Ref. Ares(2024)1562791 - 29/02/2024

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101056732

multi-disciplinary digital-enablers for NEXT-generation AIRcraft design and operations

D1.3 - DENs to Speed-up MDOs

Document authors	Kyriakos Giannakoglou (NTUA) Marina Kontou (NTUA) Varvara Asouti (FOSS) Xenofon Trompoukis (FOSS)
Document contributors	Francois Gallard (IRT) Christophe Blondeau (ONERA) Cédric Liauzun (ONERA) Pierre-Emmanuel Des Boscs (ONERA) Quentin Bennehard (ONERA) Alessandro Alaia (OPT) Frederic Alauzet (INRIA) Tiziano Ghisu (UNICA) Alistair John (USFD) Shahrokh Shahpar (RR)

Abstract

D1.3 is related with the activities performed within T1.3. This deliverable first reports the benefits from linking NEXTAIR with the EuroHPC programs REGALE and NextSim (section 1). The benefits from using multi-level approaches in MDO are addressed in section 2, while those from using mesh adaptation in laminar wing cases and adjoint methods to compute Pareto fronts are the subjects of sections 3 and 4, respectively. Finally, section 5 reports techniques for cost reduction when the (continuous or discrete) unsteady adjoint is used in an optimization loop.

Keywords

Mesh adaptation, Multi-objective optimization, Gradient-based optimization, Multi-level parameterisation, MDO, Unsteady adjoint, Compression, Uncertainty Quantification

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Information Table

PROJECT INFORMATION	
PROJECT ID	101056732
PROJECT FULL TITLE	NEXTAIR - multi-disciplinary digital - enablers for NEXT-generation AIRcraft design and operations
PROJECT ACRONYM	NEXTAIR
START DATE OF THE PROJECT	01/09/2022
DURATION	36 months
CALL IDENTIFIER	HORIZON-CL5-2021-D5-01
PROJECT WEBSITE	https://www.nextair-project.eu/

DELIVERABLE INFORMATION	
DELIVERABLE No AND TITLE	D1.3 - DENs to Speed-up MDOs
TYPE OF DELIVERABLE	R
DISSEMINATION LEVEL	PU
BENEFICIARY NUMBER AND NAME	7 - NTUA
AUTHORS	
CONTRIBUTORS	Please see first page
WORK PACKAGE No	1
WORK PACKAGE LEADER	Kyriakos Giannakoglou (NTUA)
COORDINATOR VALIDATION DATE	29/02/2024

ACRONYM	MEANING
AI	Airbus
DAv	Dassault Aviation
DLR	Deutsches Zentrum Für Luft- und Raumfahrt
FOSS	Flow & Optimization, Software & Services
INRIA	Institut National de Recherche en Informatique ey Automatique
IRT	IRT Saint Exupery
NTUA	National Technical University of Athens
ONERA	Office National D'etudes et de Recherches Aerospatiales
OPT	Optimad Engineering SrL
RR	Rolls-Royce PLC
UNICA	Universita Degli Studi di Cagliari
USFD	The University of Sheffield

1 Links with EuroHPC Activities

1.1 Link with the REGALE EuroHPC project

NTUA is, simultaneously, involved in the REGALE EuroHPC project (GA No. 956560) in which its main responsibility is to perform shape optimization studies in hydraulic turbomachinery in the framework of Pilot 1 of the project; five pilots, acting as representatives of the next generation of HPC applications are considered in REGALE. The NTUA in-house GPU-accelerated PUMA code is used for the simulations and optimization runs. This is the same code as the one used in NEXTAIR, however, due to the applications of REGALE (hydraulic turbomachines), its incompressible fluid variant is used. Nevertheless, since both the incompressible and compressible fluid flow solvers of PUMA are parts of the same overall software, the compressible flow solver used in NEXTAIR has equally profited from the work done in REGALE. Excluding code customizations dealing with the specific optimization problem studied in REGALE, the following tasks of NTUA are directly linked with the NEXTAIR project: (1) making/adapting PUMA code so as to be ready for exascale deployment and (2) implement exascale-ready workflows for uncertainty quantification (UQ) using PUMA.

The former includes a restructuring of the parallel deployment and reduction in memory requirements of PUMA so as to cope with computational grids with hundreds of millions of nodes in multi-GPU platforms. Scalability tests performed thus far (some scenarios are still on-going) for computational grids up to 80 millions of nodes, on a computational node with 8 NVIDIA V100 GPUs and/or two nodes with 6 NVIDIA A100 GPUs are promising. For a computational grid of about 12 million nodes a near linear speed up is obtained until the partition size becomes small, i.e. on more than 6 GPUs. This enhanced version of PUMA is used in NEXTAIR for the multi-disciplinary studies of WP3 and WP4.

Regarding UQ, PUMA has been coupled with the Melissa workflow manager, (76). Melissa, orchestrates the simultaneous execution of the simulations required in UQ (herein using PUMA) interacting with commonly used batch schedulers (e.g. SLURM or OAR) of supercomputing systems. It also uses iterative algorithms for computing statistical quantities (mean, variance, Sobol indices, etc.) without storing any data to disk.

1.2 Link with the NextSim EuroHPC project

AIRBUS, DLR and ONERA develop the next generation aerodynamic simulation code CODA compliant with extreme-scale parallel computing platforms. Ongoing NEXTAIR activities are contributing to validate and enhance the maturity of this CFD code with a focus on CSM-CFD coupled computations and adjoint methods.

As a first step towards coupled CSM-CFD simulation, we considered the rigid DLR SMR-HARW F25 configuration to evaluate CODA's capability for handling extensive nonlinear steady flow analyses. Two unstructured, hybrid meshes, called thereafter M1 and M2, supplied by DLR, were used at ONERA for this analysis, each of these meshes consisting of 4.0 and 9.9 million grid points, respectively.

Physical conditions are typical of a transonic regime and the Mach number, reference pressure and temperature are set to $M_\infty = 0.78$, $P_\infty = 22729\ Pa$, $T_\infty = 216.8268\ K$, respectively. The Negative Spalart-Allmaras RANS model is used, with a turbulent viscosity ratio of 0.77.

We aim at computing the steady flow on this configuration, using the time-marching linearized implicit Euler method implemented in CODA (21). In practice, this is a penalized Newton method, in which a virtual time stepping is added. The relative importance of the time step is monitored with the residual using a SER ramping method, and the CFL reaches values up to 60000 in the case considered below. Linear systems are solved using a restarted GMRES strategy with a Krylov subspace of 60 vectors, together with a classical Jacobi preconditioner. The nonlinear solver progresses through three stages, gradually enhancing the flux reconstruction scheme and the accuracy of the Jacobian matrix, as outlined in Table 1.

Stage #	1	2	3
Flux limiter	Full Limiting	Spline quintic ($K = 10$)	idem
Extended stencil	no	yes	idem
Exact jacobian (AD)	no	no	yes
Relative residual tolerance	1.0×10^{-3}	1.0×10^{-5}	1.0×10^{-8}

Table 1: Parameters used in the different CODA solver stages.

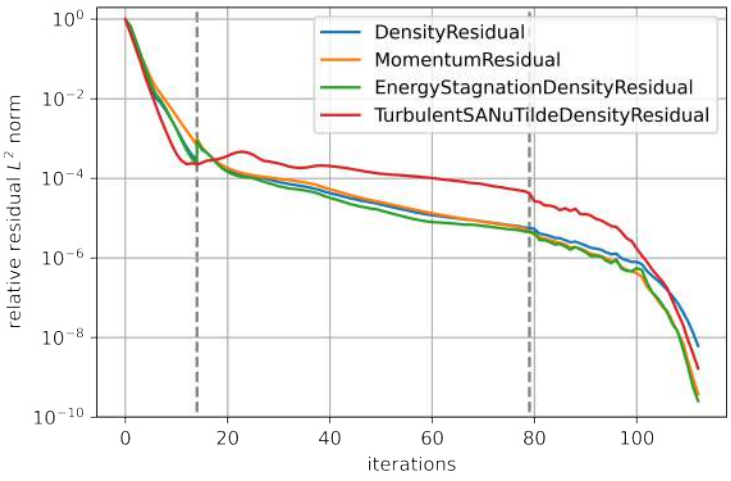


Figure 1: Relative Residuals L^2 norm across iterations, for an angle of attack $\alpha = 2°$ at $M = 0.78$. Different stages are separated by the vertical lines.

A typical convergence history of the density residual is shown in Figure 1, in which the three different

phases are highlighted. During the first stage the residual decreases rapidly, as the flow, initiated as a free-stream starts approaches a physical solution. This provides a better starting solution for the second and third phases where more accurate (but less robust) schemes are activated along with the exact Jacobian matrix evaluation (third stage).

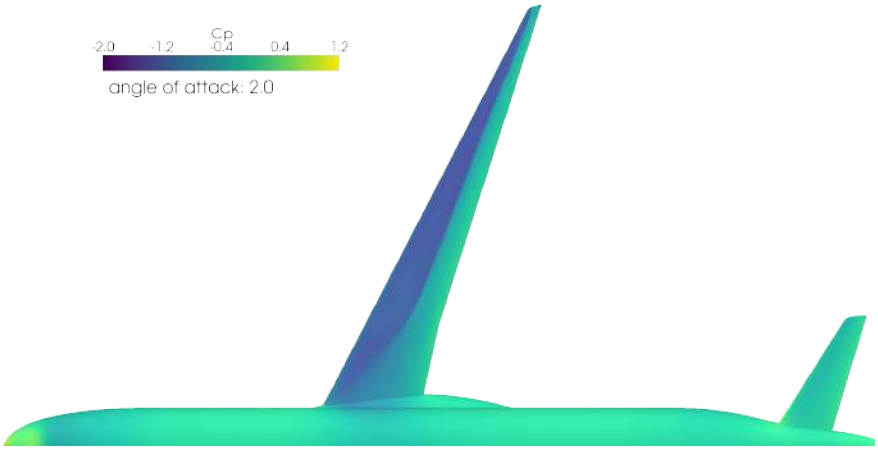


Figure 2: Top view of the DLR F25 configuration showing the pressure coefficient distribution at angle of attack $\alpha = 2°$.

In Fig. 2 the pressure coefficient distribution over the aircraft is plotted, showing notably the well established shock spanning on the entire wing. Computations were carried on a range of angles of attack in $[-2.0 : 3.0]$, and aerodynamic coefficients are shown in Fig. 3. Between meshes M1 and M2, only the third significant digit of the drag changes in all cases, and the solution can be considered to be converged in mesh size. However, for large angles of attack the flow detaches and the results on the lift varies on the second significant digit. The results are further compared with those obtained with the elsA solver (5), albeit with a different mesh (multi-block structured, comprising 12M grid points), and with different flux limiter (van Albada), convection scheme (centered Jameson while Roe is used in CODA) and linear solver (LU-SSOR).

Overall, this work focused on the resolution of the flow state at typical flight conditions around a rigid configuration, and is a necessary first step towards aeroelastic simulations.

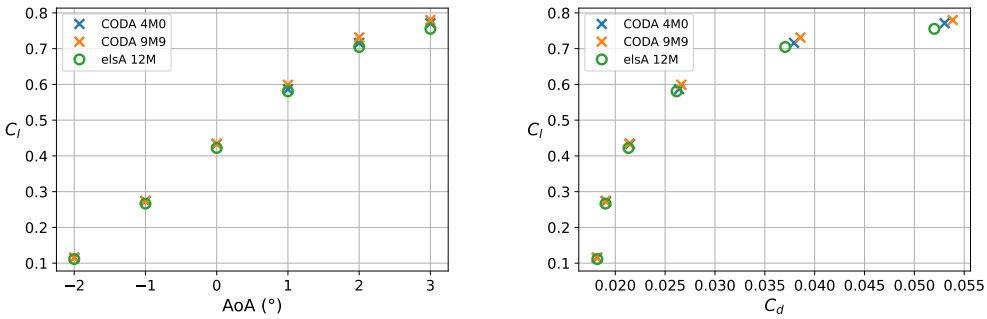


Figure 3: Lift coefficient v.s. angle of attack (right) and Lift v.s. Drag coefficient (left) for the two different meshes are shown with crosses, while the open circle denotes the results obtained using solver elsA (5) on a 12M points structured grid.

The adjoint capabilities of CODA are still under development at DLR: their current status does not allow a complete assessment of the CFD shape gradient yet. A first comparison is presented here by

ONERA for the adjoint solution with respect to the results computed by means of the legacy code elsA (ONERA-Safran property). The test case is represented by the Euler flow past a NACA0012 airfoil at $M = 0.5$ and zero incidence. In CODA, the (primal) flow field is solved by using a Discontinuous Galerkin (second order) spatial discretization. Then the GMRES algorithm is employed to solve the adjoint problem associated with the drag coefficient. Figure 4 illustrates the comparison of the components of the adjoint wall field as computed by CODA and elsA. Furthermore, for this particular test case, the adjoint solution has an analytical expression (40) and the CODA results also compare favorably to the analytic solution. Testing and validation will continue according to the newly available adjoint capabilities in CODA.

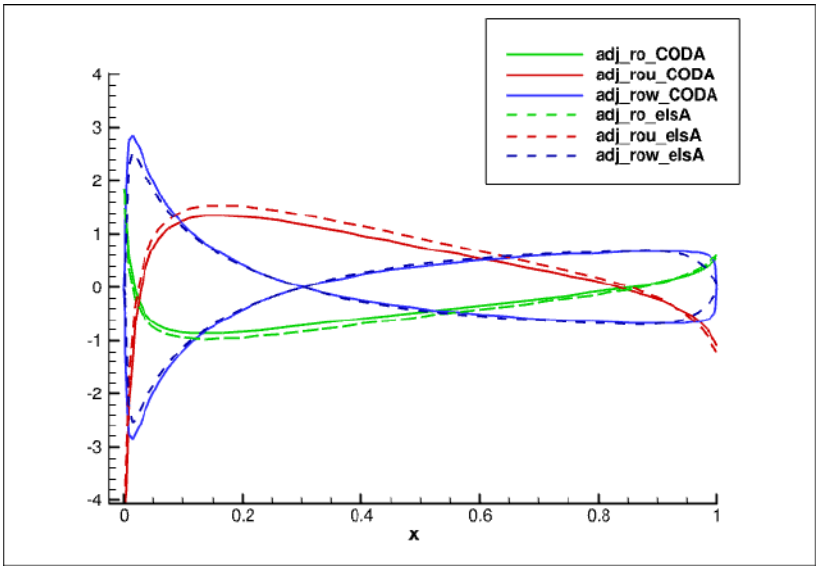


Figure 4: Comparison of the adjoint wall fields computed with CODA and elsA for the NACA0012 Euler test case at $M = 0.5$ and zero incidence.

2 Multi-Level Parameterisation & Multi-Fidelity Approaches in MDO

2.1 Multi-fidelity ML model to speed up topology optimisation of Heat Exchangers

2.1.1 Forewords

The goal of the activities performed by OPT in ST1.3.2 is the definition of a Machine Learning (ML) model to speed up the Multi-Disciplinary Optimization (MDO) of Additive Manufactured (AM) Micro-Channel Heat Exchanger (MCHE). Since, multiscale optimization of MCHE is the topic of ST6.3.2, the focus of this deliverable is the description of the ML model. A detailed description of the optimization method can be found in deliverable D6.3 and the accompanying report. ST6.3.2 focuses on the definition of an efficient multiscale approach for the numerical simulation and optimization of MCHE based on Triple Periodic Minimal Surfaces (TPMS) lattices. The goal of the optimization is to determine the topology of microchannel (i.e. lattice parameters) in order to maximize the performances of the heat exchanger. Due to the inherent multiscale nature of the problem, Conjugate Heat Transfer (CHT) simulations of MCHE require high-resolution computational models to well-resolve fluid dynamics at the scale of microchannels. Depending on the lattice geometry, computational meshes typically range from several dozens to a few hundreds of millions of cells to accurately resolve phenomena at the smallest time-space scales. While the CFD analysis of a single configuration is certainly possible with modern HPC resources, in a topology optimization loop, hundreds (if not thousands) of configurations must be spanned during the exploration of the parameter space, thus making topology optimization challenging to attain in a computational time compatible with industrial turnover times.

In order to cope with those challenges, the simulation strategy is based on homogenization theory. Instead of simulating the flow field at all time-space scale, the lattice matrix is treated as a meta-material and is simulated only at the macroscale in terms of suitable field-averaged quantities. The equivalent porous medium is characterized (locally) by a permeability tensor and heat transfer coefficients, which are assumed to be non-linear functions of the local lattice geometry and the local flow conditions. This leads to a fully non-linear Darcy-Forchheimer-type model, which requires proper closure relationships in order to be solvable. Activities carried out in ST1.3.2 focus on developing a ML technique to infer such (non-linear) closure relationships in order to link macroscopic properties of the lattice matrix to the behaviour of the flow field at the microscopic scale, thus realizing the two-way coupling between macro and micro(meso)-scales.

2.1.2 Multi-fidelity strategy to MDOs

Ultimately, the goal of the Reduced Order Model (ROM) developed in ST1.3.2 is to minimize the computational effort required by topology optimization of MCHE. The strategy developed in this task is built around on a Non-Intrusive ROM (NI-ROM) based on the Proper Orthogonal Decomposition (POD) method. In this approach, the full order model (i.e., high fidelity CFD simulation of the whole heat exchanger) is replaced by the homogenized model. Closure relationships required by macroscopic equations are obtained through fast predictions of the flow field at the scale of a single lattice cell. Such predictions are computed by the multi-fidelity surrogate starting from the (local) lattice geometry and the (local) flow conditions computed at the macroscale. From flow field predictions, relevant quantities, such as permeability tensor, Forchheimer's correction and heat transfer coefficient are extracted and fed back to the macroscale simulation. The focus of ST1.3.2 is the exploitation of multi-fidelity methods to reduce the number of CFD simulations required to train the ROM. This choice is motivated by the fact that ROMs for CFD applications operate in the so-called "data-sparsity" regime, i.e. training data are scarce due to the significant computational cost required to obtain high-fidelity CFD simulations. On the contrary, ML approaches allow to exploit heterogeneous information sources, and consequently reduce the cost associated to dataset generation.

2.1.3 Multi-fidelity POD-ROM

In order to design the ROM, a dimensionality reduction was required. POD (specifically, "the method of snapshots") was used to build the NI ROMs. NI ROMs are based on machine learning algorithms used to infer the correlation between low and high fidelity data. The most common choices are Radial Basis Function (RBF) interpolation, Gaussian Process (GP) regression, or Neural Networks (NN). By combining these regression models with the encoded representation obtained by POD, a ROM was trained to predict the fluid dynamic fields in a lattice cell given the cell geometry and operating conditions, that is the average pressure gradient across the cell as computed at the macroscopic scale.

The proposed POD multi-fidelity ROM was designed to address sparse High Fidelity (HF) problems, characterized by large geometrical variations. In our application, geometry variations of lattice cells are generated during the optimization loop. A Non-linear Auto-Regressive multi-fidelity Gaussian Process (NARGP) (Perdikaris et al., 2017) was adopted to infer the non-linear correlation between high fidelity and low fidelity POD coefficients. In other words, a multi-fidelity model was designed to improve the projection error obtained by a low-fidelity POD by leveraging few sparse high fidelity data. As opposed to other multi-fidelity models, NARGP is characterized by an enhanced level of flexibility and is also non-intrusive. Moreover, NARGP has a recursive structure which allows to incorporate an arbitrary number of fidelity levels. This is one of the generalization envisioned for the remaining part of activities for this sub task. At the present moment, only two fidelity levels were considered, both obtained through CFD simulations. HF data were obtained by running simulations on single lattice cells with variable geometry using a fine computational grid of approximately 500k computational cells. Low Fidelity (LF) data were obtained using a coarser computational mesh of 60k cells (more details on the training dataset are given in the next section). LF data were used to quickly populate the training dataset at a relatively small cost, albeit with data points characterized by a lower accuracy. Since simulations on the coarser grid tend to under-resolve heat transfer phenomena (especially for Reynolds number approaching the transitional regime), correlation between HF and LF POD coefficients is assumed to be non-linear. Hence, the use of NARGP model. Lastly, LF and HF training datasets are nested. This proved to be slightly beneficial during the training, although not mandatory.

2.1.4 Training dataset

The training dataset of microscale simulations was generated with two nested fidelity levels. Each fidelity level corresponds to simulations performed at a different mesh resolution. HF simulations were performed on a computational grid of roughly 500k computational cells. This mesh size was determined during a mesh dependency study as the best compromise between accuracy and computational cost. LF simulations were performed on a coarser mesh of 60k cells. Figure 5 shows and example of HF and LF simulations performed for two different lattice cell geometries. From the mesh dependency study, flow fields computed on the LF computational grid show to under-resolve certain flow features, especially in the range of parameters corresponding to transitional Reynolds number at the microscale. This eventually leads to slightly poorer prediction accuracy of the multi-fidelity model in such a range (see next sections).

Each simulation in the database was performed on an individual lattice cell with varying geometry and operating conditions. Both wall thickness and cell frequency were varied to generate the training data points. The main parameters used to generate dataset are summarized in the Table 2.

For each simulated geometry, two type of simulations were performed:

- **Cold runs**: used to characterize permeability tensor and Forchheimer's correction as a function of the pressure gradient across the lattice cell and its geometry.
- **Hot runs**: used to characterize the heat transfer coefficient as a function of the pressure gradient across the lattice cell and its geometry.

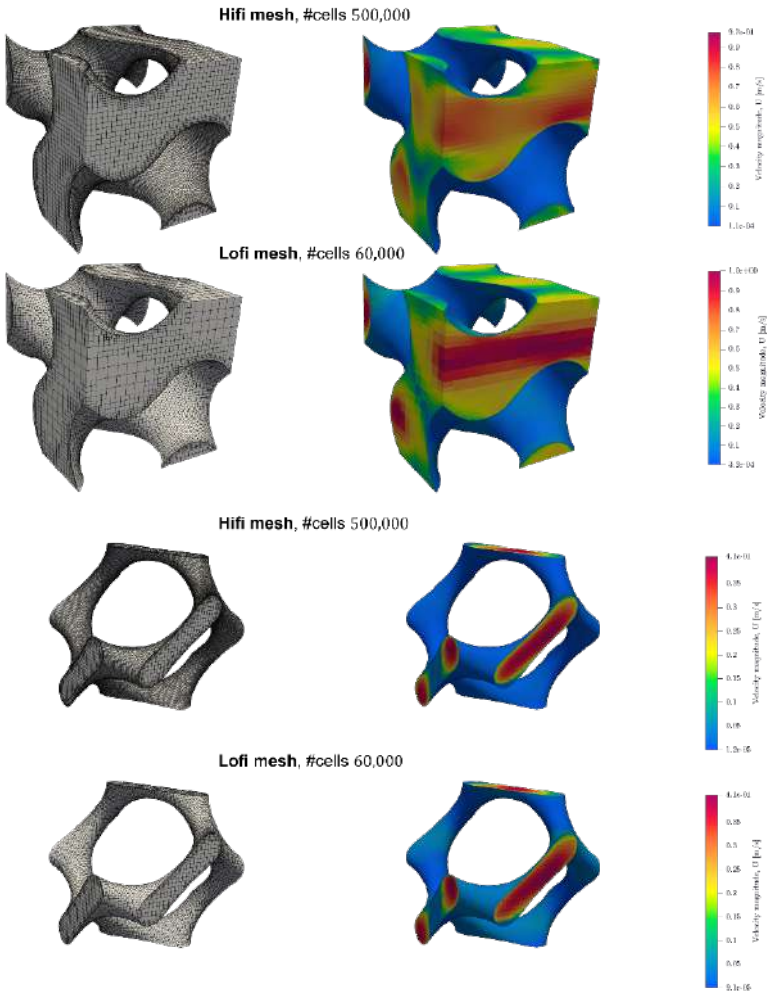


Figure 5: High and low fidelity simulations for two different lattice cells included in the training dataset.

Property	range	units
Pressure gradient across unit cell	$0 - 200$	Pa
Gyroid thickness	$0.0003 - 8.0$	mm
Gyroid x, y, z-period	$5 - 20$	mm
Coolant dynamic viscosity	0.00165	Pa s
Coolant density	1068	kg/m^3

Table 2: Geometrical and physical parameters used to create the dataset of micro-scale simulations.

Both hot and cold runs were performed using the *buoyantPimpleFoam* solver from the OpenFOAM suite (https://www.openfoam.com/). In cold runs, a pressure gradient was assigned in turn along each lattice direction to characterize the corresponding row of the permeability tensor for non-isotropic cell (i.e. lattice cell characterized by different frequency along each coordinate axis). Periodic Boundary Conditions (BCs) on pressure were enforced along the other directions. Periodic BCs on velocity were used in all directions. This setup corresponds to OpenFOAM *CyclicAMI* BCs and is commonly used to emulate fully developed flows in the bulk region of a porous medium. Permeability tensor and Forchheimer's coefficient are reconstructed using the Darcy-Forchheimer's law, starting from the applied pressure gradient and a reference velocity computed by the simulation. For small Reynolds

numbers, the reference velocity was computed as the average inlet fluid velocity at the steady state. Simulation performed at transitional Reynolds numbers (i.e. starting from Re > 700) does not exhibit a steady solution. In such cases the reference velocity was computed as the inlet velocity averaged over a time-window of 1 s at statistical convergence (i.e. starting from a simulated time of 4 s.).

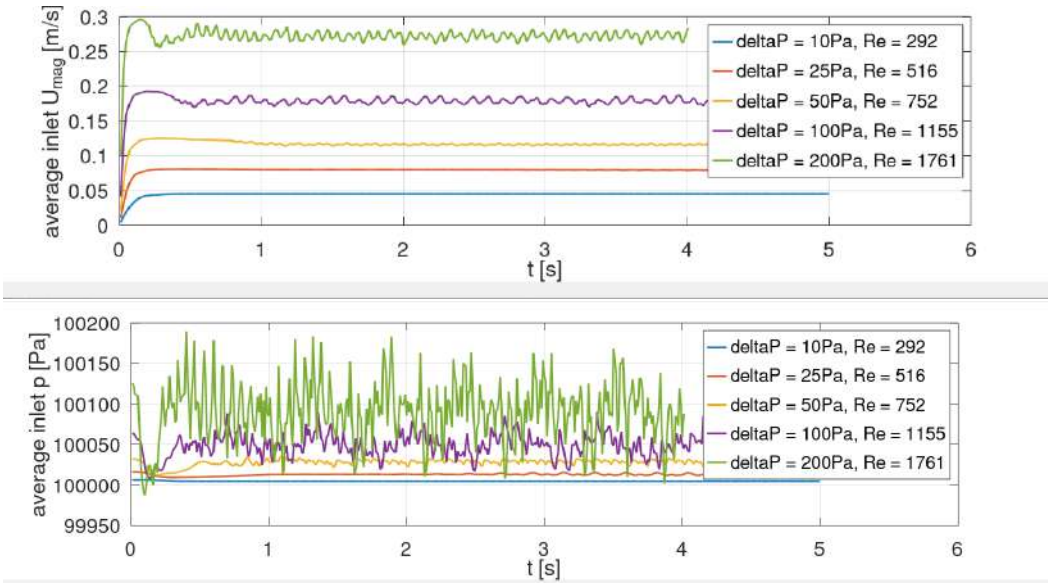


Figure 6: Example of time history of the average inlet velocity (top) and inlet pressure (bottom) for a isotropic gyroid cell with 10 mm period and 1mm wall thickness at various pressure gradient across the unit cell. Starting from Re = 700 (approx.), flow field does not exhibit a steady state.

Hot runs were performed by assigning a fixed temperature to the solid and a fixed inlet temperature to the fluid. This choice is motivated by the fact that the viscosity of the coolant does not exhibit strong variations with temperature in the range of operating conditions simulated during the optimization loop. Thus, the heat transfer coefficient can be considered approximately constant with respect to temperature variations and the thermal and dynamic solvers can be decoupled. In order to characterize the heat transfer coefficient, hot runs were started from the converged solution of the cold run. During the hot run, the heat transfer coefficient was estimated at each time step from the wall heat flux and the instantaneous temperature difference between solid and coolant. Instantaneous values of the heat transfer coefficient were averaged over the time history purged from the initial numerical transient. The dataset is composed of 105 high fidelity simulations and 294 LF simulations (gray circles and blue circles in figure 7) obtained by a Latin Hypercube Sampling (LHS) strategy. Each datapoint corresponds to a different cell frequency, wall thickness and pressure gradient imposed across the lattice cell. Among all simulations 263 LF and 97 HF simulations exhibited a steady state. Figure 6, shows a projection of the dataset of steady simulations onto the plane of the parameter space spanned by pressure gradient and wall thickness. Simulations corresponding to wider channels (lower wall thickness) performed at higher pressure drop exhibited a unsteady behavior due the Reynolds number approaching the transitional regime. Data points in this region were generated using a Poisson Disk sampling strategy, that is the minimum Euclidean distance between randomly sampled point is constrained above a certain threshold to guarantee that datapoints are distributed as uniformly as possible. In both regions, HF and LF data points are nested. In our numerical experiments, this resulted in better training performances of the NARGP model.

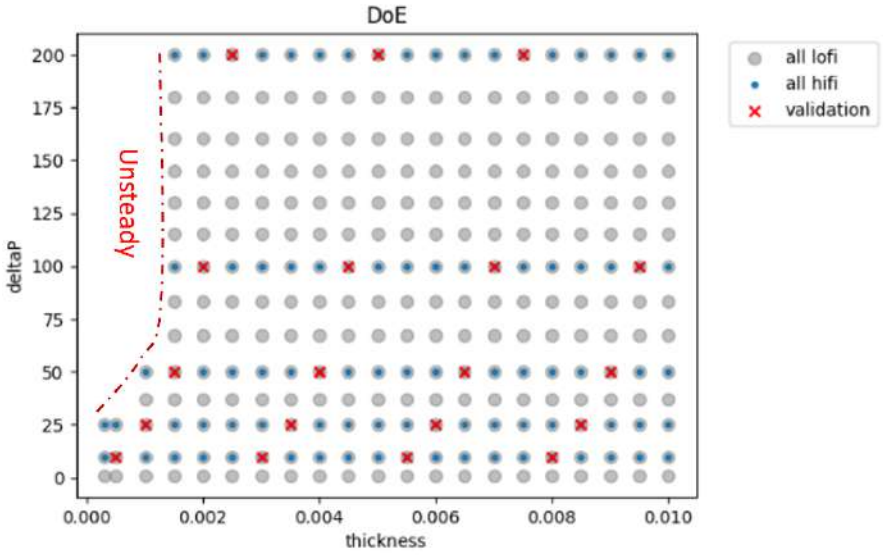


Figure 7: Distribution of data points in the region of the parameter space corresponding to steady flow field at the micro-channel scale. Blue dots represent high-fidelity points, i.e. simulations performed using a fine computational grid. Gray markers indicate low-fidelity samples. Red crosses indicate data points used for validation.

2.1.5 Solution Mapping

One of the main drawbacks of the POD method is that all the snapshots are required to have the same cardinality in order to form the snapshot matrix, that is all the simulations on the same fidelity level must be performed on the same computational mesh. In practice, however, it was impossible to use the exact same computational mesh across the whole training dataset due to the varying geometry of lattice cells. To circumvent this problem, a strategy was devised to map numerical solutions obtained on different computational meshes onto a common "template" grid. The idea is similar (in philosophy) to mesh registration. A suitable geometrical mapping is defined such that HF (LF) computational grids are mapped to a common HF (LF) "template" mesh. Next, flow fields are interpolated onto the template grid and then used to form the snapshots matrix. This procedure, of course, introduces an additional source of error due to data interpolation resulting in a prediction accuracy limited by the interpolation error. However, given the size of the computational meshes used in the experiments, it was found that such an interpolation error is usually lower than the projection error. In this work, the focus was on lattice cells belonging to the family of TPMS. Given that, the geometrical mapping was required to handle only shape variations in the same class of lattice cell topologies. Variable frequency is handled by "stretching" the template grid differently along each coordinate axis. Thickness variations are handled by parametrizing the mapping with respect to the distance from the medial axis of channels (see figure 8, for a schematic depiction).

2.1.6 Results

For the HF POD, a variable number of HF snapshots ranging from 5 to 75 were used to perform a parametric study in order to define the energy level at which the POD basis is truncated. The energy chosen to truncate the POD basis was set to $\varepsilon = 0.999999$. this value guarantees that the higher frequency modes corresponding to (almost) pure noise are removed, while retaining lower frequency modes which still carry physical information. This is shown in figure 9, where projection error for both validation and training datapoints are plotted against the energy threshold used to truncate the POD

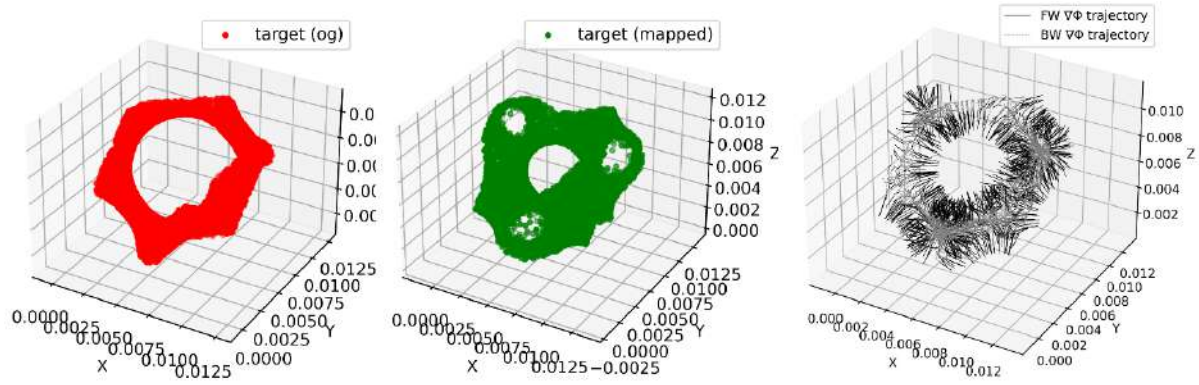


Figure 8: Registration of lattice cells with variable wall thickness. The target geometry (left panel) is mapped onto the template (middle panel). The mapping is performed by advecting the zero-contour of the level set function along the level set gradient. These trajectories (right panel) converge onto the medial axis of the geometry. By varying the integration time along these trajectories it is possible to generate cells with variable wall thickness.

basis for a variable number of HF training snapshots.

For the energy threshold defined above, the energy of each POD mode is shown in figure 10, alongside the error histogram computed over the training and validation datasets. Figure 9 and 10, show that projection error on the training dataset is in the same order of machine precision. This corresponds to a projection error on the validation dataset between $5 - 10\%$, which is also the prediction accuracy obtained on permeability and Forchheimer coefficients (see later).

Since the multi-fidelity approach is sensible to the relationship between the LF and HF information, only the coefficients associated to the first modes are regressed with the NARGP model. This stems from the fact that LF snapshots struggle to capture high frequency features due to the coarser computational grid. Consequently, the projection of LF snapshots on the last part of the POD base will not necessarily improve the prediction accuracy. This is particularly evident for unsteady simulations, i.e. simulations performed on gyroid cells with lower wall thickness (larger channels) and higher pressure drops imposed across the lattice cell. Figure 11 shows the correlation plot between HF modes and the corresponding LF modes for a gyroid cell in the unsteady regime. While the first two modes are perfectly correlated (r = 0.99 and 1 respectively), starting from mode 2, correlation between HF and LF modes is almost non-existent (r = 0.11). This effect is likely due to the fact that some flow features (such as detaching vortices) are under-resolved on the coarser grid. In the HF POD such features are still represented by modes with lower frequency (larger energy). On the contrary, in LF POD modes this information is represented only partially due to under-resolution, leading to the low value of the correlation coefficient.

This is also confirmed by the value of permeability and Forchheimer's coefficient computed from HF and LF simulations (shown in figure 12). The discrepancy observed for lower value of the wall thickness (associated to unsteady simulations) is noticeable and lead to a poor correlation between HF and LF data.

From figure 12, it should be noted that Forchherimer's coefficients are also polluted by noise in the regime of unsteady simulations. This is likely the consequence of the statistical accumulation method described in the sect. 2.1.5, and could explain the better prediction performances on the Forchheimer's coefficient obtained with a single fidelity GP as opposed to the multi-fidelity approach. For the reasons described above, only the coefficients corresponding to the first 4 modes have been regressed with the multi-fidelity model, while less energetic modes have been approximated with single fidelity GP regression models. Since increasing the number of HF modes does not bring any significant benefit, also the number of high-fidelity snapshots was reduced to four. A summary of hyperparameters used

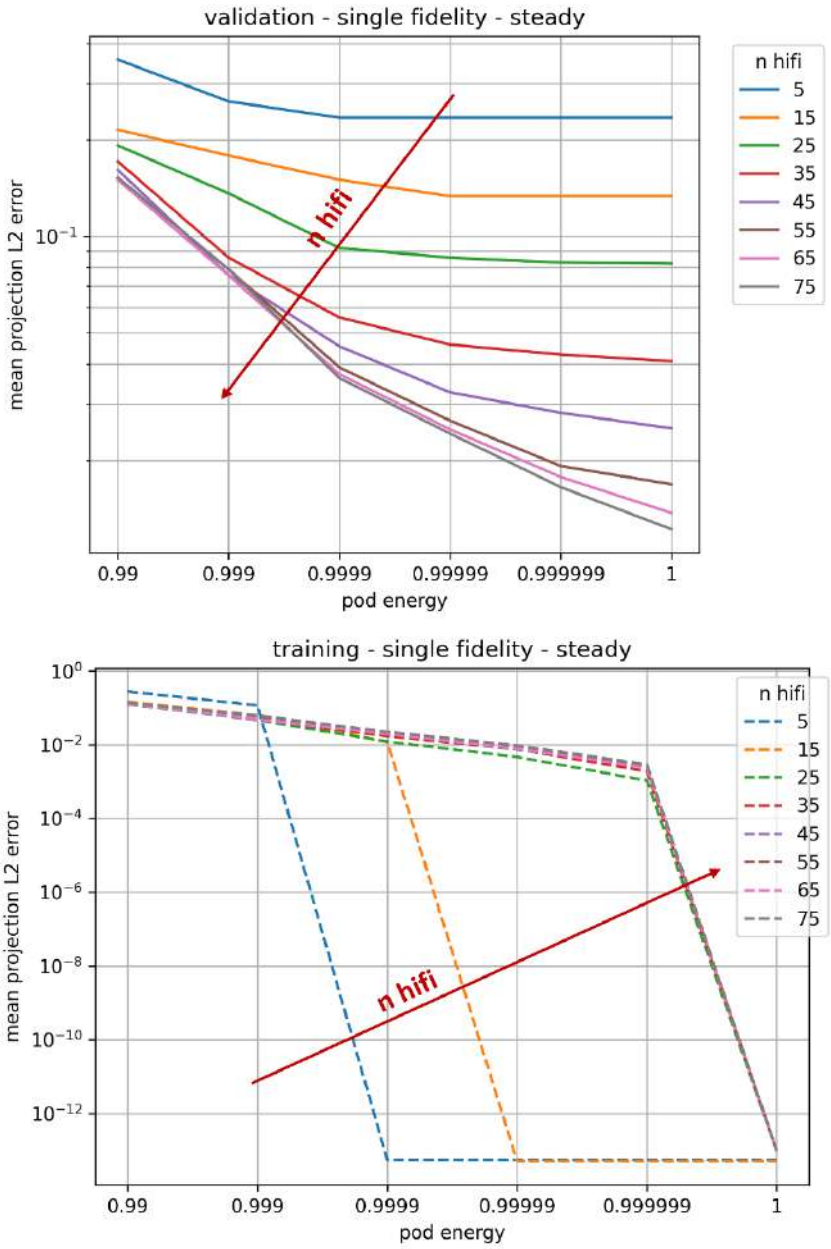


Figure 9: Projection error for validation and training data plotted as a function of the energy threshold used to truncate the POD basis using a variable number of HF training snapshots.

in the training can be found in Table 3. The single-fidelity GP column refers to the options for the single fidelity model used to fit the POD coefficients relative to less energetic modes. The same options have been used also for the single fidelity GP regression model used for comparison.

