A Newton–Krylov Solver for Robust Turbomachinery Aerodynamic Analysis

Shenren Xu*[1,2], Pavanakumar Mohanamuraly †[3], Dingxi Wang ‡[2], and Jens-Dominik Müller §[3]
[1]*Yangtze River Delta Research Institute of NPU, Northwestern Polytechnical University, Taicang 215400, P.R. China*
[2]*Shaanxi Key Laboratory of Internal Aerodynamics in Aero-engines, 1 Dongxiang Road, Xi'an 710072, P.R. China*
[3]*School of Engineering and Materials Science, Queen Mary University of London, London E1 4NS, United Kingdom*

Steady computational fluid dynamics solvers based on the Reynolds-averaged Navier–Stokes equations are the primary workhorse for turbomachinery aerodynamic analysis due to their good engineering accuracy at a low computational cost. However, even state-of-the-art steady solvers suffer from convergence slowdown or failure when applied to challenging off-design conditions. This severely limits the reliable nonlinear and linearized turbomachinery aerodynamic analysis over a wide operating range. To alleviate the convergence difficulties, a nonlinear flow solver using the Newton–Krylov method is developed. This is the first time the Newton–Krylov algorithm is used for achieving robust analysis of turbomachinery aerodynamics in the open literature. The proposed solution algorithm features (i) the exact Jacobian matrix forming, (ii) straightforward parallelization, and (iii) a reliable globalization strategy, and aims to achieve fast machine-zero convergence. The solver accuracy is validated using four test cases: an airfoil, a linear turbine cascade, a centrifugal compressor, and an axial compressor. Machine-zero convergence is achieved for all cases over a wide range of operating conditions without manual intervention. The method shows great potential for enabling automated and reliable whole-map turbomachinery aerodynamic analysis and paves the way for robust and efficient linearized aerodynamic analysis, such as adjoint, time-linearized and eigenvalue analysis.

Nomenclature

B	=	rotational transformation matrix
c_0	=	Jacobian blending coefficient
c_p	=	specific heat
e	=	internal energy
E	=	total energy

*Associate Professor (Email: shenren_xu@nwpu.edu.cn)
†PhD student
‡Professor
§Reader

$\mathbf{F}_c,\ \mathbf{F}_c^r$	=	convective flux in an absolute and relative reference frame
$\mathbf{F}_v^r$	=	viscous flux in an relative reference frame
$\mathbf{F}_\omega$	=	additional body force due to rotation
$h,\ H$	=	enthalpy, total enthalpy
k	=	thermal conductivity coefficient
m	=	Krylov vector number in GMRES solver
$\mathbf{n}$	=	flux face unit normal vector
p	=	pressure
Δt	=	time step
T	=	temperature
$\mathbf{u}$	=	absolute velocity
$\mathbf{u}_{rot}$	=	rotational speed
u_n	=	contravariant velocity
u_n^r	=	relative contravairant velocity
$\mathbf{U}, \nabla\mathbf{U}$	=	primitive flow variable and its gradient
$\mathbf{W}$	=	conservative flow variable
$\mathbf{W}_0$	=	steady state solution
$\mathbf{W}^n,\ \mathbf{W}^{n+1}$	=	flow solution at time step n and $n+1$
$\mathbf{R}$	=	steady residual
$\mathbf{R}^{uns}$	=	unsteady residual
$\mathbf{R}^{(1stO)},\ \mathbf{R}^{(2ndO)}$	=	unsteady residual of first/second-order spatial accuracy
$\mathbf{x}$	=	position vector
β	=	Newton update under-relaxation factor
γ	=	ratio of specific heat
σ	=	Courant number
$\bar{\bar{\tau}}$	=	stress tensor
ν	=	kinematic viscosity
$\tilde{\nu}$	=	Sparlart–Allmaras turbulence variable
ν_t	=	eddy viscosity
ρ	=	density
$\boldsymbol{\omega}$	=	angular velocity
$\Omega_r,\ \partial\Omega_r$	=	a control volume in a relative reference frame and its boundary

I. Introduction

Computational fluid dynamics (CFD) solvers based on the Reynolds-averaged Navier–Stokes (RANS) equations are widely used for steady-state analysis of turbomachinery aerodynamics [1–4]. Unsteady simulation techniques like unsteady RANS [5], large eddy simulation and hybrid methods [6] are also rapidly maturing, but their computational cost remains too high for routine analysis in industry. Despite their limitations [7] steady RANS solvers remain the most widely used approach.

Robustness of the steady RANS solvers remains a major challenge for computing flows in industrial applications. For benign flow conditions, commonly encountered at design condition and characterized by largely attached flows, explicit and point-implicit solvers combined with multigrid acceleration perform well [8, 9]. However, for challenging flow conditions, typified by separations and shock/boundary layer interactions, commonly encountered for turbomachines operating at off-design conditions or designed with high-loading, steady solvers experience severe convergence slowdown or even divergence. Although such convergence difficulties are commonly encountered in practice, they are seldom discussed in the literature [10, 11]. We attribute this to two primary reasons. Firstly, machine-zero convergence is usually not sought and many practitioners accept a partially converged solution. Secondly, the lack of machine-zero convergence most critically affects linearized analysis such as adjoint and time-linearized analyses, the users of which constitute only a small fraction of the turbomachinery CFD community.

Substituting steady-state solution with semi-converged or an averaged flow solution (e.g. over the last few iterations), usually results in qualitatively correct analysis, particularly for benign flow conditions. However, at off-design conditions, this approach can lead to significant uncertainty [11, 12] and severely undermines the reliability of the analysis tool. Therefore, the authors, and others such as e.g. [13] posit that convergence to machine zero is a prerequisite for the reliable analysis of turbomachinery aerodynamics, especially at off-design conditions.

