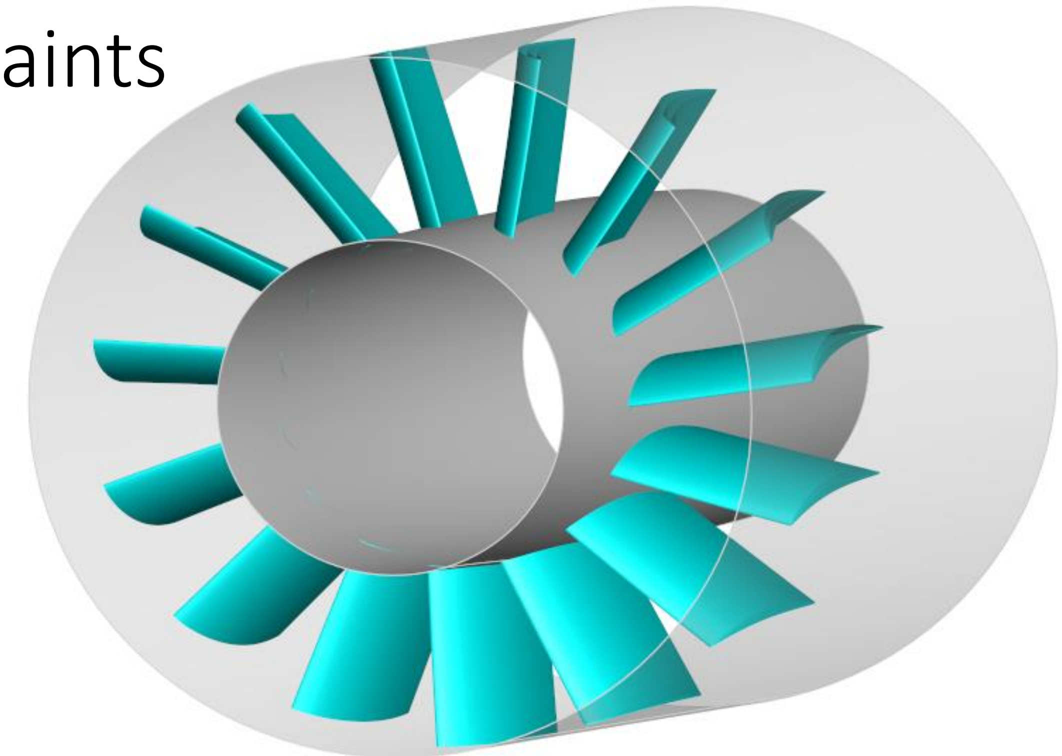
Test Case 3: TU Berlin TurboLab Stator Optimization requirements

Objectives:

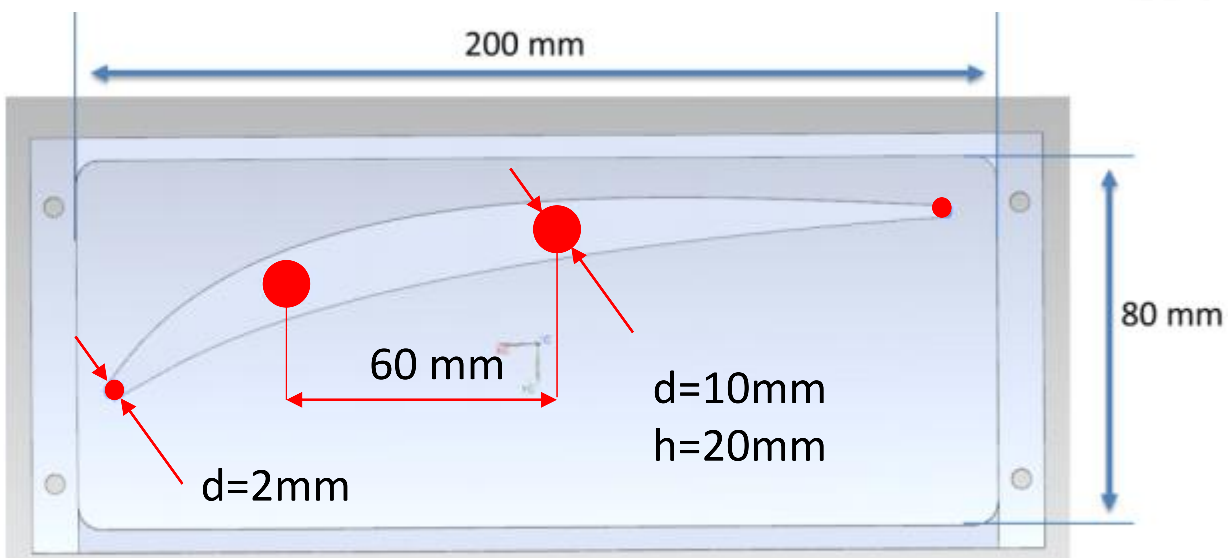
- Decrease outfow axial deviation
- Decrease total pressure loss