Results of the predicted coefficients are shown in figure 13 and 14. All the tested models show a good prediction accuracy for both permeability and Forchheimer's coefficient in range of geometric parameters and operating conditions corresponding to steady state flow at the microscopic scale. On the other hand, a reduced prediction accuracy in the range corresponding to unsteady flow is observed. This effect is due to the poor correlation exhibited by HF and LF snapshots due to flow features being under-resolved in LF simulations. For the Forchheimer's coefficient, the single fidelity (GP) model (used here for comparison) shows slightly better prediction performances than the multi-fidelity

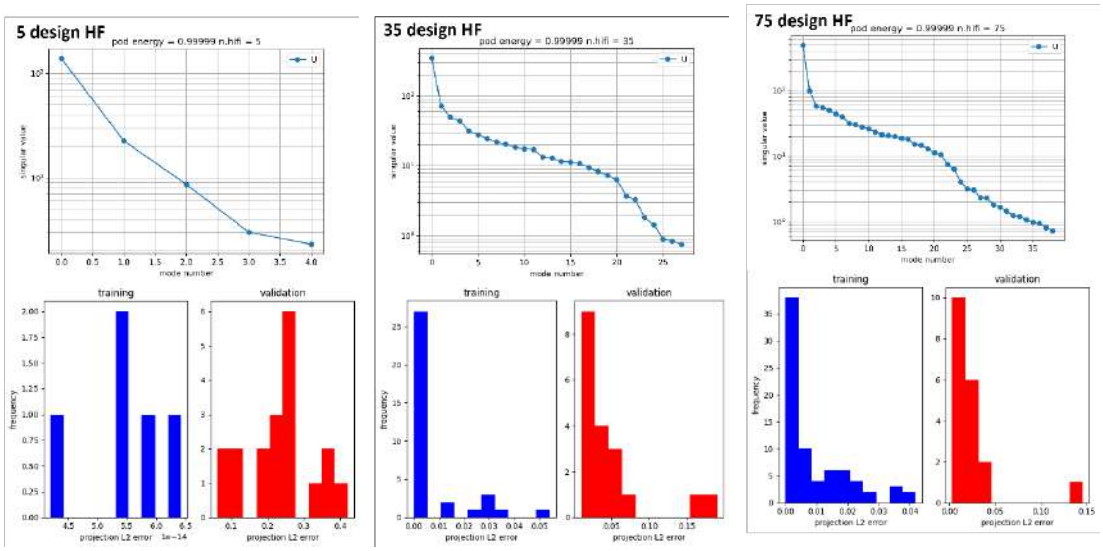


Figure 10: Validation and training error plotted against the energy threshold used to truncate the HF basis for different number of HF snapshots.

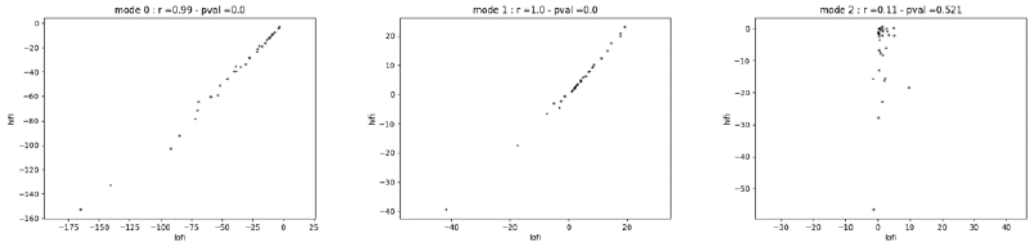


Figure 11: Correlation plot for the between low and high fidelity POD for the first three modes computed for an unsteady flow field. While the first two modes are perfectly correlated, starting from mode $\#2$, the correlation is almost non-existent, suggesting that some flow feature observed in HF simulations are under-resolved in the LF simulations.

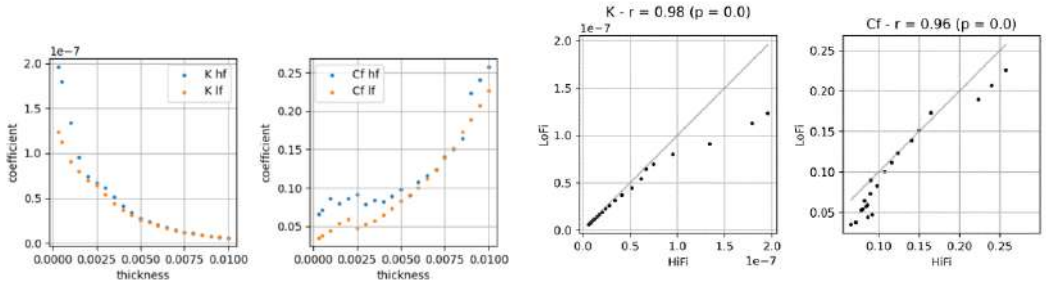


Figure 12: Permeability and Forchheimer's coefficient computed from HF (blue) and LF (orange) simulations (left panel) and their correlation plot (right panel). In the region characterized by lower wall thickness (larger channel) the discrepancy between HF and LF data is noticeable leading to a poor (linear) correlation.

model, due to the presence of noise (observed in both HF and LF simulations).

Option	LF model	HF model	Single fidelity GP
Optimization algorithm	BFGS(*)	BFGS	BFGS
Max. iterations	2500	2500	2500
Optimizations restarts (**)	10	8	10
Kernel	Matern 3/2	Matern 3/2	Matern 3/2
Gaussian noise variance	0.2 (normalized)	0.2 (normalized)	0.2 (normalized)

Table 3: Options used for the main hyperparameters in the training of the multi-fidelity model. (*) Broyden–Fletcher–Goldfarb–Shanno algorithm. (**) Random restarts of the optimization to increase robustness with respect to the initial guess.

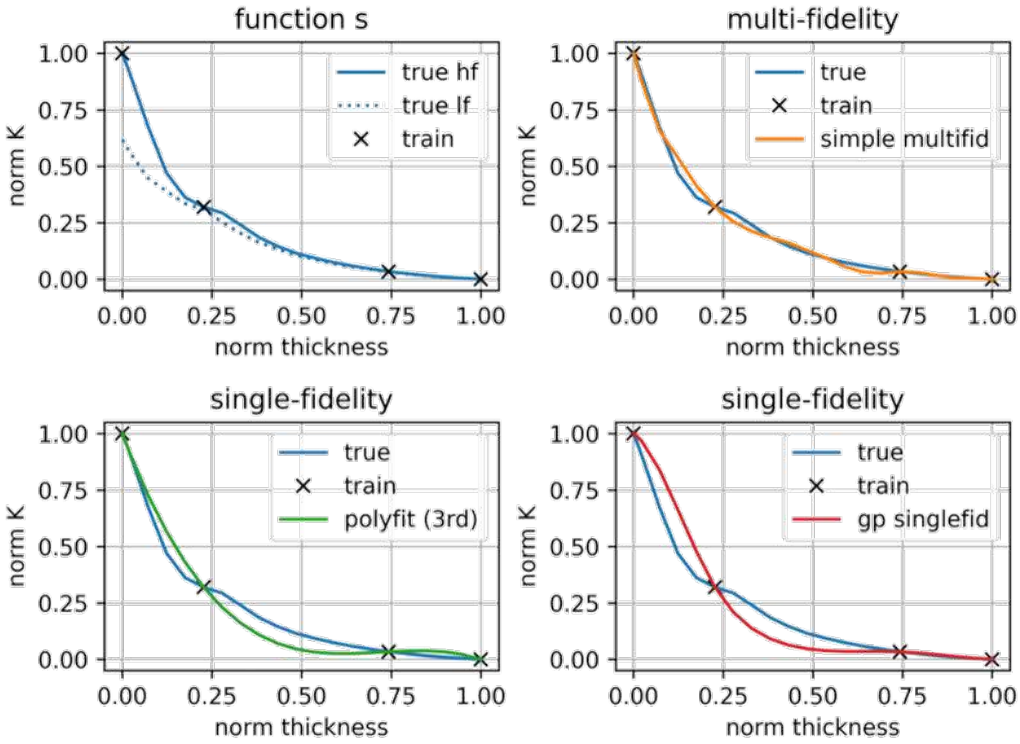


Figure 13: Predictions of permeability coefficient obtained with a multi-fidelity model and compared with a single fidelity GP model.

2.1.7 Conclusions

The purpose of the multi-fidelity POD ROM presented in this work was to improve accuracy of single-fidelity ROMs, while reducing the amount of HF information needed to train the model. In light of the results showed in section 2.1.7, several conclusions can be drawn.

First, the addition of LF snapshots improves the prediction accuracy of the model in the steady state regime. This is easily noticeable in Figure 9, where the multi-fidelity model produces prediction on permeability coefficient which are >10% more accurate than the single fidelity ROM.

Second, the presence of noise in the input data has a detrimental effect on the prediction of the Forchheimer coefficient. This is evident in figure 14, where the multi-fidelity model tries to mimic the high frequency noise from HF data, as opposed to a the single fidelity GP. To further improve the prediction accuracy for the Forchheimer's coefficient, we plan to investigate the effect of different statistical post-processing of both HF and LF simulation data. Additionally, models based on more than two fidelity levels will be investigated in order to incorporate data with an intermediate fidelity which

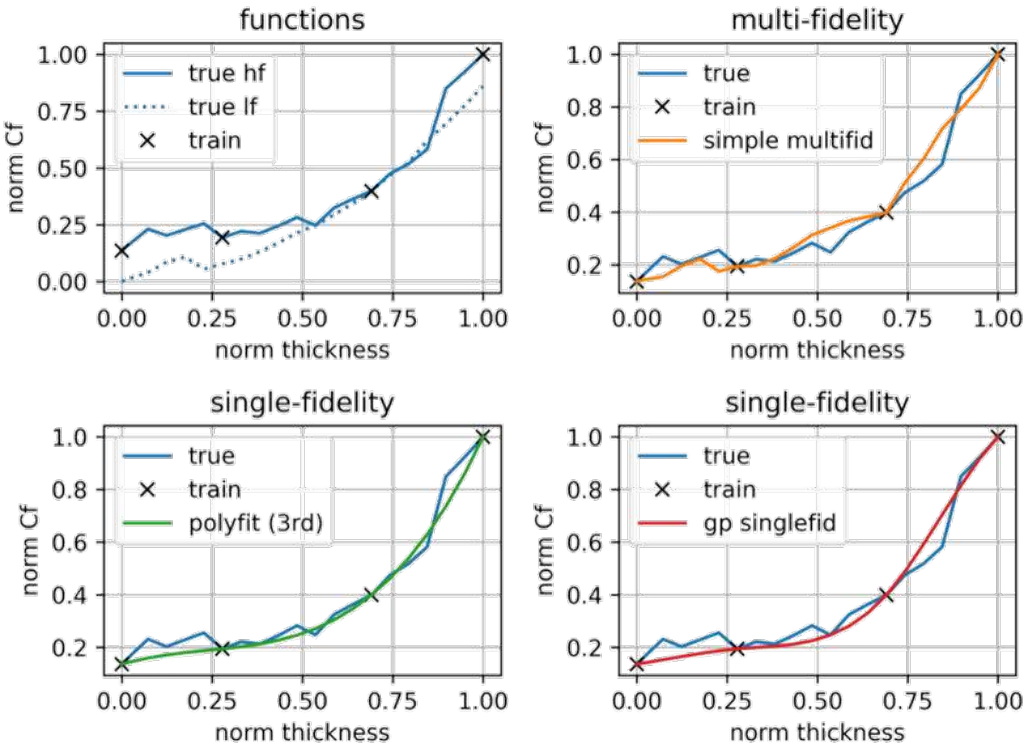


Figure 14: Predictions of Forchheimer's coefficient obtained with a multi-fidelity model and compared with a single fidelity GP model.

would still represent a significant speed-up with respect to relying solely on HF simulations.

Given all these considerations, the proposed multi-fidelity POD ROM was able to augment the capabilities of the analogue single-fidelity NI-ROM in the steady regime, both in terms of representability and prediction accuracy. The advantages in terms of computational costs are inevitably related to the choice of the LF model. Nevertheless, this experiment demonstrates that the HF approximation of a ROM could benefit from otherwise poor LF information.

2.2 Multi-level Parameterisation to Speed-up MDO

We investigate the creation of a preconditioner from a Hessian matrix approximation of the objective function to speedup the optimization. This approximation is very expensive to compute when applied to the full design space. Therefore, the proposed approach aims at building this approximation on a well-chosen subspace: the most and least important gradients of the objective. Then, a sampling of the objective function gradient is performed on this subspace. Finally, the Hessian matrix approximation is obtained. The approach is illustrated in figure 15.

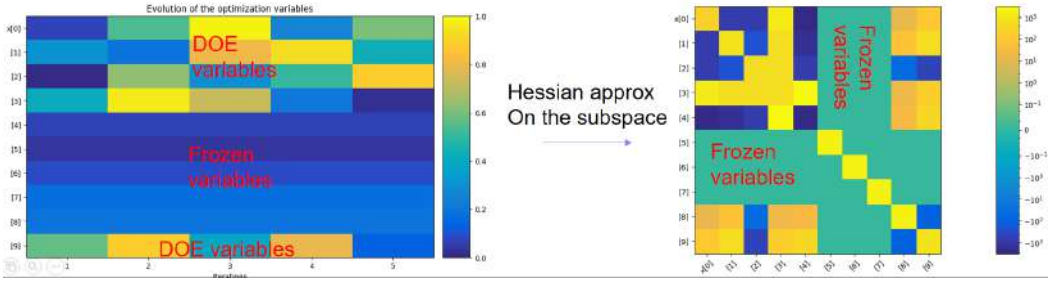


Figure 15: Construction of the Hessian matrix approximation in a reduced dimension space

Then, the preconditioner is obtained from the formula $P = H^{-(1/2)}$. This preconditioner is passed to the preconditioned variant of the gradient-based L-BFGS-B algorithm (80) developed at IRT. The overall process has been tested on mathematical formulae and does not currently provide any speedup, the reasons for this are being investigated. The subspace selection criteria may not be adequate. The active subspace method (41) is also being investigated for this purpose. Once the speedup is demonstrated, the process will be applied on the wing optimization of the DLR-F25 configuration.

2.3 Required Simulation Capabilities for the UHBR fan (TC3) (RR, UNICA, USFD)

In this task, ST1.3.2 (WP1) - Multi-Level Parameterisation & Multi-Fidelity Approaches in MDO (M1-M18), USFD, UNICA and RR have worked on the development of the required simulation capabilities for the UHBR fan multi-objective optimization (MOO) that will take place in ST4.3.1 (WP4) - Multi-fidelity aerodynamic optimization of the UHBR fan blade (M9-M27).

As outlined in the NEXTAIR project proposal (HORIZON-CL5-2021-D5-01-06, PartB document), the present task ST1.3.2 is meant to pave the way for the posterior task ST4.3.1; which states: ``UNICA, supported by USFD for the integration of the structural requirements and RR for the CFD models at different levels of fidelity, will perform parametric and MOO studies, making use of multiple levels of fidelity".

At the same time, the parametric MOO studies in ST4.3.1 are preceded by a problem setup and validation phase outlined in task ST4.2.1 (WP4) - optimization problem setup and validation, UHBR TCs (M03-M18):

TC3 - Aerodynamic and structural simulation capabilities for UHBR composite fan blade:

- RR and UNICA will focus on the aerodynamic simulations:
  - Lo-Fi simulations on the isolated fan blade using RANS solvers
  - Hi-Fi simulations (U-RANS) on the installed configuration to take into account component interactions, such as non-axisymmetric OGV-Pylon and non-axisymmetric intake. If required, the DDES and zonal LES HYDRA capabilities will be used.
- USFD will focus on structural solvers and their link to aerodynamic simulations.

So, the present report should demonstrate the acquisition of the necessary capabilities to fulfil the requirements for ST4.3.1 (WP4) described above. These requirements have been met (for (WP4), and also have been exceeded, since also capabilities for WP5 and WP6 have been developed, which are also presented here. It seems adequate to present these additional developed capabilities for WP5 and WP6 here in the present WP1 report, since, on the general level, WP1 is meant to ``ensure that the developed methods, workflows and codes, when used in WPs 3-6, will yield designs that can run in realistic time scales faster than with the state-of-the-art tools (KPIs 1-6)''. Nevertheless, it should be noted that ST1.3.2 (WP1), in particular, does not stipulate as a requirement the additional achieved capabilities for the posterior strategic tasks of WP5 and WP6 described below (underlined):

- ST5.3.2 (WP5) - Robust optimization of a UHBR Fan (M18-M36): ``Manufacturing variability data will be combined with UQ techniques (from WP1) and Hi-Fi multi-disciplinary numerical simulations to determine the uncertainty in component performance''.
- ST6.1.2 (WP6) - Evaluation of residual performance and life of Digital Twins (M06-M24): ``UNICA, will focus on meshing and CFD analyses required for estimating the performance, considering the very-nonlinear interaction of the modified shapes and their neighbouring components. This will require a multi-passage multi-stage unsteady simulation.''

Even though the preliminary development of capabilities to be used in WP5 and WP6 for the UHBR Fan (TC3) is not explicitly stipulated in WP1, from the NEXTAIR Plan, it is clear that the capability to simulate the geometrical variability of 'real', manufactured UHBR fan blades in multi-passage, multi-stage unsteady simulations is a necessary capacity to successfully carry out the planned tasks in WP5 and WP6 for this testcase. Hence, these are also presented here.

The aerodynamic and manufacturing variation simulation capabilities for WP4, WP5 and WP6 are achieved thanks to a series of fully internal Rolls-Royce software packages, and the structural capability is achieved thanks to an in-house Rolls-Royce structural finite element method (FEM) solver and two additional commercial tools:

- For the aerodynamic simulation capabilities, the Rolls-Royce in-house codes PADRAM (PArametric Design and RApid Meshing system) (69) and HYDRA-CFD (a suite of CFD solvers) (33) are used.
- For the reproduction of real manufactured fan blade geometries with manufacturing deviations, in addition to PADRAM, the in-house software P2S (Point2Surface) (52) and the optimizer SOFT (Smart optimization For Turbomachinery) (66) are used to construct the digital CAD models in a process called ``inverse-mapping'', developed at Rolls-Royce.
- The structural simulation capabilities are achieved thanks to the Rolls-Royce internal finite element method (FEM) code SC03 (2), the Cambridge Flow Solutions (CFS) meshing package Boxer (13), and the commercial tool Ansys ICEM CFD (24).

A short description of these software packages is provided in the next section.

2.3.1 Description of the Rolls-Royce In-house Software Used

Rolls-Royce counts with several software tools developed internally to tackle the company's needs for engineering analysis of its turbomachinery products and components.

Currently the in-house SOFT-PADRAM-HYDRA (SOPHY) (67) system is used in Rolls-Royce for aerodynamic optimization. The main elements (optimizer-mesher-CFD solver) can be used separately or together as a fully integrated, automatic, and flexible system (as shown Figure 16). These elements are described below.

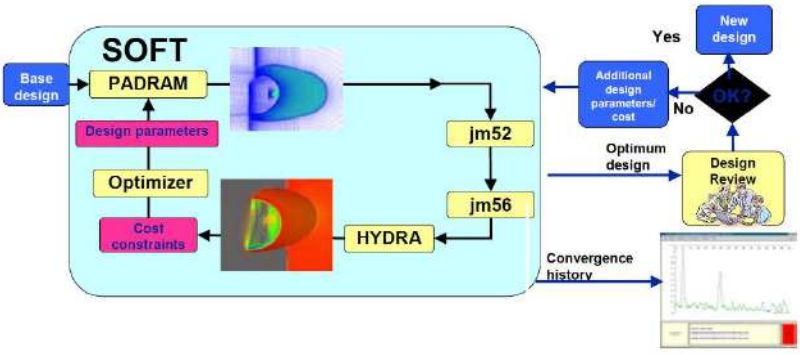


Figure 16: Workflow of the SOPHY automatic aerodynamic optimization system. From (67).

SOFT (66) provides state-of-the-art optimization libraries, and an integrating platform runnable both through a graphical user interface (GUI) and in batch mode through Python scripts that takes in template text files as inputs that define the optimization strategy, cost functions, optimization parameters, the limits of these parameters, the optimization algorithms to be applied and their sequence, optimization constraints, and the options for the generation of optimization history files, as well as post-processing capabilities such as the capacity to generate plots, contours, parallel co-ordinates, and pareto fronts. This tool is analogous to GEMSEO (19).

PADRAM (69) is an automatic parametric geometry modeller and rapid meshing system with multi-passage and multi-stage capability (this capability fulfils the requirements for ST6.1.2. In our multi-fidelity modelling problem, PADRAM receives as input a blade definition file (BDF) that expresses the geometry of blade streamline sections in cylindrical coordinates (r, θ,a z), as well as a user-defined mesh configuration file. The meshing options of the multi-block O-H-C grid available to the user is extensive, including number of nodes across several directions of the blocks, cell clustering, and

automatic application of cell clustering based on boundary conditions and desired y+ values. The mesh generation is very fast (in the order of seconds) and fully automated, so it does meet the necessary condition of not requiring user interaction during the automatic optimization cycle; another necessary requirement for the multi-objective optimization (MOO) to take place in ST4.3.1, and other work to be conducted in ST5.3.2 and ST6.1.2.

Apart from rapid and automatic mesh generation, PADRAM also offers a number of different ways to parametrically and automatically modify the geometry of a blade and re-mesh it from scratch at each iteration of the optimization cycle (54). This is done through the input and change in values of a geometrical blade design parameters (DP) file as well as another file containing the radial (span-wise) locations of the blade where these geometrical perturbations will be applied. Each design parameter (DP) that morphs the geometry — also called blade engineering parameter (EP) — controls a particular degree-of-freedom (DOF) of the blade. A partial summary of five parameters is provided in Figure 17.

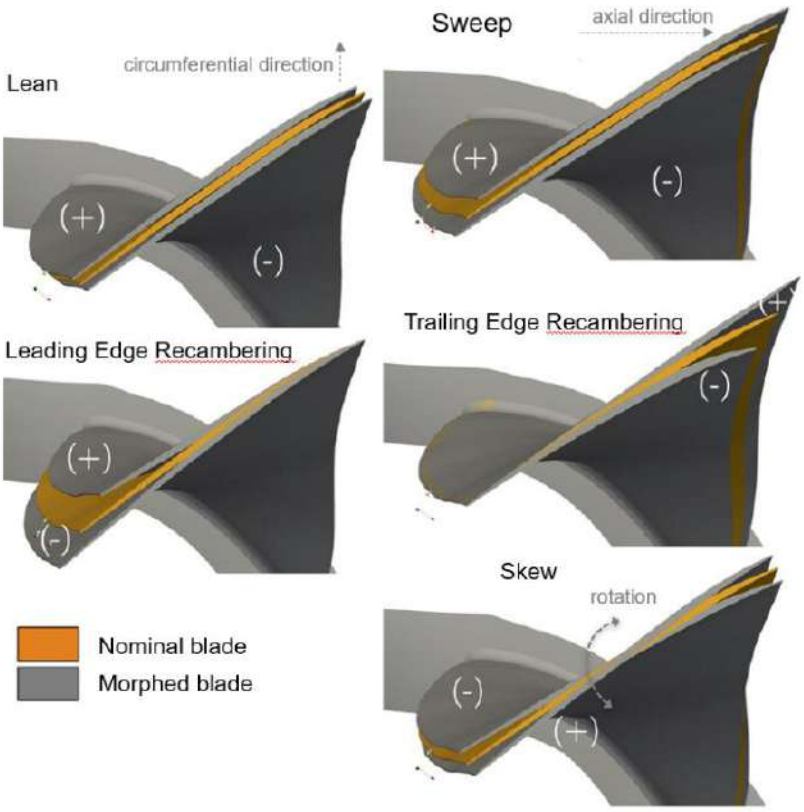


Figure 17: Partial description of the capability of the PADRAM design space to morph geometries. Adapted from (65).

HYDRA (33) is a suite of CFD flow solvers with linear, non-linear, and adjoint capability; both for steady state (non-time varying) and unsteady state (time-varying) simulations. The suite provides a consistent framework for input, output, multi-grid acceleration, parallelization, and visualization. The steady state solver can use both explicit and implicit time-stepping schemes; the explicit utilizes a standard Runge-Kutta multigrid scheme with local time-stepping, and the implicit steady solver a semi-implicit Runge-Kutta multigrid scheme with Richardson incomplete lower-upper decomposition. The adjoint solver Adjoint HYDRA enables to efficiently determine the sensitivity of one (or more than one) scalar objective functions to any number of small changes in the geometry or the boundary conditions, proving to be very useful in optimization processes.

With regard to multi-passage and multi-stage capability; full annulus or sector, single stage or

multiple-stage turbomachinery models (where blade row counts permit) can be simulated, since both rotating and stationary components can be coupled in the simulation with several boundary condition options suitable for both steady and unsteady turbomachinery computations, including periodic boundaries, mixing and sliding planes and phase-lagged periodic setups.

Regarding the turbulence modelling options, the available physical viscous models for RANS computations include Spalart-Allmaras (with or without helicity-correction), k-ϵ, and k-ω in two variants: the explicit algebraic stress model (EASM) and the shear-stress transport (SST) k-ω models. Furthermore, adaptive wall functions are available in HYDRA to cope with cases where the grid resolution is not sufficient near the solid walls $y^+ > 1$ (53). For large eddy simulations (LES), the viscous models available are the detached eddy simulation (DES) based on the Spalart-Allmaras model, the hybrid DES model, and the Smagorinsky model with near wall damping.

Lastly, a utility of post-processing tools gives the user a wide array of options to interrogate grid and flow solution data in the output files and perform post-processing functions.

The integrated use of these three tools conforms the SOPHY automatic aerodynamic optimization system, represented in Figure 16 in a workflow (67).

P2S (53, 68) is a Rolls-Royce proprietary code to perform mesh-based, automated distance-field computations between surfaces and point-clouds in any combination. Its library version is fully integrated with PADRAM and HYDRA and can perform multi-data analysis and treatment, including statistical analysis, amongst other capabilities. It is key in the automated process developed by Rolls-Royce to model the geometrical variability of manufactured blades (explained in section 2.3.2).

2.3.2 Multi-Fidelity Aerodynamic Simulation Capabilities

As a short introduction, the aerodynamic simulation capabilities developed for the UHBR Fan Testcase (TC3) comprise four dimensions of fidelity that can be combined as per the requirements of the user:

1. Single-Passage – Multi-Passage: single-blade-passage and multiple-blade-passage (whole-annulus) capability.

2. Single-Stage – Multi-Stage: simulation of the isolated fan blade stage or multiple-stage simulation of the blade in its installed configuration: fan rotor blade plus bypass outlet guide vane (BOGV), engine section stator (ESS) and straightener vane (SV); to take into account neighbouring component interactions. These neighbouring components are shown in Figure 18.

3. Turbulence Modelling: Steady-RANS – Unsteady-RANS (U-RANS), Favre-averaged Non-linear Harmonic (FNLH) and Large-Eddy Simulation (LES) (fully resolved or RANS-LES hybrid variants): capability to model in HYDRA unsteady flow influences with several turbulence models (as described in section 2.3.1) and to approximate the unsteady flow interactions in a multi-stage simulation through FLNH.

4. Modelling of Real Geometry Features Elastic rings, vortex generators and geometrical variability of manufactured blades (see section 2.3.2.

Several combinations of these different levels of fidelity are presented next as examples to demonstrate the present multi-fidelity approach.

Single-Passage, Single Stage (public test case)

The capability of simulating isolated fan geometries is here demonstrated on the NASA R37 test-case, which is a public geometry often used for validation of CFD solvers. This is the simplest, low-fidelity (Lo-Fi) testcase domain possible. A typical set-up is presented in Figure 19, where inlet and exit boundaries

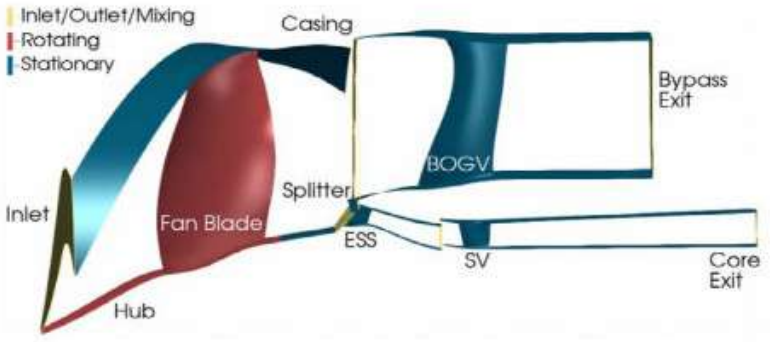


Figure 18: Domain of the UHBR fan (TC3) including downstream components. From (36).

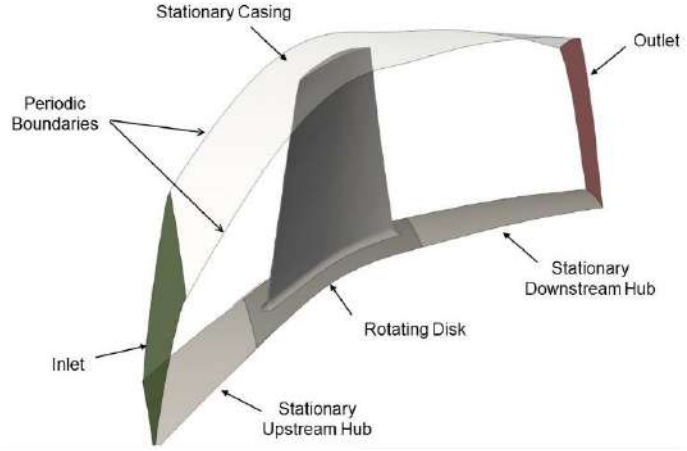


Figure 19: Typical set-up for the isolated fan simulation for the R37. From (26).

are visible, together with stationary and rotating parts. The periodic surfaces are not shown for clarity, but are placed at either side of the domain, in the circumferential direction.

The simulation workflow involves the use of the aforementioned RR in-house software: PADRAM (69) and HYDRA (33). As previously described in section 2.3.1, PADRAM is capable of creating a high-quality, highly smoothed multi-block structured mesh of numerous turbomachinery components, as well as providing a rich set of parameters to modify the design of a component. For this testcase, a multi-block structured mesh consisting of an O-mesh around the blade and H blocks for the upstream, passage and downstream domains is constructed in a rapid and robust process, thanks to the use of templates adapted to the different engine components. The tip gap was meshed using a butterfly topology in order to limit cell skewness compared to other possible options (H-block, polar mesh). A view of the mesh on a blade-to-blade surface, in the leading edge region, is shown in Figure 20.

The in-house Rolls-Royce flow solver HYDRA (for more information, see section 2.3.1) is used to solve the steady RANS equations, with the Spalart-Allmaras turbulence model, given its proven reliability for this type of compressor flows. Total pressure, total temperature and turbulent intensity radial distributions are imposed at the inlet along with fixed velocity angle (axial inlet velocity). An exit capacity boundary condition (modelling the presence of a throttled valve) is applied at the exit. No wall function was used, as the boundary layer is resolved by clustering the mesh towards the viscous walls.

A compressor characteristic with 3 levels of mesh refinement is presented in Figure 21. Pressure ratio (PR) and efficiency (EFF) are shown as a function of non-dimensional corrected mass-flow, which

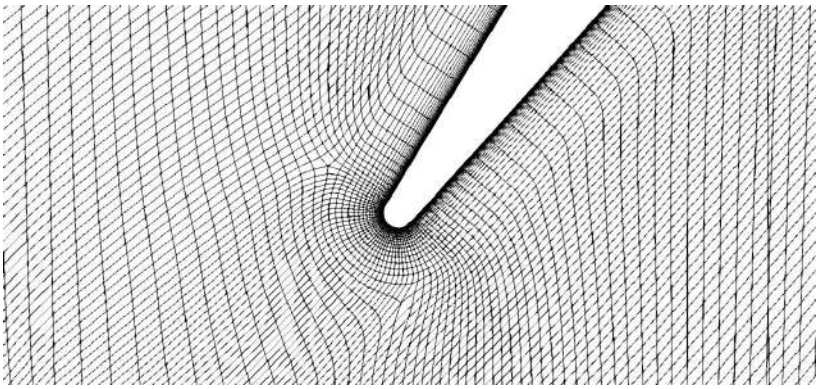


Figure 20: Close up view of the mesh in the leading edge region

is the ratio between the actual mass-flow and the value at choking, compared to the experimental values (15).

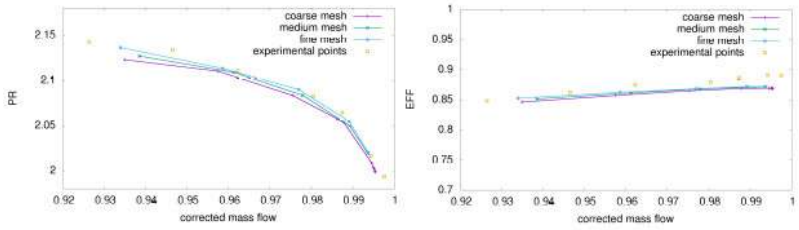


Figure 21: Simulated R37 characteristic compared to experimental data

Single-Passage, Single Stage UHBR Fan Testcase (TC3)

After an initial validation with a public geometry of the lowest-fidelity domain possible, the same level of fidelity is applied to the UHBR Fan Testcase (TC3). The only difference in the domain (in terms of stage components) is that, as opposed to the NASA R37 testcase, a splitter that diverts the airflow into the bypass outlet guide vane (BOGV) and the engine core stages is present. For meshing the CFD domain, a multi-block structured mesh consisting of an O-mesh around the blade and H blocks for the upstream, passage and downstream domains is used; and a butterfly topology for the tip gap block. A C-topology is used in the meridional plane for meshing the splitter.

For the CFD solver, the same setup described in 2.3.2 is used. HYDRA is configured to solve the steady RANS equations, with the Spalart-Allmaras turbulence model. The applied boundary conditions at the inlet and exit are subsonic non-reflective flow, where total pressure, total temperature and turbulent intensity radial distributions are imposed at the inlet along with fixed velocity angle (axial inlet velocity), and the subsonic outflow is controlled by exit capacity (modelling the presence of a throttled valve).

A visualization of the CFD simulation results along with the domain geometry and mesh is provided below in Figure 22.

Single-Passage, Multi-Stage UHBR Fan Testcase (TC3)

The capability to study the interaction of the fan with neighbouring components to increase the level of fidelity of the aerodynamic simulation is also demonstrated here. The fan geometry is shown in Figure

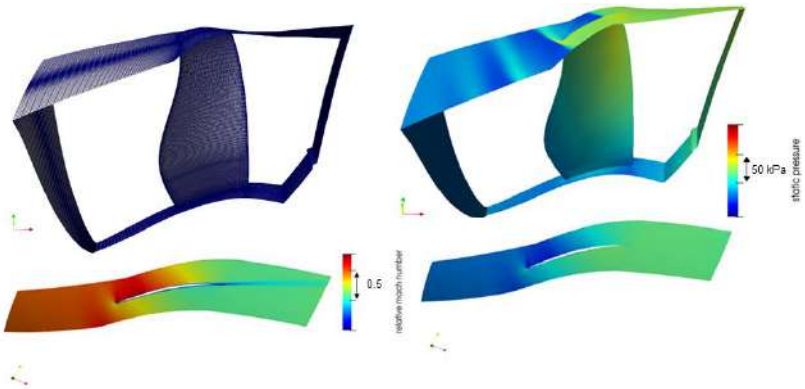


Figure 22: Single-blade, single-passage CFD simulation for the UHBR fan (TC3). Mesh, full domain static pressure field, and cross-sectional views of the static pressure field and relative Mach number. Images distorted and not to scale.

23 and consists of the rotating disk (fan blade and hub), the stationary casing and the splitter, but this time the bypass outlet guide vane (BOGV) and the engine section stator (ESS) are also modelled.

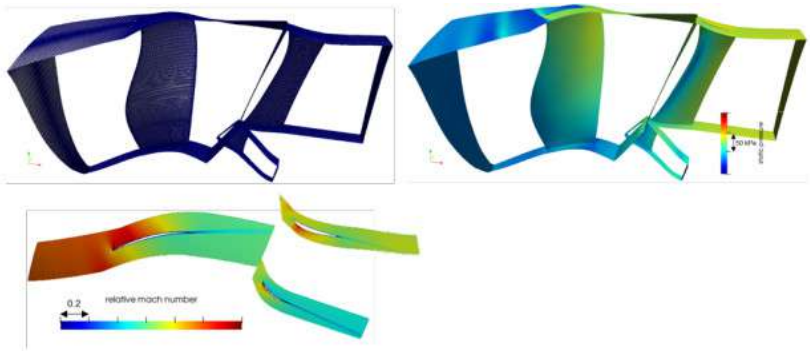


Figure 23: Single-passage, multi-stage UHBR fan (TC3) simulation. Mesh, full domain static pressure field, and cross-sectional views of the relative Mach number. Images distorted and not to scale.

Again, the different domains are meshed using the Rolls-Royce in-house mesher PADRAM, and then coupled through the use of a mixing-plane approach. An H-O-H mesh topology is employed for all blade passages, with the use of H blocks for the upstream and downstream regions. A C-topology is used in the meridional plane for the meshing of the splitter. The y+ value on all viscous walls is adjusted to be around unity for a correct resolution of the boundary layer. The flow solver is the Rolls-Royce in-house software HYDRA, with same boundary conditions and numerical algorithms of the isolated fan simulation, with the addition of the mixing-plane approach to couple the rows of the multi-row machine and the use of the helicity-corrected Spalart-Allmaras model in order to better model the near-stall points.

A visualization of the CFD simulation results along with the domain geometry and mesh is provided in Figure 23.

The validity of the numerical model was assessed running a characteristic from the near-stall to near-choke points for two different engine speeds (95% and 103%) and comparing results against experimental measurements. The comparison is shown in Figure 24 for the total pressure ratio and the total temperature rise ratio of the entire compression system.

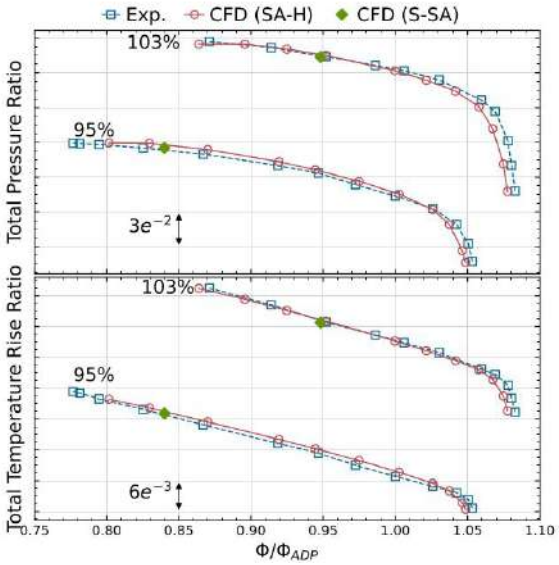


Figure 24: Simulated characteristics for the UHBR fan (TC3) domain, including the interaction of downstream components. From (36)

Multi-Passage, Unsteady UHBR Fan Testcase (TC3)

For the highest level of fidelity presented, an unsteady RANS (U-RANS), whole-annulus run of inverse-mapped, manufactured blades is described. After successful convergence of the inverse-mapping process, PADRAM's capability to mesh an entire row of blades is used, with a resulting mesh of approximately 160 million cells. The same multi-block structured H-O-H topology for the mesh domain, and C-topology for the splitter is used. For this aerodynamic simulation, HYDRA is initialized with the flow field of a previously converged steady-state solution, and three levels of multi-grid are used to ease the convergence to a solution with an explicit scheme solver. The sliding plane approach is applied to the inlet and exit boundaries of the fan stage. Periodic boundaries are used at either side of the blade domain, and a y+ of less than one is achieved on all viscous walls, both static and stationary. A closeup of the whole-annulus mesh and converged solution is provided in Figure 25.