For linearized flow analysis such as the adjoint [14] and time-linearized analysis [15], the lack of machine-zero convergence of the nonlinear solver to steady state causes deterioration in accuracy and can lead to convergence problems for the linearized solvers. Moreover, linearized analysis assumes that one achieves a steady solution $\mathbf{W}_0$ which satisfies $\mathbf{R}(\mathbf{W}_0) = 0$ to machine precision. Thus the nonlinear residual for the perturbed state $\mathbf{W}$ can be linearized using Taylor expansion shown below

$$\mathbf{R}(\mathbf{W}) = \frac{\partial \mathbf{R}}{\partial \mathbf{W}} \Delta \mathbf{W}, \tag{1}$$

assuming that the perturbations $\Delta\mathbf{W} = \mathbf{W} - \mathbf{W}_0$ is small. Using a non-converged flow solution in Eq. (1) introduces a non-zero residual ($\mathbf{R}(\mathbf{W}_0) \neq 0$) and introduces errors to the linear solution $\delta\mathbf{W}$. This shows that a reliable linearized analysis requires the underlying nonlinear flow solution to be accurate to more than engineering accuracy. In addition, the convergence difficulty of the nonlinear flow solver due to severe numerical stiffness is shown to propagate to the linearized solvers [11, 16] and therefore, the capability to reach machine-zero convergence for the nonlinear steady

problem is an important aspect in addressing convergence difficulties of the linearized solvers.

Several methods have been devised to facilitate machine-zero convergence of the nonlinear steady solvers, such as the selective frequency damping (SFD) and the recursive projection method (RPM). SFD is based on the idea of 'selectively' filtering out the error modes by identifying the persistent modes with low damping and applying strong damping to suppress them [17]. Similarly, RPM identifies slow-damping or even non-contractive modes and applies Newton iterations to the selected mode to stabilize the convergence [16]. A natural extension of the SFD and RPM stabilization technique is to apply Newton's method not just to selective modes, but to to the entire system.

Newton's method solves the nonlinear equation, $\mathbf{R}(\mathbf{W}) = 0$, iteratively by using the linearization

$$\Delta W = \mathbf{W}^{n+1} - \mathbf{W}^{n} = -\left(\frac{\partial \mathbf{R}}{\partial \mathbf{W}}\right)^{-1} \mathbf{R}(\mathbf{W}^{n}).$$

Considering the Jacobian $\dfrac{\partial \mathbf{R}}{\partial \mathbf{W}}$ as a general matrix governing the pseudo time marching of the nonlinear equations, one can treat every time marching scheme, such as the explicit, point-implicit, and implicit as different approximations of the Newton method [18, 19]. With this unified view, it is straightforward to see that the Newton solver has the strongest error damping capability compared to all other pseudo time marching schemes, thus having the best asymptotic convergence property.

The Newton method requires the construction of the exact Jacobian matrix and the solution to the resulting large sparse linear system of equations. For typical second-order accurate discretization with a stencil that depends on the neighbors and also the neighbors of neighbors, this is computationally demanding and memory intensive compared to lightweight stabilization approaches such as SFD or RPM. Therefore, the key ingredient to an efficient Newton solver lies in the efficient solution of the linear system. Krylov-subspace solvers such as the generalized minimal residual (GMRES) method are commonly used for efficiently solving large sparse linear system of equations arising from the discretized RANS equations. Combining the outer Newton iteration with the inner Krylov-subspace solvers for the linear system yields the popular Newton–Krylov (NK) [20] algorithm.

The NK algorithm has long been used for solving the RANS equations for external aerodynamic analysis in two and three dimensions [21–26]. These early works mainly dealt with relatively benign conditions where the flow is mostly attached. It is not until recently that attention has shifted to challenging edge-of-the-envelope cases, for which the convergence difficulties are much more severe [12, 13]. In such situations a robust Newton–Krylov (NK) solver is shown to be vital for obtaining machine-zero converged flow solutions.

Turbomachines operate over a wide range of conditions. For some applications, good overall efficiency over a wide working range is more important than optimal peak efficiency at a single design point. Besides, except for simple academic cases, turbomachines rarely operate without secondary flows or separations and the flows are intrinsically three-dimensional and complex, especially at off-design. In many design optimization studies, improvement of

performance at severe off-design conditions is sought. Therefore, the ability to achieve machine-zero convergence over a wide operating range is critical for the reliable nonlinear and linearized aerodynamic analysis of turbomachines. However, algorithmic development aimed at this goal has received little attention in the literature.

In the present work, we develop an NK RANS solver for nonlinear turbomachinery flow analysis, with the aim of improving the solver's robustness over a wide operating range. In addition to presenting the basic algorithms of an NK RANS solver, we discuss in detail the computation of the entries of the Jacobian matrix, the solution of the resulting linear system, the globalization strategy of the nonlinear iterations, as well as aspects pertinent to turbomachinery flow calculations: the rotating reference frame and periodic boundary conditions. The remainder of the paper is organized as follows. First, the algorithmic development and implementation details of the steady NK RANS solver are discussed in detail in Sec. II. Validation results on an airfoil, a linear turbine cascade, a centrifugal compressor and an axial compressor are reported in Sec. III. Conclusions are drawn in Sec. IV along with an outlook for further improvement of the performance of the nonlinear steady solver as well as the extension to linearized solvers.

II. Nonlinear flow solver

The flow solver developed in this work, called NutsCFD (Newton Unstructured Turbomachinery flow Solver for Computational Fluid Dynamics), solves the RANS equations on arbitrary unstructured meshes using a node-based finite volume method. The core algorithm of NutsCFD is explained in this section, covering first the governing equations and then the spatial and temporal discretization, with emphasis on aspects related to turbomachinery flows such as the rotating reference frame, and periodic boundary conditions. Parallelization using domain decomposition is explained with emphasis on advantage offered by the two-halo partitioning approach. The algorithm development and implementation details for the Newton–Krylov method are then discussed in detail, including the forming of the Jacobian matrix, the preconditioning technique, and the efficient solution of the linear system. Finally, the globalization technique which adaptively controls the solution process is explained.