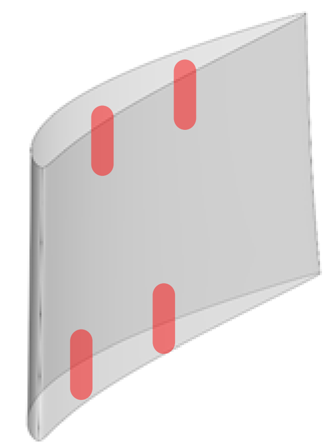
Considering 3 operating points

TurboLab
Manufacturing Constraints

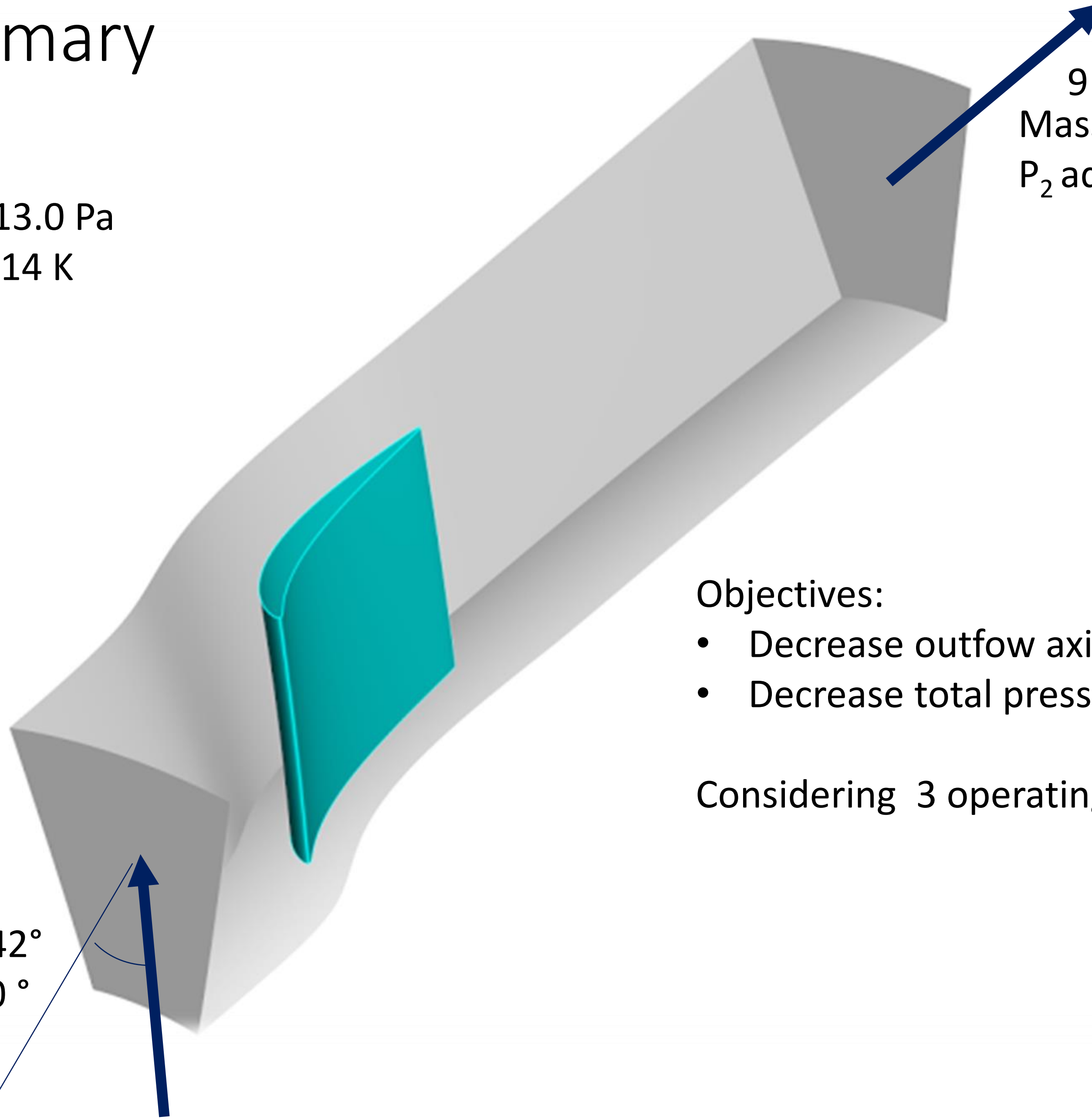


- $N_{blades} = 15$
- Chord length fixed
- Casing fixture





TurboLab: Boundary conditions and summary



9 kg/s +/- 0.1
Massflow imposed
P_2 adapted

Inlet P_0: 102713.0 Pa
Inlet T_0: 294.314 K

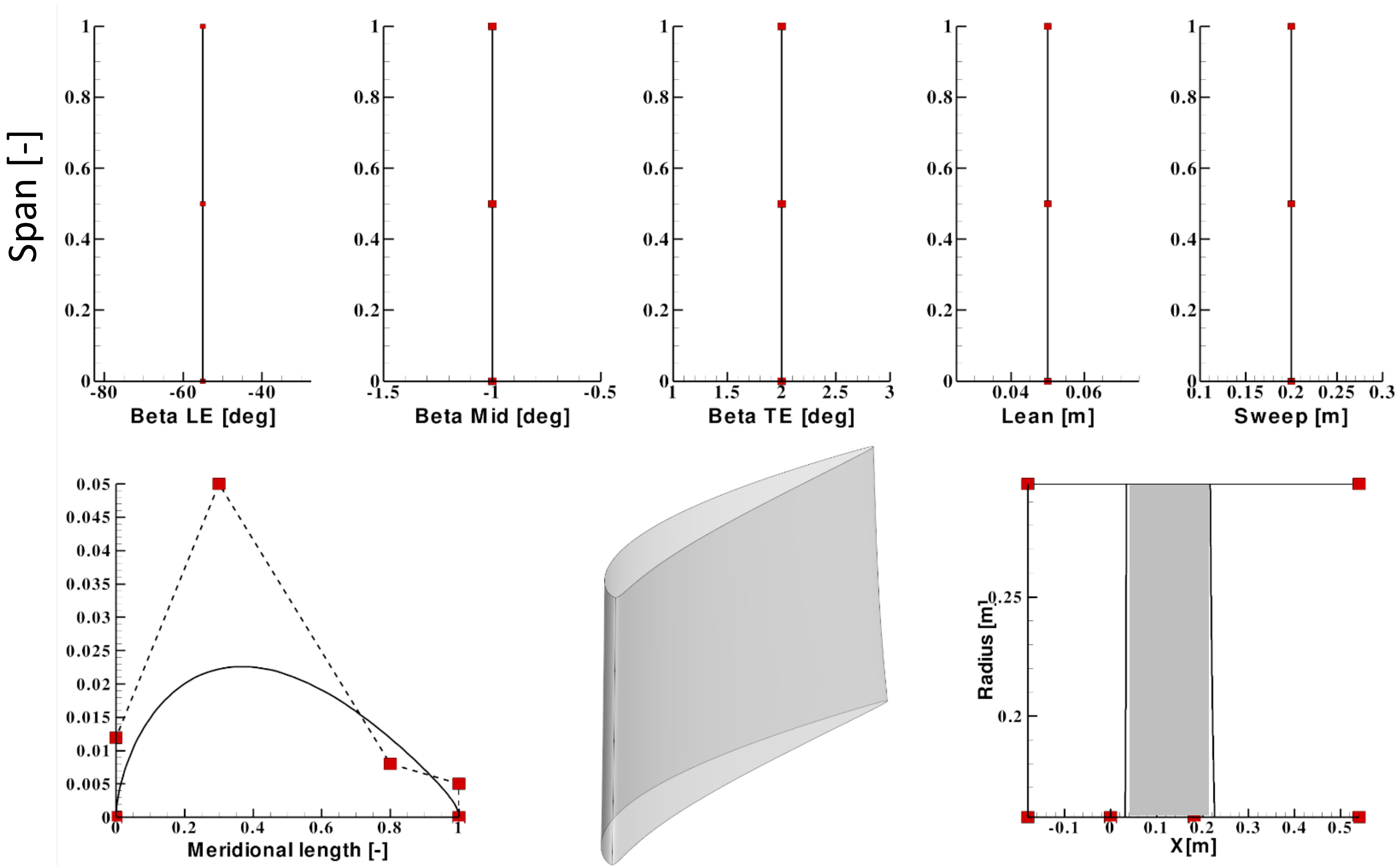
Objectives:

- Decrease outfow axial deviation
- Decrease total pressure loss

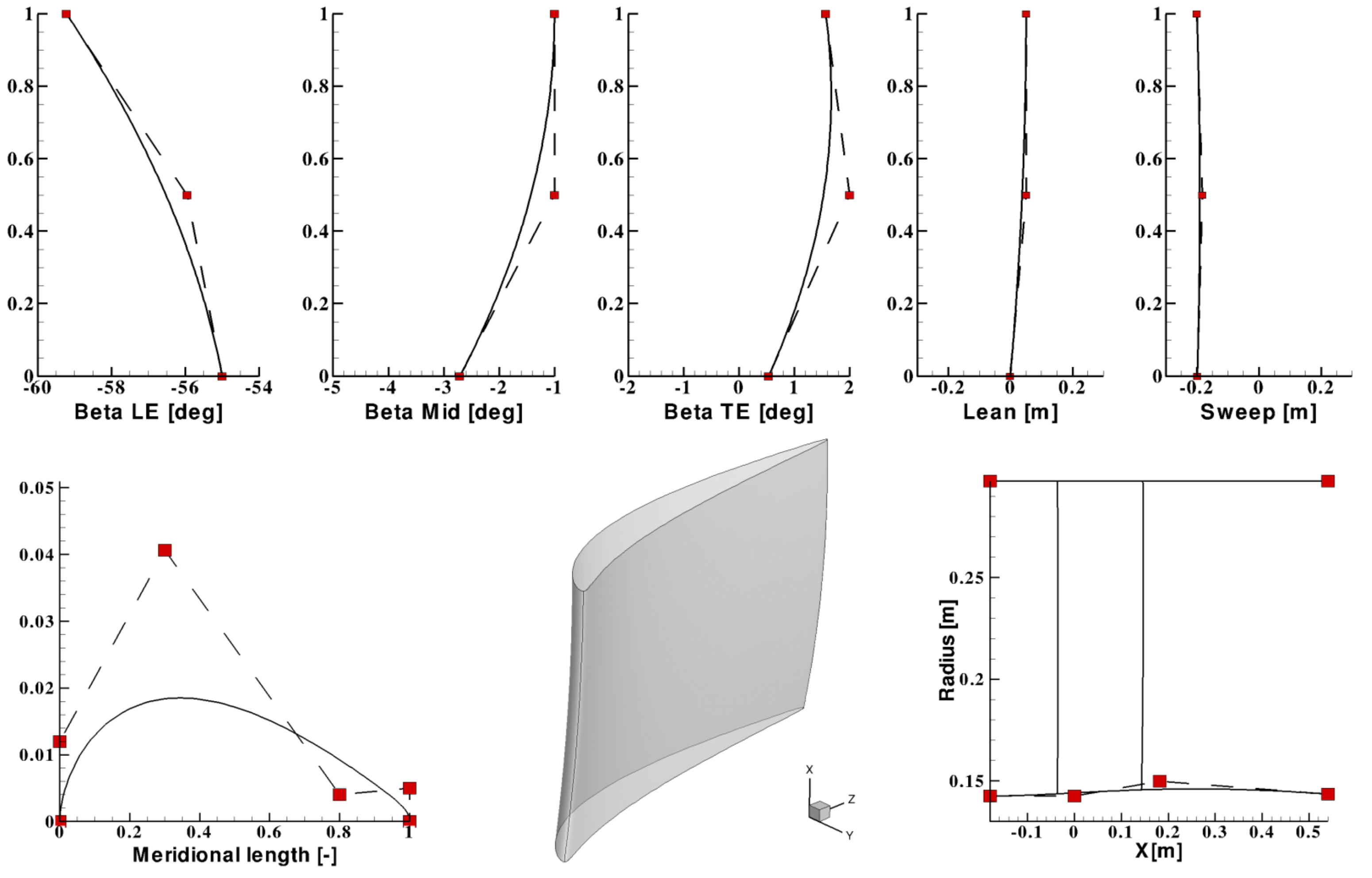
Considering 3 operating points

Inlet whirl angle: 42°
Inlet pitch angle: 0 °

Parametrization
21 Design variables