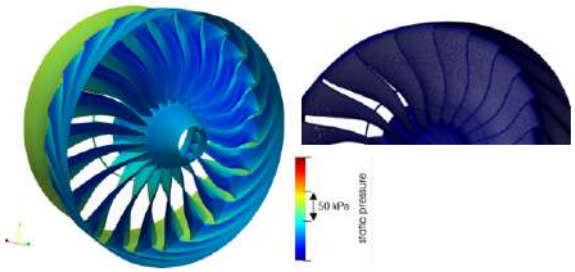


Figure 25: All-annulus fan simulation (TC3).

Modelling real-geometry features

- Elastic rings

In order to prevent aero-structural phenomena that affect UHBR fans (such as flutter), some fan blade models are provided with an elastic ring mounted on the root. New capabilities have been developed within the Rolls-Royce mesher PADRAM to model the gap derived from such configuration. Figure 26 shows a three-dimensional view of blade gap meshed thanks to the new PADRAM capability. The mesh inside the gap is correctly clustered towards the walls for a correct boundary layer resolution.

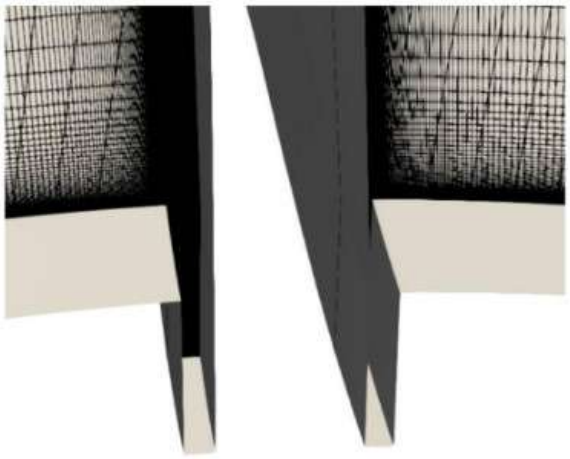


Figure 26: Blade gap meshing capability.

- Simulation of Vortex Generators (VG) inside the Blade Passage

  Secondary flows are an important source of losses in turbomachinery, as well as of instabilities. VGs can be used to alter secondary flows, thus the capability of simulating a blade domain with a VG, as well as of modifying their shape and relative placement is of great importance. To simulate blade passages with VGs, the overset/Chimera capability available in HYDRA has been used. The blade passage (without the VG) and a VG domain are meshed separately with PADRAM, and then coupled through the overset/Chimera capability available in HYDRA. A view of the surface mesh for the VG and the VG domain (red) inserted in the blade passage domain (blue) are shown in Figure 27. This approach allows high quality meshes to be generated

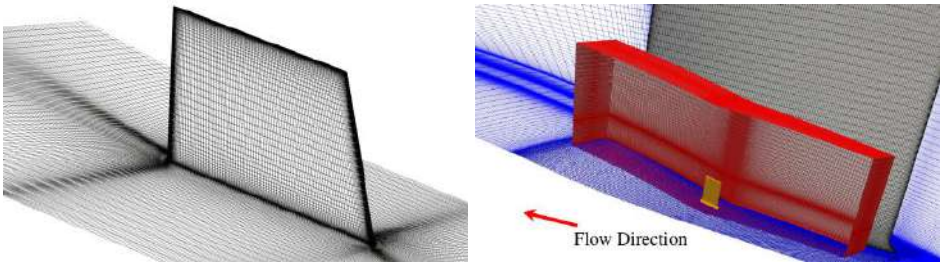


Figure 27: VG surface mesh and VG domain coupled with the passage domain.

separately for the 2 domains, independently of the blade and VG geometries, lending itself well to the use in optimization (WP4). Figure 28 presents the limited streamlines (useful to visualize the presence of crossflow, an important secondary flow feature) on the hub as modified by the presence of a VG.

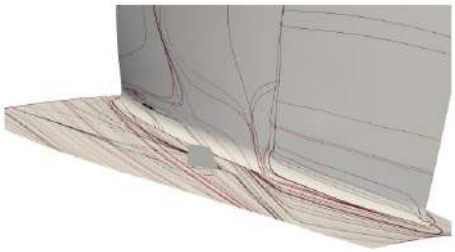


Figure 28: Limit streamlines at the hub for a case with (black) and without VG (red).

- Modelling and simulation of real, manufactured UHBR fan stages with manufacturing deviations

  A graphical user interface (GUI) to run inverse mapping processes has been recently developed at Rolls-Royce. An inverse mapping process is the process of producing a digital 3D CAD model of a real, manufactured blade. In other words, it is to 'reverse-engineer' or 'map' the geometry of an already manufactured blade, that comes in the form of Coordinate Measuring Machine (CMM) data points, measured at the factory with a touch-trigger probe, into a digital CAD file. This concept is illustrated in Figure 29. The resulting measurements and CAD file can then be used to determine the geometrical deviation of the manufactured blade from the 'design intent' blade. Also, CFD can be run upon it to analyse if and how the manufacturing defects or deviations from the design intent (within the manufacturing tolerances), are affecting the blades' and overall engine fan performance.

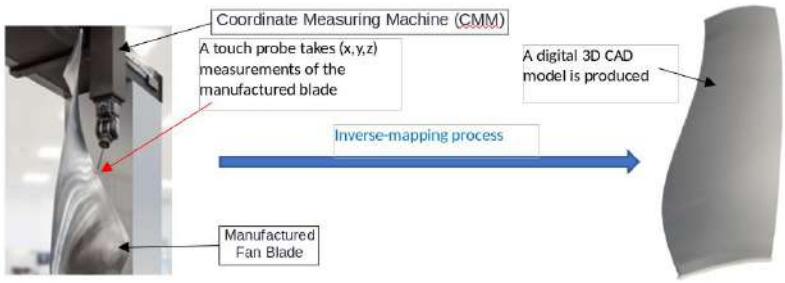


Figure 29: Blade scanning with CMM machine.

For this, to obtain the digital CAD model of an existing blade, an optimization process presented in (68) is used where PADRAM, P2S, and SOFT are run to try to morph or shape the existing 'design intent, nominal' blade geometry into the actual 'manufactured' blade geometry (represented by the CMM datapoints taken at the factory, which is the 'target' of the optimization). After convergence, this process results in the wanted CAD file.

For this, PADRAM, P2S and SOFT are run sequentially with different optimization stages, each modifying different degrees of freedom of the blade shape, like lean, sweep, skew, thickness, or leading and trailing edge angle adjustments; all in an attempt to bring down the cost function, which is the aggregate of the distance field between the 'design intent, nominal' geometry and the actual manufactured geometry, calculated by P2S.

This process of running PADRAM, P2S and SOFT is governed and automated through Python scripts in the source code of the GUI.

After successful completion of the process, distance-field contour maps can be produced to assess the matching accuracy of the 3D CAD model with the CMM datapoints taken at the

factory. An example of these distance-field contour maps is shown in Figure 30, where the CMM data colour changes according to its proximity to the morphed blade. Before the process is run, and the blade has not been modified yet (i.e., the CAD corresponds to the design intent shape), the CMM data shows extreme red and blue colours, corresponding to high distance values. However, after the process is finished, these colours have changed to green, indicating that the distance between the surface of the generated CAD model and the CMM datapoints is near zero.

Moreover, to measure the aforementioned geometrical agreement, detailed inspection of critical zones of the geometry that affect fan performance (like blade leading edge and trailing edge) can also be conducted by zooming into the CAD model at the area of interest.

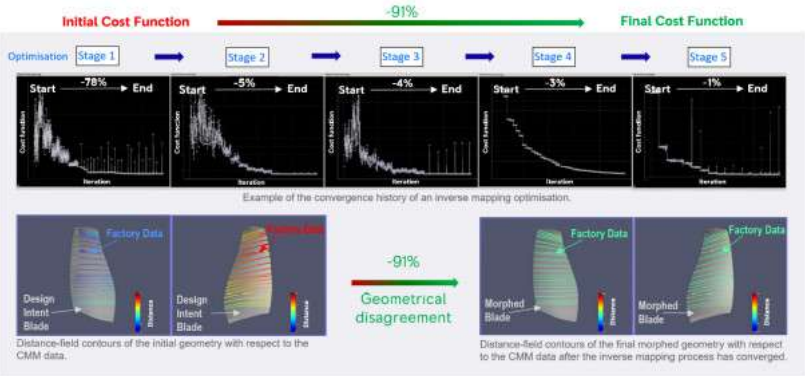


Figure 30: Example of convergence and results of an inverse mapping run. Fan blade images distorted and not to scale.

Thanks to the PADRAM Design Space (69), manufacturing variations from design intent can also be quantified for each blade. Deviations of design and aerodynamic parameters of interest can be plotted radially and analysed to establish correlation between fan increase or decrease in performance, and its geometrical variability. The aerodynamic performance of the fan can come either from tested experimental data or from CFD simulations. By way of example, Figures 31, 32 and 33 provide radial distributions of design parameters of interest like sweep, lean, skew, leading and trailing edge recambering, and blade metal angles (inlet, exit and stagger) as well as other features of interest that can influence the aerodynamic performance of the fan like the passage throat area or magnitude and location of the blade's maximum thickness.

2.3.3 UHBR Fan Structural Simulation Capabilities

The capabilities for FEM (structural) simulation of a fan blade have been developed and tested starting from the process that was previously developed by UNICA for turbine blades. The present capability is fully automated and ready to be used inside a multi-disciplinary optimization process for the fan blade.

The structural simulation must be performed only after a preliminary aerodynamic CFD simulation has been carried out with a PADRAM-HYDRA setup starting from a hot-conditions BDF. Before performing the structural run, the blade and hub surfaces are exported from a PADRAM setup based on a cold-conditions BDF. The geometric definition of the fan blade is completed by adding the root and firtree parts from a CAD file. The different parts are imported into BOXER where a meshing run is performed to generate the solid computational grid. The meshing process takes about 30 minutes on 24 cores on one of Rolls-Royce's High-Performance Computers (HPC). After meshing, the solid grid is imported into Ansys ICEM (24) to perform a format change, which makes the mesh readable for the Rolls-Royce FEM solver SC03 (2).

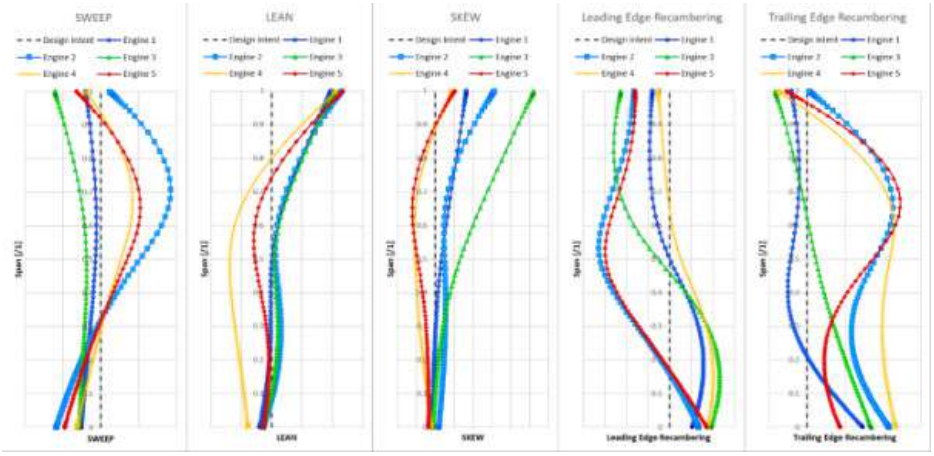


Figure 31: Example of averaged radial distribution of aerodynamic shape parameters of interest for the fans of 5 different manufactured engines. As in Figure 32, the smooth distribution is achieved by plotting the interpolation of 5 PADRAM Engineering Parameters across the streamline sections with a 3rd order Bézier spline.

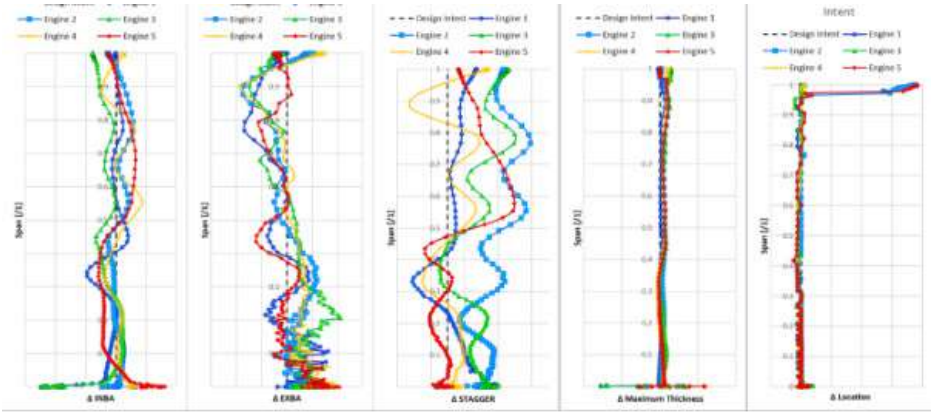


Figure 32: Example of averaged PADRAM Design Space Engineering Parameters radial distribution for the fans of 5 different manufactured engines. The smooth distribution across streamline sections is achieved with a 3rd order Bézier spline interpolation of the PADRAM Engineering Parameters given at 5 control points.

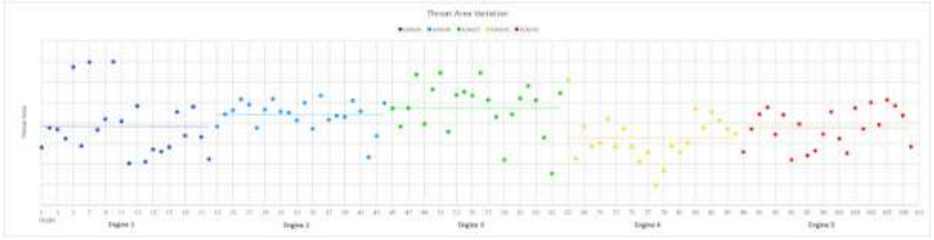


Figure 33: Throat area variation for every blade of 5 different manufactured engines. In dashed lines, the arithmetic mean of the throat area values for that particular engine is represented.

Figure 34 shows the complete structural mesh for the fan. The static pressure field from the preliminary CFD simulation is applied on the 'BLADE' surface. The 'FIRTREE' surface is where the fixed

constraint boundary condition is applied. The structural solver (SC03) reads the mesh and picks up the static pressure field on the fan surface from the converged PADRAM-HYDRA CFD simulation. This static pressure field is extracted from the CFD solution at 20 different heights along the span of the blade as shown in Figure 34. Being that the pressure field is extracted from a hot-conditions geometry, while the structural run needs to be performed on a cold-conditions one, the pressure values are automatically interpolated from the former to the latter prior to being used as a boundary condition for the structural simulation. After the boundary conditions are applied, including the rotational speed value, the fan material properties are imported into SC03 and the structural run is performed. The FEM simulation takes about another 30 minutes on 24 computer cores in one of Rolls-Royce's high-performance computer (HPC) visualization nodes. A script is available for the simulation to be submitted to Rolls-Royce's HPC systems' computing queue and thus performed on multiple nodes. This allows for a significant speed up and for parallel submission of multiple cases. Figure 35 shows the results from a FEM simulation on the fan in terms of Von Mises Stresses.

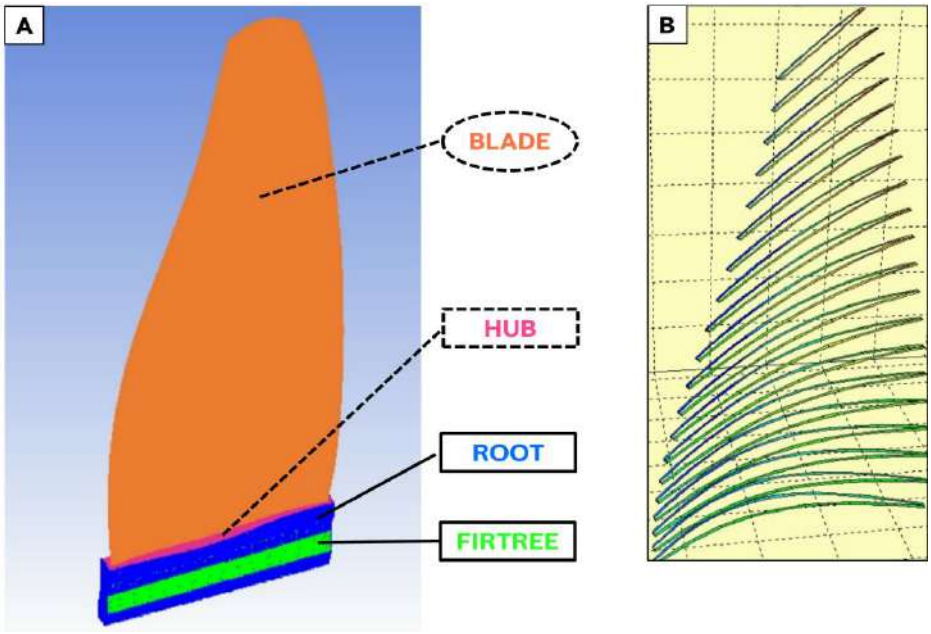


Figure 34: Fan solid mesh for structural simulation (A) and pressure boundary condition for the "BLADE" component extracted from CFD (B). The components of the geometry indicated with dashed lines in (A) have been exported from the PADRAM-HYDRA setup. Pictures distorted.

The overall process can be described in the following workflow (Figure 36):

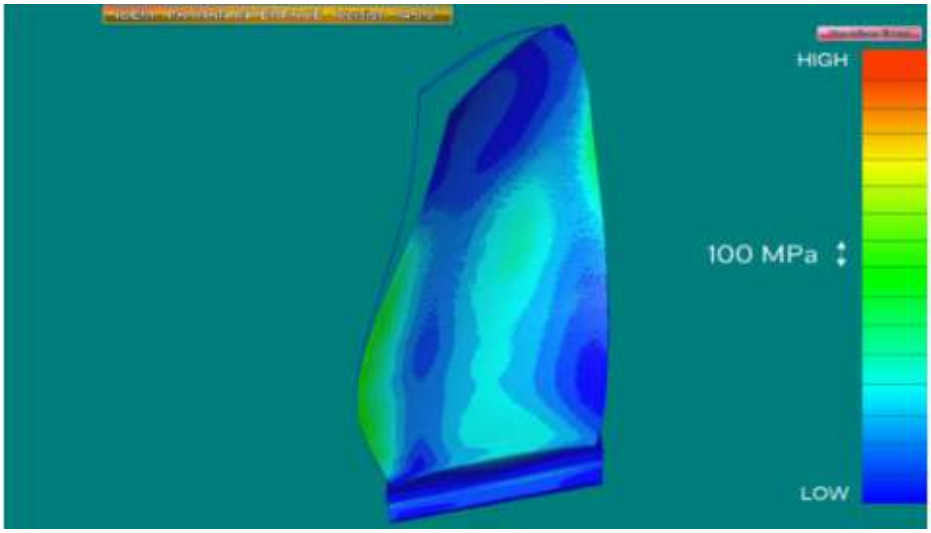


Figure 35: FEM simulation results in terms of Von Mises Stresses (MPa). Contours are being displayed on the resulting deformed geometry of the fan blade. The undeformed shape is also shown with a blue outline. Deformation is multiplied by a scaling factor. Picture distorted.

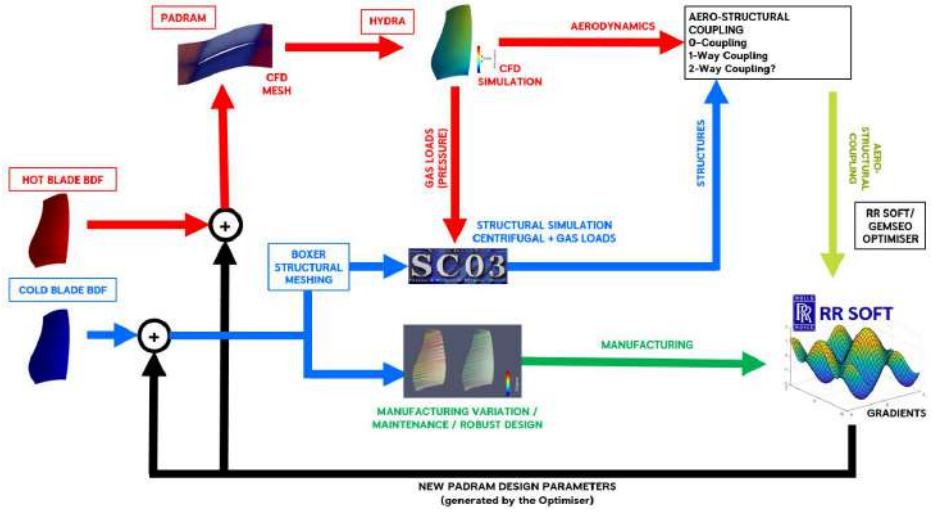


Figure 36: Aero-structural UHBR Fan (TC3) multi-disciplinary optimization (MDO) workflow.

3 Geometry & Mesh Adaptation for Laminar Wing Design

For the sake of completeness and due to strong overlapping between the two topics (transition modeling and its adjoint on the one hand, grid adaptation on the other) it was decided to include, in this report too, the material presented by INRIA in D1.2 too. On the other hand, there are some results in the part presented by DAV which include results (such as Fig. 47) from collaborative work with INRIA.

The objective of the study carried out by INRIA is the assessment of an anisotropic mesh-converged solution for the SA-neg-noft2-BCM transitional model. The mesh adaptation process is performed using the mesh-adaptive solution platform composed of WOLF, the anisotropic local remeshing software FEFLO.A (37, 38, 39) and the field interpolator INTERPOL (1).

Two cases are considered.

- 2D Zero-Pressure-Gradient Flat Plate run with the a featured-based (FB) error estimator;
- 2D subsonic flow past the NLF4016 airfoil run with the a goal-oriented (GO) error estimator.

3.1 2D Zero-Pressure-Gradient Flat Plate

The first test case considered is the subsonic flow over a 2D flat plate with zero pressure gradient. The flow conditions are: Mach number $M = 0.1443$ and Reynolds number $Re = 3.34 \times 10^6$, with the latter based on the unit flat plate length. The turbulence intensity specified at free-stream is $Tu_\infty = 0.18\%$. This case corresponds to the Schubauer and Klebanoff (64) and was already used to calibrate the BCM-model (51). The FB error estimator considered is based on the L^4 norm of the interpolation error of the local Mach number.

3.1.1 Assessment of an anisotropic metric-based mesh-converged solution with FB error estimator

Figure 37 shows on the left, results from a 500-vertices to a $80\,000$-vertices adapted mesh: (i) the mesh-convergence of the drag coefficient C_D for the fully turbulent calculation (blue curve); (ii) the mesh-convergence of the drag coefficient C_D for the transitional calculation (red curve).

First, the fully turbulent RANS adaptive process is performed. Second, the solution computed on a 2000-vertices adapted mesh is used to initialize the transitional calculation. We can see that both calculations are converged at $20\,000$ vertices. Figure 37 on the right shows the comparison between the two skin-friction coefficients on a $20\,000$ adapted mesh. Transition occurs shortly before $Re_x = 3 \times 10^6$, according to literature.

3.1.2 Transition point mesh-convergence

Our main objective was to assess mesh-convergence solution for the BCM-model, specifically in terms of the position of the transition point. Figure 37 on the right shows that, during the adaptation process, the transition point does not oscillate but converge to a specific position. This means that this model is robust and guarantees mesh-convergence solutions. Particularly, the reader can note that the transition points does not change anymore since $20\,000$ vertices.

The resulting adapted meshes are shown in Figure 38. On the left, adapted mesh and solution for the SA model. On the right, adapted mesh and solution for the SA-BCM model. Both meshes are stretched along the wall-normal direction by a factor 20, to highlight the leading edge and the transitional region. We can see that the adaptation process captures automatically all discontinuities in the flow. Specifically, the leading edge is a discontinuity in terms of boundary condition. The transition point is a discontinuity in the solution, when passing from laminar to turbulent flow.
A zoom of the transition point is shown in Figure 39. Both meshes are stretched along the wall-normal direction by a factor 5, to highlight the refinement at the transition point. We can see in detail as the

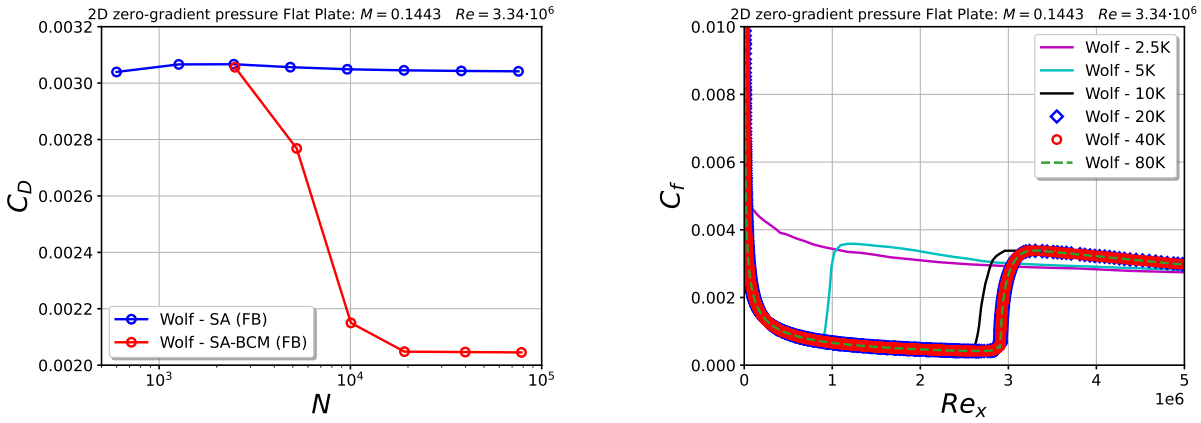


Figure 37: 2D Zero-Pressure-Gradient Flat Plate FB anisotropic metric-based mesh adaptation. Left: volution of the Drag C_D coefficient during the mesh-convergence process. N is the number of vertices of the adapted meshes. Right: skin-friction coefficient C_f on a $20\,000$ vertices-adapted meshes. Comparison between fully turbulent calculation and transitional calculation.

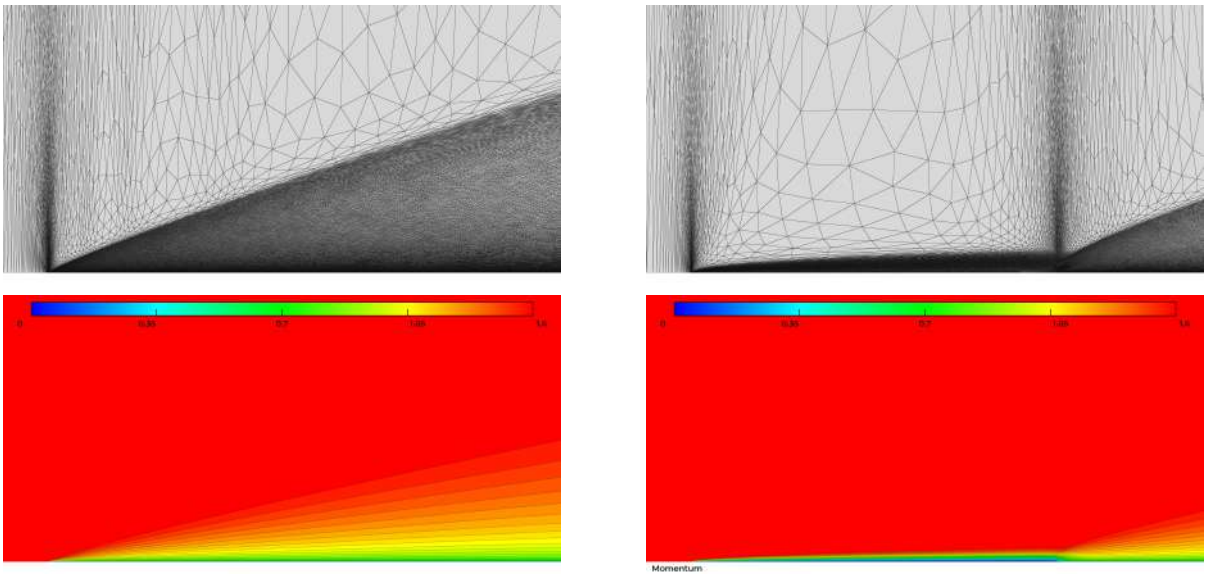


Figure 38: 2D Zero-Pressure-Gradient Flat Plate FB anisotropic metric-based mesh adaptation. Left: adapted mesh and streamwise velocity component field for turbulent solution. Right: adapted mesh for and streamwise velocity component field for transitional solution. Both meshes are stretched along the wall-normal direction by a factor 20, to highlight the leading edge and the transitional region.

adaptation process dispenses the proper number of points according to the complexity of the flow. The laminar boundary layer requires much less points w.r.t. the turbulent one, to be correctly captured. Then, we can see the transition to turbulence already in the mesh: the upstream region for the SA-BCM calculation is laminar, velocity gradients are lower and boundary layer is thin. Downstream, velocity gradients are higher and boundary layer is thick. Such physical considerations are well highlighted by the refinement region. In the transition region, adaptation become more isotropic to capture transition. While being highly anisotropic before and after the transition region.

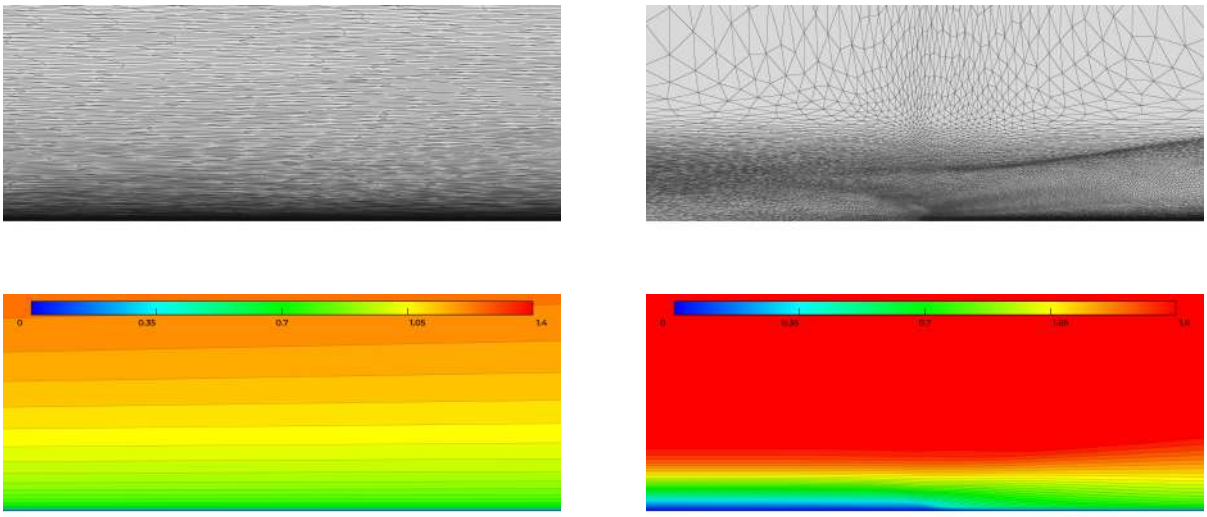


Figure 39: 2D Zero-Pressure-Gradient Flat Plate FB anisotropic metric-based mesh adaptation. Left: adapted mesh for turbulent solution. Right: adapted mesh for transitional solution. Both meshes are stretched along the wall-normal direction by a factor 5 with respect to the streamwise direction.

3.2 2D subsonic flow past the NLF4016 airfoil

The case considered is the flow around the NLF4016 airfoil at Mach number $M = 0.1$ and Reynolds number $Re = 4 \times 10^6$ based on the airfoil chord length c. Several experimental data are available for Re ranging from 1×10^6 to 9×10^6, and M from 0.1 to 0.4 (72). In these experiments a specific roughness was opportunely sized for each Reynolds number to fix the transition point at $0.075c$ on both surfaces.

3.2.1 Assessment of an anisotropic metric-based mesh-converged solution with GO error estimator

The NLF4016 airfoil is run with the goal-oriented error estimator using the drag as the targeted functional. The SA-BCM model, with fixed free-stream turbulence intensity Tu_∞ is used to calculate the adapted RANS solution. The free-stream temperature is set at $T_\infty = 300\,\mathrm{K}$ and the free-stream Turbulence Intensity at $Tu_\infty = 0.15\,\%$, according to the 1st AIAA Transition Modeling Workshop. However, it should be stressed that an-induced transition by roughness is totally different by a by-pass one induced by a fixed free-stream turbulence. Therefore, there is no guarantee that the transition point is the same as that measured in the experiments.

As previously done for the 2D flate plate calculation, the adaptive process for the SA-BCM model is initialized with a corresponding SA adapted solution at $10\,000$ vertices. The process is stopped when reaching a $300\,000$ vertices adapted mesh. A polar from 0° to 5° Angle of Attack each 1° is investigated. The mesh and the solution, for AoA = 0° at $30\,000$ vertices for the SA and the SA-BCM model are shown in Figure 40. We can see that also in this case, the adaptive process manages to adapt the mesh to the transition from laminar to turbulent flow.

The drag coefficient C_D evolution of SA and $SA - BCM$ calculations during the adaptive process is shown in Figure 41. We can see that both calculations are mesh-converged at $150\,000$ vertices.

3.2.2 Transition point mesh-convergence

Figure 42 shows the convergence of the transition point during the adaptation process. As for the 2D flat plate, the transition point converges to a specific location and it does not change anymore since the $150\,000$ vertices-adapted mesh. This mesh-convergence is achieved for all the AoAs investigated.

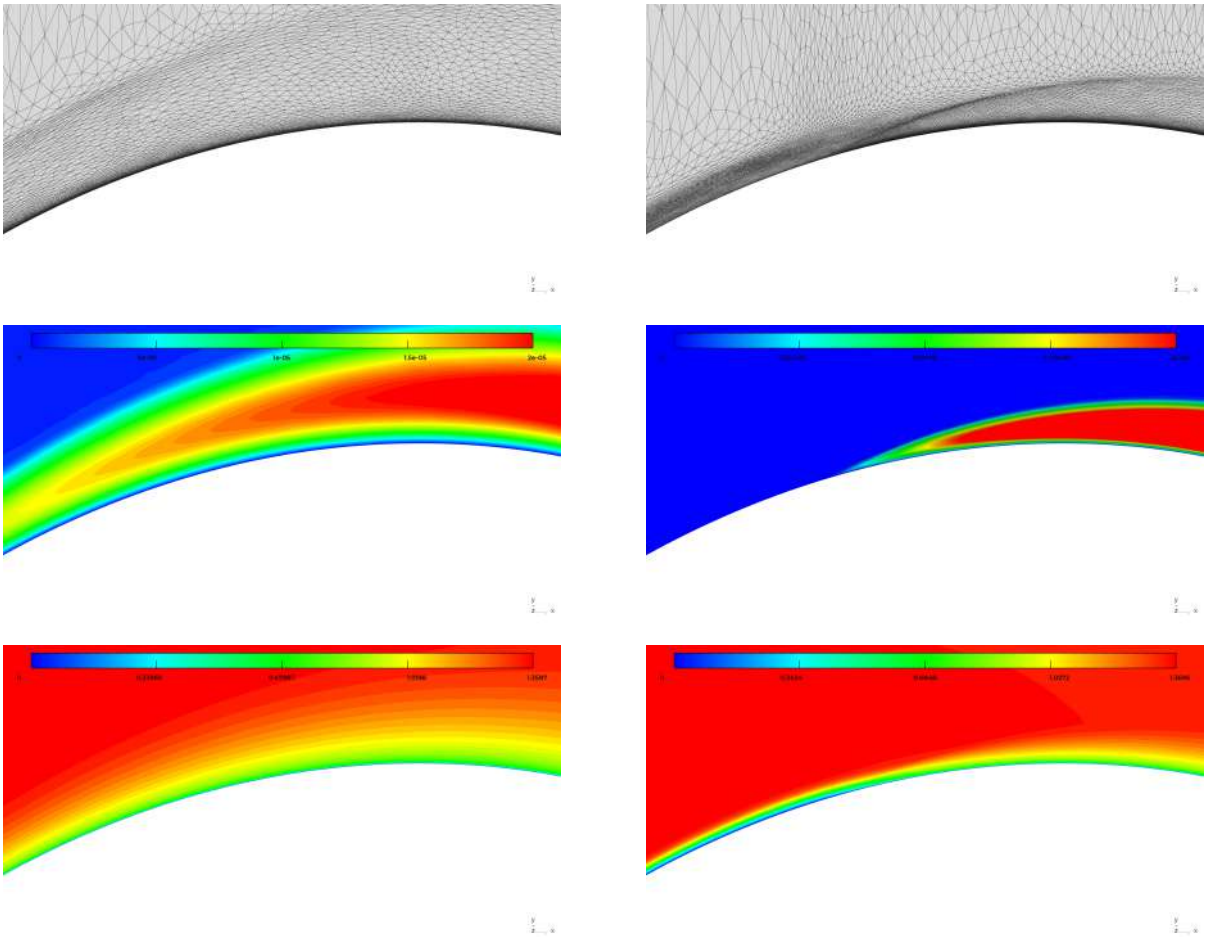


Figure 40: 2D flow past the NLF4016 airfoil at AoA = 0° using GO anisotropic metric-based mesh adaptation for $30\,000$ vertices. Adapted mesh for the fully turbulent solution (left). Adapted mesh for the transitional solution (right) . Pseudo-viscosity field $\tilde{\nu}$ (middle) and streamwise velocity component field (bottom). Both meshes are stretched along the wall-normal direction by a factor 5 with respect to the streamwise direction.

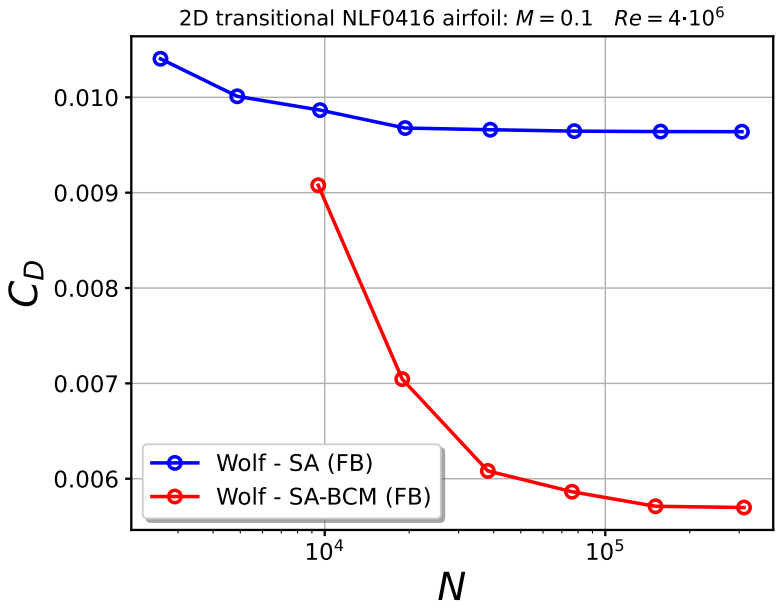


Figure 41: 2D flow past the NLF4016 airfoil at AoA = $0°$. GO anisotropic metric-based mesh adaptation. Evolution of the drag coefficient C_D during the adaptation process. Comparison between fully turbulent calculation (blue curve) and transitional calculation (red curve).