A. Governing equations

1. Mean flow

The equations describing the conservation of mass, momentum and total energy, in a reference frame rotating with a constant angular velocity $\boldsymbol{\omega}$, are given as

$$\frac{d}{dt}\int_{\Omega_r} \mathbf{W} dV + \oint_{\partial\Omega_r} (\mathbf{F}_c^r - \mathbf{F}_v^r) dS + \int_{\Omega_r} \mathbf{F}_\omega dV = \mathbf{0},$$

where ρ is the fluid density, $\mathbf{u}$ is the absolute velocity, $\mathbf{W} := (\rho,\ \rho\mathbf{u},\ \rho E)^T$ is the vector of conservative variables, e is the internal energy, and $E := e + \frac{1}{2}\rho\|\mathbf{u}\|^2$ is the total energy. The vector of the convective flux in the rotating and

stationary reference frames, $\mathbf{F}_c^r$ and $\mathbf{F}_c$, are

$$\mathbf{F}_c^r = \mathbf{F}_c - \rho U_n^{rot} \begin{Bmatrix} 1 \\ \mathbf{u} \\ E \end{Bmatrix}, \qquad \mathbf{F}_c = \rho U_n \begin{Bmatrix} 1 \\ \mathbf{u} \\ H \end{Bmatrix} + \begin{Bmatrix} 0 \\ p\mathbf{n} \\ 0 \end{Bmatrix},$$

where $H := E + p$ is the total enthalpy, p is the pressure, $\mathbf{n}$ is the unit normal vector of the flux face, $U_n := \mathbf{u} \cdot \mathbf{n}$ is the contravariant velocity, and U_n^{rot} is the peripheral velocity due to rotation, defined as

$$U_n^{rot} = \mathbf{u}_{rot} \cdot \mathbf{n}, \qquad \mathbf{u}_{rot} = \boldsymbol{\omega} \times \mathbf{x}.$$

$\mathbf{F}_v^r$ is the vector of the viscous flux in the rotating reference frame

$$\mathbf{F}_v^r = \begin{Bmatrix} 0 \\ \bar{\bar{\tau}} \cdot \mathbf{n} \\ \mathbf{u} \cdot \bar{\bar{\tau}} \cdot \mathbf{n} + k\mathbf{n} \cdot \nabla T \end{Bmatrix},$$

where $\bar{\bar{\tau}}$ is the stress tensor, k is the thermal conductivity coefficient, and T is the fluid temperature. $\mathbf{F}_\omega = (0, (\rho\boldsymbol{\omega} \times \mathbf{u})^T, 0)^T$ is an additional term due to Coriolis force.

2. Turbulence model

We model the flow turbulence using the negative variant of the Spalart–Allmaras (SA) model, called SA-neg [27]. Unlike the original SA model [28], SA-neg admits negative values as solution to the turbulence variable $\tilde{\nu}$ during the strong transient as well as the converged solution, especially on under-resolving coarse meshes. This approach avoids clipping the turbulence variable to a non-negative value, thus removing the hinderance to machine-zero convergence. The detailed governing equation is fully explained in [27] and only a few key aspects are discussed here.

The SA-neg model is based on the Boussinesq eddy viscosity assumption, where the eddy viscosity ν_t is determined using the turbulence variable $\tilde{\nu}$ and kinematic viscosity ν as

$$\nu_t = \begin{cases} \tilde{\nu} f_{v_1} & \text{if} \quad \tilde{\nu} > 0 \\ 0 & \text{if} \quad \tilde{\nu} \leq 0 \end{cases}, \qquad f_{v_1} = \frac{\chi^3}{\chi^3 + c_{v_1}^3}, \qquad \chi = \frac{\tilde{\nu}}{\nu}.$$

When $\tilde{\nu} > 0$, the governing equation is

$$\frac{d}{dt}\int_{\Omega_r} \tilde{\nu} dV + \oint_{\partial\Omega_r} (F^r_{c,sa} - F^r_{v,sa}) dS = \int_{\Omega_r} S_{sa} dV, \tag{2}$$

with

$$F^r_{c,sa} = U^r_n \tilde{\nu}, \quad F^r_{v,sa} = \frac{(\nu + \tilde{\nu})\nabla\tilde{\nu} \cdot \mathbf{n}}{\sigma}, \quad S_{sa} = \frac{1}{\sigma} C_{b_2} (\nabla\tilde{\nu})^2 + P - D,$$

where $U^r_n := (U_n - U^{rot}_n)$ and P and D are production and wall destruction terms whose definition can be found in [27].

When $\tilde{\nu} \leq 0$, the viscous flux term and the source term in Eq. (2) of the SA-neg model are defined as

$$F^r_{v,sa} = \frac{(\nu + \tilde{\nu} f_n)\nabla\tilde{\nu} \cdot \mathbf{n}}{\sigma}, \quad S_{sa} = \frac{1}{\sigma} C_{b_2} (\nabla\tilde{\nu})^2 + P_n - D_n,$$

where $f_n := (c_{n_1} + \chi^3)/(c_{n_1} - \chi^3)$, and P_n and D_n are modified production and wall destruction terms defined as

$$P_n = c_{b_1}(1 - c_{t_3}) S\tilde{\nu}, \quad D_n = -c_{w_1}(\tilde{\nu}/d)^2.$$

with S the magnitude of the vorticity and d the distance to wall. The values of the constants used are from [27].

Once the turbulence model is defined, the mean flow equation can be closed by including the Reynolds stresses in the stress tensor $\bar{\bar{\tau}}$ using kinematic viscosity ν and the eddy viscosity ν_t

$$\bar{\bar{\tau}} = 2\rho(\nu + \nu_t)\dot{\gamma}$$

where $\dot{\gamma}$ is the shear rate tensor. In addition, the effect of eddy viscosity on the thermal conductivity coefficient is accounted for by the relation

$$k = c_p \left(\frac{\rho\nu}{Pr} + \frac{\rho\nu_t}{Pr_T} \right),$$

with c_p the specific heat at constant pressure and $Pr = 0.72$ and $Pr_T = 0.9$ the Prandtl and turbulent Prandtl numbers.