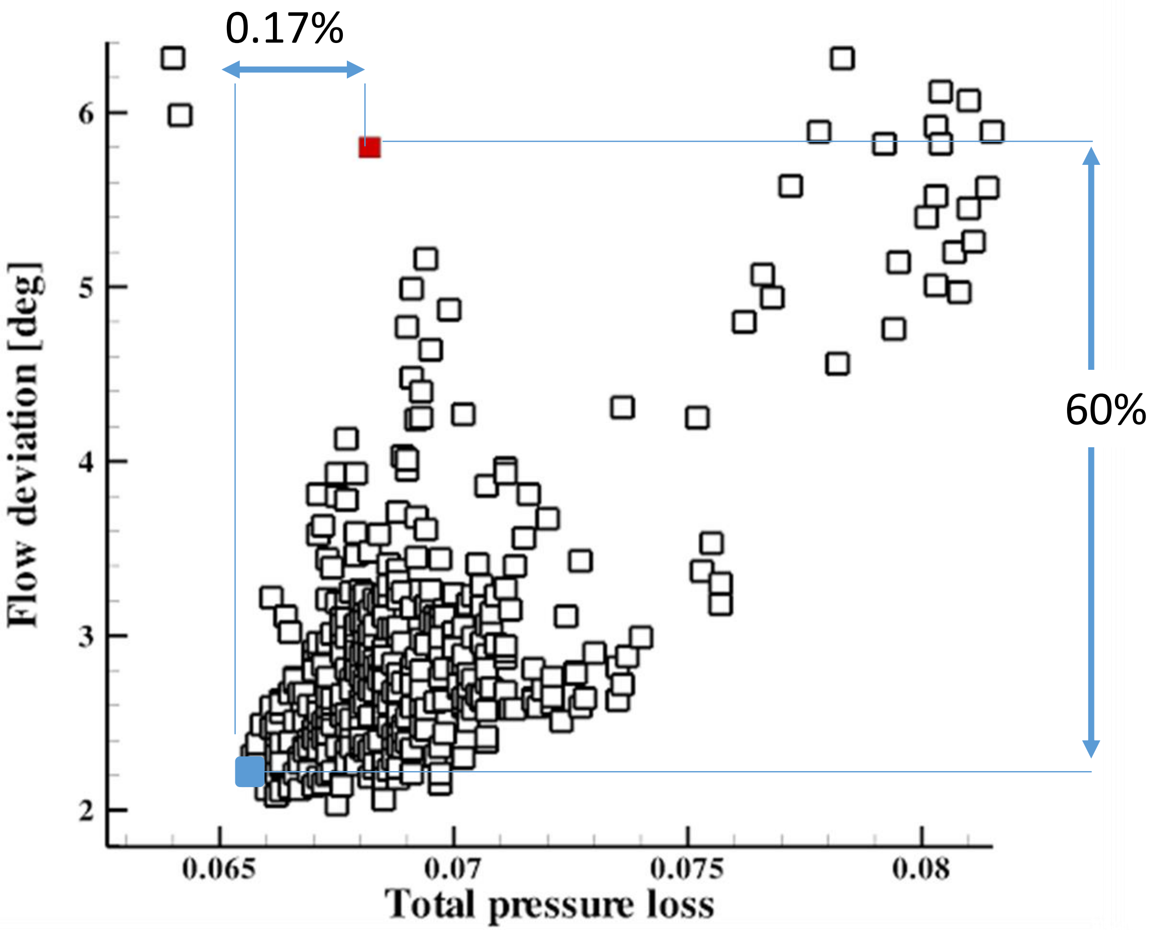


Turbolab Parameterization



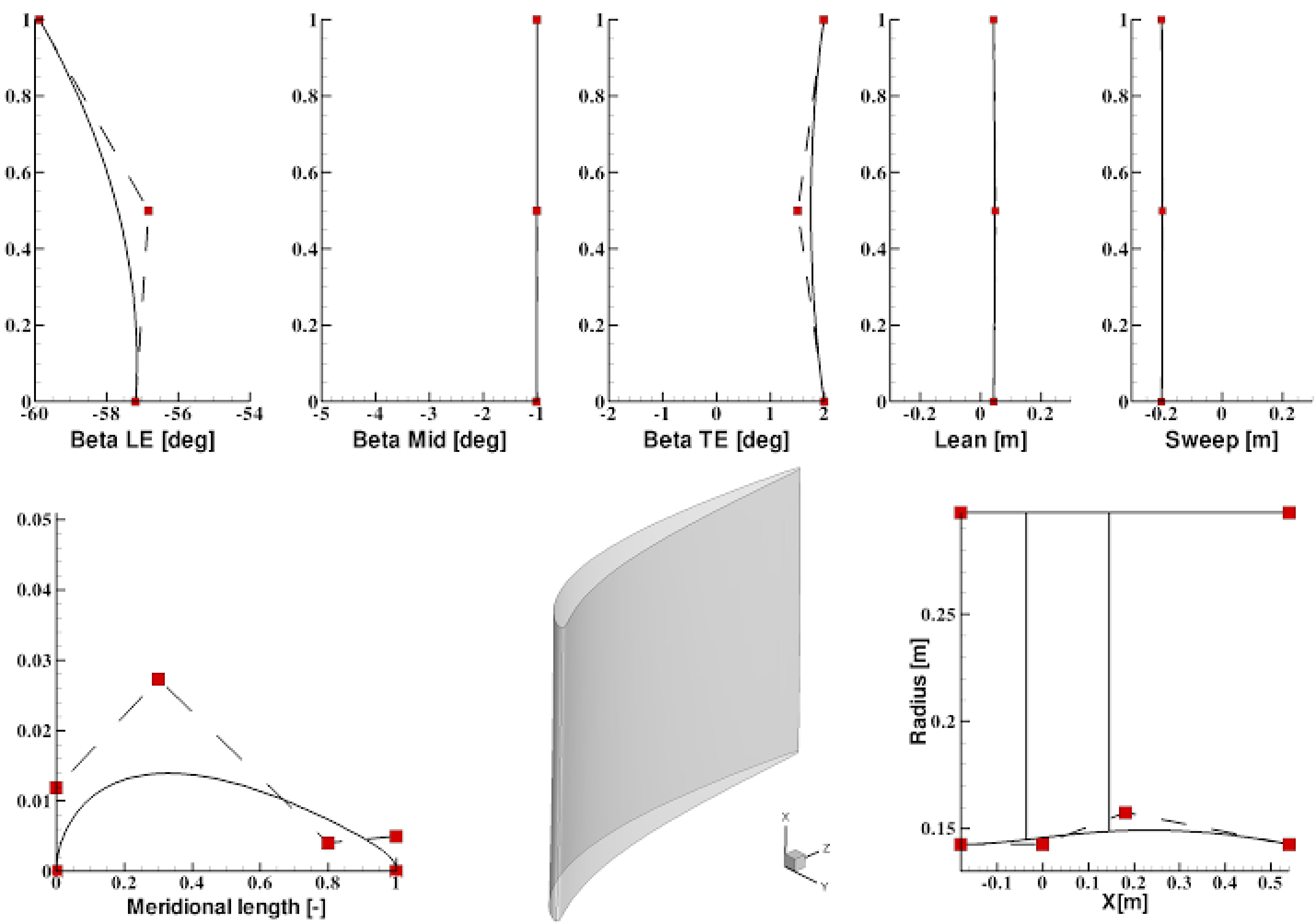
Optimization Results

$$\int_{hub}^{casing} \alpha_{whirl}^2$$

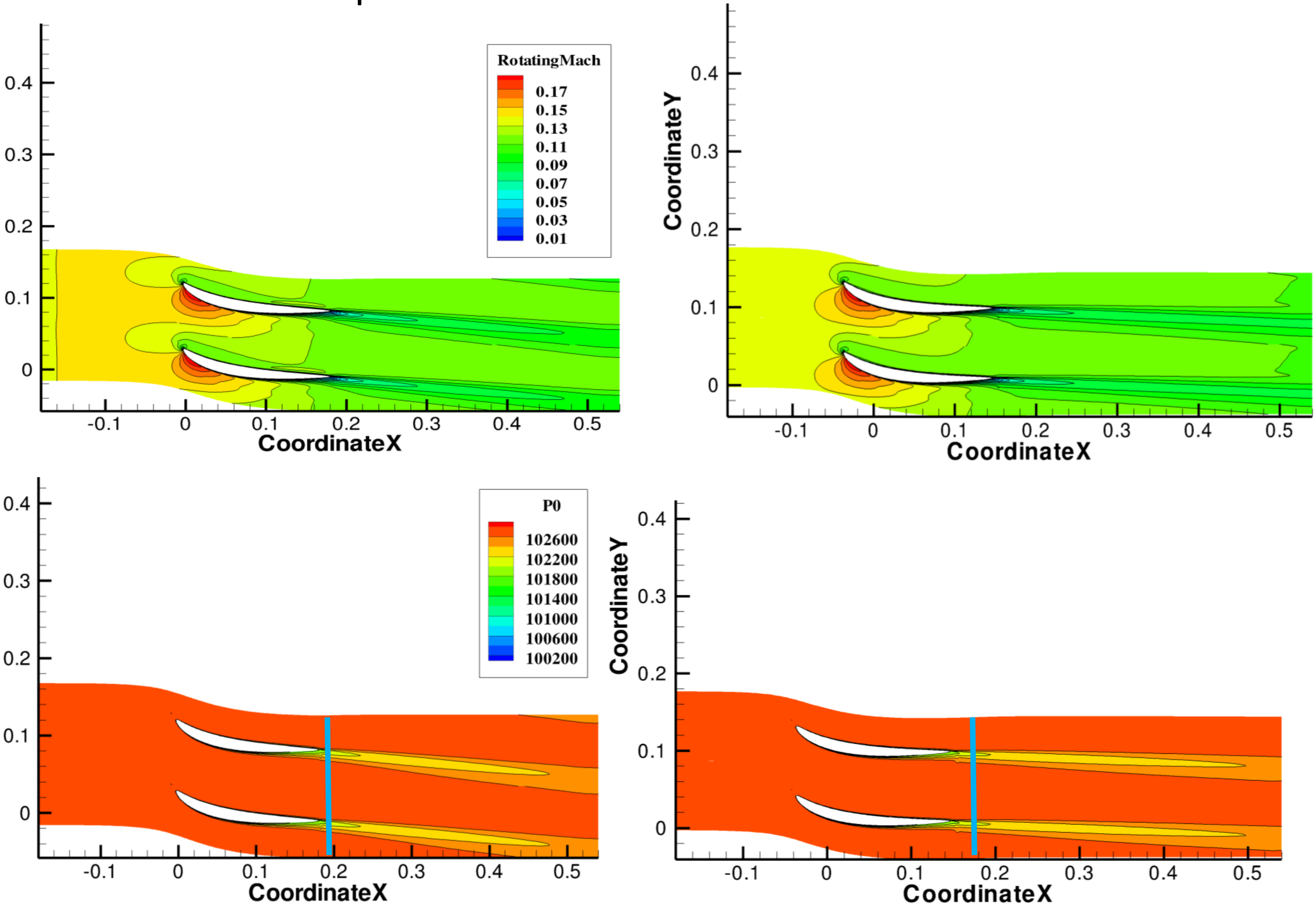


$$Loss_{P_0} = \frac{p_{01} - p_{02}}{p_{01} - p_1}$$