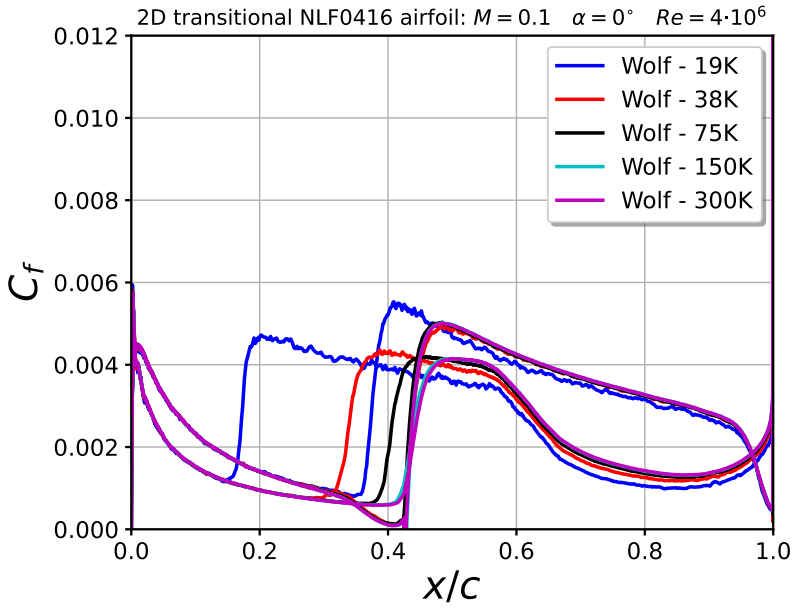


Figure 42: 2D flow past the NLF4016 airfoil at AoA = $0°$. GO anisotropic metric-based mesh adaptation. Evolution of the skin-friction coefficient C_f during the adaptation process. The solution is mesh-converged at the $150\,000$ vertices adapted mesh.

3.2.3 Adapted calculations at different Angles of Attack

The adaptive calculations at different Angles of Attack (AoAs) are shown in Figure 43. In order to accelerate the convergence, each adaptive simulation is initialized at its corresponding adapted SA solution at $10\,000$ vertices. The process is confirmed to be robust. As expected the drag C_D and lift C_L coefficients increase with increasing AoAs. Specifically, from Figure 44 , we can observe the transition point shifting towards upstream locations on the suction side. This clearly means that transition occurs earlier at higher AoAs.

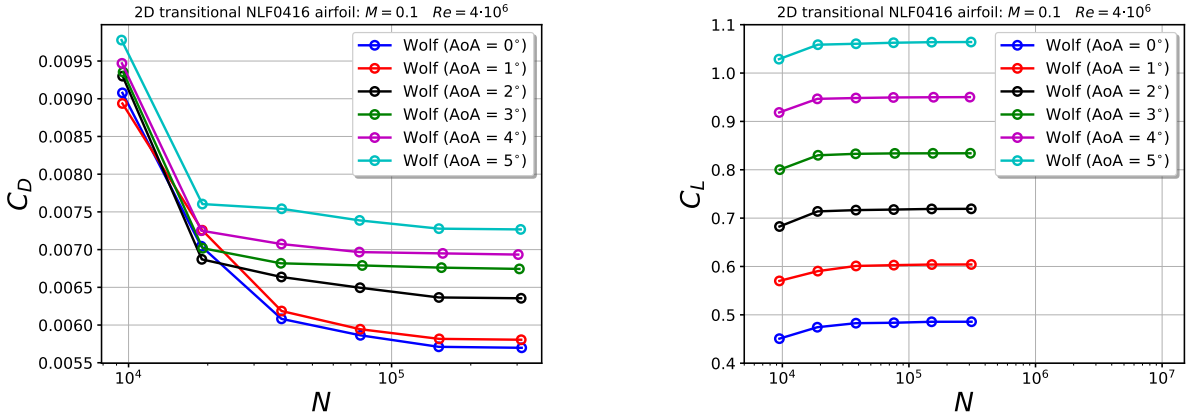


Figure 43: 2D flow past the NLF4016 airfoil at different Angles of Attack. GO anisotropic metric-based mesh adaptation. Evolution of the drag C_D (left) and the lift C_L (right) coefficients, during the adaptation process. N is the number of vertices of the adapted meshes.

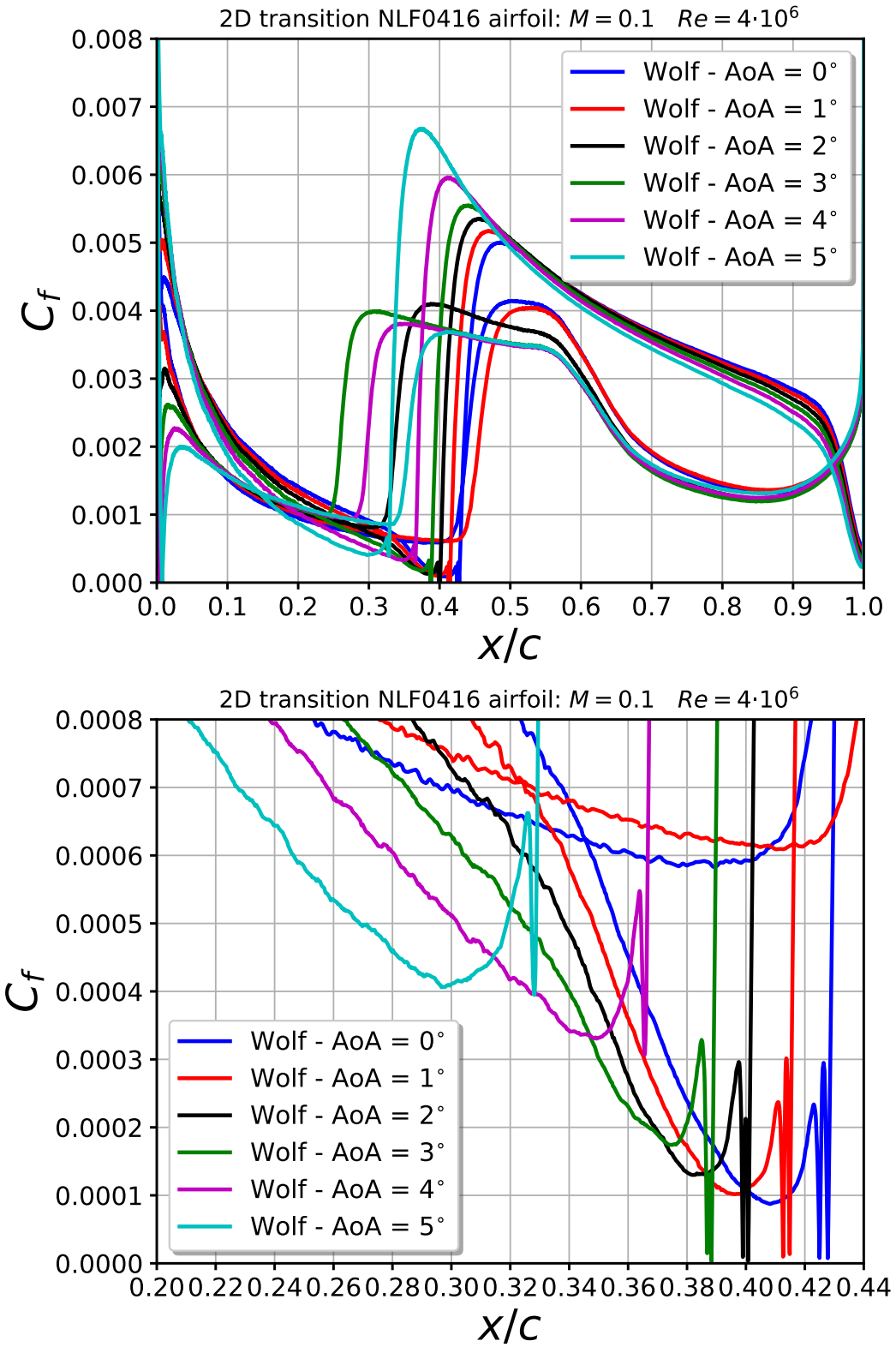


Figure 44: 2D NLF0416 GO anisotropic metric-based mesh adaptation. Evolution of the skin-friction coefficient C_f with the Angle of Attack AoA. The adapted meshes are composed of about $300\,000$ vertices.

3.3 DAV Contribution

In order to reduce pressure fluctuations which have a negative impact on the transition onset of the boundary layer, it is important to ensure a smooth curvature distribution. During the optimization process a set of NURBS control points are chosen as local design variables for the shape definition at each iteration. The current CAD model definition is at least C^2 continuous at each control point. However to minimize potentially induced pressure fluctuations during the optimization process and enhance both convergence and shape quality requirements for laminar flow, the use of a smoothing operator taking into account higher order continuity, i.e. beyond C^2, is an important asset. Figure 45-left shows an example of the curvature distribution of various wing profiles as a function of their NURBS control point index which defines the shape starting from the trailing edge lower side through a maximum at the leading edge along to the trailing edge upper side. In this figure a given initial profile is discontinuous at control point index 11. The figure illustrates a possible new distribution while applying a smoothing operator seeking to reduce the overall lattice curvature level while maintaining C2 continuity and the same operator maintaining a quasi G3 continuity at each control point. Figure 45-right shows the corresponding impact on the pressure level distribution as a function of the chordwise position and the corresponding control point index has also been indicated. It shows the beneficial impact regarding the pressure distribution near the leading edge of using a higher order continuity smoothing operator.

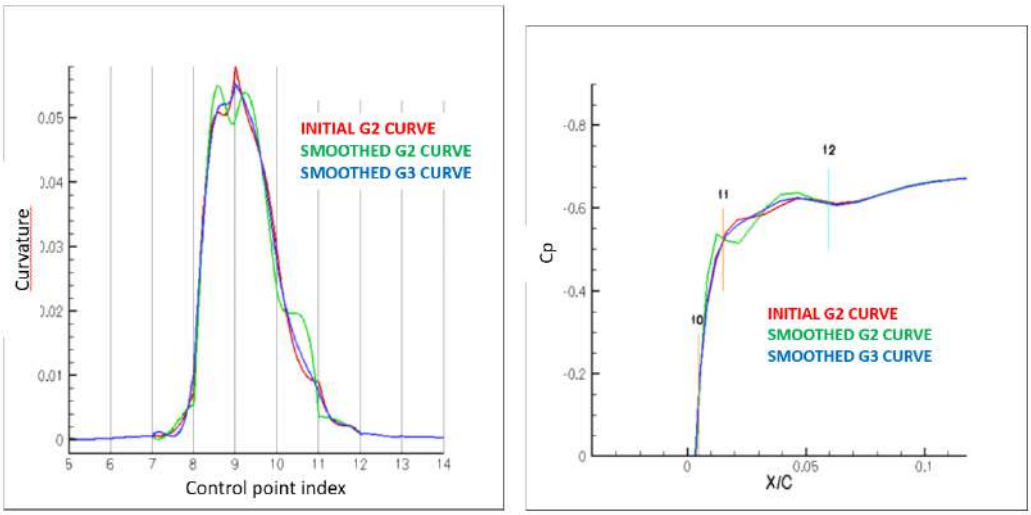


Figure 45: Left: example of wing profile curvature distribution as a function of control point index. Right: example of chordwise wing profile pressure coefficient with corresponding control point index positions.

3.3.1 DAV – INRIA mesh adaptation

The aim of this coollaborative work was to demonstrate Riemannian mesh adaptation on RANS with transitional model. The primal code is AETHER (DAV RANS code) and our choice for modelization is SA-noft2-Gamma-Retheta Transition Model (see also deliverable report D1.2, *"Assessment of transition models and their adjoints"*). The FB error estimator (INRIA WOLF code) considered is the L4 norm of the interpolation error of the local Mach number. Our first test case deals with flat-plate. The Schubauer and Klebanoff flat plate experiment is a useful validation test case for transition models. This case has a low freestream turbulence intensity and corresponds to natural transition. Figure 46 gives the history of convergence for skin-friction coefficient C_f.

Black color deals with NASA fine mesh (545 x 385 = 209825 vertices). Red color deals with the starting mesh (very coarse, 875 vertices)). Blue color deals with intermediate meshes. Green color deals with

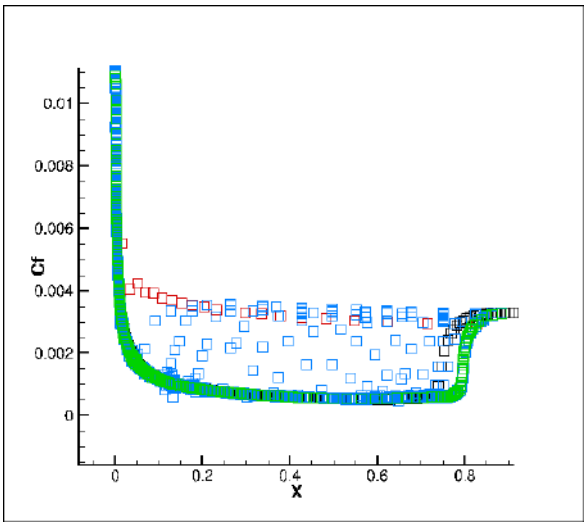


Figure 46: Convergence for skin-friction Coefficient C_f.

the results of mesh adaptation convergence (iteration 18, 4888 vertices). Mesh convergence process consists of iterations with same complexity (same number of vertices) and increase of complexity, as shown in Figure 47. Converged result for complexity 2500 is close to NASA fine mesh in terms of C_f (and so, in terms of transition point).

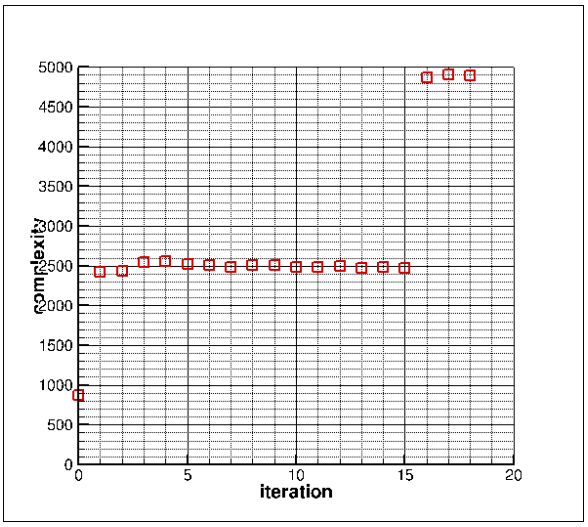


Figure 47: Complexity of the mesh adaptation process.

Results for starting mesh, for iteration 15, for iteration 18 and results for NASA fine mesh are presented in figures 48, 49, 50 and 51, respectively. It is concluded that Riemannian mesh adaptation led us both to a better for aerodynamics solution quality and a dramatically gain in terms of number of vertices.

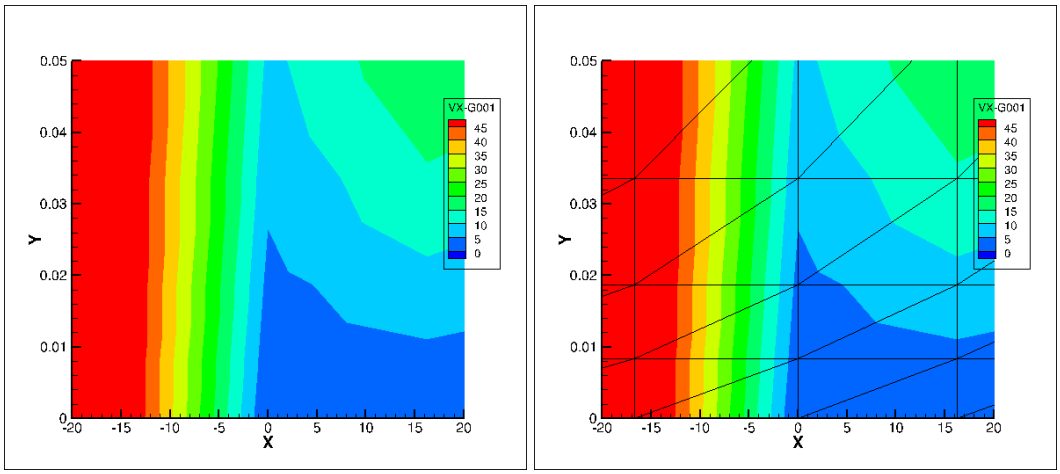


Figure 48: Results for starting mesh.

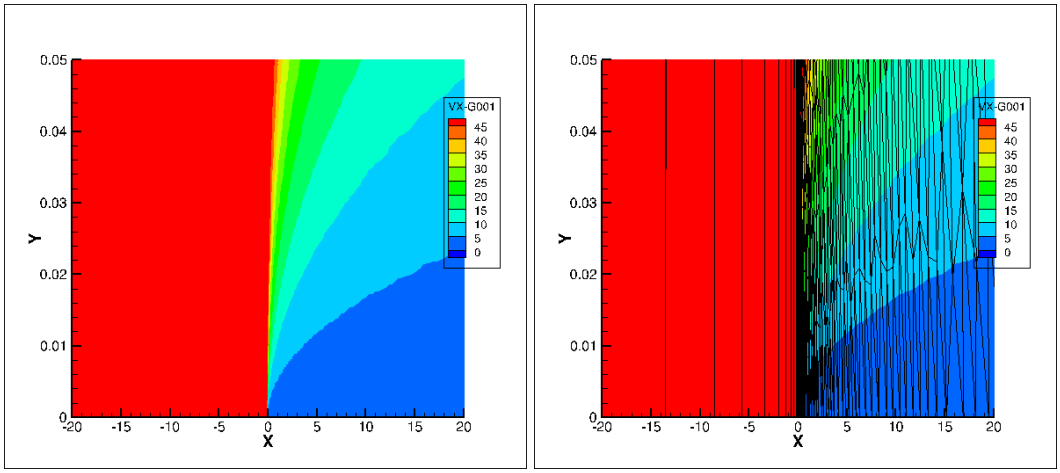


Figure 49: Results for iteration 15.

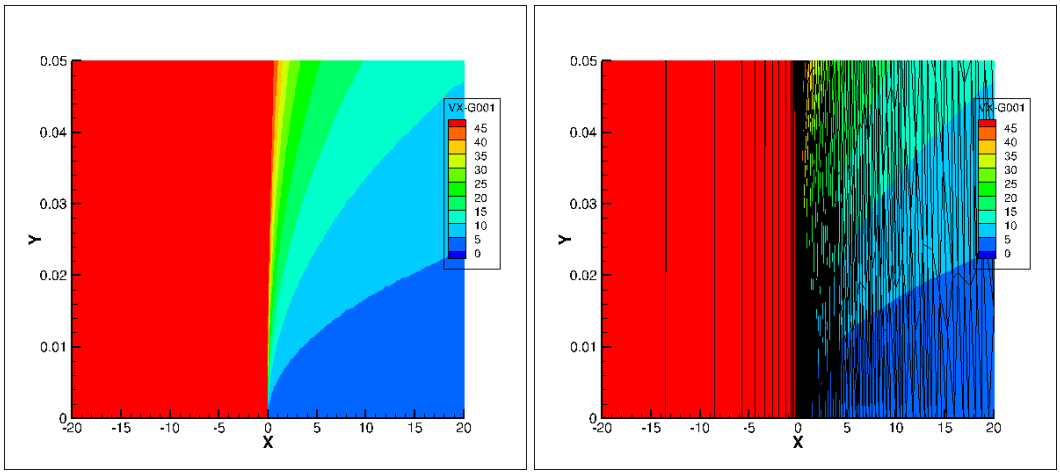


Figure 50: Results for iteration 18.

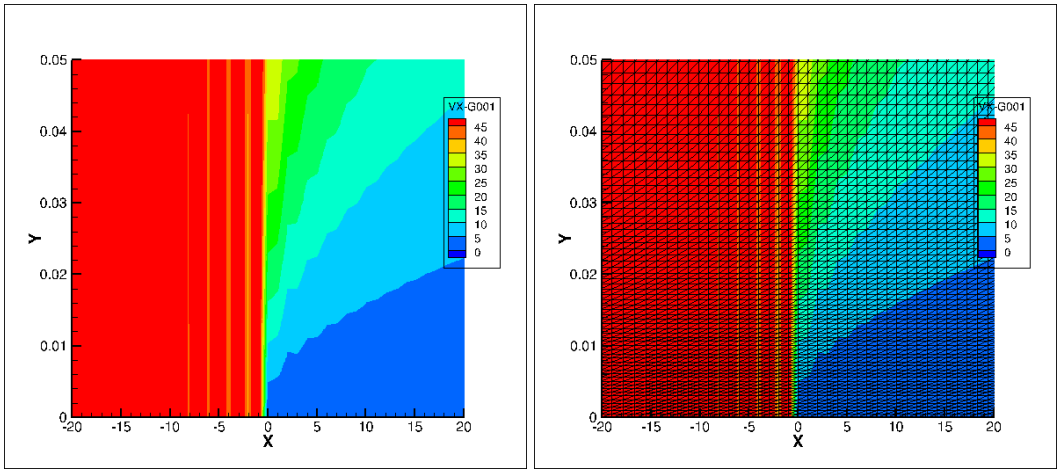


Figure 51: Results for NASA fine mesh.

4 Adjoint-based Computation of Pareto Fronts

4.1 Efficient adjoint-based Pareto tracing algorithms

NTUA and FOSS worked in tandem on the extension and further development of a gradient-based Pareto tracing algorithm supported by the adjoint method. This algorithm is put in the form of a prediction-correction scheme. Once a first point that belongs to the Pareto front has been found, the algorithm moves from one Pareto point to the next by considering the local curvature of the front. To do so, a constrained minimization problem is defined and solved, where equality constraints are imposed to all but one objective function which is to be minimized. The objective functions-related equality constraints are user-defined values. Formulating the Karush-Kuhn-Tucker (KKT) conditions and satisfying the requirements of the implicit function theorem results in a linear system, the solution of which computes the new point which might slightly deviate from the Pareto front due to curvature effects, (63). A correction step follows so as to bring the predicted point onto the Pareto front. Any constrained optimization solver can be used as corrector; herein a Sequential Least Squares Programming (SLSQP) algorithm, (29) is used.

The prediction-correction scheme is used together with the NTUA in-house GPU-accelerated PUMA code which solves the flow and adjoint equations, (28). For the needs of the NEXTAIR project, the RANS equations for compressible fluids coupled with the one-equation Spalart-Allmaras (SA) turbulence model (73) and the two equation $\gamma - \tilde{R}e_{\theta t}$ transition model, (32, 8) (when required), are used. The continuous adjoint of PUMA includes the adjoint to the turbulence and transition models as well as the adjoint to the Eikonal equation computing distances from the walls. By doing so, the method computes accurate gradients which are highly needed by the Pareto front tracing technique.

4.1.1 Problem Formulation

A MOO problem with M objectives to be minimized subject to M_e equality and M_{ie} inequality constraints reads

$$\begin{aligned} \min\ & \boldsymbol{f}(\boldsymbol{b}) = \min\{f_1(\boldsymbol{b}), ..., f_M(\boldsymbol{b})\} \\ \text{s.t. } & \boldsymbol{c}(\boldsymbol{b}) = 0 \\ & \boldsymbol{g}(\boldsymbol{b}) \leq 0 \end{aligned} \tag{4.1.1}$$

where $\boldsymbol{b} \in R^N$ is the optimization (or design variables') vector. In what follows, boldfaced characters indicate a vector.

In gradient-based optimization, the computation of a point on a convex Pareto front can be made by minimizing the weighted-sum of the M objectives as

$$\min F_w(\boldsymbol{b}) = \sum_{j=1}^{M} w_j f_j(\boldsymbol{b}) \tag{4.1.2}$$

subject to the constraints of equation 4.1.1, with $w_j \geq 0$ for $j \in [1, M]$. A straightforward manner to populate the front with an adequate number of points, is to repetitively solve problem 4.1.2 using different weight (w) combinations. Such a procedure is computationally expensive, so the prediction-correction scheme, described below, is used instead. In our case, problem 4.1.2 is solved only once, with $w_l = 1$ and $w_j = 0, j \in [1, M], j \neq l$ (any other combination of weights could be used instead) in order to compute a first point on the Pareto front. Alternatively, the first point can be computed by once solving problem 4.1.3 as discussed below. In this case, the user of the method should determine target values for all but one of the objective functions.

4.1.2 Prediction-Correction Scheme

The prediction step starts by re-formulating problem 4.1.1 so as to minimize the first (or any other, instead) objective function while imposing $M-1$ equality constraints (over and above to the problem constraints), namely

$$\begin{aligned} &\min\ f_1(\boldsymbol{b}) \\ &\text{s.t. } f_k(\boldsymbol{b}) = \hat{f}_k \\ &\quad \boldsymbol{c}(\boldsymbol{b}) = 0 \\ &\quad \boldsymbol{g}(\boldsymbol{b}) \leq 0 \end{aligned} \tag{4.1.3}$$

where $\hat{f}_k$ are user-defined values for the remaining k objective functions $\forall k \in [2, M]$. Problem 4.1.3 allows the selection of a specific Pareto point (by appropriately selecting $\hat{f}_k$) provided that the designer knows that such a point exists in the objective space and respects all constraints. On the other side, if there is no a-priori knowledge of the objective space, solving problem 4.1.2 is the only way. The Lagrangian of problem 4.1.3 is

$$\min \mathcal{L}(\boldsymbol{b}, \hat{\boldsymbol{\lambda}}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f_1(\boldsymbol{b}) - \sum_{k=2}^{M} \hat{\lambda}_k (f_k(\boldsymbol{b}) - \hat{f}_k) - \sum_{i=1}^{M_e} \lambda_i c_i(\boldsymbol{b}) - \sum_{j=1}^{M_{ie}} \mu_j g_j(\boldsymbol{b}) \tag{4.1.4}$$

where $\hat{\boldsymbol{\lambda}}$, $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$ are the vectors of the Lagrange multipliers ($\hat{\lambda}_k$, λ_i, μ_j) for the prescribed (target) objective functions and the equality and inequality constraints, respectively. The optimal solution should satisfy the KKT conditions, namely

$$\begin{aligned} &\nabla_{\boldsymbol{b}} \mathcal{L}(\boldsymbol{b}, \hat{\boldsymbol{\lambda}}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \nabla_{\boldsymbol{b}} f_l(\boldsymbol{b}) - \sum_{k=2}^{M} \hat{\lambda}_k \nabla_{\boldsymbol{b}} f_k(\boldsymbol{b}) - \sum_{i=1}^{M_e} \lambda_i \nabla_{\boldsymbol{b}} c_i(\boldsymbol{b}) - \sum_{j=1}^{M_{ie}} \mu_j \nabla_{\boldsymbol{b}} g_j(\boldsymbol{b}) = 0 \\ &f_k(\boldsymbol{b}) - \hat{f}_k = 0\ ,\ \forall k \in [2, M] \\ &c_i(\boldsymbol{b}) = 0\ ,\ \forall i \in [1, M_e] \\ &g_j(\boldsymbol{b}) \leq 0\ ,\ \forall j \in [1, M_{ie}] \\ &\mu_j g_j(\boldsymbol{b}) = 0\ ,\ \forall j \in [1, M_{ie}] \\ &\mu_j \leq 0\ ,\ \forall j \in [1, M_{ie}] \end{aligned} \tag{4.1.5}$$

The first four equations of the above system can be re-written as

$$\boldsymbol{H}(\boldsymbol{b}, \hat{\boldsymbol{\lambda}}, \boldsymbol{\lambda}, \boldsymbol{\mu}, \hat{\boldsymbol{f}}) = \boldsymbol{0} \tag{4.1.6}$$

By defining a new augmented vector of unknowns $\boldsymbol{z} = [\boldsymbol{b}, \hat{\boldsymbol{\lambda}}, \boldsymbol{\lambda}, \boldsymbol{\mu}]^T$, formed by all design variables and all Lagrange multipliers, and applying the implicit function theorem

$$\boldsymbol{H}(\boldsymbol{z}, \hat{\boldsymbol{f}}) = \boldsymbol{0} \Leftrightarrow \boldsymbol{z} = \boldsymbol{h}(\hat{\boldsymbol{f}}) \tag{4.1.7}$$

where $\hat{\boldsymbol{f}}$ is the array of the $M-1$ imposed objective functions; the last part of Eq. 4.1.7 shows that solution $\boldsymbol{z}$ depends on the selected values of $\hat{\boldsymbol{f}}$. It is, thus, straightforward to express the total derivative of $\boldsymbol{H}$ (and set it to zero, since $\boldsymbol{H} = \boldsymbol{0}$ for any value-set of $\hat{\boldsymbol{f}}$) as

$$\frac{\partial \boldsymbol{H}}{\partial \hat{\boldsymbol{f}}} + \frac{\partial \boldsymbol{H}}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{h}}{\partial \hat{\boldsymbol{f}}} = \boldsymbol{0}$$

or

$$\frac{\partial \boldsymbol{h}}{\partial \hat{\boldsymbol{f}}} = -\left[\frac{\partial \boldsymbol{H}}{\partial \boldsymbol{z}}\right]^{-1} \frac{\partial \boldsymbol{H}}{\partial \hat{\boldsymbol{f}}} \tag{4.1.8}$$

with

$$\frac{\partial \boldsymbol{H}}{\partial \boldsymbol{z}} = \begin{bmatrix} \nabla^2_{\boldsymbol{bb}}\mathcal{L} & -(\nabla_{\boldsymbol{b}} f_k)^T & -(\nabla_{\boldsymbol{b}} c_k)^T & -(\nabla_{\boldsymbol{b}} g_k)^T \\ -\nabla_{\boldsymbol{b}} f_k & 0 & 0 & 0 \\ -\nabla_{\boldsymbol{b}} c_k & 0 & 0 & 0 \\ -\nabla_{\boldsymbol{b}} g_k & 0 & 0 & 0 \end{bmatrix} \text{ and } \frac{\partial \boldsymbol{H}}{\partial \hat{\boldsymbol{f}}} = \begin{bmatrix} \boldsymbol{0} \\ -I \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}$$

The dimensions of the $\frac{\partial \boldsymbol{H}}{\partial \boldsymbol{z}}$ matrix are $(N+M-1+M_e+M_{ie}) \times (N+M-1+M_e+M_{ie})$ since $\nabla^2_{\boldsymbol{bb}}\mathcal{L}$ is a $N \times N$ matrix and $\nabla_{\boldsymbol{b}} f_k, \nabla_{\boldsymbol{b}} c_k, \nabla_{\boldsymbol{b}} g_k$ are vectors with N rows and $M-1, M_e, M_{ie}$ columns, respectively.

The gradients of objective and constraint functions ($\nabla_{\boldsymbol{b}} f_k, \nabla_{\boldsymbol{b}} c_k, \nabla_{\boldsymbol{b}} g_k$) are computed using the continuous adjoint method. The Hessian of the Lagrangian $\nabla^2_{\boldsymbol{bb}}\mathcal{L}$ is approximated by the Broyden-Fletcher-Goldfarb-Shanno (BFGS) or Symmetric rank-one (SR1) updating formulas; herein, SR1 is used. After solving the linear system of Eqs. 4.1.8 for $\boldsymbol{z}$, a prediction of the next point $\boldsymbol{z}^{pred}$ is given by a first-order Taylor expansion as

$$\boldsymbol{z}^{pred} = \boldsymbol{z} + \frac{\partial \boldsymbol{z}}{\partial \hat{\boldsymbol{f}}} \Delta \hat{\boldsymbol{f}} \tag{4.1.9}$$

which simultaneously updates the design vector ($\boldsymbol{b}^{pred}$) and the Lagrange multipliers. $\Delta \hat{\boldsymbol{f}}$ is the array including the decided steps in the $M-1$ objective functions. Since this is a first-order expansion, Eq. 4.1.9 cannot guarantee that $\boldsymbol{b}^{pred}$ (includes in $\boldsymbol{z}^{pred}$) belongs to the Pareto front and a correction step in needed to bring this new point on the Pareto front. Eq. 4.1.9 concludes the prediction step.

The new point $\boldsymbol{b}^*$ on the Pareto front is obtained by solving a minimization problem from an updated target point using $\boldsymbol{b}^{pred}$ as an initial guess. This corresponds to the so-called correction step and is carried out using SLSQP to solve the following constrained problem

$$\begin{aligned} &\text{min } f_1(\boldsymbol{b}) \\ &\text{s.t. } f_k(\boldsymbol{b}) - \hat{f}_k = 0\ ,\ \forall k \in [2, M] \end{aligned}$$

The SLSQP algorithm solves the above optimization problem iteratively and the solution at the $(p+1)^{\text{st}}$ iteration $\boldsymbol{b}^{p+1}$ is obtained from $\boldsymbol{b}^p$ as

$$\boldsymbol{b}^{p+1} = \boldsymbol{b}^p + \alpha^p\, \boldsymbol{d}^p$$

where $\boldsymbol{d}^p$ is the search direction and α^p the step size. Compared to the standard SQP algorithm, in SLSQP the quadratic programming sub-problem is replaced by a least squares one, (62).

4.1.3 Mathematical Example

Let us assume the following (convex) two-objective optimization problem

$$\min \begin{cases} f_1(\boldsymbol{b}) = \sum_{i=1}^{M} (b_i - 3)^2 \\ f_2(\boldsymbol{b}) = \sum_{i=1}^{M} (b_i - 4)^2 \end{cases} \tag{4.1.10}$$

with $b_i \in [1, 6], \forall i \in [1, N]$ and $N = 6$.

To assess the proposed prediction-correction scheme in terms of computational cost, one time unit (TU) is the cost of computing both objective function values. Given that, for this problem, the derivatives are computed analytically, their cost is also set equal to one TU for all of them (as if the adjoint method was used for their computation).

The analytical expression of the Pareto front is known[1], so for obtaining the first point on the Pareto front (being one of the two front edges), problem 4.1.3 with $\hat{f}_2 = 6$ was solved at the cost of 10 TUs.

[1] All points with $(f_1, f_2) = N(\omega^2, (1-\omega)^2) 2\ \omega \in [0, 1]$.

A target of 11 Pareto points was set, using a step of $f_2(\boldsymbol{b}) - \hat{f}_2 = 0.6$ among points. Each one of the 9 subsequent points (using the prediction-correction scheme) required on average 7 TUs. For the last point, 33 TUs were required, resulting in a total computational cost of 106 TUs ($10+ \ 9 \times 7+33$ for obtaining the 11 Pareto points. It must be stressed that the $f_2(\boldsymbol{b})$ values are computed with increased accuracy w.r.t. the set target values $\hat{f}_2$; a percent error less than 0.01% for all points is obtained.

To verify the results obtained from the gradient-based method, these are compared with the analytically derived Pareto front, Fig. 52 left. It can be seen that the proposed method is able to accurately capture the Pareto front. The cost of 106 TUs is very low compared to the one that any stochastic method would ask for solving this problem. The right part of the same figure, shows the position of the predicted points in the objective space (from which each correction step is initiated) compared to the finally computed Pareto points. The closeness of the points resulted from the prediction step to the Pareto front enables the fast computation of the front point and makes the prediction-correction scheme computationally efficient.

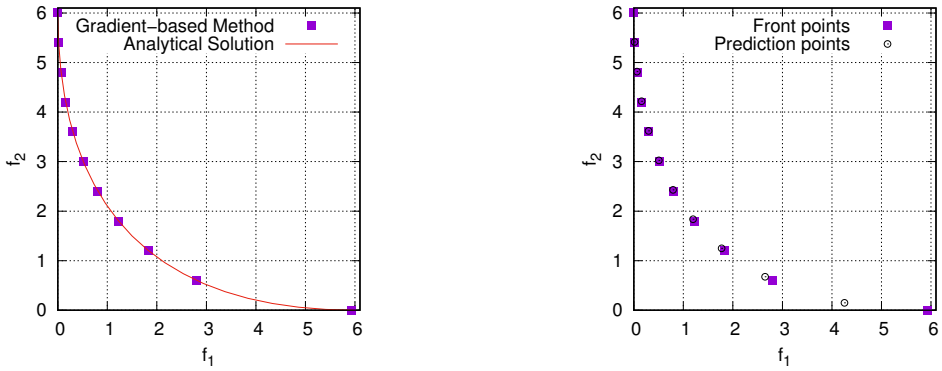


Figure 52: Left: Comparison of the analytically derived Pareto front (red line) and the developed gradient-based method (filled pink squares). Right: Predictions (empty black circles) and finally computed front points (filled pink squares); most of the prediction points cannot be distinguished from the final ones.

4.1.4 CFD-based Applications

The prediction-correction scheme is applied to the two-objective shape optimization of isolated airfoils for maximum lift and minimum drag coefficient. In both examined cases, the MOO problem is defined as

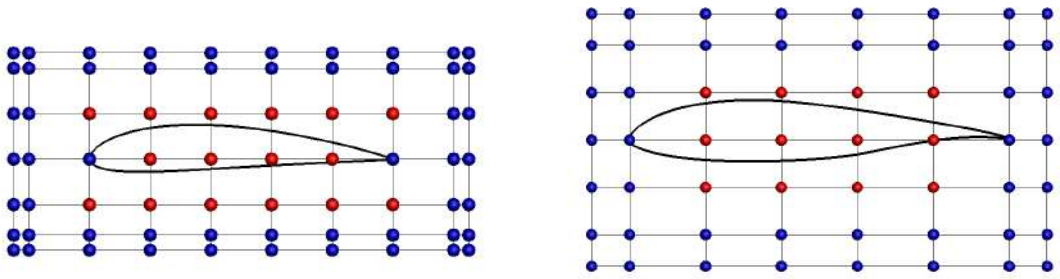
$$\min \begin{cases} f_1(\boldsymbol{b}) = \quad c_D(\boldsymbol{b}) \\ f_2(\boldsymbol{b}) = -\, c_L(\boldsymbol{b}) \end{cases} \tag{4.1.11}$$

In specific, two applications namely the shape optimization of the NACA4415 airfoil at inviscid flow conditions and the NLF(1)--0416 airfoil at transitional flow conditions are considered. In either case, the airfoil shape is parameterized using NURBS control lattices, a 10×7 for NACA4415 and a 8×7 for NLF(1)--0416 as shown Fig. 53.

In these cases, one TU corresponds to the cost of numerically solving the flow equations including the turbulence and transition models (if needed) using PUMA. Also, the assumption that the solution of the adjoint equations has practically the cost of solving the flow equations is made. Thus, the cost of solving the primal and the two adjoint equations for the two objective functions is equal to 3 TUs.

Shape Optimization of the NACA4415 Airfoil

The first application deals with the shape optimization the NACA4415 airfoil, at transonic flow



(a) NACA4415 **(b)** NLF(1)--0416

Figure 53: Parameterization of the (a) NACA4415 and (b) NLF(1)--0416 airfoils. Control points in blue remain fixed while red ones are allowed to move in the the normal-to-the-chord direction.

conditions (inviscid flow), with AoA $= 2.0^\circ$ and $M_\infty = 0.7$. The red control points (of the 10×7 lattice) that are allowed to vary in the normal-to-the-chord direction lead to $N = 16$ design variables in total.

The cost to obtain the first point on the Pareto front (bottom left, Fig. 54) using an adjoint-assisted gradient-based method for solving problem 4.1.2 with $w_1 = 1$ and $w_2 = 0$ is equal to 45 TUs. For computing 10 more points on the front, two scenarios were considered:

1. repetitively solve problem 4.1.2 with user-defined $\hat{f}_2$ (herein c_L) values, initializing the algorithm from the previously computed point on the Pareto front,
2. apply the above mentioned prediction-correction algorithm.