B. Spatial discretization

The governing equations are discretized using the method of lines and thus the spatial and temporal discretizations can be treated separately. The governing equations, including both the mean flow and turbulence equations, can be written formally as

$$\frac{d\mathbf{W}}{dt} + \mathbf{R}(\mathbf{W}) = \mathbf{0}, \tag{3}$$

where $\mathbf{W} := (\rho, \rho\mathbf{u}^T, \rho E, \tilde{\nu})^T$ is the vector of the conservative variables and $\mathbf{R}$ is the residual vector. The residual for each control volume is computed by summing up the fluxes across each flux face as well as the volumetric source terms.

The discretization of the convective and viscous fluxes, $\mathbf{F}_c^r$ and $\mathbf{F}_v^r$, as well as the periodic boundary condition, are explained in the following subsections.

1. Convective flux

The convective flux for the mean flow is computed using the Roe scheme [29]. Second-order spatial accuracy is achieved by linearly extrapolating the flow variables from the node to the flux face center, using gradient calculated with the Green–Gauss formula. Denoting the left and right states by $\mathbf{W}_L$ and $\mathbf{W}_R$, the convective flux can be computed as

$$\mathbf{F}_{c,face}^r = \frac{\mathbf{F}_c^r(\mathbf{W}_L) + \mathbf{F}_c^r(\mathbf{W}_R)}{2} + \frac{1}{2}\left|A_{Roe}^r\right|(\mathbf{W}_R - \mathbf{W}_L), \tag{4}$$

where A_{Roe}^r is the Jacobian matrix of the relative convective flux $\mathbf{F}_c^r$ with respect to $\mathbf{W}$, evaluated using the Roe-averaged variables on the flux face. The Jacobian matrices of the convective flux in the stationary and rotating reference frames, $A_{Roe} = \dfrac{\partial \mathbf{F}_c}{\partial \mathbf{W}}$ and $A_{Roe}^r = \dfrac{\partial \mathbf{F}_c^r}{\partial \mathbf{W}}$, are related by

$$A_{Roe}^r = A_{Roe} - U_n^{rot} I.$$

Consequently, the eigenvalues of A_{Roe}^r are the eigenvalues of A_{Roe} shifted by U_n^{rot} and the wave speeds in the dissipation term of the Roe flux are shifted from $\{U_n, U_n, U_n, U_n + c, U_n - c\}$ for the stationary reference frame to $\left\{U_n^r, U_n^r, U_n^r, U_n^r + c,\ U_n^r - c\right\}$ for the relative reference frame. A Roe flux calculation implementation using a non-rotational frame thus requires substituting the contravariant velocity U_n with U_n^r, in order to compute the convective flux in the rotating reference frame.

The convective flux for the SA-neg equation is discretized using the first-order upwind scheme

$$\mathbf{F}_{c,sa}^r = \frac{1}{2}(U_{n,i}^r \tilde{\nu}_i + U_{n,j}^r \tilde{\nu}_j) + \frac{1}{2}|U_{n,j}^r + U_{n,i}^r|(\tilde{\nu}_j - \tilde{\nu}_i)$$

where $U_{n,i}^r$, $U_{n,j}^r$ and $\tilde{\nu}_i$, $\tilde{\nu}_j$ are the relative contravariant velocity and turbulence variable for left and right nodes across the flux face. No entropy fix is used for the dissipative part as we found that causes excessive dissipation. Similar observation was reported in [9].

2. Viscous flux

Computation of the viscous flux for both the mean flow and turbulence equations requires the primitive flow variables, $\mathbf{U} := (\mathbf{u}^T, p, T, \tilde{\nu})^T$, and their gradients, $\nabla\mathbf{U}$, at the flux face. The flow variables on the face are taken as the arithmetic average of the left and right nodes. The first step in the calculation of the gradient at the flux face is to compute the

arithmetic average of the gradients for the left and right nodes

$$\overline{\nabla \mathbf{U}} = \frac{\nabla \mathbf{U}_i + \nabla \mathbf{U}_j}{2},$$

where $\nabla \mathbf{U}_i$ and $\nabla \mathbf{U}_j$ are the respective gradient of the primitive variables, evaluated using the Green–Gauss formula. To avoid odd-even oscillations, the tangential part is replaced by the directional derivative [30, 31] as

$$\nabla \mathbf{U} = \overline{\nabla \mathbf{U}} - \left(\overline{\nabla \mathbf{U}} \cdot \mathbf{t} + \frac{\mathbf{U}_j - \mathbf{U}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|} \right) \mathbf{t}, \qquad \mathbf{t} = \frac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|}.$$

3. Periodic boundary condition

Periodic boundary condition is necessary for computing steady flows in turbomachines using a single passage. We explain our implementation of the periodic boundary condition with reference to the mesh shown in Fig. 1. The rotational angle from the master (red) to shadow (blue) periodic patch is θ. For a mesh with matching periodic boundaries, each

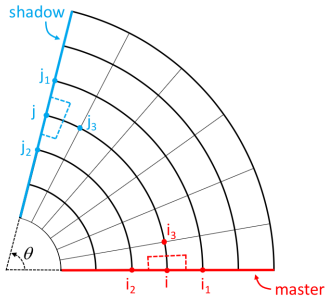

Fig. 1 A sketch of the computational mesh with rotational periodic boundaries.

node on the master patch forms a pair with one node on the shadow patch, e.g., node i and j in Fig. 1. The full residual for node i, $\mathbf{R}_i$, is obtained by summing the partial residuals $\mathbf{r}_i$ and $\mathbf{r}_j$ computed for nodes i and j, as follows

$$\mathbf{R}_i = \mathbf{r}_i + B^{-1}\mathbf{r}_j,$$

where B,

$$B = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) & 0 & 0 & 0 \\ 0 & sin(\theta) & cos(\theta) & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

accounts for the rotation from the master to the shadow periodic patch. The partial residuals $\mathbf{r}_i, \mathbf{r}_j$ are the sum of the fluxes across their respective interior flux faces (red and blue dashed lines in Fig. 1). The unknowns of node j are redundant as they depend on those of node i. During the solution update, only $\mathbf{W}_i$ is updated, and $\mathbf{W}_j$ is assigned the updated value $\mathbf{W}_i$ after the transformation $\mathbf{W}_j = B\mathbf{W}_i$. Translational periodicity is achieved with $\theta = 0$.