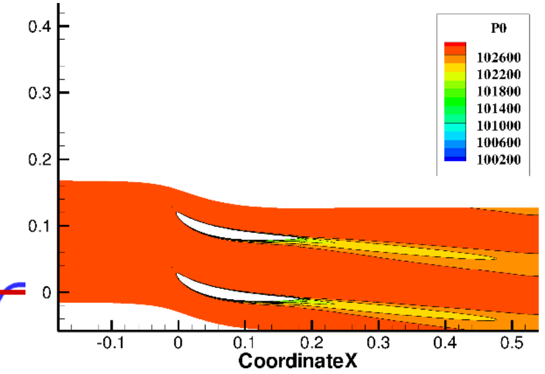
Optimized Blade

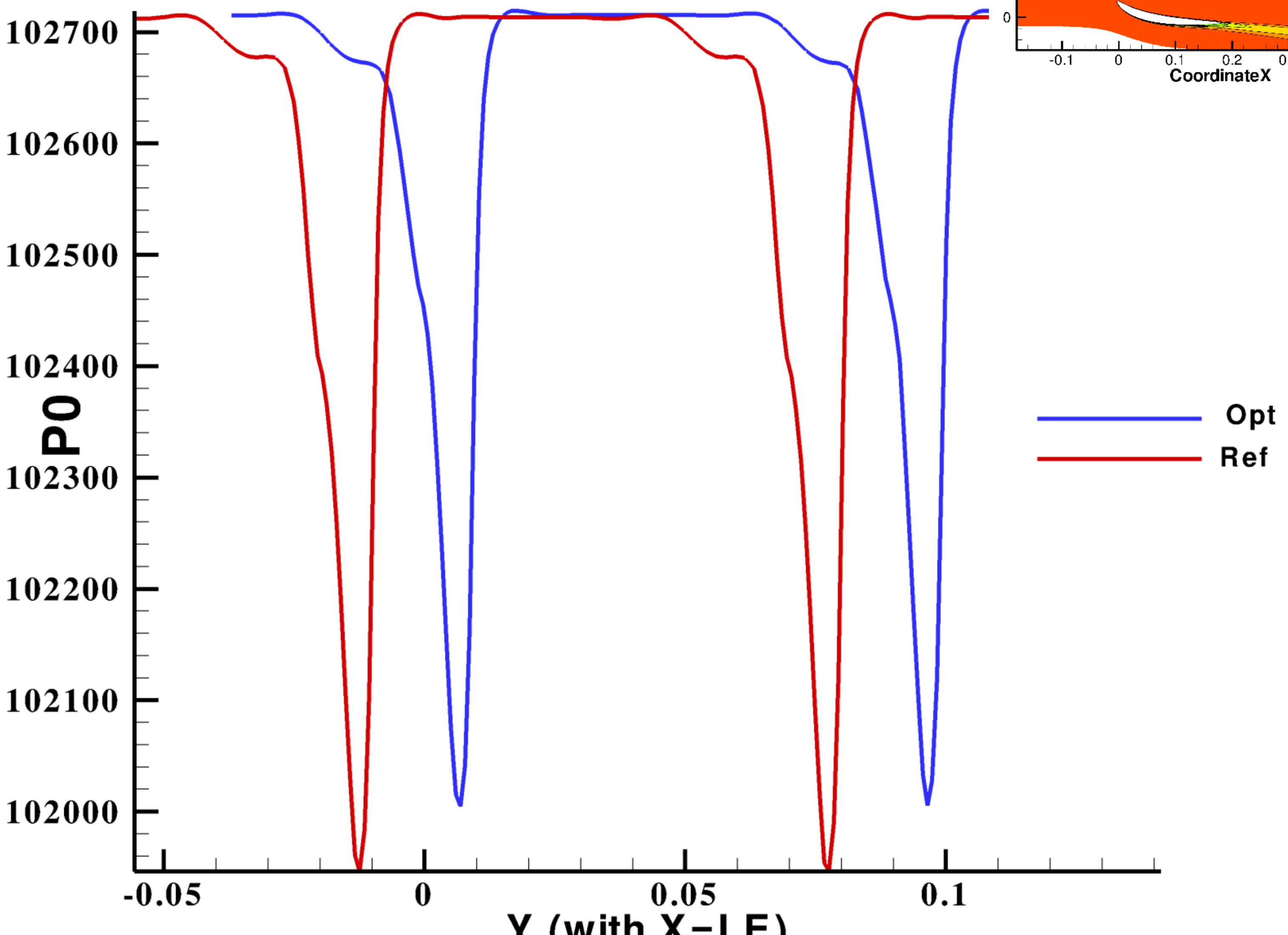


Baseline Vs Optimized

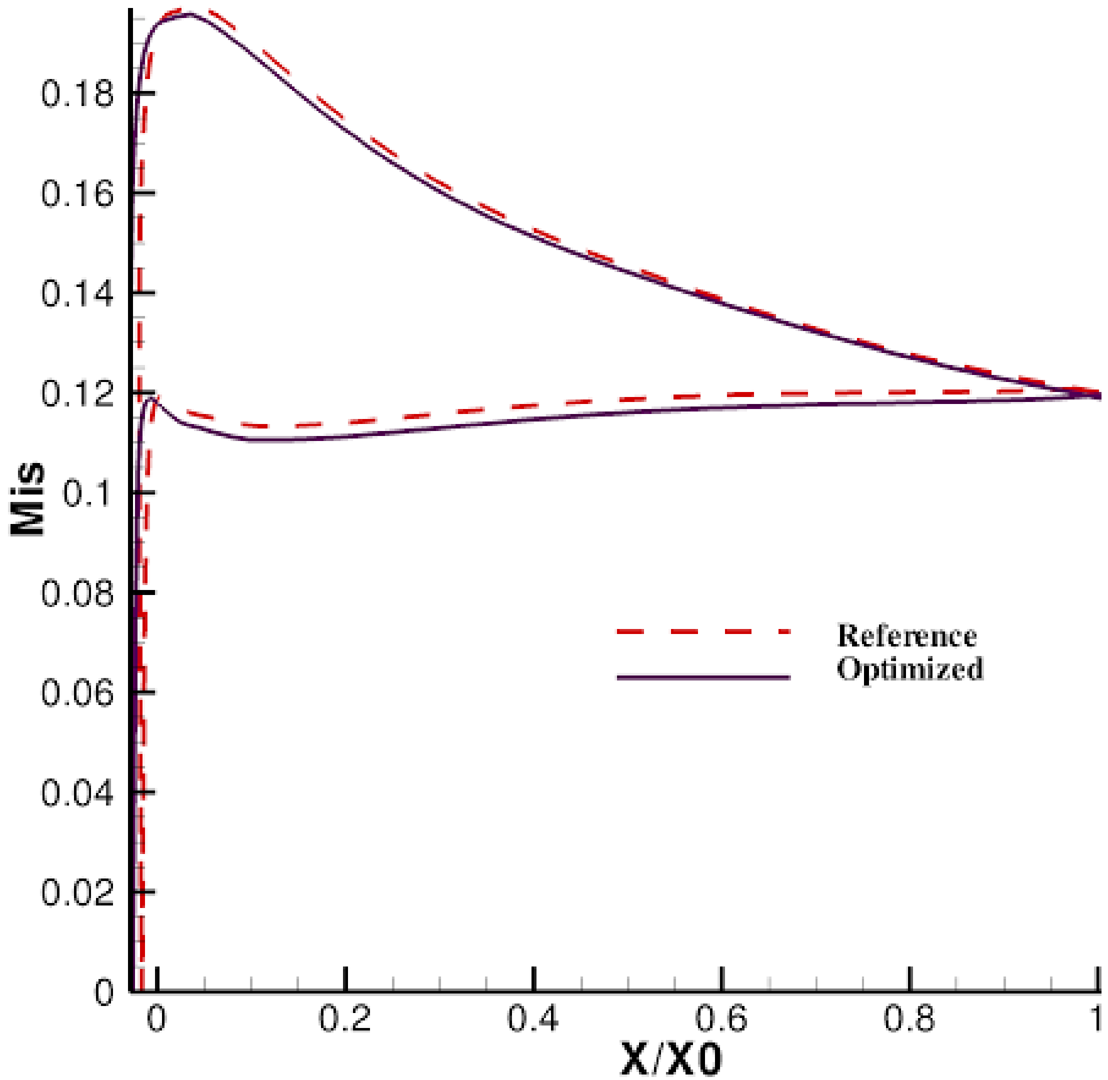


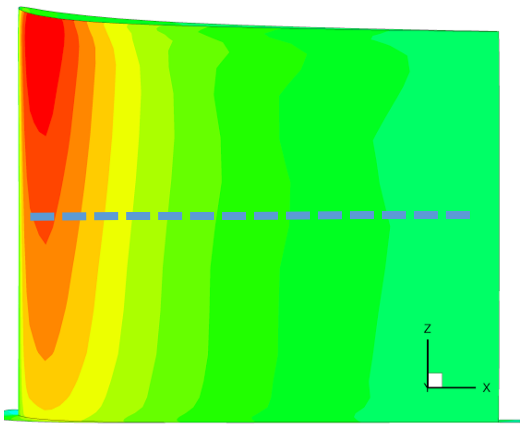