In either case, the target $\hat{f}_2$ values are the same. For the former, computing each front point costs, on average, 21 TUs leading to a total of 255 TUs for 11 Pareto points. On the other hand, the prediction-correction algorithm needs about 12 TUs per point or 165 TUs, in total, which makes it more efficient in terms of computational cost. The resulted fronts are compared in Fig. 54. The airfoil geometries of some of the front points as well as the Mach number field around them are given in Fig. 55.

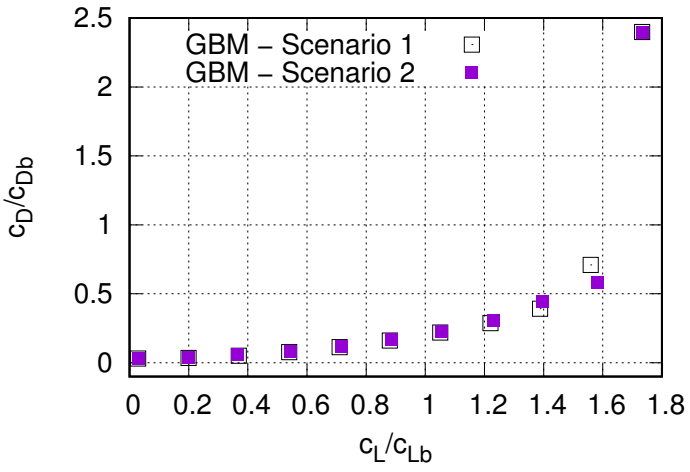
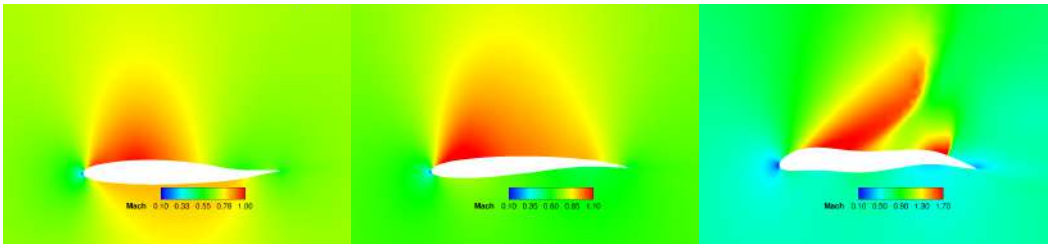


Figure 54: Comparison of the fronts resulted from the repetitive solution of problem 4.1.2 (Scenario 1; empty black squares) and the developed prediction-correction gradient-based method (Scenario 2; filled pink squares). Values on the axes are non-dimensionalized with the objective function values of the baseline geometry. Results obtained using the PUMA code.



(a) Point 1; min c_L, min c_d **(b)** Point 6 **(c)** Point 11; max c_L, max c_d

Figure 55: Mach number field around the airfoils of points 1, 6 and 11 (starting from the bottom left point, Fig. 54) resulted from the prediction-correction algorithm. Mach number legend differs among the various points.

Shape Optimization of the NLF(1)-0416 Airfoil

In the second application, the NLF(1)--0416 airfoil is optimized at transitional flow conditions with AoA $= 2.03°$, $M_\infty = 0.1$, $Re = 4 \cdot 10^6$ and $Tu = 0.15\%$. The gradients of the objective functions are accurately computed using the continuous adjoint of PUMA that differentiates not only the turbulence model but also the transition model as described in D1.2. The airfoil is parameterized using a 8×7 lattice and the total number of design variables is $N = 12$.

In this case, only the predictor-corrector method is used as a gradient-based one. The cost of obtaining the first point on the Pareto (bottom left) is equal to 15 TUs. With, more or less, the same cost, all subsequent points can be computed, leading to a total of 165 TUs. This front is compared with the front obtained using a stochastic population-based method at the cost of 500 TUs, Fig. 56. It can be seen that the gradient-based method may compute points in areas of the design space not explored from the stochastic algorithm as well as points dominating those found by the stochastic algorithm. In the ``vertical'' area of the front (top right) the points of the stochastic algorithm dominate those of the gradient-based method.

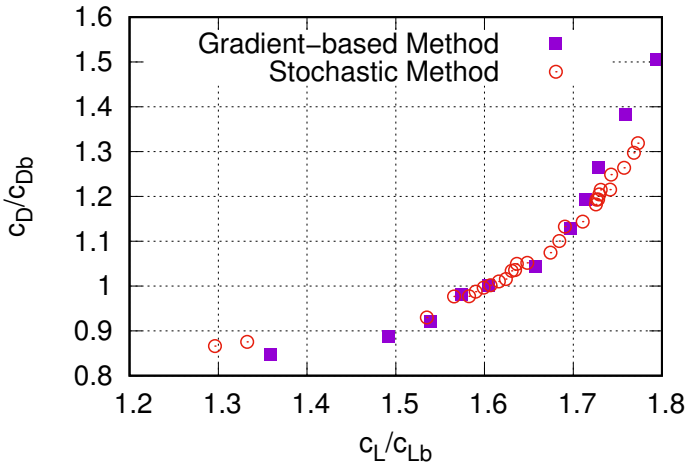


Figure 56: Comparison of the Pareto front resulted from the stochastic population-based method (empty red circles) and the developed gradient-based method (filled pink squares). Values on the axes are non-dimensionalized with the objective function values of the baseline geometry.

Figure 57 presents three airfoil geometries from the front computed by the gradient-based algorithm. These correspond to front points 1, 6 and 11, starting from the bottom left-point, i.e. the one with minimum c_L and c_D values.

(a) Point 1; min c_L, min c_d **(b)** Point 6 **(c)** Point 11; max c_L, max c_d

Figure 57: Airfoil geometries for points 1, 6 and 11 resulted from the gradient-based algorithm.

To conclude, an adjoint-based method that computes Pareto front in multi-disciplinary optimization has been devised and demonstrated. This method is now ready to be used in the application-centric work packages of NEXTAIR by NTUA and FOSS.

4.2 Stochastic Tangential Pareto Dynamics

Inria PLATON team has worked on developing methods for multi-objective stochastic programming and uncertainty quantification. Introducing stochastic uncertainty into optimization problems allows for the algorithmic search for risk averse solutions to engineering problems. This requires us to transform the quantities of interest into random variables. Using the framework of stochastic programming, we work to minimize the mathematical expectation of the set of quantities of interest. Computing the average can be computationally costly, requiring tens of function evaluations for each optimization step, and so new methods are needed in order to manage computational costs. We decompose the problem into two parts, sampling and identification. To generate samples, we first debias the stochastic multigradient direction of descent by addressing the problem at the source, the correlation in the estimation of the squared Jacobian used in the stochastic subproblem. Then, as opposed to restarting the algorithm to generate samples, we explore the Pareto front directly by adding a nonvanishing noise term tangential to the Pareto front to diffuse across the whole front. This can be seen as sampling from a potential function which has minima along the Pareto front. We go on to identify the Pareto front by building a nearest neighbor model with points generated *only* during the course of optimization. This is calibrated using the Hypervolume indicator. Finally, we estimate joint probability of each generated point being undominated and provide joint confidence intervals around each point using a bootstrap estimator.

4.2.1 Problem Formulation and Background

We direct the reader to the stochastic multi-objective optimization framework. Given a probability space $(\Theta, \mathcal{F}, \mu)$ and a set of k quantities of interest $F := \{f_i(\mathbf{x}, \mathbf{W}(\theta))\}_{i=1,\dots,k}$, $f_i : \mathcal{X} \subseteq \mathbb{R}^d \times \mathbf{W}(\theta)(\Theta) \mapsto \mathbb{R}$, our goal is to find *all* $\mathbf{x}^*$ such that

$$\mathbf{x}^* \in \underset{\mathbf{x}\in\mathcal{X}}{\operatorname{argmin}}\{\mathbb{E}[f_i(\mathbf{x}, \mathbf{W}(\theta))], ..., \mathbb{E}[f_k(\mathbf{x}, \mathbf{W}(\theta))]\}. \tag{4.2.1}$$

We denote the set of expectations $G := \{g_i(\mathbf{x})\}_{i=1,\dots,k}$ with each $g_i = \mathbb{E}[f_i(\mathbf{x}, W)]$. This formulation is sufficiently general for risk averse optimization, were it is desirable to minimize a function and its higher moments (58, 59). For example, we can simultaneously minimize both the mean and variance by taking $f_1 = f(\mathbf{x}, \mathbf{W}(\theta))$ and $f_2 = (f(\mathbf{x}, \mathbf{W}(\theta)) - \mathbb{E}[f(\mathbf{x}, \mathbf{W}(\theta))])^2$. If we have analytical expressions for each of the values, $\mathbb{E}[f_i(\mathbf{x}, \mathbf{W}(\theta))]$, we would have no need to invoke stochastic programming and could treat this problem as we would a deterministic one. Ignoring the random nature of the problem by naively substituting samples of the quantities of interest, $f_i(\mathbf{x}, \mathbf{W}(\theta)_t)$, for their true value in a classical multi-objective method could lead to convergence to a suboptimal point. We will use a Robbins Monro type algorithm, stochastic multigradient descent, to find the minima of equation 4.2.1 using single samples of the quantities of interest and their gradients at each iteration (46). This approach has the advantage of provably converging to the Pareto front without requiring the approximation of averages or an external model (46).

Pareto Optimal Probability

Identifying members of the Pareto set is also more complex in the stochastic case. If we do not have direct access to the values $\mathbb{E}[f_i(\mathbf{x}, \mathbf{W}(\theta))]$ we must estimate them. This estimation has an associated variance, complicating our reasoning and motivating a probabilistic concept of Pareto dominance. Given a set of individual solutions $\mathcal{I} := \{\mathbf{y}_i\}_{i=1,\dots,N}$ where $\mathbf{y}_i = F(\mathbf{x}_i, \mathbf{W}(\theta)_i)$ are samples from F we will use the Pareto optimal probability as defined in (57).

Definition 1 (Pareto Optimal Probability)**.**

$$\mathbb{P}_{\mathcal{I}}(\mathbf{I}_k \notin \overline{P}) = 1 - \mathbb{P}_{\mathcal{I}}(\underset{j \neq k}{\cap} \mathbf{I}_j \prec \mathbf{I}_k) \tag{4.2.2}$$

Where P denotes the Pareto set. Evaluating this probability requires knowing the joint distribution over all elements of $\mathcal{I}$. Since this is not generally tractable, it has been approximated using the subadditivity of $P_{\mathcal{I}}$ (56). Using, for example, $P_{\mathcal{I}}(\cap_{j \neq k} \mathbf{y}_k \succeq \mathbf{y}_j) \leq \max_{\mathbf{y}_j \neq \mathbf{y}_k} P_{\mathbf{y}_k, \mathbf{y}_j}(\mathbf{y}_k \succ \mathbf{y}_j)$, gives a lower bound on the Pareto optimal probability. We estimate the Pareto optimal probability using a bootstrapping approach to capture more information about the possible Pareto optimal points.

Stochastic Multigradient Descent Algorithm

We now turn our attention to the stochastic formulation of the multigradient descent algorithm, SMGDA. There has been some question as to the convergence of the SMGD algorithm, even in the strongly convex case (79). Using the tools we have developed, we give new proof of its convergence which extend to the nonconvex biased case. Improved understanding of the behavior of the SMGD algorithm allows us to debias the computation of the stochastic multigradient. Which can now be made to converge across the whole Pareto front.

Instead of finding Pareto stationary points by calculating the gradients of G, $\nabla\, \mathbb{E}[f_i(\mathbf{x}, \mathbf{W}(\theta))]$, and using a deterministic approach, we replace the gradient calculation with an estimate from a *single sample* of $\mathbf{W}(\theta)$, denoted $\mathbf{W}(\theta)_i$.

	Objectives	Gradients
Deterministic	$g(\mathbf{x})$	$\nabla g(\mathbf{x})$
Stochastic	$\mathbb{E}[f(\mathbf{x}, \mathbf{W}(\theta))]$	$\nabla f(\mathbf{x}, \mathbf{W}(\theta)_i)$

Where we assume that $\mathbb{E}[\nabla f(\mathbf{x}, \mathbf{W}(\theta))]$ is an unbiased estimator of the true gradient with finite second moment.

It is unclear how to find a direction of descent from individual gradients in a multiobjective optimization problem. One approach, pioneered in the deterministic case in (16), has been extended to the stochastic case, the multigradient. We search for a direction of descent at each iteration by solving the subproblem (16),

$$\nabla_x^{\mathcal{C}}\{F\}(\mathbf{x}_t) = \underset{\mathbf{d}}{\operatorname{argmin}} \max_i \langle \nabla f_i(\mathbf{x}_t, \mathbf{W}(\theta)_t), \mathbf{d} \rangle + \frac{1}{2}||\mathbf{d}||_2^2. \tag{4.2.3}$$

The logic behind this strategy is a straightforward extension of single objective optimization theory. Indeed, we would search for a direction of descent, $\mathbf{v}$, such that $\langle \nabla f(\mathbf{x}_t, \mathbf{W}(\theta)_t), \mathbf{v} \rangle \leq 0$ by solving the equation, $\mathbf{v}_t = \operatorname{argmin}_{\mathbf{d}} \langle \nabla f(\mathbf{x}, \mathbf{W}(\theta)), \mathbf{d} \rangle + \frac{1}{2}||\mathbf{d}||_2^2$. In the multi-objective case, if the point $\mathbf{x}_t$ is not on the Pareto set, we should have that $\langle \mathbb{E}[\nabla f_i(\mathbf{x}_t, \mathbf{W}(\theta)_t)], \mathbb{E}[\nabla_x^{\mathcal{C}}\{F\}(\mathbf{x}_t, \mathbf{W}(\theta)_t)] \rangle \leq 0$ for *all* i. If this criterion is not met by one of the objectives, then we cannot produce a simultaneous improvement on all objectives and we have identified a member of the Pareto set.

It is often easier to solve for the direction of descent by taking the dual of eq. 4.2.3 , giving the multi-gradient the form

$$\nabla_x^{\mathcal{C}}\{F\}(\mathbf{x}, \mathbf{W}(\theta)) = \{\sum_{i=1}^{k} \alpha_k^* \nabla f_i(\mathbf{x}, \mathbf{W}(\theta)) \mid \alpha^* = \underset{\alpha \in \Delta^{k-1}}{\operatorname{argmin}} \alpha^\top J(\mathbf{x}, \mathbf{W}(\theta)) J(\mathbf{x}, \mathbf{W}(\theta))^\top \alpha\} \tag{4.2.4}$$

Where Δ^{k-1} is the $k-1$ dimensional simplex. Iterates take the form $\mathbf{x}_{t+1} = \mathbf{x}_t - \varepsilon_t \nabla_x^{\mathcal{C}}\{F\}(\mathbf{x}_t, \mathbf{W}(\theta)_t)$. We also require that ε_t be a sigma suite, e.g. it meets the following two critera,

$$\sum_t \varepsilon_t = \infty \qquad\qquad \sum_t \varepsilon_t^2 < \infty \tag{4.2.5}$$

It has been shown that this approach converges to a point on the Pareto front under heavy assumptions. There is, however, a bias in the calculation of the stochastic multigradient. This prevents the algorithm from converging to an arbitrary point along the Pareto front. Part of our original contribution is to show that this algorithm converges to the Pareto front in the nonconvex and biased case. This is used in our sampling approach in order to efficiently sample points along the whole of the uncertain Pareto front.

4.2.2 Description of The Algorithm

Our approach decomposes the problem of estimating the uncertain Pareto front by decomposing it into sampling and identifying stages. To ameliorate the bias in the SMGD algorithm and generate samples along the Pareto front we augment the standard multigradient descent iterates in two ways. First, we introduce a novel debiasing strategy which allows the algorithm to converge to the whole of the Pareto front. Then, to explore the Pareto front as well as overcome any remaining bias, we add a noise term which will allow the algorithm to explore directions tangential to the Pareto front.

Because of the added noise term, this approach efficiently explores the area around the minima and will not stay stuck in a saddle point, leading to a dense characterization of the Pareto front with less wasteful computation when compared to multiple restarts. In the second phase, once we have a collection of samples from a noise ball around the Pareto front, we must determine the set of Pareto optimal design points and estimate the location of the true Pareto front. However, we do not have samples of curves from which to make confidence intervals. Instead, we have a collection of points, each a sample from a possible Pareto front. To form a cohesive picture from these snapshots, we form preliminary estimation the Pareto front using local averages. Then, we create both bootstrap confidence intervals and estimate the fitness of each potential design point to give a characterization of the whole of the Pareto front with uncertainty. Our strategy does not require the full objective functions or gradients to ever be evaluated, making the extra gradient descent steps worthwhile and computationally tractable.

Debiasing SMGDA

To calculate the stochastic-multigradient we would like to use $\mathbb{E}[J_F(\mathbf{x}, \mathbf{W}(\theta))]\,\mathbb{E}[J_F(\mathbf{x}, \mathbf{W}(\theta))]^\top$, which would be equivalent to using $J_G(\mathbf{x})$, the deterministic Jacobian matrix. Instead, we only have access to samples of the matrix $J_F(\mathbf{x}, \mathbf{W}(\theta)) J_F(\mathbf{x}, \mathbf{W}(\theta))^\top$. We notice that the two are not unrelated quantities as we have the relation,

$$\mathbb{E}[J_F(\mathbf{x}, \mathbf{W}(\theta)) J_F(\mathbf{x}, \mathbf{W}(\theta))^\top] = \Sigma_{J_F(\mathbf{x}, \mathbf{W}(\theta))} + \mathbb{E}[J_F(\mathbf{x}, \mathbf{W}(\theta))]\,\mathbb{E}[J_F(\mathbf{x}, \mathbf{W}(\theta))]^\top \tag{4.2.6}$$

where $\Sigma_{J_F(\mathbf{x}, \mathbf{W}(\theta))}$ denotes the covariance matrix of the gradients of F.

To debias the stochastic multigradient we first compute an *online* estimate of $\Sigma_{J_F(\mathbf{x},\mathbf{W}(\theta))}$ using the recursion,

$$\widehat{\mu}_{t+1} = \gamma_t \widehat{\mu}_t + (1-\gamma_t) J_f(\mathbf{x}_t, \mathbf{W}(\theta)_t) \tag{4.2.7}$$

$$\widehat{\Sigma}_{J_F,t+1} = \gamma_t \widehat{\Sigma}_{J_F,t} + (1-\gamma_t)\big(J_F(\mathbf{x}_t, \mathbf{W}(\theta)_t) J_F(\mathbf{x}_t, \mathbf{W}(\theta)_t)^\top - \widehat{\mu}_t \widehat{\mu}_t^\top\big) \tag{4.2.8}$$

using individual samples of $J_F(\mathbf{x}, \mathbf{W}(\theta))$ and $\gamma_t \in (0,1)$. We then calculate α^* using the modified formulation.

$$\widehat{\alpha}^*(\boldsymbol{b}) = \underset{\alpha \in \Delta^{k-1}}{\operatorname{argmin}} \, \alpha^\top \big(J_F(\mathbf{x}, \mathbf{W}(\theta)) J_F(\mathbf{x}, \mathbf{W}(\theta))^\top - \widehat{\Sigma}_{J_F(\mathbf{x},\mathbf{W}(\theta))}\big)\alpha. \tag{4.2.9}$$

We finally calculate a debiased form of the stochastic gradient

$$\widehat{\nabla_x^{\mathcal{C}}\{F\}}(\mathbf{x}, \mathbf{W}(\theta)) := \sum_i \widehat{\alpha}_i^* \nabla f_i(\mathbf{x}, \mathbf{W}(\theta)). \tag{4.2.10}$$

This approach has the advantage of not requiring excess computation in individual rounds and placing less assumptions on the required behavior of α^* while also effectively reducing the bias in the calculation of $\nabla_x^{\mathcal{C}}\{F\}(\mathbf{x}, \mathbf{W}(\theta))$.

Tangential Pareto Dynamics

To find and explore the Pareto front we generate a sequence using the recurrence relation,

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \varepsilon_t \widehat{\nabla_x^{\mathcal{C}}\{F\}}(\mathbf{x}_t) + \sqrt{2\varepsilon_t \beta^{-1}} T_t, \tag{4.2.11}$$

with $T_t \sim \mathcal{N}(0, PP^\top)$ with P such that T_t is tangential to the Pareto front, $\widehat{\nabla_x^{\mathcal{C}}\{F\}}(\mathbf{x}_t)$ is the direction of descent calculated with debiased parameters $\widehat{\alpha}^*$, and β an inverse temperature parameter set by the practitioner.

Since in this setting we do not have a priori information, particularly about the Pareto front, we have to estimate the direction tangent to the Pareto front and the location of the Pareto front. To do this, we can use the *stochastic multi-gradient* information since the gradient will always have a component perpendicular to the front. At each iteration, choosing $\gamma_t \in (0,1)$, we calculate a moving average,

$$\overline{\nabla_x^{\mathcal{C}}\{F\}}_{t+1} = \gamma_t \overline{\nabla_x^{\mathcal{C}}\{F\}}_t + (1-\gamma_t)\widehat{\nabla_x^{\mathcal{C}}\{F\}}. \tag{4.2.12}$$

And the projection matrix,

$$P_{t,i,j} = \delta_{i,j} - \frac{1}{||\overline{\nabla_x^{\mathcal{C}}\{F\}}_t||^2} \overline{\nabla_x^{\mathcal{C}}\{F\}}_{t,i} \overline{\nabla_x^{\mathcal{C}}\{F\}}_{t,j}, \tag{4.2.13}$$

where $\delta_{i,j}$ denotes the dirac delta function. Setting $T_t = P_t Z_t$ with $Z \sim \mathcal{N}(0, \mathbb{1}_{d\times d})$ allows us to take a step forward as defined in eq. 4.2.11 .

The goal of this approach is to explore the whole of the Pareto front, and allowing ε_t or γ_t to go to zero would create a unique limiting distribution centered at $\mathbf{x}_\infty$. In order to insure that all points are reached we are willing to allow for a slightly higher amount of noise in the sampled points and so we define a pair of sequences

$$\{\varepsilon_t\}_{t=0,\dots,\infty} := \varepsilon_0 \geq \varepsilon_1 \geq \dots \geq \varepsilon_\infty > 0 \tag{4.2.14}$$

$$\{\gamma_t\}_{t=0,\dots,\infty} := \gamma_0 \geq \gamma_1 \geq \dots \geq \gamma_\infty > 0 \tag{4.2.15}$$

Which prevents the algorithm from having a limiting distribution centered on a single point.

Algorithm 1 Summary of Sampling Algorithm

Input $\{\varepsilon_t\}, \{\gamma_t\}, \mathbf{x}_0$.
Initialize $\widehat{\mu}_0 = \mathbf{0}$, $\widehat{\Sigma}_{J_F,0} = \mathbb{1}_{d\times d}$, $t = 1$, $\overline{\nabla_x^{\mathcal{C}}\{F\}}_0 = 0$.
while Running **do**
  Query Stochastic Oracle for $J_F(\mathbf{x}_t, \mathbf{W}(\theta)_t)$ ▷ Referred to as J_{F_t} below.
  $\widehat{\mu}_t \leftarrow \gamma_t \widehat{\mu}_{t-1} + (1-\gamma_t) J_{F,t}$
  $\widehat{\Sigma}_{J_F,t} \leftarrow \gamma_t \widehat{\Sigma}_{J_F,t-1} + (1-\gamma_t)(J_{F,t}J_{F,t}^\top - \widehat{\mu}_t\widehat{\mu}_t^\top)$
  $\widehat{\alpha}^* \leftarrow \operatorname{argmin}_{\alpha\in\Delta^{k-1}} \alpha^\top \big(J_{F_t}J_{F_t} - \widehat{\Sigma}_{J_F,t}\big)\alpha$
  $\widehat{\nabla_x^{\mathcal{C}}\{F\}}_t \leftarrow J_{F_t}^\top \alpha^*$
  $\overline{\nabla_x^{\mathcal{C}}\{F\}}_t \leftarrow \gamma_t \overline{\nabla_x^{\mathcal{C}}\{F\}}_{t-1} + (1-\gamma_t)\widehat{\nabla_x^{\mathcal{C}}\{F\}}_t$
  $P_t \leftarrow \mathbb{1}d \times d - \frac{\overline{\nabla_x^{\mathcal{C}}\{F\}}_t \overline{\nabla_x^{\mathcal{C}}\{F\}}_t^\top}{||\overline{\nabla_x^{\mathcal{C}}\{F\}}_t||_2^2} \overline{\nabla_x^{\mathcal{C}}\{F\}}_t \overline{\nabla_x^{\mathcal{C}}\{F\}}_t^\top$
  $Z_t \sim \mathcal{N}(\mathbf{0}, \mathbb{1}_{d\times d})$
  $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \varepsilon_t \widehat{\nabla_x^{\mathcal{C}}\{F\}}_t + \sqrt{2\varepsilon_t\beta^{-1}} P_t Z_t$
  $t \leftarrow t+1$
end while

Inferring the Pareto set and Pareto Front

Having a set of samples, $\{(\mathbf{x}_1, \mathbf{F}(\mathbf{x}_1, \mathbf{W}(\theta)_1)), ..., (\mathbf{x}_n, \mathbf{F}(\mathbf{x}_n, \mathbf{W}(\theta)_n))\}$, we must now determine the Pareto front and the Pareto set. We do this by first mapping our samples into objective space, then assessing their Pareto optimality to determine the Pareto front, and finally looking at the preimage to determine the Pareto set.

For the first task, we must find a mapping

$$\widehat{G}(\mathbf{x}) : \mathbb{R}^d \mapsto \mathbb{R}^k, \mathbf{x} \mapsto [\mathbb{E}[f_1(\mathbf{x}, \mathbf{W}(\theta))], ..., \mathbb{E}[f_k(\mathbf{x}, \mathbf{W}(\theta))]].$$

Unfortunately, we only have access to previously collected samples using algorithm 1, which are almost surely not directly on the Pareto front. We approximate their expected values using k-nearest-neighbor regression, a flexible fully nonparametric approach with few hyperparameters that converges under light assumptions (7). Calculating the number of neighbors is equivalent to finding the best smoothing of our data. Using too few neighbors results in a granular and overly noisy Pareto front. Too many, though, and we overwrite the local nature of the estimate and instead compute a global average. To track the quality of our estimate and calibrate the number of neighbors we sequentially add neighbors and search for an elbow in the *hypervolume indicator*, a commonly used performance metric in multi-objective optimization (23, 81). A rapid change in the hypervolume indicator corresponds to an oversmoothing of the local average, and so to prevent oversmoothing we set the number of neighbors to be equal to half of the 'elbow' value, detected through the kneedle algorithm (75).

Our next task, assessing the relative dominance between points, is now a probabilistic one. Using the mapping, $\widehat{G}(\mathbf{x})$, we would like to know if an individual $\mathbf{I}_k$ in the set of n tuples $\mathcal{I} := \{(\mathbf{x}_1, \widehat{G}(\mathbf{x}_1)), ..., (\mathbf{x}_n, \widehat{G}(\mathbf{x}_n))\}$ is undominated, e.g. $\nexists \mathbf{I}_j \in \mathcal{I}$ s.t. $\mathbf{I}_j \prec \mathbf{I}_k$. Since there is noise in the estimation of the mean, a direct comparison between vectors in k dimensions is likely to lead to suboptimal conclusions. Instead, we use the Pareto optimal probability defined in eq. 1, which requires the joint pdf across individuals, $\mathbb{P}_{\mathcal{I}}$. For this we take a bootstrapping approach. For each $\mathbf{x}_i$ we first resample individual $\mathbf{I}_i$ m times, generating a set of bootstrap estimates $\{\widehat{\mathbf{I}}_i^{(1)}, ..., \widehat{\mathbf{I}}_i^{(m)}\}_{i=1}^n$ for each point. We then approximate the Pareto optimal probability using the bootstrap estimator $\mathbb{P}_{\mathcal{I}}(\mathbf{I}_k \notin \overline{P}) \approx 1 - \frac{1}{m}\sum_{p=1}^m \mathbb{1}(\exists j \text{ s.t. } \widehat{\mathbf{I}}_j^{(p)} \prec \widehat{\mathbf{I}}_i^{(p)})$.

To estimate the set of optimal design points, we take as $P(\widehat{G}\mathcal{X})$ the set

$$P(\widehat{G}\mathcal{X}) = \{\mathbf{x}_i \mid \mathcal{P}_i > c\} \tag{4.2.16}$$

Where c is a user defined cutoff. With this estimation of the set of Pareto optimal points and corresponding bootstrap estimates of $\widehat{G}(\mathbf{x})$. We can also generate bootstrapped confidence intervals in objective space by discretizing the space into a set of k dimensional boxes, $B_{1,\dots,k}$, where in each dimension there are $l(k)$ boxes. A confidence interval for dimension d can be made by ordering the elements in the union of all the boxes in the d^{th} row/column $D := \{\cup B_{\dots,d,\dots}\}$, and taking appropriate quantiles.

$$CI(d) = [D_\alpha, D_{1-\alpha}] \tag{4.2.17}$$

Where D_α is the α quantile of the samples in collection D. The preimage of D can be used to compute a confidence interval in design space. For a robust estimate of both the Pareto front and Pareto set, we use the median of D.

4.2.3 Mathematical Example

We look towards a stochastic version of a classic bi-convex problem

$$f_1(\mathbf{x}, \mathbf{W}(\theta)) = (\mathbf{x} - \frac{5}{\sqrt{2}})^\top(\mathbf{x} - \frac{5}{\sqrt{2}}) \tag{4.2.18}$$

$$f_2(\mathbf{x}, \mathbf{W}(\theta)) = \mathbf{x}^\top\mathbf{x} \tag{4.2.19}$$

With $\mathbf{x} \in \mathbb{R}^2$, $\mathbf{W}(\theta) \sim \mathcal{N}(\mathbf{0}, \mathbb{1}_{2\times 2})$. We see that $\mathbb{E}[f_1(\mathbf{x}, \mathbf{W}(\theta))] = (\mathbf{x} - \frac{5}{\sqrt{2}})^\top(\mathbf{x} - \frac{5}{\sqrt{2}}) + 1$ and $\mathbb{E}[f_2(\mathbf{x}, \mathbf{W}(\theta))] = \mathbf{x}^\top\mathbf{x} + 1$. This problem has a Pareto set which forms a straight line in design space inbetween $\mathbf{0}$ and $\frac{5}{\sqrt{2}}\mathbf{1}$. With 1000 evaluations of the quantities of interest, the generated 95% confidence intervals cover a large portion of both the Pareto set and Pareto front over 71 unique points. See figure 58 for a visual representation, where we have juxtaposed our estimation with the true Pareto front. We also include the average area of the confidence box. If the average area is low,

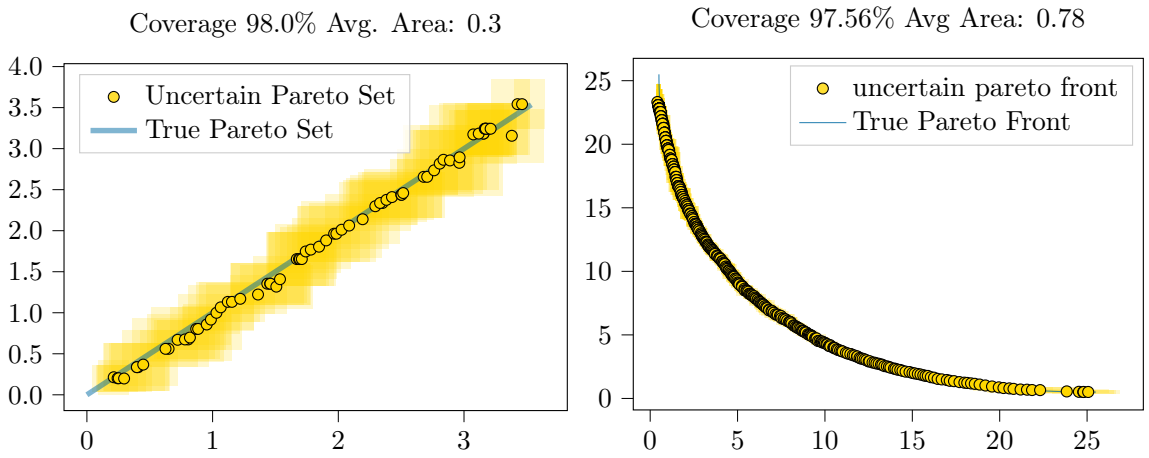


Figure 58: 95% Confidence intervals (yellow shading) generated through the proposed approach with 1000 evaluations of the quantity of interest cover almost the whole Pareto front and Pareto front (set).

while the coverage is high, then we say that the Pareto front (set) is estimated with high precision.

4.3 The modified Normal Boundary Intersection algorithm for efficient Pareto fronts computation (IRT)

4.3.1 Overall objectives

The Normal Boundary Intersection optimization algorithm allows Pareto fronts to be computed by solving a set of single objective optimization problems (9). This is an advantage because it allows efficient algorithms to be used for these subproblems, even in high dimension, such as gradient-based optimizers using adjoint derivatives. One of the main drawbacks of the classical weighted sum of objectives, which is similar to the creation of single objective subproblems, is that the points obtained on the Pareto front are not evenly distributed. The NBI algorithm ensures that the points obtained are evenly spaced. It is therefore more computationally efficient for a given target accuracy.

However, not all points obtained by the NBI algorithm are Pareto optimal. Therefore, the algorithm was improved by Shukla in (70) so that weak Pareto optimality of solutions is guaranteed without losing its ability to generate a set of evenly spaced solutions. Another improvement by Shukla allows to avoid solving some subproblems if it can be known in advance that their solutions are not on the Pareto front.

4.3.2 Description of the algorithm

The main idea of the algorithm is to first compute the extrema of the Pareto front, i.e. the individual minima of each objective. This is shown in Figure 59 as F1* and F2* for a two objective problem.

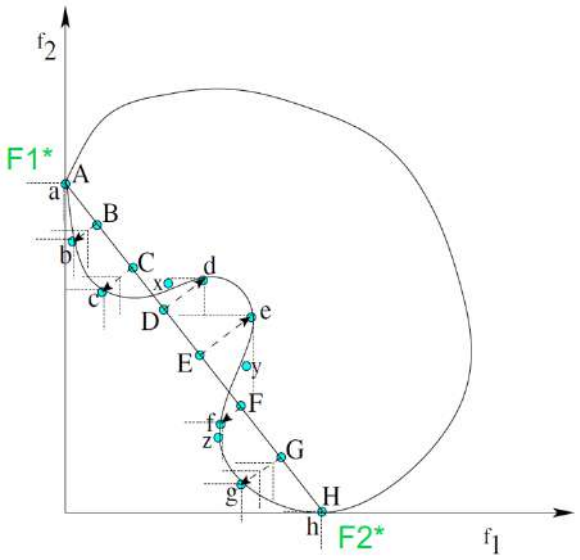


Figure 59: Main idea of the mNBI algorithm: the uniform Pareto front discretization from (9).

Figure 60 describes the main computation steps of the algorithm.

The normals to the convex hull of each minimum are discretized in a uniform way, thanks to the approach proposed in (9). The β vectors are sampled. This defines N sub-optimization problems with a new variable t to maximise, and the multi-objective function f appears in a new inequality constraint involving the normal direction and the anchor point on the convex hull of each minimum. Figure 59 shows well the solution obtained thanks to this constraint. The use of inequality constraints by Skula instead of the original formulation by Das and Dennis, which used equality constraints, allows to obtain Pareto optimal points in situations D and E, which is a significant advantage.

In the following sections we validate the mNBI algorithm on 3 different analytic benchmark problems. This also allows to illustrate the key mNBI features:

- user controlled discretization of the Pareto Front,

1. Compute the convex hull of the individual minimas

$$F^* = (f_1^*, f_2^*, \ldots, f_m^*)^\top$$

2. Compute the Phi matrix and the Normal direction

$$\Phi = (F(\mathbf{x}_1^*), F(\mathbf{x}_2^*), \ldots F(\mathbf{x}_m^*))$$

$$\hat{n} = -\Phi e, \text{ where } e = (1, 1, \ldots, 1)^T$$

3. Discretize the Beta vectors following the original NBI paper with a user specified number of points N

$$\beta = \{(b_1, b_2, \ldots, b_m)^\top \mid \sum_{i=1}^m b_i = 1\}$$

4. Solve the N scalar sub problems

$$\begin{aligned} &\max_{(\mathbf{x},t)} \quad t, \\ &\text{subject to } \Phi\beta + t\hat{n} \geq F(\mathbf{x}), \\ &\qquad\qquad \mathbf{x} \in S, \end{aligned}$$

Figure 60: mNBI Algorithm main steps

- obtaining convex and non-convex Pareto fronts,
- the use of Gradient-based and gradient-free sub-optimization algorithms.

4.3.3 Numerical experiments on the Binh-Korn problem

The Binh-Korn problem is a famous and simple bi-objective optimization problem from (4).

In Figure 61 we illustrate the obtained Pareto fronts with two variants of the algorithm, one using the gradient-based SLSQP (mNBI/SLSQP (29) variant) and the other using the gradient-free COBYLA (mNBI/COBYLA (55)) algorithms for subproblem solution. It clearly shows the nice even repartitioning of the obtained Pareto fronts at convergence. The post-processing also shows the intermediate points in the optimization history that are not dominated in green, but are probably not optimal. The algorithm is also computationally very efficient, as very few intermediate iterations (in blue) are needed. All Pareto fronts are consistent, which also validates the approach.

4.3.4 Scalability with respect to discretization: gradient enhanced MOO

The next figure compares the scalability of the algorithm, in terms of the number of points on the Pareto fronts, of the mNBI/SLSQP and mNBI/COBYLA variants. Analytic derivatives of the Binh-Korn problem are provided for mNBI. The analytic derivatives of the subproblem objective and constraints are computed in our implementation. The gradient-based variant shows a dramatic reduction in computation time (a factor of 10 in dimension 60).

4.3.5 Binh-Korn : mNBI vs NSGA2 genetic algorithm

In the following experiment, we compare mNBI/COBYLA with the famous NSGA2 genetic algorithm (12), which serves as a reference. The figure 63 compares the results obtained by the two derivative-free algorithms for two different computational budgets: 340 and 630 function evaluations. Overall, the Pareto fronts obtained are consistent. The one obtained by NSGA-2 is more refined, but many more points are far from the Pareto front in the optimization history. Also with NSGA-2 some areas are not well refined, this is not controlled. mNBI/COBYLA has a slightly better precision, the obtained optimal points are better than those obtained by NSGA-2. However, this problem is very simple and in low dimension, in the following sections we will scale the dimension of the problem, which will change the conclusions.