C. Temporal discretization

In order to find the steady solution, Eq. (3) is discretized implicitly in time as

$$\frac{\mathbf{W} - \mathbf{W}^n}{\Delta t} + \mathbf{R}(\mathbf{W}) = \mathbf{0}, \tag{5}$$

where $\mathbf{W}^n$ is the flow solution at time step n and unknown $\mathbf{W}$ is the solution at $t = t_n + \Delta t$. Since the steady solver is only concerned about the final steady solution, the physical time step Δt can be replaced with the node-wise local time step Δt_i multiplied with a global Courant number for step n, σ^n. We call $\mathbf{R}(\mathbf{W})$ the steady residual and define the unsteady residual $\mathbf{R}^{uns}(\mathbf{W}; \mathbf{W}^n)$ as

$$\mathbf{R}_i^{uns}(\mathbf{W}; \mathbf{W}^n) := \frac{\mathbf{W}_i - \mathbf{W}_i^n}{\sigma^n \Delta t_i} + \mathbf{R}_i(\mathbf{W}).$$

The governing equation can be rewritten using the introduced unsteady residual as

$$\mathbf{R}^{uns}(\mathbf{W}; \mathbf{W}^n) = 0.$$

By introducing the state perturbation $\Delta\mathbf{W} := \mathbf{W} - \mathbf{W}^n$, the above equation can be solved using the Newton method as

$$\mathbf{P}\Delta\mathbf{W} = -\mathbf{R}^{uns}(\mathbf{W}; \mathbf{W}^n), \tag{6}$$

with

$$\mathbf{P} := \frac{\partial \mathbf{R}^{uns}(\mathbf{W}; \mathbf{W}^n)}{\partial \mathbf{W}} = \text{diag}\left\{\frac{1}{\sigma^n \Delta t_i}\right\} + \frac{\partial \mathbf{R}(\mathbf{W})}{\partial \mathbf{W}}$$

The matrix $\dfrac{\partial \mathbf{R}(\mathbf{W})}{\partial \mathbf{W}}$ is the Jacobian of the steady residual. Since maintaining time accuracy is unnecessary, only one inner nonlinear step is taken, that is, Eq. (6) is solved once for each outer nonlinear iteration, with $\mathbf{W}$ always initialized with $\mathbf{W}^n$, and the flow solution $\mathbf{W}^{n+1}$ updated as

$$\mathbf{W}^{n+1} = \mathbf{W}^n + \Delta\mathbf{W}. \tag{7}$$

The reason the unsteady form of the governing equation is used is that it prevents the nonlinear iterations from getting trapped in a local minimum of $\|\mathbf{R}(\mathbf{W})\|_2$, which would stall the steady state solver convergence [12]. In addition, ensuring that the left-hand side matrix $\mathbf{P}$ is an exact linearization of the right hand side unsteady residual guarantees that the linear solution to Eq. (6) will be in the descent direction of $\|\mathbf{R}^{uns}(\mathbf{W}; \mathbf{W}^n)\|_2$ [12].

Once the spatial and temporal discretizations are established, there are three main steps to complete a Newton update.

They are (i) forming the Jacobian matrix, (ii) solving the large sparse linear system of equations, and finally, (iii) steering the nonlinear solution process, or globalization. These steps are explained in detail in the following subsections.

1. Forming the Jacobian matrix

The forward mode of the algorithmic differentiation (AD) tool Tapenade [32] is used to differentiate the subroutine that computes the unsteady residual $\mathbf{R}(\mathbf{W}; \mathbf{W}^n)$ with respect to $\mathbf{W}$. A naive implementation to compute the entries of the Jacobian would be to in turn perturb each component of the vector $\mathbf{W}$ at every node and propagate this seed through a tangent-linear model, which requires $6 \times N$ executions of the differentiated residual subroutine for a mesh with N nodes. To accelerate the computation of the Jacobian, a distance-two graph coloring algorithm [33] is employed which divides all grid points into groups of distinct colors such that no two points of the same color are distance-two neighbors of each other. The open source tool ColPack [34] is used in this work. Flow variables for all the grid points in each group can then be perturbed simultaneously and a total of $6 \times N_{color}$ executions of the differentiated residual subroutine are required to form the Jacobian matrix. For all the unstructured meshes used in this work (including two-dimensional quadrilateral and three hexahedral meshes), the number of colors never exceeded 66, and therefore computing one instance of the exact Jacobian requires no more than 400 executions of the differentiated residual subroutine.

Alternative to the AD approach, analytical approach to derive the Jacobian matrix [35] is also a viable way to obtain the exact Jacobian. However, compared to the AD approach, this is more tedious and error-prone, especially for an unstructured mesh solver, and thus is not adopted in the current work.

Two additional aspects need to be considered when forming the Jacobian matrix for cases with periodic boundaries. Firstly, coloring is applied to all grid points except those on the shadow periodic patch. During the execution of the differentiated residual subroutine, the seeding of the flow variables for the nodes on the shadow periodic patch is always made consistent with their twins on the master periodic patch. For example, when flow variable for node i on the master periodic patch is assigned unit seed value ($\delta\mathbf{W}_i = 1$) the shadow node j value is set to $\mathbf{W}_j = B$. Secondly, the j-th row of blocks in the Jacobian matrix, corresponding to node j on the shadow periodic boundary, are assigned 6×6 identity matrix on the diagonal and 6×6 zero matrices on all off-diagonal positions. This prevents the Jacobian matrix from becoming singular due to the nullification of the residual for the shadow nodes, without having to re-number the nodes to form the Jacobian without the shadow nodes altogether.