Baseline Vs Optimized





Isentropic Mach Number at mid-span



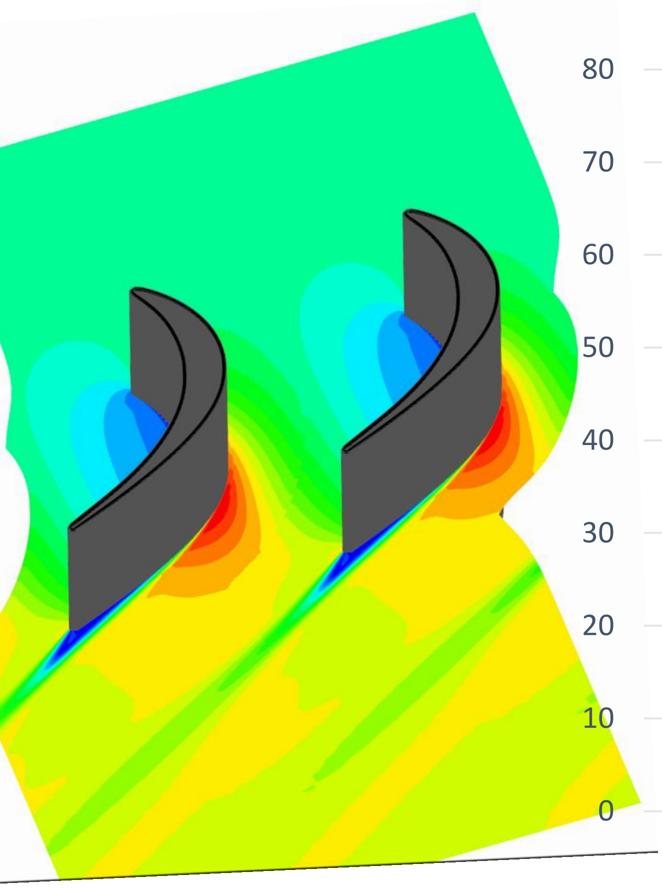


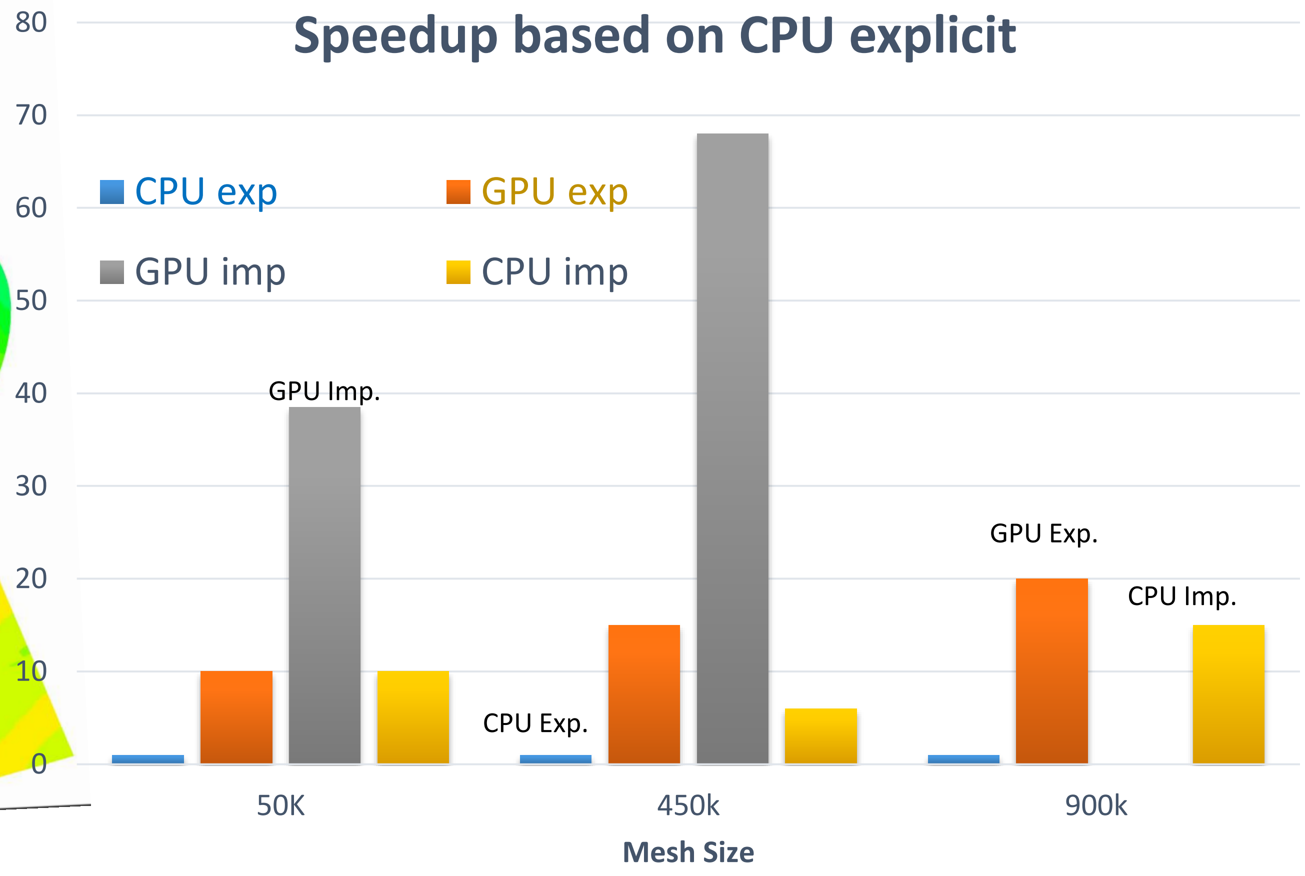
Conclusion

- Optimization
- GPU Solver with implicit time stepping
- *On-demand* (incomplete) Factorization
- 10x speedup
- Aerodynamic shape optimization

Future Work

Benchmark Case: Transonic Turbine Stator T106c





VON KARMAN INSTITUTE FOR FLUID DYNAMICS

TU Delft Delft University of Technology

Thanks for your attention

Mohamed Hassanine Aissa
Turbomachinery & Propulsion Department
72, chaussee de Waterloo
B1640 - Rhode Saint Genese - **Belgium**
Email: **aissa@vki.ac.be**

acknowledgements:

PARALUTION
Support

NVIDIA
Hardware

MARIE CURIE ACTIONS