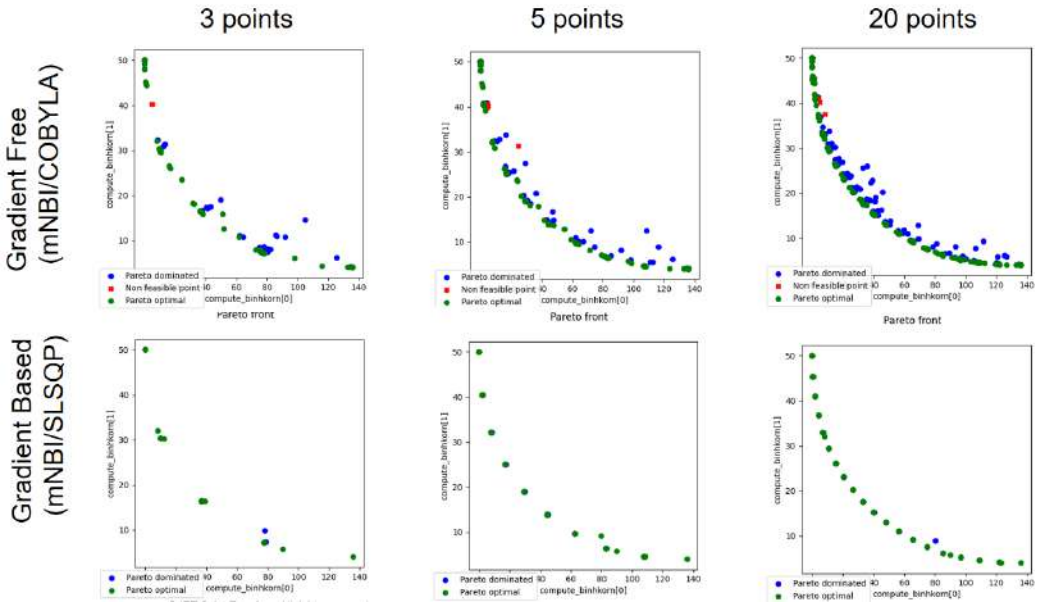


Figure 61: Pareto fronts obtained by the mNBI/SLSQP and mNBI/COBYLA algorithm on the Binh-Korn problem,

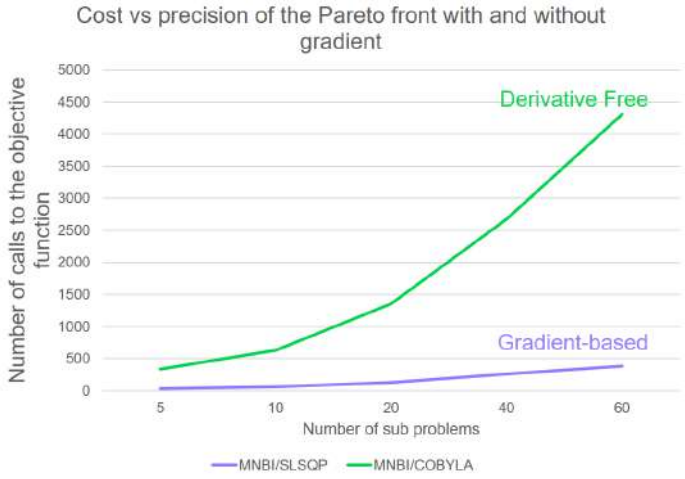


Figure 62: Pareto fronts obtained by the mNBI on the Binh-Korn problem with the gradient-based SLSQP and gradient-free COBYLA algorithms for the sub-problems resolution.

4.3.6 Fonseca Flemming problem: non-convex Pareto front

The Fonseca-Flemming bi-objective benchmark problem (17) is interesting because the Pareto front is not convex. This can be a difficulty for some classes of multi-objective optimizers. As shown in 64, this is not a difficulty for mNBI, which efficiently computes the expected Pareto front.

4.3.7 Fonseca Flemming problem : mNBI vs NSGA2 genetic algorithm scalability

An interesting feature of the Fonseca-Flemming problem is that its input dimension can be increased. This scalability makes it possible to compare the efficiency of the mNBI/COBYLA algorithm with that of the NSGA-2 algorithm when this dimension is increased. This is particularly important because the target applications in NEXTAIR are mostly in high dimension.

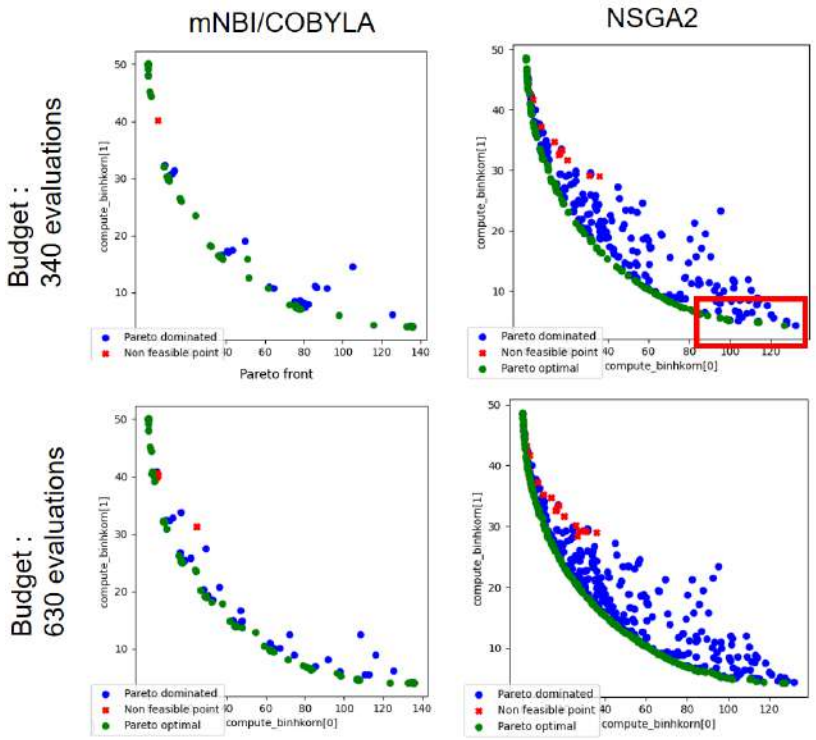


Figure 63: Pareto fronts obtained by the derivative free mNBI/COBYLA and the NSGA2 genetic algorithm on the Binh-Korn problem

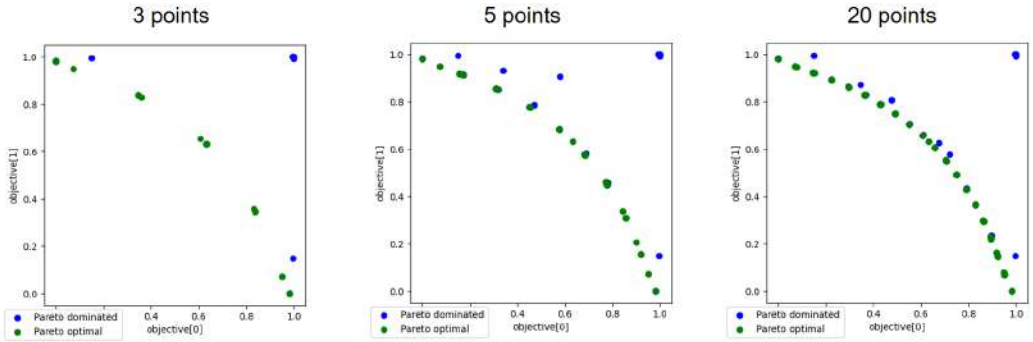


Figure 64: Pareto fronts obtained by the mNBI-COBYLA algorithm on the Fonseca Flemming problem

Figure 65 shows the obtained Pareto fronts in input dimension 3, 4 and 5, which is still low. We limited this input dimension because it is visible that the performance of NSGA-2 decreases rapidly with the dimension, both in terms of cost and precision of the obtained Pareto front. The computational budget was limited to 500 objective function calls, which was always a limit for NSGA-2, while mNBI/COBYLA did not require more than 142 evaluations. The input dimension is only limited by the ability of the sub-optimizer to solve the problem for mNBI, so it could easily be thousands if analytic derivatives are provided. It is not surprising that Genetic Algorithms scale poorly with the dimension of the input space, both for analytic and wing design problems (42).

4.3.8 Poloni's problem: avoiding holes in the Pareto front

The Poloni's problem is described in (11) as it is a classical benchmark for genetic algorithms. Its main feature of interest is the presence of a hole in the Pareto front.

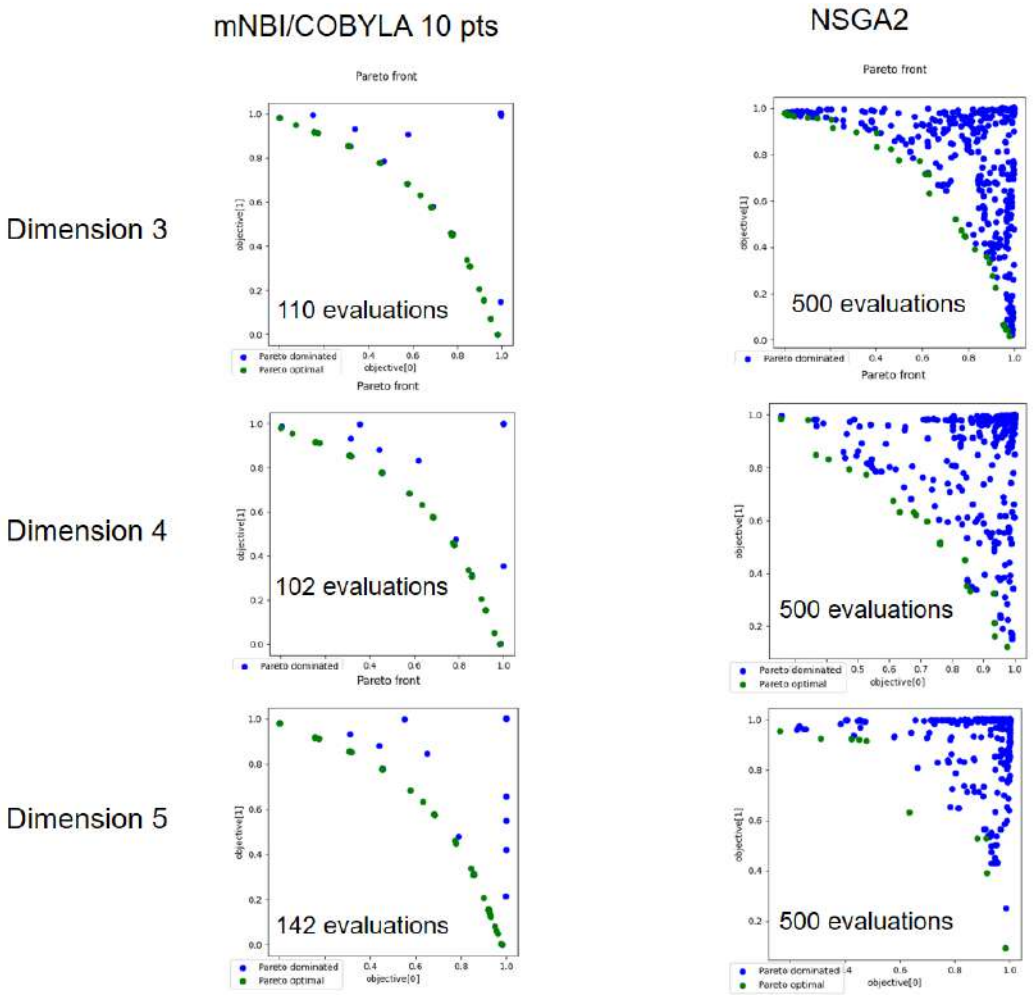


Figure 65: Pareto fronts obtained by the mNBI-COBYLA and NSGA2 algorithms on the Fonseca Flemming problem for varying dimensions of the design variables.

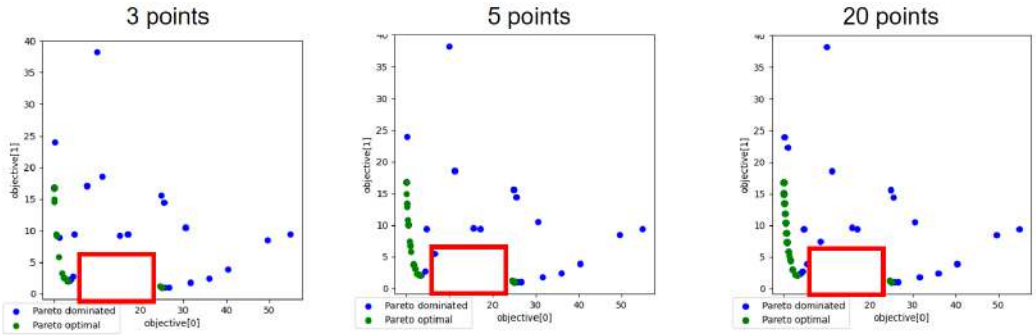


Figure 66: Pareto fronts obtained by the mNBI-COBYLA and mNBI algorithms on the Polini's problem, with varing target number of points on the Pareto front.

Figure 66 shows the Pareto front obtained with the mNBI/COBYLA algorithm, with varying numbers of target points. The algorithm successfully obtains the Pareto front. A very interesting feature of the mNBI implementation is that it does not compute all the a priori prescribed subproblems (defined by the β matrix) if some regions are found to be dominated. This is particularly useful for this problem when the number of target points is high, as it reduces the computational cost.

4.3.9 Discussion and conclusion

The mNBI multi-objective optimization algorithm was implemented in Python. The algorithm has been validated and its properties illustrated on benchmark problems:

- user controlled Pareto front discretization,
- ability to greatly reduce computation time when gradients are available,
- more accurate and cheaper than GA (NSGA2) for moderate dimensions and upper (>5),
- avoids dominated regions in the Pareto front.

In conclusion, the mNBI algorithm is a very promising and well validated technique that is ready for use on real high-dimensional test cases.

As future work, we will implement a feature that allows the user to refine a specific region of the Pareto front. This is of course possible with the mNBI algorithm by selecting existing points on the Pareto and refining the sampling on the β matrix in the convex hull of the selected points. mNBI will be integrated into GEMSEO (20). It will be open sourced in the next version of GEMSEO.

5 Unsteady Adjoints Running at Realistic Time Scales, in MDO

5.1 ONERA contribution

ONERA developed a modular time/spatial parallel Time Spectral solver and the associated adjoint solver, to introduce unsteady/dynamic constraints from Hi-Fi simulations in the design process. To this aim, a CFD solver independent high-level Python abstraction layer was developed and extended to deformable grids. All the operations deriving from the space discretization were performed by the elsA CFD code.

5.1.1 Modular TSM solver

Theoretical background

The semi-discrete governing fluid equations resulting from a Finite Volume discretization process can be written as

$$\frac{\partial \boldsymbol{\mathcal{V}}\mathbf{W}}{\partial t} + \mathbf{R}(\mathbf{W}, t) = \mathbf{0} \tag{5.1.1}$$

where $\mathbf{W}$ is the fluid conservative variables vector, t is time, $\boldsymbol{\mathcal{V}}$ is a block-diagonal matrix containing the cell volumes and $\mathbf{R}(\mathbf{W}, t)$ is the residual computed from the fluxes and eventual source terms. The TSM method is used to seek for time periodic phenomena of pulsation ω. It consists in first projecting the conservative variables and residual vector in the Fourier space and in truncating the Fourier Series to keep the first N harmonics. The aplication of the Inverse Fourier Transform to equation (5.1.1) written in the Fourier domain allows then the expression of the latter equation at the $M = 2N + 1$ instants of the time sampling of one period. Such a process yields the expression of the time derivative term at the n^{th} instant

$$\left.\frac{\partial \mathbf{W}}{\partial t}\right|_{t_n} = D_t(\mathbf{W}_n) = \sum_{j=0}^{M-1} d_{nj}\mathbf{W}_j \tag{5.1.2}$$

$$\text{with} \begin{cases} d_{nn} &= 0 \\ d_{nj} &= \frac{\omega}{2}(-1)^{(n-j)} \csc\left(\frac{\pi(n-j)}{M}\right) \end{cases}$$

The fluid equation at the n^{th} instant can then be written as a steady equation

$$|\boldsymbol{\mathcal{V}}|\frac{\partial \mathbf{W}_n}{\partial \tau_n} + \mathbf{R}_n + |\boldsymbol{\mathcal{V}}|D_t(\mathbf{W}_n) = \mathbf{0}, \quad 0 \leq n < M \tag{5.1.3}$$

where $\mathbf{W}_n$ is the fluid conservative variables vector at the n^{th} instant and $|\boldsymbol{\mathcal{V}}|$ is a block-diagonal matrix containing the cell volumes. The pseudo time term $|\boldsymbol{\mathcal{V}}|\frac{\partial \mathbf{W}_n}{\partial \tau_n}$ is added in order to help in stabilizing the time integration scheme. It is discretized by a first-order scheme

$$|\boldsymbol{\mathcal{V}}|\frac{\Delta \mathbf{W}_n}{\Delta \tau_n} + \mathbf{R}_n + |\boldsymbol{\mathcal{V}}|D_t(\mathbf{W}_n) = \mathbf{0}, \quad 0 \leq n < M \tag{5.1.4}$$

where $\Delta \mathbf{W}_n = \mathbf{W}_n^{q+1} - \mathbf{W}_n^q$ is the solution increment between iterations q and $q+1$, and $\Delta\tau_n$ is the pseudo-time increment computed using a local time-stepping strategy and based on a CFL number. The backward-Euler implicit scheme is derived by differentiating the residual and the time-spectral source term at $\mathbf{W}_n^{q+1}$. Let $\mathbf{J}_n = \partial \mathbf{R}_n / \partial \mathbf{W}_n$ be the Jacobian matrix of the residual vector, we have

$$\mathbf{R}_n^{q+1} \approx \mathbf{R}_n^q + \mathbf{J}_n \Delta \mathbf{W}_n \tag{5.1.5}$$

The time spectral derivative operator D_t being linear it follows directly that

$$D_t(\mathbf{W}_n^{q+1}) = D_t(\mathbf{W}_n^q) + D_t(\Delta \mathbf{W}_n) \tag{5.1.6}$$

Substituting (5.1.5) and (5.1.6) in (5.1.4) we obtain the fully implicit TSM formulation

$$\left(\frac{|\boldsymbol{\mathcal{V}}|}{\Delta\tau_n}+\mathbf{J}_n+|\boldsymbol{\mathcal{V}}|D_t(.)\right)\Delta\mathbf{W}_n=-\mathbf{R}_n^q-|\boldsymbol{\mathcal{V}}|D_t(\mathbf{W}_n^q)=-\mathbf{R}_{TSM}(\mathbf{W}_n^q) \tag{5.1.7}$$

Then the fully coupled system with all flow time instances gathered in vector $\mathbf{W}=(\mathbf{W}_0,\mathbf{W}_1,\cdots,\mathbf{W}_{M-1})^T$ can be written as

$$[\mathbf{A}]\Delta\mathbf{W}=-\mathbf{R}_{TSM}(\mathbf{W}^q) \tag{5.1.8}$$

with the resulting Jacobian matrix given by

$$[\mathbf{A}]=\begin{pmatrix}\frac{|\boldsymbol{\mathcal{V}}^0|}{\Delta\tau_0}+\mathbf{J}_0 & |\boldsymbol{\mathcal{V}}^1|\mathbf{d}_0^1 & \dots & |\boldsymbol{\mathcal{V}}^{M-1}|\mathbf{d}_0^{M-1}\\ |\boldsymbol{\mathcal{V}}^1|\mathbf{d}_1^0 & \frac{|\boldsymbol{\mathcal{V}}^1|}{\Delta\tau_1}+\mathbf{J}_1 & \dots & |\boldsymbol{\mathcal{V}}^{M-1}|\mathbf{d}_1^{M-1}\\ \vdots & \vdots & \ddots & \vdots\\ |\boldsymbol{\mathcal{V}}^{M-1}|\mathbf{d}_{M-1}^0 & |\boldsymbol{\mathcal{V}}^{M-1}|\mathbf{d}_{M-1}^1 & \dots & \frac{|\boldsymbol{\mathcal{V}}^{M-1}|}{\Delta\tau_{M-1}}+\mathbf{J}_{M-1}\end{pmatrix} \tag{5.1.9}$$

where $\mathbf{d}_n^j=diag(d_{nj})=d_{nj}\mathbf{I}$. In the expression above we have emphasized that the volume matrix $|\boldsymbol{\mathcal{V}}^n|$ may depend on the time instance for the case of an ALE formulation with a deforming mesh.

Imlementation

A TSM solver requires several operations which can be split into the ones relevant to a pure CFD simulation and those relevant to the TSM approach itself. A solver was then implemented in a Python program with an interface with the CFD code elsA. The Python program is in charge of performing the time integration scheme, in computing the time source terms by handling the coupling between the different instants and in assembling the TSM residual ($\mathbf{R}_{TSM}$ in equation (5.1.7)). The CFD code should be able to provide the residual vector from flow variables at a given instant, the jacobian matrix or the matrix-vector product for each instant and the local time step for each time instance. The elsA code was then modified to make these later operators available from an external Python script. One of the main advantage of such a modular implementation is the capability to investigate quickly and rather easily different resolution strategies. Furthermore both time and space parallelization was implemented taking benefit from the space parallelization of the CFD code. Indeed, because of the large number of degrees of freedom in the space-time problem (the number of time instances M times the number of degrees of freedom of the space problem), a particular parallel implementation is often needed. In order to keep a fixed number of unknowns per process, the classical space discretization managed by the CFD code is augmented with a time parallelization as illustrated in figure 67. The distribution of the spatial unknowns over K processes is duplicated by the number of instants. The vector of the unknowns of the whole space-time problem $\mathbf{W}$ is then distributed over a total of $M\times K$ processes indexed by the couple (n,k) where $n\in[0,M-1]$ and $k\in[0,K-1]$. Parallel communications may occur either in the time direction via one of the K time communicators (yellow) or in the space direction via one of the M space communicators (blue).

Two resolution strategies of the TSM problem were implemented. The first one consists in solving the equation 5.1.7 using an implicit iterative Block Jacobi as described by Sicot et-al (71). The solution increment is computed using an iterative way from

$$\left(\frac{|\boldsymbol{\mathcal{V}}|}{\Delta\tau_n}+\mathbf{J}_n\right)\Delta\mathbf{W}_n^q=-\mathbf{R}_n^q-|\boldsymbol{\mathcal{V}}|D_t(\mathbf{W}_n^q)-|\boldsymbol{\mathcal{V}}|D_t(\Delta\mathbf{W}_n^{q-1}) \tag{5.1.10}$$

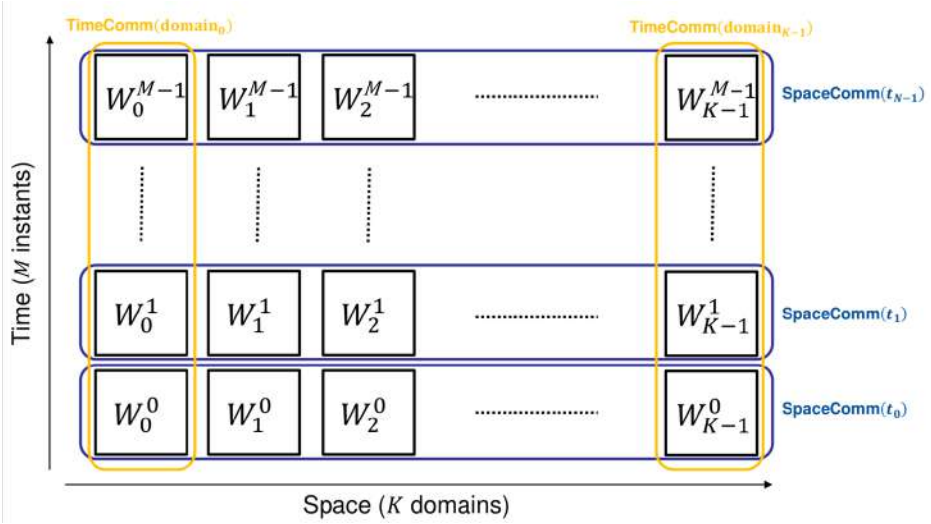


Figure 67: Distribution of the space-time vector $\mathbf{W}$ over $M \times K$ processes where $\mathbf{W}_k^m$ refers to the unknowns belonging to spatial domain k and the instant m. Parallel communications occurs only within one of the K temporal communicator (yellow) or one of the M space communicators (blue).

matching a total jacobian matrix

$$[\mathbf{A}] = \begin{pmatrix} \frac{|\boldsymbol{\mathcal{V}}^0|}{\Delta\tau_0} + \mathbf{J}_0 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \frac{|\boldsymbol{\mathcal{V}}^1|}{\Delta\tau_1} + \mathbf{J}_1 & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \frac{|\boldsymbol{\mathcal{V}}^{M-1}|}{\Delta\tau_{M-1}} + \mathbf{J}_{M-1} \end{pmatrix} \tag{5.1.11}$$

According to this approach, the solution increment of one time instance can be computed independently of the other instants, the coupling between the instants being accounted for only in the right hand side term. Nevertheless, the CFL must remain low, especially for a high number of instants, to converge towards a solution. The second implemented approach consists in handling straightforwardly the whole space-time vector $\mathbf{W}$ and in solving equation 5.1.8 using a preconditioned GMRES algorithm. It can be less dependent on the CFL number than the Block Jacobi resolution, but its robustness depends on the preconditioning technique. Moreover, a parallel implementation of the GMRES algorithm is needed for a large number of instants or of mesh cells.

NACA64 2D test case

The selected test case to validate the implementation of the TSM solver is a symmetric *NACA64A010* airfoil under a prescribed harmonic pitch motion. The motion is sinusoidal with an angle of attack defined by the function

$$\alpha(t) = \alpha_0 + \widehat{\alpha}\sin(\omega t) \tag{5.1.12}$$

where α_0 is the mean angle of attack and $\widehat{\alpha}$ is the amplitude of the pitch motion.

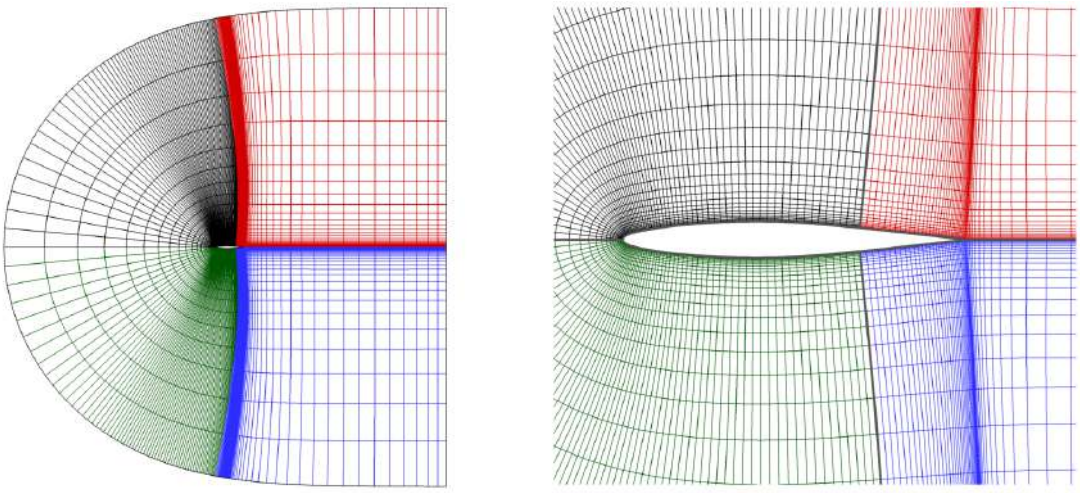


Figure 68: Structured 4-block 2D C-mesh of the NACA64A010 airfoil.

Experimental results are reported by the AGARD group in (10). We have selected the *dynamic index 55* case corresponding to a transonic Mach number of 0.796 and an excitation frequency of 34.4Hz. The fluid domain is discretized with a structured 2D C-mesh of 257x33 cells. The inviscid Euler equations are considered for the fluid model. An upwind second order Roe spatial discretization scheme associated to a MUSCL reconstruction and a Van Albada limiter for the convective fluxes is used. The flow conditions are reported in Table 4.

Table 4: Flow conditions for dynamic index 55.

M	0.796	Mach number
p_∞	$133912\,\text{Pa}$	free stream pressure
p_t	$203321\,\text{Pa}$	total pressure
q	$59395\,\text{Pa}$	dynamic pressure
f	$34.4\,\text{Hz}$	pitching frequency
α_0	$-0.21°$	mean incidence
$\widehat{\alpha}$	$1.02°$	pitching amplitude

Numerical experiments

The phase portrait of the aerodynamic coefficients as functions of the angle of attack considering the whole time history of the reference unsteady analysis are plotted in Figure 69. The plain bold curves correspond to the established periodic response in the 8^{th} time period, after a transient of 7 periods. The colored markers correspond to the instantaneous values obtained for a varying number of harmonics.

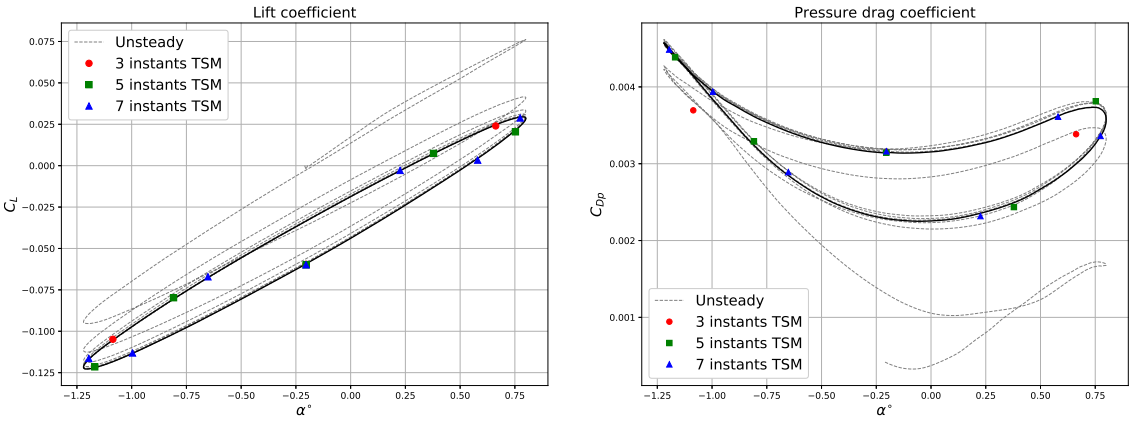


Figure 69: Aerodynamic coefficients versus angle of attack plots. The instantaneous values for a varying number of harmonics are superimposed to the reference unsteady time history.

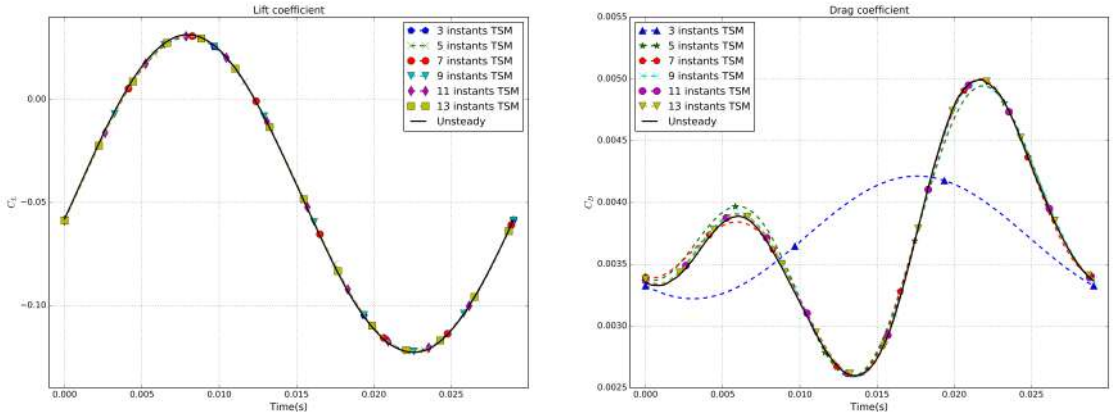


Figure 70: Accuracy of the time-spectral reconstruction for lift and drag coefficients for a spectral approximation from 1 to 6 harmonics.

The previous simulations were performed considering rigid rotations of the whole mesh, meaning that the airfoil motion was taken into account using an entrainment speed applied to the whole mesh. In order to be able to handle structural deformations, the ALE formulation was implemented in the modular TSM solver to take into account that the fluid mesh is not the same at each instant. Such formulation implies that the residuals vectors $\mathbf{R}_n$ and the jacobian matrix $\mathbf{J}_n$ provided by the CFD code must account for the additional flux due to the mesh deformation speed and for the specific boundary conditions for a deforming mesh. The time operator D_t should also be applied to the quantity $\mathcal{V}\mathbf{W}$ and not only to the conservative variables vector $\mathbf{W}$. A first step of the validation of the ALE formulation implementation into the modular solver consisted in checking that the grid deformation speed is correctly taken into account by the CFD code by comparing two steady simulations. One simulation is a classic one just applying a nonzero angle of attack to the far-field boundary conditions. The second consisted in applying no angle of attack but an equivalent constant vertical grid velocity to the whole mesh. It has been checked that both simulations provided similar results. This validation step was also performed in the case of the 3D DLR-F25 wing using a multi-block structured mesh. The second validation step consisted in carrying out TSM simulations, applying the same oscillation motion (5.1.12) to the airfoil and keeping the far boundaries fixed. The mesh is then computed for each instant using a grid deformation tool (figure 71).

A first set of simulations was performed for a time discretization based on 1 harmonic, i.e. for a time sampling of 3 instants. The flow variable distributions at the three instants are in a good agreement as can be seen in figure 72 showing the iso-contours of the density for both the rigid and deformable meshes. The discrepancies between the iso-lines can be explained by the different positions of the airfoil (initial position for the rigid mesh since the rotation is taken into account by the far-field boundary conditions, and rotated position for the deformable mesh). Figure 73 shows the time evolution over one period of the global coefficients computed for the rigid and deformable meshes, and using both unsteady and TSM simulations. These coefficients are computed in the wind axes. As expected according to the simulations presented above, the TSM with 3 instants is accurate enough to catch the lift evolution. A good agreement can indeed be observed between the four simulations. But the TSM with 3 instants is not accurate enough to capture the drag evolution. If a rather good agreement between the unsteady drag evolutions computed with the rigid and deformable meshes can be observed, discrepancies between the 2 TSM simulations can be noticed. Further investigations are on going to try to explain them.

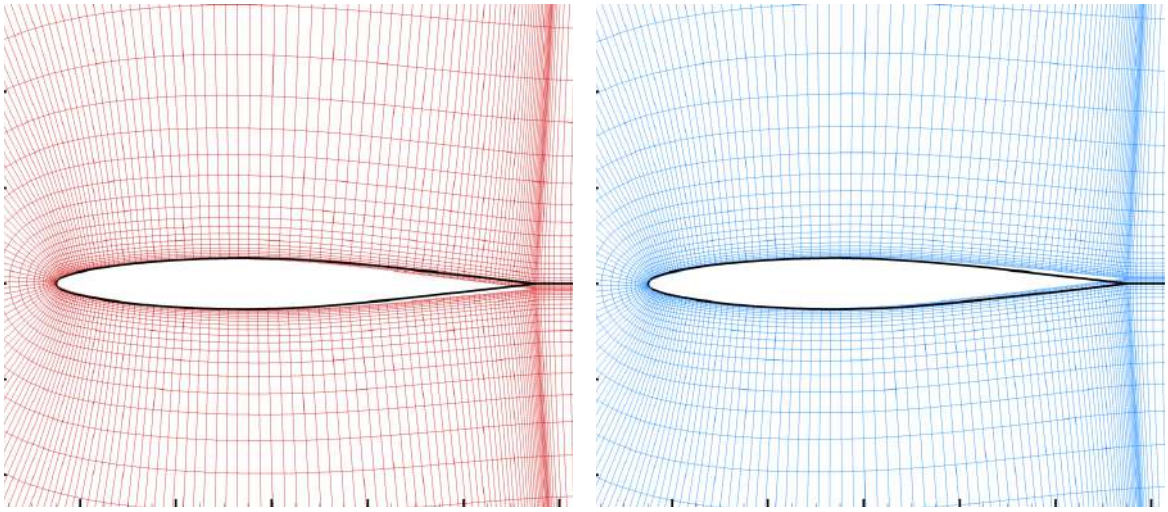


Figure 71: Deformed meshes at the instants 2 (left) and 3 (right) of a TSM simulations with 3 instants

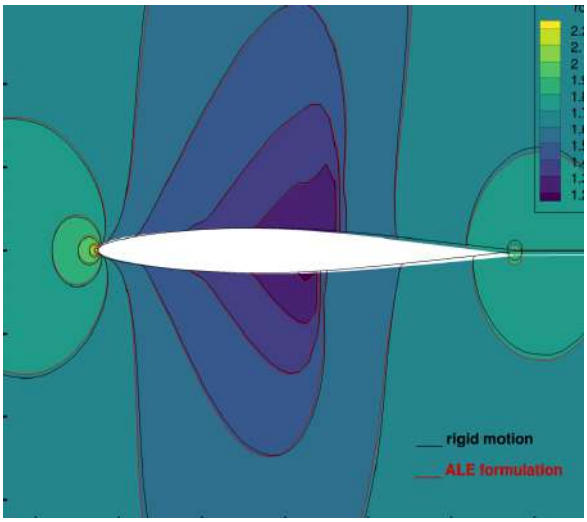


Figure 72: density contours at the 2^{nd} instant computed for the rigid and deformable meshes

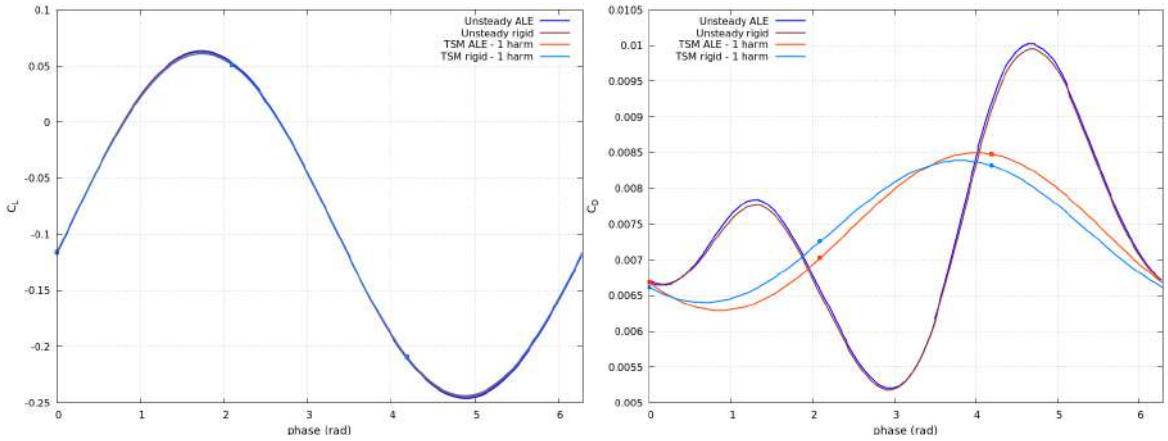


Figure 73: time evolution over one period of the lift (left) and drag (right) coefficients resulting from unsteady and TSM simulations

Numerical simulations were also performed for the TSM problem with 4 harmonics and 9 instants for its capability to predict accurately the drag evolution.

Some investigations about the numerical algorithms were carried out. The TSM problem with 3 instants (1 harmonics) was solved using a Newton like non-linear algorithm with an additional pseudo time term (CFL). The resulting linear system to compute the solution increment at each non-linear iteration was solved using a block Jacobi algorithm or a preconditioned GMRES algorithm. The semi implicit block-Jacobi technique, as presented by Sicot et-al (71), allows the resolution instant by instant with an update of the time terms at each Jacobi sub-iteration. The CFL number had to remain constant at a rather low value (no greater than 10 for this case) for the whole simulation in order the latter to converge. The linear system which arises at each Jacobi sub-iteration was partially solved using either a relaxation LUSGS method or a preconditioned GMRES. On the other hand, the linear system at each non-linear iteration can be solved using a preconditioned GMRES. This linear system takes into account the degrees of freedom of all the instants. The GMRES algorithm must then be applied to a much larger system than that solved using the Jacobi algorithm. But the CFL number can vary and be adapted to the residual in order that the pseudo time terms vanish at the end of the resolution process. The convergences towards the solution are very similar for both rigid and deformable meshes, whatever the used resolution algorithm. The TSM problem with 3 instants is numerically easy to solve. The convergence of the residual is similar for all the resolution algorithms as long as the CFL number remains constant. This parameter has the greatest influence on the convergence speed. Its evolution proportional to a power of the inverse of the residual norm allows a rapid increase of itself and then reduces drastically the number of needed non-linear iterations as can be seen in figure 74 left.