2. Solving the large sparse linear system of equations

GMRES is used to solve the large sparse system of linear equations at each nonlinear iteration and ILU(0) is used as a right preconditioner. Although higher ILU fill-in levels have been reported in the literature to achieve optimal speed-up, our experience shows that the optimal fill-in level is very case-dependent and no universally applicable optimal choice can be identified. Therefore, ILU(0) is used for all the cases reported in this work. ILU factorization is based on the

blended Jacobian matrix, Jac^{blend}, which is defined as

$$Jac^{blend} = c_0 \frac{\partial \mathbf{R}^{(2ndO)}}{\partial \mathbf{W}} + (1 - c_0) \frac{\partial \mathbf{R}^{(1stO)}}{\partial \mathbf{W}}, \quad 0 < c_0 < 1.$$

$\mathbf{R}^{(2ndO)}$ and $\mathbf{R}^{(1stO)}$ denote the unsteady residuals that are based on the second- or first-order accurate spatial discretization. The first-order spatial discretization is achieved by disabling the linear reconstruction when computing the convective flux, i.e., set $\mathbf{W}_L = \mathbf{W}_i$ and $\mathbf{W}_R = \mathbf{W}_j$ in Eq. (4). The resulting ILU preconditioner based on the blended Jacobian is found to be much more effective than the one based on $\frac{\partial \mathbf{R}^{(2ndO)}}{\partial \mathbf{W}}$ only [22, 36, 37]. Ideally, c_0 should automatically adapt during each nonlinear iteration, but as shown in [37], the effectiveness of the preconditioner only weakly depends on c_0 as long as it is not near unity. Therefore, a constant value of $c_0 = 0.5$ is chosen.

When using GMRES to solve the linear system of equations, the choice of stopping criterion has a major impact on the overall nonlinear solution efficiency. In the following discussion, we use GMRES(m, tol) to denote GMRES with two stopping criteria, maximum m vectors or a relative residual drop of tol, whichever is reached first. GMRES is used without restarting. In [38] it is reported for an NK solver that GMRES(30, 0.1) is overall optimal for the two-dimensional turbulent cases computed, where the relatively small number of GMRES vectors can probably be attributed to the use of a strong ILU(4) preconditioner in that work. Wong [25] used GMRES(50, 0.01) with ILU(1) in an NK solver for three-dimensional external flow calculations at cruise condition on unstructured grids. Mavriplis [39] used GMRES(100, 0.01) for computing the flow around a wing-body model at cruise condition. Our numerical experiments for the cases used in this paper show that in general $m = 100$ is enough to reach the relative drop of $tol = 0.1$. However, it is well known that as the problem stiffness increases, GMRES may experience convergence stall if m is chosen too low [37]. In this work, to ensure the robustness of the solver, we use GMRES(500, 0.1). For all cases tested, the relative residual drop always reaches $tol = 0.1$ before the maximum vector number of 500 is reached. The downside of using a large vector basis is that the CPU time grows significantly due to the orthogonalization procedure in GMRES, whose cost scales quadratically with m. Future work will explore the use of more advanced Krylov-subspace solvers, such as the generalized conjugate residual solver with deflated restarting (GCRO–DR) [37, 40] to keep the size of the vector basis small.

3. Jacobian-forming versus Jacobian-free approach

The Jacobian-forming approach adopted in this work naturally incurs heavy memory overhead for storing the large sparse Jacobian matrix. A popular alternative is the Jacobian-free Newton–Krylov (JFNK) [20] approach, where the second-order Jacobian matrix is never formed and stored. This is possible because the matrix-vector product $\left(\frac{\partial \mathbf{R}}{\partial \mathbf{W}}\right) \Delta \mathbf{W}$ required by the GMRES solver can be computed without having the exact Jacobian matrix explicitly available. For example, it can be computed exactly by applying AD to the flux computation, or approximately using Frechet derivatives.

There are two reasons for choosing the Jacobian-forming approach. First, the aim of this work is not only to stabilize and accelerate the nonlinear flow solver itself, but also to support linearized analysis tools. Forming the matrix and storing it in memory is very advantageous, as the system matrix of various linearized analysis tools can be obtained in a very straightforward way. For example, for the adjoint equation $\left(\frac{\partial \mathbf{R}}{\partial \mathbf{W}}\right)^T \mathbf{v} = \mathbf{f}$, one only needs to transpose the Jacobian matrix to obtain the system matrix. To solve the time-linearized equation $\left(\frac{\partial \mathbf{R}}{\partial \mathbf{W}} + j\omega I\right)\mathbf{u} = \mathbf{b}$, one only needs to apply a complex-valued diagonal shift to the Jacobian matrix. Adopting the Jacobian-forming approach in the nonlinear solver makes the extension to such linearized analysis tools extremely convenient.

Secondly, even with the JFNK approach, it is actually not always matrix-free. For example, in order to form the ILU preconditioner, an approximate Jacobian based on a lower-order discretization, easier to form and requiring less memory, still needs to be computed and stored in memory. Although basing the ILU on this approximate Jacobian reduces the memory overhead, its preconditioning effect is found to be less than the ILU based on the exact Jacobian [41].

4. Globalization strategy

A good globalization strategy for adaptively varying the solver parameters is critical for the robust convergence of Newton–Krylov solvers. A review of various globalization methods is given in [20]. The globalization strategy of [12], a hybrid method combining the line search and pseudo-transient continuation with CFL ramping is adopted in this work. Pseudo-transient continuation refers to finding the steady solution by solving the unsteady equation with an increasingly large physical/pseudo time step Δt. As the time step approaches infinity, the time derivative term in the unsteady Eq. (5) diminishes and the transient solution of the unsteady equation approaches the root of $\mathbf{R}(\mathbf{W}) = 0$.