TSM simulations were also performed with a deformable mesh (ALE formulation) and a time sampling of 9 instants (4 harmonics). Only the Block Jacobi technique associated with a LU-SGS algorithm was used. Similar convergences can be observed with both rigid and deformable meshes for CFL numbers less than 3 (figure 74 right). The time evolutions of the lift and drag coefficients over one period for both rigid and deformable meshes are in a good agreement one with the other and with the unsteady simulations (figures 75).

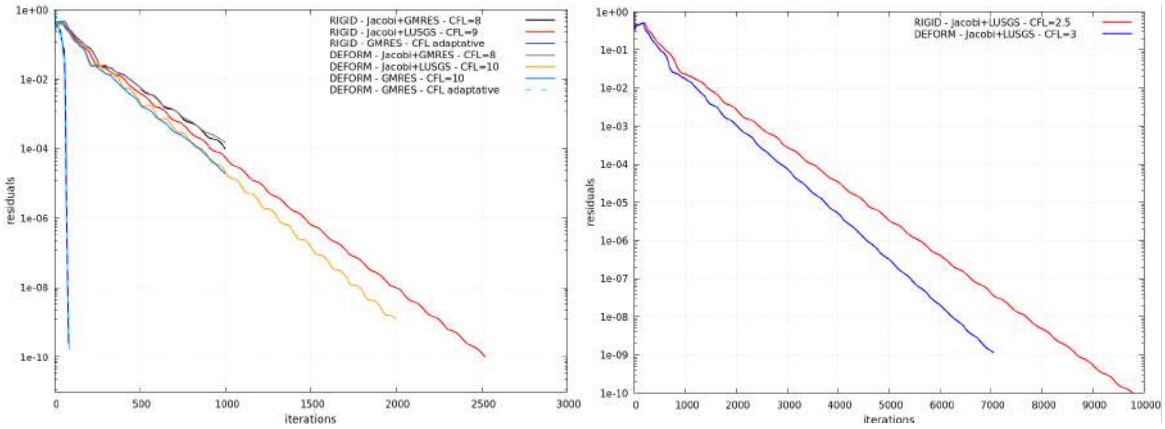


Figure 74: Residual evolutions for the TSM simulations with 3 (left) and 9 instants (right).

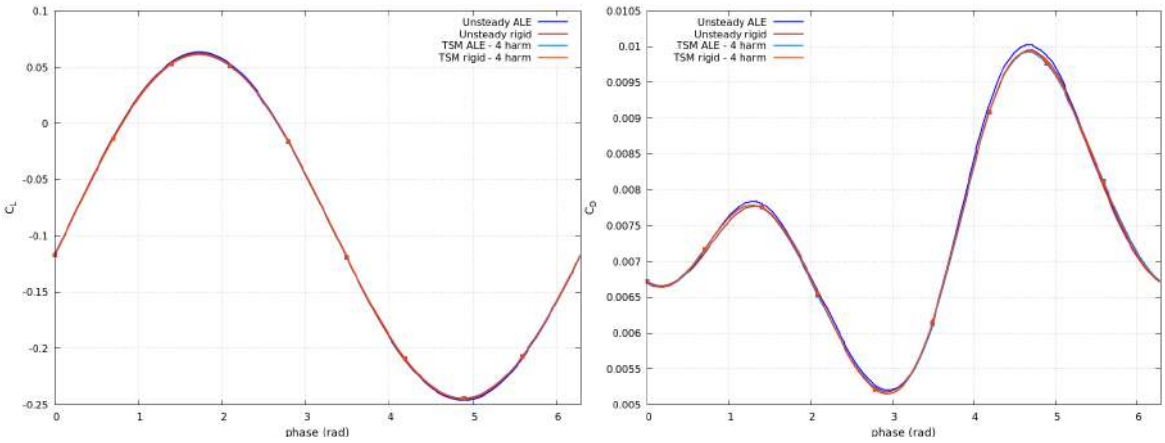


Figure 75: time evolution over one period of the lift (left) and drag (right) coefficients resulting from unsteady and tsm simulations.

5.1.2 TSM adjoint solver

Theoretical background

Let $\mathbf{p}$ be a set of n_p shape design parameters, $\mathbf{X}(\mathbf{p})$ the corresponding parameterized fluid mesh and $\mathcal{J}(\mathbf{p}) = J(\mathbf{W}_n(\mathbf{p}), \mathbf{X}(\mathbf{p}))$ any scalar aerodynamic function of interest like drag, lift or moment coefficient. Note that the flow variables at the $2N+1$ instants in the period appear in function $\mathcal{J}$. The flow field $\mathbf{W}_n$ and the volume mesh $\mathbf{X}$ are linked by the discrete residual equations

$$\mathbf{R}_{TSM}(\mathbf{W}_n, \mathbf{X}) = \mathbf{R}_n(\mathbf{W}_n, \mathbf{X}) + |\mathcal{V}| D_t(\mathbf{W}_n) = \mathbf{0} \tag{5.1.13}$$

In this section we only discuss applications of the ALE formulation for a rigid entrainment velocity field. We then consider a constant mesh geometry and the associated volume matrix $|\mathcal{V}|$ is therefore taken constant in the following. Anyway, the extension to a general ALE formulation is straightforward. Direct differentiation of (5.1.13) with respect to a design parameter p gives

$$\frac{d\mathbf{R}_{TSM}}{dp} = \sum_{j=0}^{M-1} \frac{\partial \mathbf{R}_{TSM}}{\partial \mathbf{W}_j} \frac{d\mathbf{W}_j}{dp} + \frac{\partial \mathbf{R}_{TSM}}{\partial \mathbf{X}} \frac{d\mathbf{X}}{dp} \tag{5.1.14}$$

A strong requirement here is that the total variation of the residual must vanish for any design parameter change, i.e. $d\mathbf{R}_{TSM}/dp = \mathbf{0}$. Inserting the definition of $\mathbf{R}_{TSM}$ in (5.1.14) leads to the following tangent linear system:

$$\mathbf{J}_n \frac{d\mathbf{W}_n}{dp} + |\boldsymbol{\mathcal{V}}| \sum_{j=0}^{M-1} d_{nj} \frac{d\mathbf{W}_j}{dp} = -\frac{\partial \mathbf{R}_n}{\partial \mathbf{X}} \frac{d\mathbf{X}}{dp} - \delta_p |\boldsymbol{\mathcal{V}}| \sum_{j=0}^{M-1} d_{nj} \mathbf{W}_j \tag{5.1.15}$$

where $\delta_p |\boldsymbol{\mathcal{V}}| = \frac{\partial |\boldsymbol{\mathcal{V}}|}{\partial \mathbf{X}} \frac{d\mathbf{X}}{dp}$ represents the matrix of analytic sensitivities of cell volumes to a change in p. On the left-hand side we recognize the full Jacobian matrix $[\mathbf{A}]$ in (5.1.9).

We introduce the general form of an objective function for a periodic response as the weighted sum $J = \sum_{n=0}^{M-1} c_n J_n(\mathbf{W}_n, \mathbf{X})$. The total derivative of the objective function is then written as

$$\frac{d\mathcal{J}}{dp} = \sum_n c_n \frac{\partial J_n}{\partial \mathbf{X}} \frac{d\mathbf{X}}{dp} + \sum_n c_n \frac{\partial J_n}{\partial \mathbf{W}_n} \frac{d\mathbf{W}_n}{dp} \tag{5.1.16}$$

We also have the following equality that holds $\forall\, \boldsymbol{\lambda}_n \in \mathbb{R}^{n_a}$

$$c_n \boldsymbol{\lambda}_n^T \frac{d\mathbf{R}_{TSM}}{dp} = 0, \quad 0 \leq n < M \tag{5.1.17}$$

and combining with (5.1.14) and (5.1.16) leads to

$$\begin{aligned} \frac{d\mathcal{J}}{dp} = & \sum_n c_n \frac{\partial J_n}{\partial \mathbf{X}} \frac{d\mathbf{X}}{dp} + \sum_n c_n \boldsymbol{\lambda}_n^T \frac{\partial \mathbf{R}_{TSM}}{\partial \mathbf{X}} \frac{d\mathbf{X}}{dp} \\ & + \sum_n c_n \left(\frac{\partial J_n}{\partial \mathbf{W}_n} + \boldsymbol{\lambda}_n^T \mathbf{J}_n \right) \frac{d\mathbf{W}_n}{dp} + \sum_n c_n \boldsymbol{\lambda}_n^T \left(|\boldsymbol{\mathcal{V}}| \sum_j d_{nj} \frac{d\mathbf{W}_j}{dp} \right) \end{aligned} \tag{5.1.18}$$

The adjoint system is obtained by canceling the factor of $d\mathbf{W}_n/dp$ which defines the adjoint vectors $\boldsymbol{\lambda}_n$ as the solutions of the following coupled linear systems

$$\mathbf{J}_n^T \boldsymbol{\lambda}_n + |\boldsymbol{\mathcal{V}}| \sum_{j=0}^{M-1} d_{jn} \boldsymbol{\lambda}_j = -\left(\frac{\partial J_n}{\partial \mathbf{W}_n} \right)^T, \quad 0 \leq n < M \tag{5.1.19}$$

The transpose of the system matrix (5.1.9) of the TSM solver naturally appears in the adjoint linear system. Using the skew-symmetric property of the time-spectral derivative matrix $(d_{nj})_{0 \leq n,j < M}$ we have $d_{jn} = -d_{nj}$. Finally, the total derivative of the objective function turns out to be

$$\frac{d\mathcal{J}}{dp} = \sum_n c_n \left(\frac{\partial J_n}{\partial \mathbf{X}} + \boldsymbol{\lambda}_n^T \frac{\partial \mathbf{R}_n}{\partial \mathbf{X}} \right) \frac{d\mathbf{X}}{dp} + \sum_n c_n \boldsymbol{\lambda}_n^T \left(\delta_p |\boldsymbol{\mathcal{V}}|\, D_t(\mathbf{W}_n) \right) \tag{5.1.20}$$

Numerical experiments

We apply the TSM adjoint solver to the same NACA64 2D test case (4-block structured C-type mesh) presented in the section devoted to the TSM solver. As already mentioned, the Block-Jacobi iterative solution strategy can no longer be applied because the adjoint system lacks of diagonal dominance. As pointed out by Mundis and Mavriplis in (49), with an increasing number of harmonics the full Jacobian matrix in 5.1.9 proceeds farther and farther from diagonal dominance. This results in linear systems of increasing complexity in terms of size and numerical stiffness. This undesirable property becomes problematic for relaxation-based iterative solvers whose convergence relies on diagonal dominance. To circumvent the need for diagonal dominance, Krylov subspace methods (e.g. the GMRES algorithm (60)) were considered in more recent works (77, 50, 74, 48, 48, 34). In practice, we decided to switch to more efficient Krylov solvers embedded in the *hpddm* package (27) of the linear

algebra library PetSc (3). The interested reader may refer to (25) for details about the nested Krylov strategy applied to CFD adjoint systems. Whatever, the linear solution method used (i.e. stationary methods or Krylov methods), a preconditioner is always needed to decrease the condition number of the system and allow for better convergence. In the present work, we focus on the so-called Block Jacobi preconditioner. However, several preconditioning strategies have been proposed in the literature, among which the *STI* (Spatial-Temporal diagonal-block Inversion) preconditioner (48), the *approximate-factorization* preconditioner (50), the *block-circulant* preconditioner ((47), Chap. 3), etc.

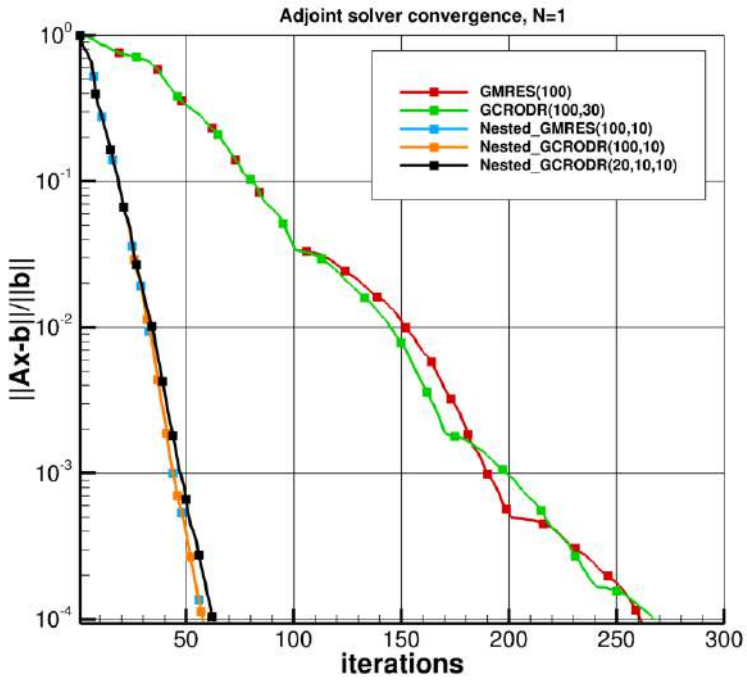


Figure 76: Convergence of various Krylov solvers applied to the TSM adjoint linear system, for a single harmonic spectral approximation. Stationary preconditioned GCRO-DR and GMRES-DR perform similarly. The same conclusion is drawn for nested Krylov solvers. For this simple problem, even a small outer Krylov subspace of 20 vectors turns out to be very efficient.

Various advanced Krylov solvers with deflation and restarting like GMRES-DR (Generalized Minimal Residual Method with Deflated Restarting) and GCRO-DR (Generalized Conjugate Residual method with inner Orthogonalization and Deflated Restarted) combined with stationary or iterative nonlinear preconditioners have been tested. The associated convergence histories of the relative residual norm are plotted in Figure 76. GMRES-DR and GCRO-DR perform equivalently, while their flexible counterparts exhibit a speedup factor of about 5. The numerical parameters of the Krylov solvers are reported in the figure legend.

The efficiency of the Krylov solver is then evaluated for an increasing number of harmonics in the spectral approximation of the unsteady solution. In Figure 77 the associated residual norm convergence history of the GCRO-DR and FGCRO-DR adjoint solvers is plotted. Clearly the Block-Jacobi preconditioner, even if associated to a nested Krylov solver, cannot achieve wave number independence in terms of iterations. As mentioned previously, designing such a preconditioner is still an active area of research. Reusing existing building blocks was our main goal in setting up this TSM modular adjoint solver prototype and improvements will be considered in the near future regarding the preconditioning strategy.

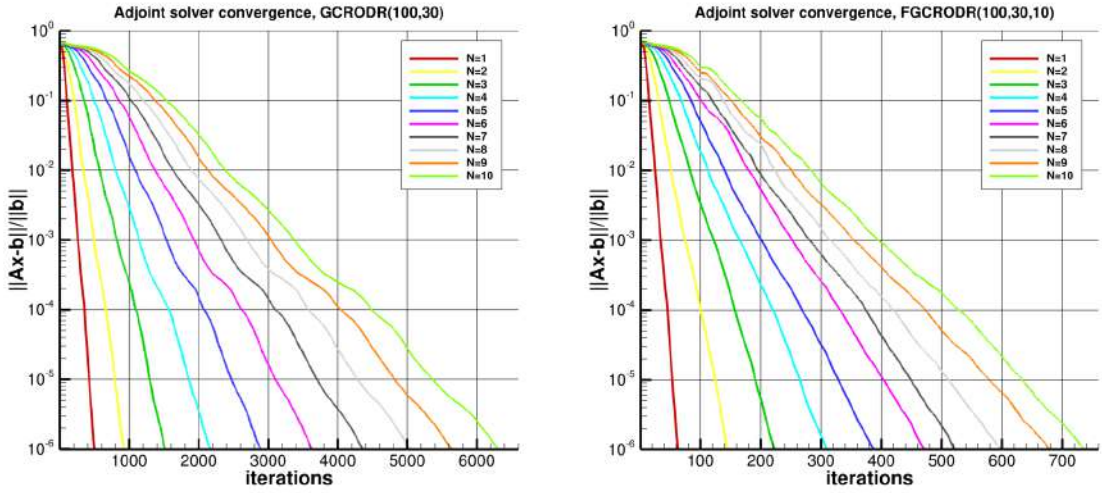
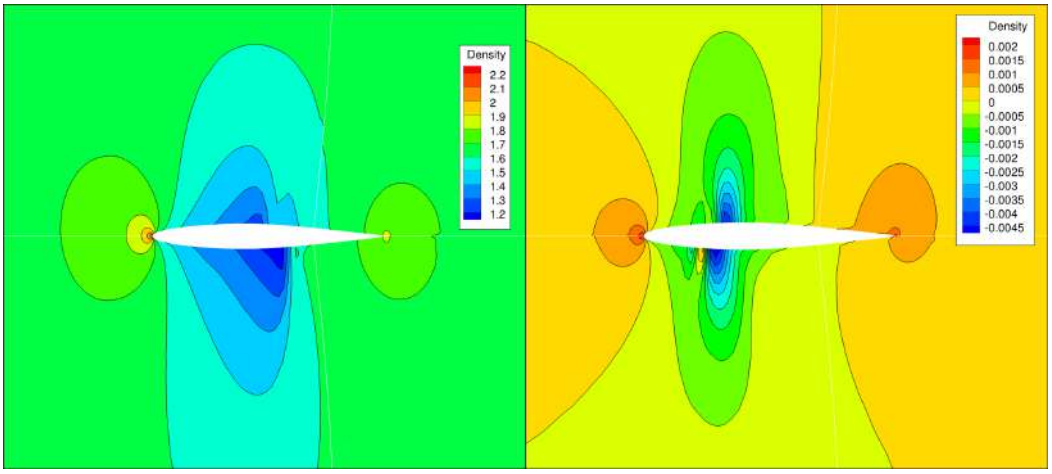
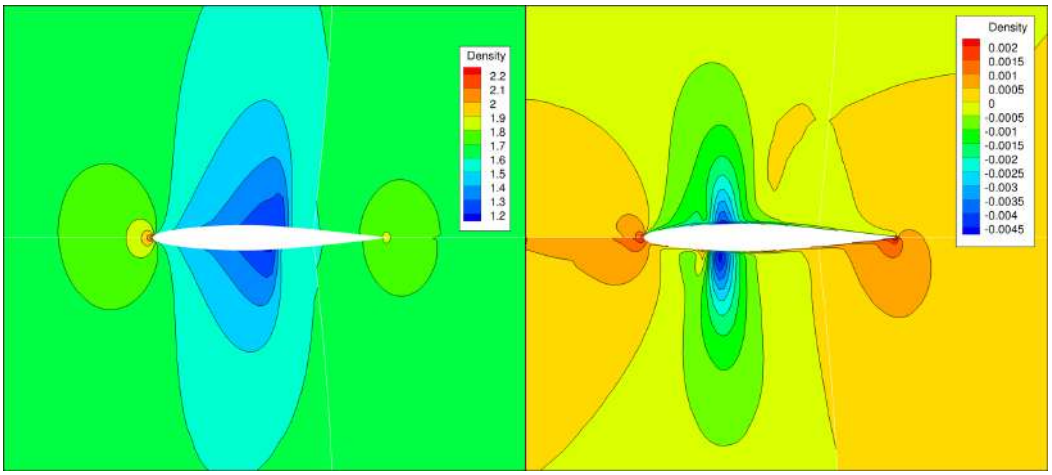


Figure 77: Convergence of the standard (left) and nested (right) GCRO-DR Krylov solvers applied to the TSM adjoint linear system for an increasing number of harmonics. The maximum size of the outer Krylov subspace is 100, the size of the recycled subspace is 30 and for the flexible solver the inner Krylov subspace is limited to 10 vectors.

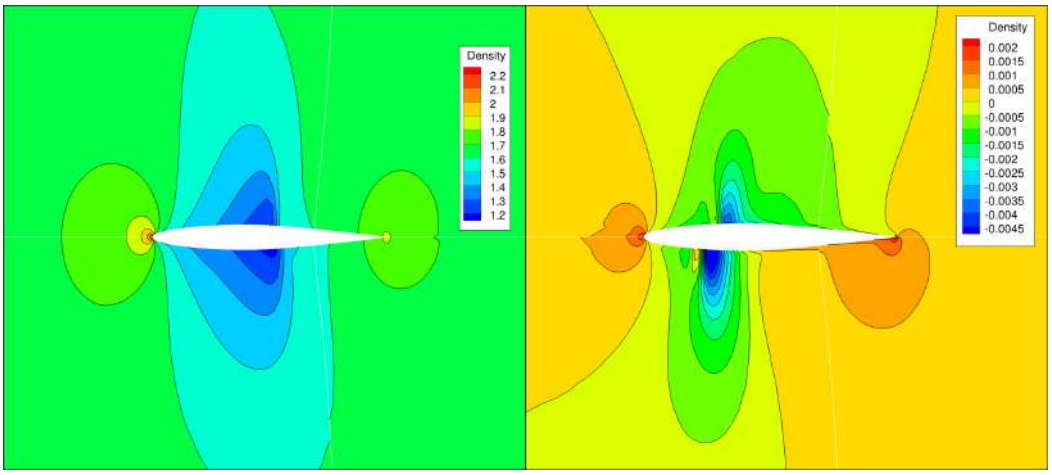
As an example, Figure 78 presents the primal and adjoint density flow solutions at the three discretization instants of the single-harmonic approximation of the unsteady motion of our NACA64 test case.



(a) Density primal flow solution, instant=0 **(b)** Density adjoint flow solution, instant=0



(c) Density primal flow solution, instant=1 **(d)** Density adjoint flow solution, instant=1



(e) Density primal flow solution, instant=2 **(f)** Density adjoint flow solution, instant=2

Figure 78: Primal and adjoint density flow solutions for a 1-harmonic spectral approximation of the forced harmonic pitching motion of the NACA64 airfoil.

Numerical validation

The basic building blocks of the TSM adjoint solver have been validated numerically by carrying out a systematic comparison with finite difference approximations. In addition, the duality checks between the full primal TSM system matrix and its adjoint counterpart show a relative precision of 8 significant digits. To further assess the accuracy of the assembled TSM adjoint total derivatives, we have implemented a simple parameterization of the NACA64 airfoil shape. Our objective is to compare with gradients of the aerodynamic pressure drag coefficient C_{Dp} obtained by finite differences. The parameterization is performed using the PADGE CAD modeler kindly provided by Airbus in the context of NEXTAIR. The PADGE framework also provides the wall grid sensitivities with respect to the design variables. We can then easily apply a volume mesh deformation operator to this quantity to obtain the volume mesh derivative $d\mathbf{X}/dp$ required for the assembly of the total gradient of the objective function (see equation 5.1.20).

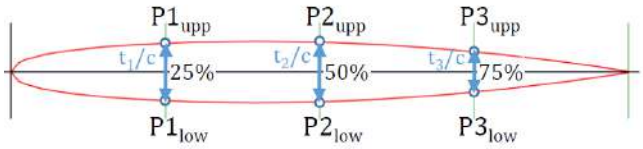
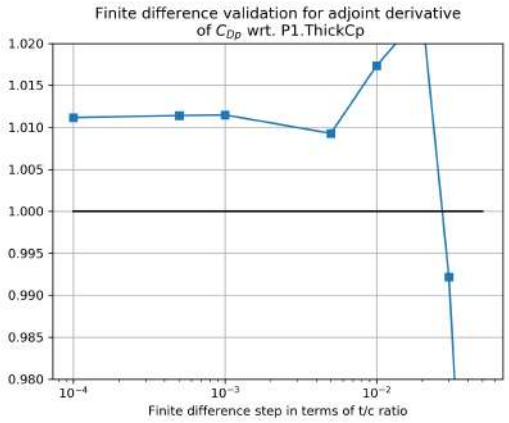
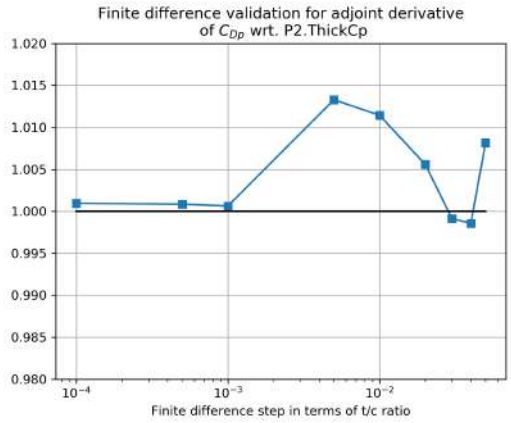


Figure 79: Simple shape parameterization of the NACA64 airfoil for validation of adjoint derivatives. Three design variables control the thickness-to-chord ratio at sections located at 25, 50 and 75% of the chord.

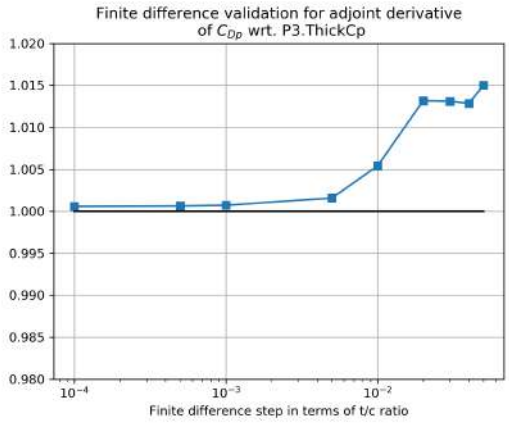
We first compute the TSM solution using a single harmonic for the spectral approximation (i.e. 3 instants in the temporal period). To get the best accuracy, we ask for a relative decrease of 10 orders of magnitude for the residuals of the TSM primal and adjoint solvers. In figure 80 the time-average derivatives of the pressure drag coefficient obtained by the finite differences and the adjoint strategies are compared. Reported values are scaled by the adjoint ones. The latter then appear as horizontal lines at a constant unit value. Typically, the finite difference approximations show good convergence for increments lower than $1. \times 10^{-3}$. In this range, a very good agreement is observed with less that 1% of maximum discrepancy.



(a) t/c at 25%



(b) t/c at 50%



(c) t/c at 75%

Figure 80: Finite difference validation for the 3 instant time-average adjoint derivative of C_{Dp}. The three parameters control the relative thickness of the airfoil at 25, 50 and 75% of chord respectively. Blue plain curves correspond to FD derivatives scaled by adjoint ones.

5.1.3 Conclusion

In this section a modular implementation of scalable, parallel in time and space, TSM primal and adjoint solvers were presented. The implementation is versatile as this framework can be easily connected to several CFD solvers like elsA or CODA. We also exploited the modularity of this implementation, thereby simplifying the use of an external linear algebra package such as PetSc. The prototype of adjoint solver is a new capability that was not available yet at ONERA. As such, we put great attention to the numerical validation of this new solver by performing systematic comparison with finite differences on a simplified airfoil shape parameterization.

5.2 Data compression techniques by NTUA

NTUA developed and used compression techniques (applied to the primal unsteady solver results) to relax the storage requirements of the unsteady adjoint method in larger scale industrial applications, such as those involved in NEXTAIR. Given that the unsteady adjoint equations are integrated backward in time, the adjoint solver should have access to the instantaneous flow fields at each and every time-step. Here we refer to the flow field time-series which are previously computed by advancing the

primal equations forward in time on the same geometry. Mainstream treatments of this issue are:

(a) Full storage of the primal solution time-series, provided that the available hardware can support this; storage is maximized, but with zero recomputation penalty. For industrial applications, this is usually not feasible.

(b) No-storage of the primal solution time-series; at each time-step of the adjoint solver, the instantaneous flow solution must be recomputed, starting from the initial state. However, depending on the problem size, the optimization turn-around time may become prohibitively high. This is discussed here just for the sake of completeness.

(c) Use of the Binomial Check-Pointing (22) technique, which is the state-of-the-art method in industrial applications. Check-pointing strategically stores the flow solution at a number of optimally placed and regularly updated time-instants (check-points) and recomputes the rest. While this alleviates huge storage requirements, it introduces an overhead due to flow recomputations. Depending on the size of the problem and the check-pointing settings, this overhead might become non-negligible, especially in large-scale applications. In the following discussion, the (binomial) check-pointing technique is abbreviated to *BChP(S_p)*, where S_p stands for the user-selected number of check-points.

In order to avoid flow recomputations, a viable alternative is to store the primal solution (or, at least, part of it, as in check-pointing) in compressed form (30, 78). Lossless or lossy compression algorithms can be used for this purpose. As the gain from lossless compression is expected to be limited (31), lossless and lossy compression techniques are herein synergistically used. In this regard, the *iPGDZ*$^+$ (lossy) compression kernel, developed by the NTUA group in (43, 44, 45), is used herein by adapting this to the unsteady adjoint code of PUMA, as this was originally developed in OpenFOAM, for incompressible flows. *iPGDZ*$^+$ is demonstrated herein in compressible flows for the first time. *iPGDZ*$^+$ comprises three compression phases, successively based on: (a) the incremental Proper Generalized Decomposition (iPGD) (43), (b) ZFP (35) and (c) Zlib (18), see section 5.2.1. It should be noted that in adjoint-based optimization, it is important to ensure that errors in the flow fields, due to a lossy compression/decompression, will affect the accuracy of neither the adjoint solution nor the computed Sensitivity Derivatives (SDs). This is, among other, included in this report.

In this report, compression of the flow solution time-series is carried out using the *Compressed Full Storage* (*CFS*) technique (43, 44), with *iPGDZ*$^+$ serving as the main building block. *CFS* leverages the *iPGDZ*$^+$ algorithm to store the entire time-history of the flow solution in memory or on a hard-disk. Sections 5.2.1 and 5.2.2 briefly outline the *iPGDZ*$^+$ algorithm and the *CFS* technique, which is applicable to both periodic and non-periodic unsteady flows.

Even greater storage savings can be achieved by combining the *iPGDZ*$^+$ algorithm with check-pointing, which is referred to as the *Compressed Coarse-grained Check-Pointing* (*3CP*) technique. Developed recently by NTUA (45), *3CP* can decrease storage requirements by nearly an order of magnitude compared to *CFS*, albeit at the expense of a few recomputations. One should keep in mind that the *3CP* technique was developed to handle large-scale applications in which data size exceeds the available memory capacity, even after compression by *CFS*. Given that the entire time-history of the flow solution compressed by the *CFS* can be stored on the hard-disk in the cases under examination, *3CP* is not explored in this report. Generally, when data is stored on the hard-disk, the primary objective of compression techniques is to reduce the cost of data transmission to and from the hard-disk. In this respect, *3CP* is not expected to offer any advantage over *CFS*.

5.2.1 The *iPGDZ*$^+$ Algorithm

The Proper Generalized Decomposition (PGD) algorithm (6) was initially proposed to compress structured data (such as, CFD solutions on structured grids). For instance, on a structured grid, the curvilinear

coordinates might give three families of spatial modes. Thus, eq. (5.2.1) would be formed by the product of four modes (one temporal and three spatial. Since we are dealing with unstructured grids, we decided to have a single spatial mode X^{μ}, which stands for all nodes (recall that a vertex-centered finite volume is used in PUMA), with nodes numbered with their ID (in the unstructured grid description). Thus, each field $f(i,k)$ is approximated by the sum of M products of spatial $X(i)$ and temporal $T(k)$ modes, where i is the (integer) vertex-ID and k the time-step counter, reading

$$f\,(i,k) \simeq \sum_{\mu=1}^{M} X^{\mu}(i)\,T^{\mu}(k) \tag{5.2.1}$$

The number M of modes is user defined. Standard PGD requires the whole time-history of the flow to be available prior to its compression. It is though obvious that, if this storage capacity was available, there is no reason for the fields to be compressed. This is why the incremental variant of PGD (iPGD) was devised (43); this updates the computed modes each time a new instantaneous field is computed, i.e. once the flow solution at the next time-step becomes available. In specific, once the solution field at the new $(L+1)$ time-step becomes available, a new element T^{m}_{L+1}, $m \in [1,M]$, is computed and modes T^{m}_{k}, $k \in [1,L]$ and X^{m}_{i}, $i \in [1,I]$, where I is the total number of grid cells, are simultaneously updated. The iterative computation of (X^{m},T^{m}), $m \in [1,M]$ targets the minimization of the error function (43)

$$E_m = \frac{1}{2}\underbrace{\sum_{i=1}^{I}\left[\sum_{\mu=1}^{m} X_i^{\mu}T_{L+1}^{\mu} - f_{i,L+1}\right]^2}_{\substack{\text{Error corresponding to the new}\\ \text{time-instant } (L+1)}} + \frac{w}{2}\underbrace{\sum_{i=1}^{I}\sum_{k=1}^{L}\left[\sum_{\mu=1}^{m} X_i^{\mu}T_{k}^{\mu} - f_{i,k}^{iPGD}\right]^2}_{\substack{\text{Error corresponding to all previous}\\ \text{time instants } (1,\ldots,L)}} \tag{5.2.2}$$

The $f_{i,k}^{iPGD}$ fields are reconstructed on the fly as $\sum_{\mu=1}^{M}\widetilde{X}_i^{\mu}\widetilde{T}_k^{\mu}$, $i \in [1,I]$, $k \in [1,L]$, using the previous approximation to $f_{i,k}$, based on $\widetilde{X}$ and $\widetilde{T}$. w a user-defined weight factor; in all set-ups of this report, $w=1$. The unknown modes $\left(X_i^{m},\ T_k^{m}\right.$ and $\left.T_{L+1}^{m}\right)$ are computed by iteratively solving the $\partial E_m/\partial X_i^{m} = \partial E_m/\partial T_k^{m} = \partial E_m/\partial T_{L+1}^{m} = 0$ equations, in a segregated manner (43).

After compressing a time-series with iPGD, the spatial and temporal modes are further compressed by successively using ZFP (35) and Zlib (18). Practically, the modes are stored in compressed form. First, each mode of size l is transformed into a 2D array of size $4\lceil l/4\rceil$ ($\lceil\ \rceil$ is the ceiling function) and, then, lossly compressed using the *fixed-precision* mode of ZFP with a user-defined number (P) of bits (same for all modes). The resulting data stream is losslessly compressed using the fastest level of the Zlib library, prioritizing compression/decompression speed over data reduction. The main steps of the *iPGDZ*$^{+}$ algorithm are sketched in fig. 81.

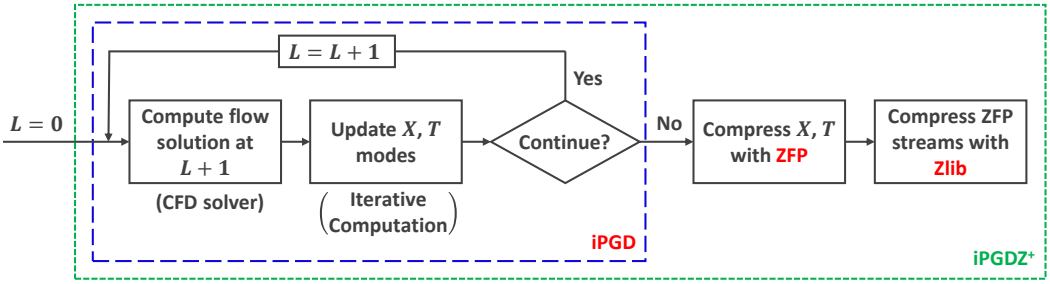


Figure 81: Main steps of the *iPGDZ*$^{+}$ compression algorithm.

The ZFP and Zlib algorithms are briefly presented in this paragraph. ZFP (35) may lossly compress integer and floating-point data stored in d-dimensional arrays, with $d \in [1,4]$. These are partitioned into

blocks of 4^d values each and compressed. The ZFP follows the typical structure of lossy compression algorithms, including decorrelation, quantization, and encoding stages. On the other hand, Zlib (18) is a general purpose lossless compression library that implements the Deflate algorithm (14). The latter compresses a stream consisting of a series of blocks of data using a combination of the LZ77 algorithm and Huffman encoding (61) on each block. Zlib handles all data types as a continuous stream of bytes. Repeated sequences of bytes within a stream are replaced with references to a single copy of that data existing earlier in the uncompressed stream.

It is important to note that none of the constituents of the *iPGDZ*$^+$ algorithm requires parallel communications, as they only work with the data available on each MPI rank. Hence, no scalability issues may arise by increasing the MPI ranks.

5.2.2 The *CFS* Technique

In the *CFS* technique by NTUA, data are compressed by the previously described *iPGDZ*$^+$ algorithm and the whole time-history of the flow solution is stored in memory or in hard-disk. For the *iPGDZ*$^+$ to retain its compression accuracy as the total number of time-steps increases, a higher number of modes (M) becomes necessary, which increases the compression/decompression cost per time-step. To address this challenge, as proposed in (43), an additional action is optionally taken. The time-horizon is partitioned into consecutive, non-overlapping time-windows, with a user-defined number K of time-steps each (likely excluding the last one), as depicted in fig. 82. Each time-window will have a different set of modes (of the same number M, though), as it is compressed independently from the rest.

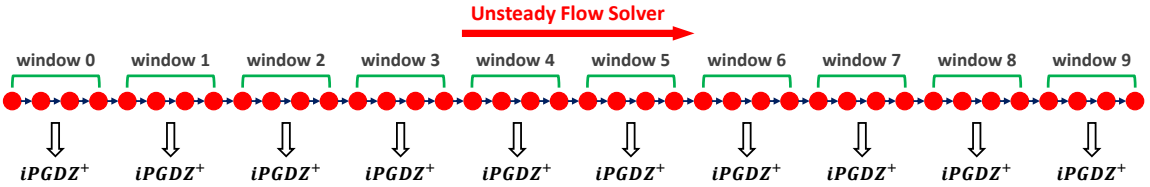


Figure 82: Implementation of *iPGDZ*$^+$ within *CFS*. In this quite simple explanatory example, the time-horizon with 40 time-steps in total is partitioned into 10 time-windows.

In section 5.2.3, all tested *CFS* variants use a precision of $P = 16$ bits for ZFP. Thus, *CFS* will be denoted as *CFS*(M,K) to specify the number M of modes and the number K of time-steps per time-window (if the time-horizon splits into time-windows).

5.2.3 Applications

The first application to demonstrate the capabilities of the *CFS* compression algorithm is the prediction of the flow around a pitching airfoil as well as the computation of the gradient of aerodynamic objective functions using continuous adjoint. The selected airfoil profile is that of NACA64010 with a forced pitching motion around its chord quarter ($c/4$) with a rotation angle determined by $\theta = A_m + A_0 sin\left(\frac{2\pi}{T}t\right)$ where $A_m = 0.0^o$, $A_0 = -2.0^o$ and $T = 0.2sec.$ The farfield conditions of the case are $M_\infty = 0.82$, angle of attack $a = 0.0^0$ and Reynolds number based on the airfoil chord $Re = 4.1 \cdot 10^6$. The computational mesh around the airfoil can be seen in figure 83. It consists of $86K$ triangular and $16K$ quadrilateral elements, with $59K$ nodes in total. The latter are placed next to the airfoil to accurately simulate the developed boundary layer.

The NTUA in-house GPU-accelerated PUMA flow analysis and adjoint code was used to compute the unsteady flow field during three periods of airfoil pitching. The Spalart--Allmaras turbulence model was employed here, while each period (T) was split into 50 time steps. The computed flow fields were

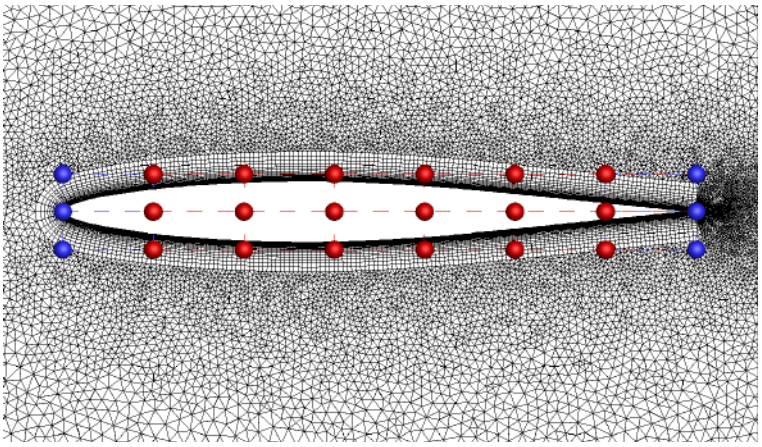


Figure 83: Computational mesh around the NACA64A010 airfoil and the control grid used.

compressed based on the *CFS* algorithm according to two different settings. In the first, 3 modes were used to compress the flow fields, with 10 successive time instants forming a time--window, abbreviated to *CFS*(3,10). The second consists of time-windows with larger size (50) which thus employ a higher number of modes (15), *CFS*(15,50). In practice, both sets use 45 modes in total, with the same disk space requirements for data storage. Figures 84, 85 show the comparison of the time histories of drag and lift coefficient computed either directly from PUMA or by first compressing and then decompressing the flow fields using the *CFS*(3,10) and *CFS*(15,50). It can be seen that, the time histories computed from the reconstructed fields are in very good agreement with those computed directly by PUMA.