For a given intermediate solution and the Courant number at time step n, $\mathbf{W}^n$ and σ^n, the linear system in Eq. (6) is first solved inexactly with GMRES with $tol = 0.1$. The approximate linear solution, $\Delta\mathbf{W}$, is used to update the solution to obtain the flow solution at time step $n+1$, $\mathbf{W}^{n+1}$. Directly updating the flow solution as in Eq. (7) may lead to unphysical updated states, and a relaxation factor β is used to stabilize the nonlinear iteration

$$\mathbf{W}^{n+1} = \mathbf{W}^n + \beta\Delta\mathbf{W}. \tag{8}$$

β is determined by reducing its value from the initial value of $\beta_{start} = 1$ to $\beta_{end} = \left(\frac{1}{1.2}\right)^6$, i.e., recursively dividing β by 1.2. Within 7 steps, if a physically valid solution that reduces the resulting unsteady residual is found, then the β value that reduces the unsteady residual the most is used in Eq. (8). This optimal β value is used to adapt the Courant number for the next nonlinear iteration. If the returned value is $\beta = 1$, it means the line search is successful, implying the nonlinear update $\Delta\mathbf{W}$ is 'healthy', and the Courant number is incremented; otherwise, it remains the same. In the case where a valid β is not found, i.e., for all the β values tested, either all solutions are unphysical, or a solution producing a reduction of $\|\mathbf{R}^{uns}(\mathbf{W};\mathbf{W}^n)\|_2$ is not found, then it implies the current Courant number is too large and should be

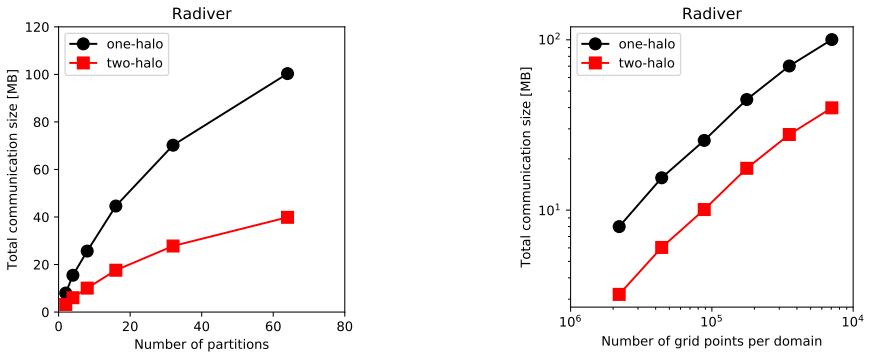

Fig. 2 Comparison of total MPI communication message size across all communication ranks against the number of partitions (left) and the number of grid points per domain (right) for one- and two-halo partitioning for the Radiver case.

reduced for the next step. The strategy described above can be summarized as

$$\sigma^{n+1} = \begin{cases} k_1 \cdot \sigma^n & \text{if } \beta = 1; \\ \sigma^n & \text{if a valid } \beta \text{ is found but } \beta < 1; \\ k_2 \cdot \sigma^n & \text{if a valid } \beta \text{ is not found.} \end{cases}$$

The values for the two constants are $k_1 = 1.2$ and $k_2 = 0.1$.

D. Domain decomposition and parallelism

1. Domain decomposition with the two-halo approach

Message-passing parallel implementations partition the mesh into contiguous regions 'owned' by a processor. The second-order accurate spatial discretization achieved via linear reconstruction uses the state gradient extrapolation and limiters on either side of a flux face to compute second-order fluxes. For faces at the partition boundary some data may be owned by a neighboring partition. Missing data can either be made available through message passing of the missing quantity, or through duplication of the underlying grid and flow solution data and re-computation, by using overlapping layers in the partitioned meshes. For second-order finite volume solvers, one can use either one or two layers of overlapping, called one-halo and two-halo overlaps [42]. The approach chosen has an effect on the convergence rate of linear solvers as some aspects, e.g. the ILU preconditioning is only performed for an individual partition to reduce computational cost.

One- and two-halo approaches differ in the amount of data communicated and the additional memory overhead due to the halo nodes. To illustrate this better, in Fig. 2, a comparison of average MPI communication message size required for one- and two-halo partitioning is shown for the Radiver cases (see Sec. III.C). For both approaches, the graph

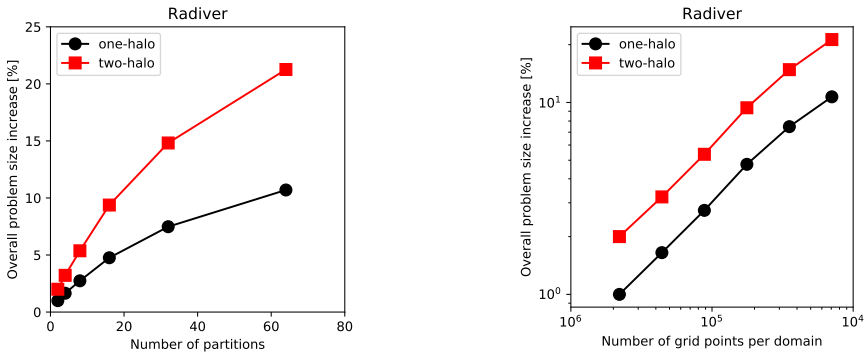

Fig. 3 Comparison of the increase in overall problem size against the number of partitions (left) and the number of grid points per domain (right) for one- and two-halo partitioning for the Radiver case.

partitioning tool Metis [43] is used to partition the nodal graph of the global mesh first, and they differ in the way the halo information is organized and the computation and parallel communication are scheduled. The communication size shown is the sum of the total MPI messages that are sent/received by all MPI ranks during a single explicit time-marching iteration step.

The one-halo approach requires approximately twice the message size compared to the two-halo approach. However, the gain in message size reduction comes at the cost of additional storage of the two-halo off-processor data. The increase in memory consumption with partition number can be estimated by aggregating the halo nodes of each MPI rank and dividing by the number of nodes in the mesh. This increase is plotted in Fig. 3 for both cases for different number of partitions. The increase in problem size of two-halo partitioning is approximately twice that of one-halo. Therefore, one has to make a trade-off between memory use and message size. In this work we trade reduced communication cost for memory usage and implemented the two-halo approach.