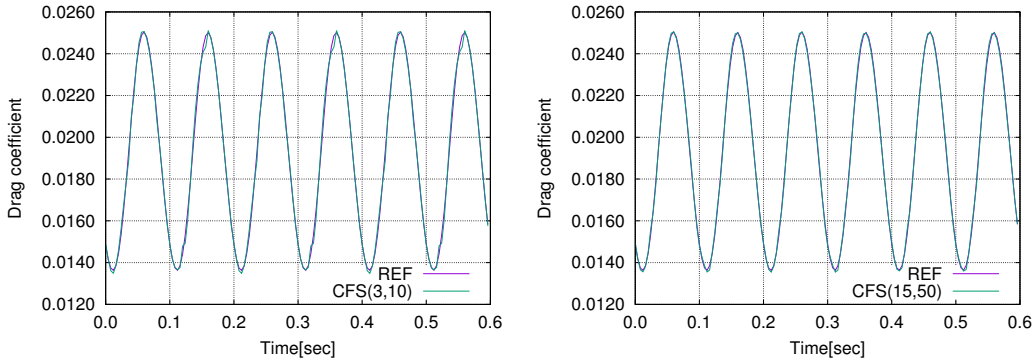


Figure 84: Comparison of the time histories of the instantaneous drag coefficient computed by PUMA, and the reconstructed flow fields using *CFS*(3,10) (left) and *CFS*(15,50) (right).

Figure 86 shows the computed Mach number fields, either by PUMA or by post--processing the reconstructed flows fields, at different time instants. Some discrepancies can be seen close to the shock wave (for instance in the first and last row of figure 86 over the suction side, especially in case of *CFS*(3,10)). Though these discrepancies were expected, it is very important that the shock wave intensity and its interaction with the boundary layer are well captured by both *CFS*(3,10) and *CFS*(15,50). It is important to comment on the reduction of the disk space requirements achieved by using data compression. As written before, both *CFS*(3,10) and *CFS*(15,50) have the same disk space requirements, which in this case, are almost the 5% of those needed to store the computed flow fields at all (150) time instants. So, there is a reduction by 95% in the disk space footprint (from $954MB$ to $49MB$).

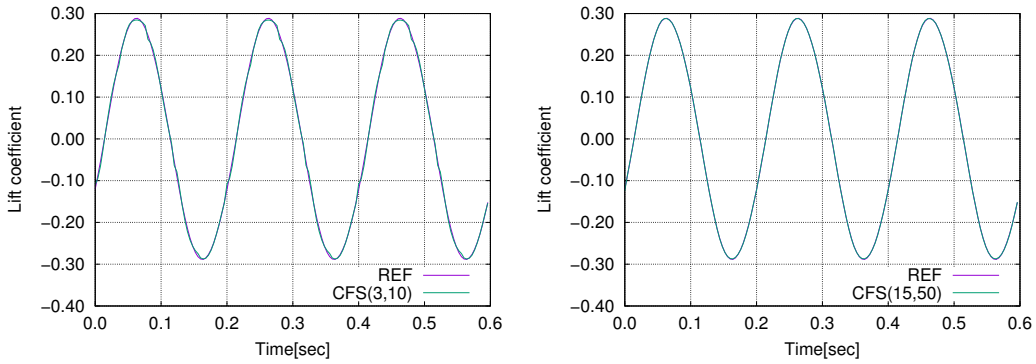


Figure 85: Comparison of the time histories of the instantaneous lift coefficient computed by PUMA, and the reconstructed flow fields using *CFS*(3,10) (left) and *CFS*(15,50) (right).

In order to assess the benefits from data compression in unsteady adjoint, the adjoint--based drag and lift coefficient sensitivities computed from the reconstructed flow fields, were compared with those computed by the flow solver. For the needs of this study, the airfoil shape was parameterized using the 8x3 control grid shown in figure 83, where, the blue control points were fixed while the red ones could move in the y-direction. Figures 87, 88 show the comparison between the adjoint--based drag and lift coefficient sensitivities computed by the above methods. It can be seen that both *CFS* settings lead to highly accurate sensitivities, with those based on *CFS*(15,50) being more accurate. Note that, in this case, all flow field instants have been compressed and stored to the disk, so no flow re-computation is needed by the adjoint solver.

The second application deals with the unsteady flow around a next generation HARW (High Aspect Ratio Wing) transport aircraft, based on the DLR-F25 model, provided by DLR. This aircraft model is representative of a short medium range transport aircraft featuring a wing aspect ratio of more than 15. For the studies carried out here, the wing-body-HTP (Horizontal Tail Plane) configuration is considered. The farfield Mach number is $M_\infty = 0.78$, while the Reynolds number based on a mean aerodynamic chord of $3.535m$ is $Re = 2.0887 \cdot 10^7$. The angle of attack oscillates sinusoidally around $a = 1.5^o$ with an amplitude of $a_m = 0.5^o$ and period $T = 1.0s$, as $a = a_0 + a_m sin\,(2\pi t/T)$. The computational mesh used was provided by DLR and consists of $\sim 4.0M$ nodes, with $\sim 2.3M$ tetrahedra, $\sim 80K$ pyramids, $\sim 12K$ prisms and $\sim 3.5M$ hexahedra. As in the first application, the flow analysis and adjoint solver of PUMA was used.

Three compression settings, with low to high compression rates, are evaluated, namely *CFS*(5,10), *CFS*(3,10) and *CFS*(3,25). Figures 89(left), 90(left) present the lift and drag coefficient time histories computed using PUMA for three simulation periods. After reconstructing the flow fields using *CFS*(5,10), *CFS*(3,10) and *CFS*(3,25), the resulting lift and drag coefficient time histories were compared with those computed by PUMA. The corresponding absolute errors are presented in figures 89(right) and 90(right). As expected, the more modes used for compression, the more accurate the reconstructed lift and drag coefficient time histories. The absolute error of the reconstructed drag and lift coefficients based on *CFS*(5,10) is less than 0.1%. It is impressive, though, that the reconstructed drag and lift coefficients based on *CFS*(3,25) are highly accurate too. For instance, concerning drag coefficient, the error of the reconstructed results does not exceed 1.0% or 3 drag counts.

Figure 91 presents a series of instantaneous pressure coefficient fields computed on the aircraft wing suction side using PUMA (top row) or reconstructed using *CFS*(3,25) (bottom row). Differences between the two fields can barely be seen.

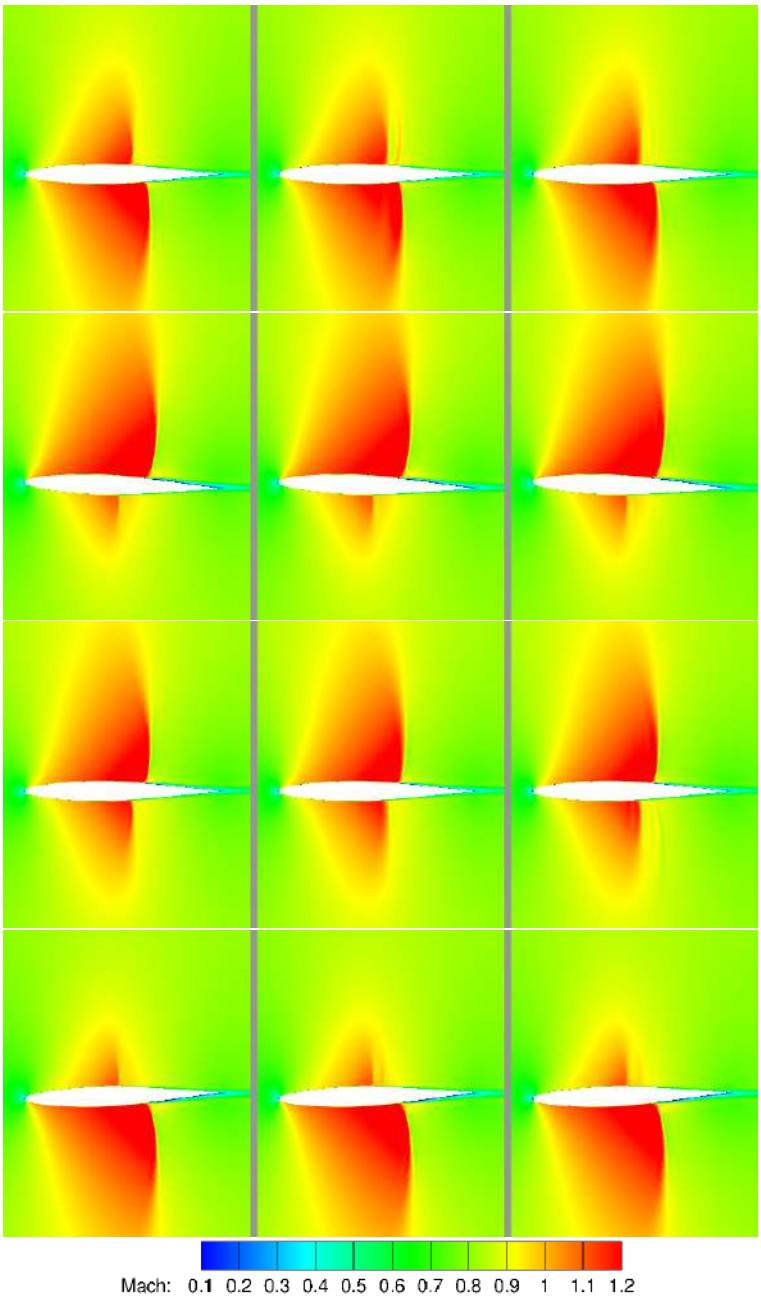


Figure 86: Instantaneous Mach number fields around the NACA64A010 airfoil computed by PUMA (left column) or the reconstructed flow using *CFS*(3,10) (middle) and *CFS*(15,50) (right), at four different time instants. The first (top) row corresponds to $\theta = 0.0^o$ (nose moving upwards), the second to $\theta = -2.0^o$, the third to $\theta = 0.0^o$ (nose moving downwards) and the fourth (bottom) to $\theta = 2.0^o$.

In order to evaluate the impact of data compression on the unsteady adjoint solver performance, the aircraft's wing was parameterized and, then, the unsteady adjoint solver of PUMA was launched four times using either the flow fields computed by PUMA or the reconstructed ones (three settings). The computed drag coefficient sensitivities were compared. To parameterize the wing and get a reasonably small number of design variables for which the sensitivity derivatives are computed, the $4\times7\times3$ control grid of figure 92 was used, with the two rows of control points close to the wing--fuselage intersection remaining fixed and the others controlling the twist of the 5 (blue) wing cross sections.

Figure 93(left) shows the computed time--averaged drag coefficient sensitivities w.r.t. the twist

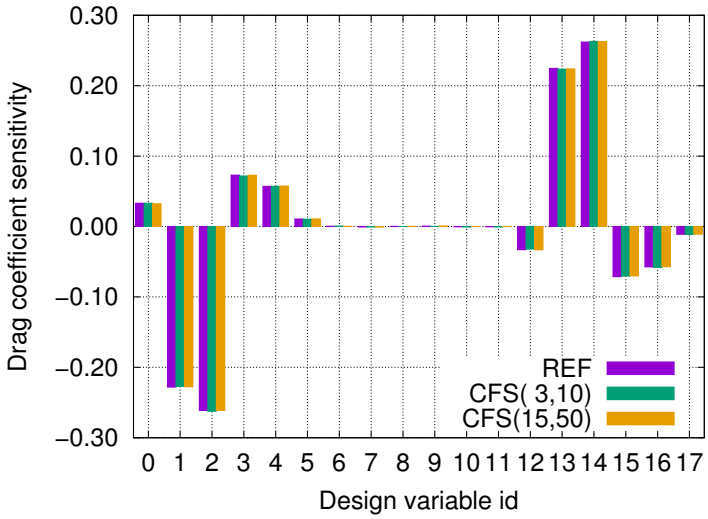


Figure 87: Comparison of the adjoint--based drag coefficient sensitivities computed by PUMA (REF) or using the reconstructed flow fields with the three *CFS* settings.

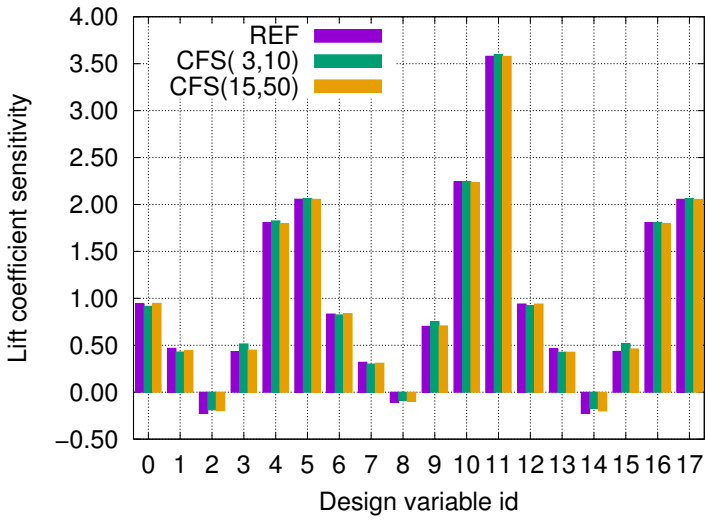


Figure 88: Comparison of the adjoint--based lift coefficient sensitivities computed by PUMA (REF) or using the reconstructed flow fields with the three *CFS* settings.

angles of the 5 cross sections as computed based on the flow fields from PUMA. The error of the corresponding sensitivities based on the reconstructed flow fields is shown in figure 93(right). Using the reconstructed flow fields, there is no significant loss in the accuracy of sensitivities, with an error which is less than 0.05%, 0.10% or 0.50% using *CFS*(5,10), *CFS*(3,10) or *CFS*(3,25), respectively. At the same time, the disk space requirements are reduced from $32GB$ (if all flow fields computed by PUMA had to be stored) to $2.7GB$ in case of *CFS*(5,10), $1.6GB$ in case of *CFS*(3,10), or only $638MB$ in case of *CFS*(3,25). Consequently, the last case results in $\sim 98\%$ reduction in the hard-disk footprint.

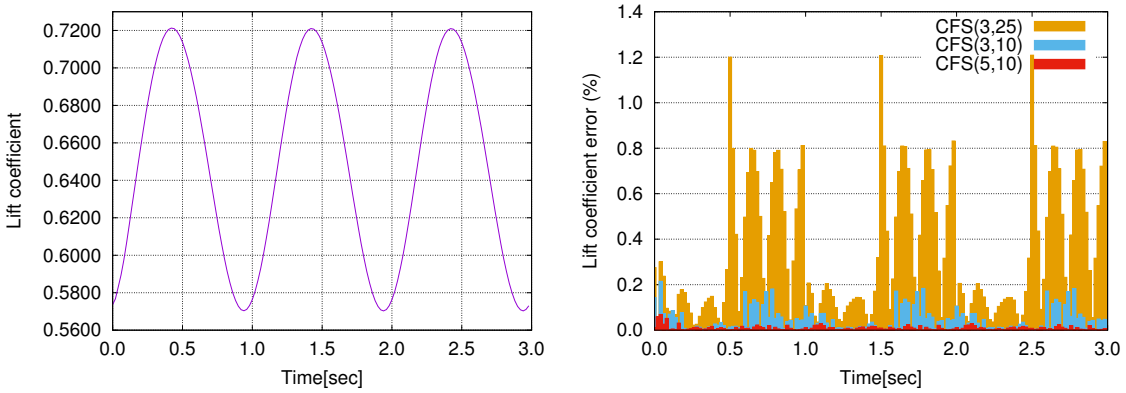


Figure 89: Instantaneous lift coefficient time history computed using PUMA (left), and the absolute error of the reconstructed lift coefficient time histories based on *CFS*(5,10), *CFS*(3,10) and *CFS*(3,25).

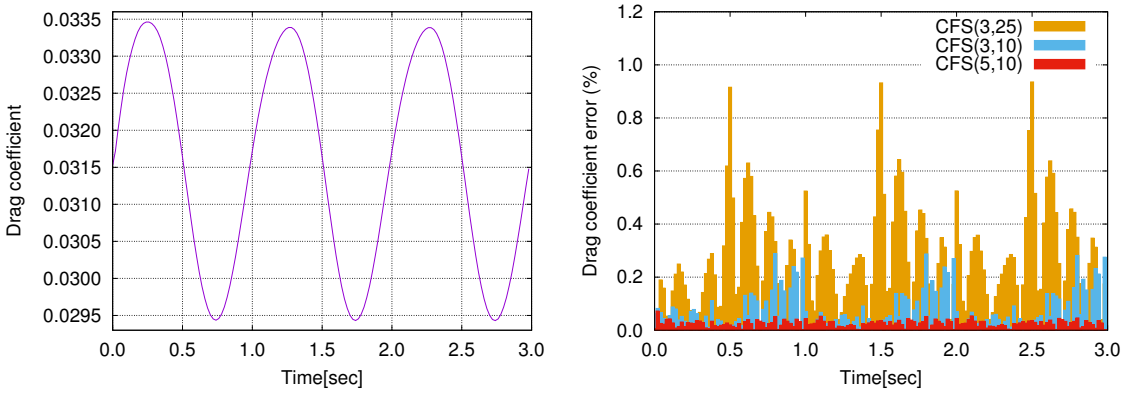


Figure 90: Instantaneous drag coefficient time history computed using PUMA (left), and the absolute error of the reconstructed drag coefficient time histories based on *CFS*(5,10), *CFS*(3,10) and *CFS*(3,25).

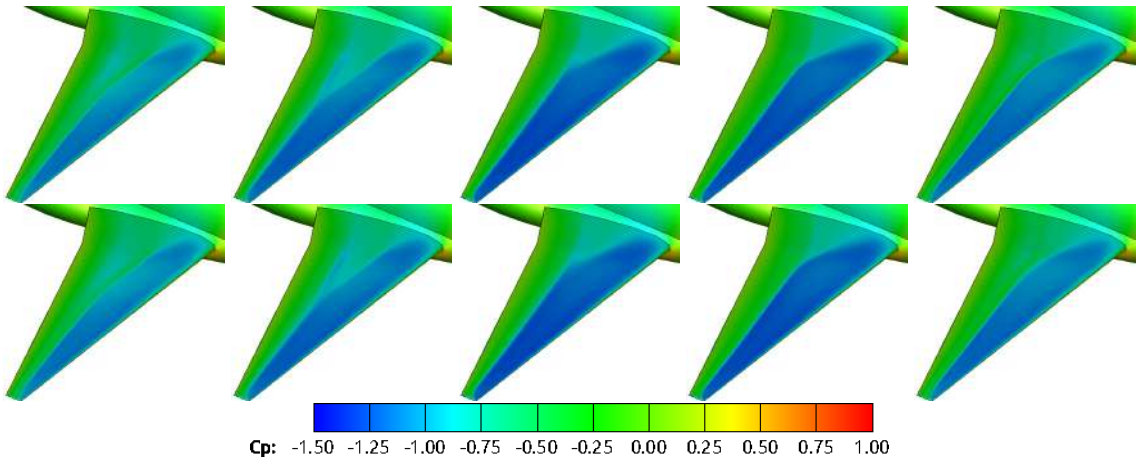


Figure 91: Instantaneous pressure coefficient fields computed on the aircraft wing suction side using PUMA (top) or reconstructed using *CFS*(3,25) (bottom) for different time instants, namely at $\frac{0}{5}T$, $\frac{1}{5}T$, $\frac{2}{5}T$, $\frac{3}{5}T$ and $\frac{4}{5}T$ from left to right.

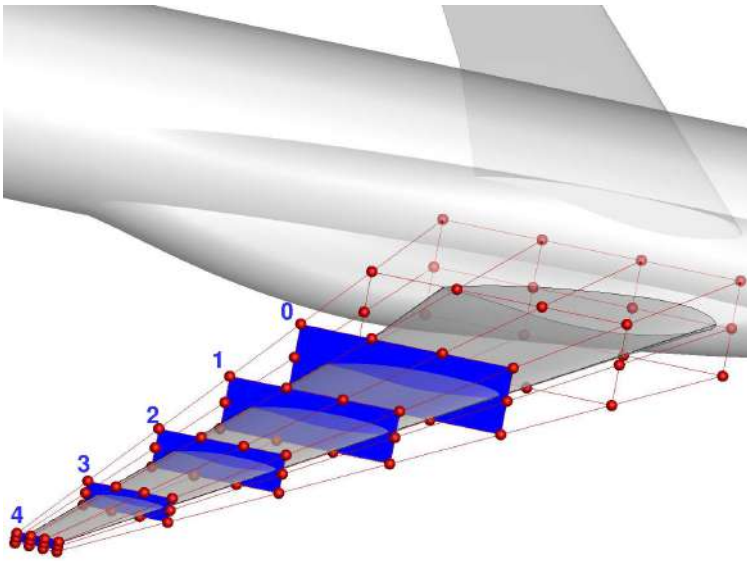


Figure 92: View of the control grid used to modify the twist of 5 wing sections.

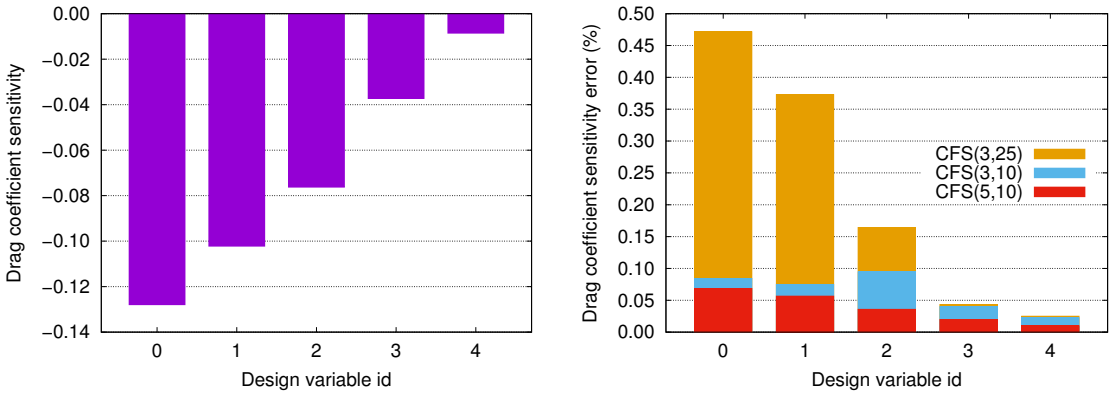


Figure 93: Adjoint--based, time--averaged drag coefficient sensitivities computed by PUMA (left). Absolute error of the corresponding sensitivities computed from the reconstructed flow fields (right).

5.3 IRT contribution

IRT Saint Exupery has developed an interface to ordinary differential equation solvers for GEMSEO. The adjoint interface to these ODE solvers is under development, using the PetsC (www.petsc.org) TSadjoint library. This will allow the derivatives of the ODE solution to be computed with respect to the design variables. The work is still ongoing, all the interfaces have been developed, but the validation is not yet working. We are facing IT problems, the Python interface to the PetsC library is very powerful in terms of features, but difficult to debug and not well documented. The validation results will be presented in the technical report of the first reporting period.

Bibliography

(1) F. Alauzet and M. Mehrenberger. P1-conservative solution interpolation on unstructured triangular meshes. *International Journal for Numerical Methods in Engineering*, 84(13):1552--1588, 2010.

(2) I. Armstrong and T. Edmunds. Fully automatic analysis in the industrial environment. In *Proceedings of the Second International Conference of Quality Assurance in Finite Element Analysis*, Stratford-upon-Avon, England, 1989.

(3) Satish Balay, Shrirang Abhyankar, Mark F. Adams, Steven Benson, Jed Brown, Peter Brune, Kris Buschelman, Emil M. Constantinescu, Lisandro Dalcin, Alp Dener, Victor Eijkhout, Jacob Faibussowitsch, William D. Gropp, Václav Hapla, Tobin Isaac, Pierre Jolivet, Dmitry Karpeev, Dinesh Kaushik, Matthew G. Knepley, Fande Kong, Scott Kruger, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Lawrence Mitchell, Todd Munson, Jose E. Roman, Karl Rupp, Patrick Sanan, Jason Sarich, Barry F. Smith, Stefano Zampini, Hong Zhang, Hong Zhang, and Junchao Zhang. PETSc Web page. `https://petsc.org/`, 2023.

(4) To Thanh Binh and Ulrich Korn. Mobes: A multiobjective evolution strategy for constrained optimization problems. In *The third international conference on genetic algorithms (Mendel 97)*, volume 25, page 27, 1997.

(5) L Cambier, M Gazaix, S Heib, S Plot, M Poinot, JP Veuillot, JF Boussuge, and M Montagnac. An overview of the multi-purpose elsa flow solver. *AerospaceLab*, (2):p--1, 2011.

(6) F. Chinesta, R. Keunings, and A. Leygue. *The Proper Generalized Decomposition for Advanced Numerical Simulations: A Primer*. Springer, 2014.

(7) Thomas M. Cover and Peter E. Hart. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory*, 13:21--27, 1967.

(8) V. D'Alessandro, S. Montelpare, and R. Ricci. Assessment of a Spalart-Allmaras model coupled with local correlation based transition approaches for wind turbine airfoils. *Applied Sciences*, 11(4), 2021.

(9) Indraneel Das and John E Dennis. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM journal on optimization*, 8(3):631--657, 1998.

(10) Sanford S. Davis. NACA 64A010 (NASA Ames model) oscillatory pitching. *AGARD report*, 702, 1982.

(11) Kalyanmoy Deb. Multi-objective optimisation using evolutionary algorithms: an introduction. In *Multi-objective evolutionary optimisation for product design and manufacturing*, pages 3--34. Springer, 2011.

(12) Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182--197, 2002.

(13) A. A. Demargne, R. O. Evans, P. J. Tiller, and W. N. Dawes. Practical and reliable mesh generation for complex, real world geometries. In *AIAA 52nd Aerospace Sciences Meeting*, National Harbor, Maryland, USA, 2014.

(14) P. Deutsch. DEFLATE Compressed Data Format Specification version 1.3. *RFC 1951*. , 1996.

(15) N. Dunham. CFD validation for propulsion system components. Technical report, AGARD advisory report, 1998.

(16) Jean-Antoine Désidéri. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5):313--318, 2012.

(17) Carlos M Fonseca and Peter J Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary computation*, 3(1):1--16, 1995.

(18) J. Gailly and M. Adler. Zlib Compression Library, 2023.

(19) F. Gallard, C. Vanaret, D. Guenot, V. Gachelin, R. Lafage, B. Pauwels, and A. Gazaix. GEMS: A python library for automation of multidisciplinary design optimization process generation. In *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2018.

(20) Francois Gallard, Charlie Vanaret, Damien Guenot, Vincent Gachelin, Remi Lafage, Benoit Pauwels, Pierre-Jean Barjhoux, and Anne Gazaix. Gems: a python library for automation of multidisciplinary design optimization process generation. In *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, page 0657, 2018.

(21) Stefan Görtz, Tobias Leicht, Vincent Couaillier, Michael Méheut, Pascal Larrieu, and Steeve Champagneux. Coda: A european perspective for a next-generation cfd, analysis and design platform. In *NATO AVT-366 Workshop on Use of Computational Fluid Dynamics for Design and Analysis: Bridging the Gap Between Industry and Developers*, May 2022.

(22) A. Griewank and A. Walther. Algorithm 799: Revolve: An Implementation of Checkpointing for the Reverse or Adjoint Mode of Computational Differentiation. *ACM Transactions on Mathematical Software*, 26(1):19--45, 2000.

(23) Andreia P. Guerreiro, Carlos M. Fonseca, and Luís Paquete. The hypervolume indicator: Problems and algorithms. *CoRR*, abs/2005.00515, 2020.

(24) Ansys Inc. Ansys icem cfd. r2. Technical report, Ansys Inc., 2021.

(25) Mehdi Jadoui, Christophe Blondeau, Emeric Martin, Florent Renac, and Francois-Xavier Roux. Comparative study of inner--outer krylov solvers for linear systems in structured and high-order unstructured cfd problems. *Computers & Fluids*, 244:105575, 2022.

(26) A. John, G. Vivarelli, N. Qin, and S. Shahpar. Using feature-based mesh adaptation to improve the adjoint optimisation of transonic compressor blades. In *Proceedings of ASME Turbo Expo 2020: Turbomachinery Technical Conference and Exposition*, Virtual event, 2020.

(27) Pierre Jolivet and Pierre-Henri Tournier. Block iterative methods and recycling for improved scalability of linear solvers. In *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 190--203. IEEE, 2016.

(28) M.G. Kontou. *The Continuous Adjoint Method with Consistent Discretization Schemes for Transitional Flows and the Use of Deep Neural Networks in Shape Optimization in Fluid Mechanics*. PhD thesis, National Technical University of Athens, Athens, 2023.

(29) D. Kraft. A software package for sequential quadratic programming. Technical Report DFVLR-FB 88-28, Institut für Dynamik der Flugsysteme, 1988.

(30) N. Kühl, H. Fischer, M. Hinze, and T. Rung. An Incremental Singular Value Decomposition Approach for Large-Scale Spatially Parallel & Distributed but Temporally Serial Data - Applied to Technical Flows. *ArXiv*, abs/2302.09149, 2023.

(31) N. Kukreja, J. Hückelheim, M. Louboutin, J. Washbourne, P. Kelly, and G. Gorman. Lossy Checkpoint Compression in Full Waveform Inversion: A Case Study with ZFP v0.5.5 and the Overthrust Model. *Geoscientific Model Development*, 15:3815--3829, 2022.

(32) R. Langtry and F. Menter. Correlation-based transition modeling for unstructured parallelized computational fluid dynamics codes. *AIAA Journal*, 47:2894--2906, 12 2009.

(33) L. Lapworth. Hydra CFD: a framework for collaborative CFD development. In *International Conference on Scientific and Engineering Computation*, Singapore, 2004.

(34) Joshua I. Leffell, Jayanarayanan Sitaraman, Vinod K. Lakshminarayan, and Andrew M. Wissink. Towards efficient parallel-in-time simulation of periodic flows. In *54th AIAA Aerospace Sciences Meeting*, number AIAA 2016-0066, 2016.

(35) P. Lindstrom. Fixed-Rate Compressed Floating-Point Arrays. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2674--2683, 2014.

(36) D. I. Lopez, T. Ghisu, T. Kipouros, S. Shahpar, and M. Wilson. Extending highly loaded axial fan operability range through novel blade design. *ASME Journal of Turbomachinery*, 144, 2022.

(37) A. Loseille and R. Löhner. On 3d anisotropic local remeshing for surface, volume and boundary layers. In *Proceedings of the 18th international meshing roundtable*, pages 611--630. Springer, 2009.

(38) A. Loseille and R. Lohner. Anisotropic adaptive simulations in aerodynamics. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, page 169, 2010.

(39) A. Loseille and R. Löhner. Boundary layer mesh generation and adaptivity. In *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, page 894, 2011.

(40) Carlos Lozano and Jorge Ponsin. Exact inviscid drag-adjoint solution for subcritical flows. *AIAA Journal*, 59(12):5369--5373, 2021.

(41) Trent W Lukaczyk, Paul Constantine, Francisco Palacios, and Juan J Alonso. Active subspaces for shape optimization. In *10th AIAA multidisciplinary design optimization conference*, page 1171, 2014.

(42) Zhoujie Lyu, Zelu Xu, and JRRA Martins. Benchmarking optimization algorithms for wing aerodynamic design optimization. In *Proceedings of the 8th International Conference on Computational Fluid Dynamics, Chengdu, Sichuan, China*, volume 11, page 585, 2014.

(43) A.-S.I. Margetis, E.M. Papoutsis-Kiachagias, and K.C. Giannakoglou. Lossy Compression Techniques Supporting Unsteady Adjoint on 2D/3D Unstructured Grids. *Computer Methods in Applied Mechanics and Engineering*, 387:114152, 2021.

(44) A.-S.I. Margetis, E.M. Papoutsis-Kiachagias, and K.C. Giannakoglou. The iPGDZ^{+} Technique for Compressing Primal Solution Time-series in Unsteady Adjoint -- Applications and Assessment. In *ECCOMAS Congress 2022 -- 8th European Congress on Computational Methods in Applied Sciences and Engineering*, 2022.

(45) A.-S.I. Margetis, E.M. Papoutsis-Kiachagias, and K.C. Giannakoglou. Reducing Memory Requirements of Unsteady Adjoint by Synergistically Using Check-Pointing and Compression. *International Journal for Numerical Methods in Fluids*, 95(1):23--43, 2023.

(46) Quentin Mercier. *Optimisation multicritère sous incertitudes : un algorithme de descente stochastique*. Theses, October 2018.

(47) Johann Moulin. *On the flutter bifurcation in laminar flows : linear and nonlinear modal methods*. PhD thesis, 2020. Thèse de doctorat dirigée par Sipp, Denis et Marquet, Olivier Mécanique des fluides et des solides, acoustique Institut polytechnique de Paris 2020.

(48) Nathan L. Mundis and Dimitri J. Mavriplis. Wave-number Independent Preconditioning for GMRES Time-spectral Solvers. *53rd AIAA Aerospace Sciences Meeting*, (January):1--21, 2015.

(49) Nathan L Mundis and Dimitri J Mavriplis. Toward an optimal solver for time-spectral fluid-dynamic and aeroelastic solutions on unstructured meshes. *Journal of Computational Physics*, 345:132--161, 2017.

(50) Nathan L. Mundis and Dimitri J. Mavriplis. Toward an optimal solver for time-spectral fluid-dynamic and aeroelastic solutions on unstructured meshes. *Journal of Computational Physics*, 345(April):132--161, 2017.

(51) R. Mura and S. C. Cakmakcioglu. A revised one-equation transitional model for external aerodynamics-part i: Theory, validation and base cases. In *AIAA Aviation 2020 Forum*, page 2714, 2020.

(52) Rolls-Royce plc. Point2surface userguide. internal document. Technical report, Rolls-Royce plc., 2018.

(53) Rolls-Royce plc. Hydra userguide. internal document. Technical report, Rolls-Royce plc., 2020.

(54) Rolls Royce plc. Padram userguide. internal document. Technical report, Rolls-Royce plc., 2022.

(55) Michael JD Powell. *A direct search optimization method that models the objective and constraint functions by linear interpolation*. Springer, 1994.

(56) Mickael Rivier. *Low-cost methods for constrained multi-objective optimization under uncertainty*. PhD thesis, Institut Polytechnique de Paris, 2020.

(57) Mickael Rivier, Nassim Razaaly, and Pietro Marco Congedo. Non-Parametric Measure Approximations for Constrained Multi-Objective Optimisation under Uncertainty. working paper or preprint, September 2022.

(58) R.Tyrrell Rockafellar and Stanislav Uryasev. Conditional value-at-risk for general loss distributions. *Journal of Banking & Finance*, 26(7):1443--1471, 2002.

(59) Andrzej Ruszczyński and Alexander Shapiro. *6. Risk Averse Optimization*, pages 253--332.

(60) Youcef Saad and Martin H Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856--869, 1986.

(61) K. Sayood. *Introduction to Data Compression*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 4th edition, 2012.

(62) K. Schittkowski. Nonlinear programming methods with linear least squares subproblems. In John M. Mulvey, editor, *Evaluating Mathematical Programming Techniques*, pages 200--213, Berlin, Heidelberg, 1982. Springer Berlin Heidelberg.

(63) S. Schmidt and V. Schulz. Pareto-curve continuation in multi-objective optimization. *Pacific Journal of Optimization*, 4:243--257, 2008.

(64) G. Schubauer and P. Klebanoff. Contributions on the mechanics of boundary-layer transition. Technical Report NACA-TR-1289, NASA, 1955.

(65) P. Seshadri, S. Shahpar, P. Constantine, G. Parks, and M. Adams. Turbomachinery active subspace performance maps. *ASME Journal of Turbomachinery*, 140(4), 2018.

(66) S. Shahpar. SOFT: a new design and optimisation tool for turbomachinery. In *Proceedings of Evolutionary Methods for Design, Optimisation and Control*, Athens, Greece, 2002.

(67) S. Shahpar. SOPHY: An integrated CFD based automatic design optimisation system. In *International Symposium on Air Breathing Engines (ISABE)*, Munich, Germany, 2005.

(68) S. Shahpar. Building digital twins to simulate manufacturing variation. In *Proceedings of the ASME Turbo Expos*, Virtual event, 2020.

(69) S. Shahpar and L. Lapworth. PADRAM: Parametric design and rapid meshing system for turbomachinery optimisation. In *ASME Turbo Expo and International Joint Power Generation Conference*, Atlanta, USA, 2003.

(70) Pradyumn Kumar Shukla. On the normal boundary intersection method for generation of efficient front. In *Computational Science--ICCS 2007: 7th International Conference, Beijing, China, May 27-30, 2007, Proceedings, Part I 7*, pages 310--317. Springer, 2007.

(71) Frédéric Sicot, Guillaume Puigt, and Marc Montagnac. Block-jacobi implicit algorithms for the time spectral method. *AIAA Journal*, 46(12):3080--3089, 2008.

(72) D. M. Somers. Design and experimental results for a natural-laminar-flow airfoil for general aviation applications. Technical report, 1981.

(73) P. Spalart and S. Allmaras. A one-equation turbulence model for aerodynamic flows. *La Recherche Aérospatiale*, 1:5--21, 1994.

(74) Xinrong Su and Xin Yuan. Implicit solution of time spectral method for periodic unsteady flows. *International Journal for Numerical Methods in Fluids*, 63(7):860--876, jul 2010.

(75) Emmet Tam. *kneedle: Kneedle Algorithm; detecting knees in graphs*, 2022. R package version 1.0.0.

(76) T. Terraz, A. Ribes, Y. Fournier, B. Iooss, and B. Raffin. Melissa: Large scale in transit sensitivity analysis avoiding intermediate files. In *The International Conference for High Performance Computing, Networking, Storage and Analysis (Supercomputing)*, pages 1--14, Denver, United States, November 2017.

(77) M. A. Woodgate and K. J. Badcock. Fast prediction of transonic aeroelastic stability and limit cycles. *AIAA Journal*, 45(6):1370--1381, 2007.

(78) L. Yang and S. Nadarajah. Data Compression Algorithms for Adjoint Based Sensitivity Studies of Unsteady Flows. In *5th Joint US-European Fluids Engineering Division Summer Meeting*. 5th Joint US-European Fluids Engineering Division Summer Meeting, 07 2018.

(79) Shiji Zhou, Wenpeng Zhang, Jiyan Jiang, Wenliang Zhong, Jinjie GU, and Wenwu Zhu. On the convergence of stochastic multi-objective gradient manipulation and beyond. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 38103--38115. Curran Associates, Inc., 2022.

(80) Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on mathematical software (TOMS)*, 23(4):550--560, 1997.

(81) Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M Fonseca, and Viviane Grunert Da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on evolutionary computation*, 7(2):117--132, 2003.