The second advantage of the two-halo approach is that it results in a clear two-step procedure for the residual calculation, namely, (i) halo communication of flow variables (MPI) $\rightarrow$ (ii) residual and preconditioner evaluation (no MPI). Consequently, the Jacobian matrix computation also proceeds without any MPI communication, via a similar workflow: (i) halo communication of flow variables (MPI) $\rightarrow$ (ii) Jacobian forming (no MPI). The AD tool thus can be used to differentiate the residual evaluation subroutine, without having to deal with any MPI call. The additional storage of the two-halo approach is an acceptable trade-off for a much simpler implementation of the calculation of the Jacobian.

2. Adjustment due to periodicity

When using Metis to partition the global mesh with periodic boundaries, each periodic edge is first collapsed to a vertex (by merging the two end vertices) in the input graph to Metis. This approach avoids the complexity of setting up additional MPI communication for off-processor periodic vertices. The periodic edges are un-collapsed after partitioning. Then we search for the halo-1 and halo-2 nodes and edges using the graph and partitioned vertices.

3. Preconditioner parallelization

The GMRES implementation used in this work is global to avoid impairing convergence and stability when using many partitions. Two popular paradigms of applying the preconditioner in the distributed manner are additive-Schwarz [44] and approximate Schur [45] methods. The former computes the preconditioning matrix based on the diagonal block of the system matrix that is local to each parallel partition and the off-diagonal block matrices are omitted. Due to this decoupling, the preconditioning effectiveness often degenerates as the number of partitions increases. The approximate-Schur preconditioning is devised to maintain a strong coupling to allow better scalability for massively parallel computations. The approximate-Schur approach combined with a flexible Krylov solver has been shown to outperform additive-Schwarz in previous studies [26]. However, the approximate-Schur preconditioner is less straightforward to implement compared to additive-Schwarz.

In our implementation, we use the additive-Schwarz approach to parallelize ILU preconditioner. The ILU(0) preconditioner are computed for each partition without considering the exchange of information between partitions. The decoupled approach allows a simple and essentially sequential ILU implementation for each partition at the cost of deteriorating convergence rate with increased number of partitions. The deterioration is investigated in detail for the Radiver case in sec. III.C.

III. Results

The NutsCFD solver is used to calculate the flow for (1) a two-dimensional airfoil, (2) a two-dimensional turbine cascade, (3) a three-dimensional centrifugal compressor, and (4) a three-dimensional axial compressor. Flow calculations ranging from benign attached flows to adverse flows with shock-boundary-layer interaction and separation are considered to demonstrate both the accuracy and convergence robustness of the solver. For all calculations, the flow is assumed to be fully turbulent. The steady state solution is found when the norm of the steady residual has been reduced by ten orders of magnitude. In all residual convergence plots in the paper, we use the norm of the steady residual, i.e., $\|\mathbf{R}(\mathbf{W})\|_2$, rather than $\|\mathbf{R}^{uns}(\mathbf{W};\mathbf{W}^n)\|_2$, as the convergence of the former is a true measure of the finding of the steady state solution.

A. Case 1: NACA0012 airfoil

1. Case overview

The first test case is the steady state flow around the NACA0012 airfoil for both subsonic and transonic conditions at different angles of attack. The farfield is placed at 100 chords away from the airfoil. The flow conditions, listed in Table 1, cover both the subsonic and transonic flow regimes. For the transonic flow case ‘f’, the inlet velocity is just below the onset point of the transonic buffet, and is thus expected to present a stiff numerical system which poses a significant challenge to the convergence of the steady solver. All flow calculations for this case were performed using

one processor.

Table 1 Flow conditions simulated for NACA0012 airfoil

Flow condition	a	b	c	d	e	f
Mach number	0.15	0.15	0.15	0.76	0.76	0.76
Angle of attack (°)	0	10	15	0	2	3
Reynolds number	$3 \cdot 10^6$	$3 \cdot 10^6$	$3 \cdot 10^6$	$15 \cdot 10^6$	$15 \cdot 10^6$	$15 \cdot 10^6$

Three meshes of different grid densities, all consisting of structured quadrilateral elements, are used, and their detail is listed in Table 2. For the finest level_0 mesh, the first layer cell height satisfies $y^+ < 1$ for both subsonic and transonic cases where the y^+ value is estimated based on [46].

Table 2 Three meshes used for NACA0012 airfoil

Mesh information	level_0	level_1	level_2
Number of grid points	134,976	37,968	20,608
First layer cell height [m]	10^{-6}	10^{-5}	10^{-4}

2. Accuracy validation, subsonic cases

The flows for all six conditions are calculated first using the level_0 fine mesh, initialized with a uniform flow field based on the farfield condition. The Mach number contours for the subsonic steady state flows around the airfoil are shown in Fig. 4. The pressure coefficient along the airfoil surface is compared with the experimental results in Fig. 5 where good agreement can be seen.

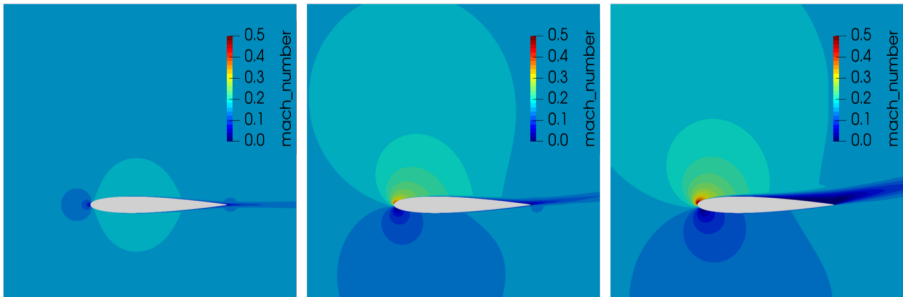

Fig. 4 Mach number contour for the steady subsonic flow around the NACA0012 airfoil using the fine mesh, cases a-c: $M = 0.15$, and $\alpha = 0°, 10°$ and $15°$ (left to right).

3. Convergence behavior, subsonic cases

Shown in Fig. 6 are the convergence histories for the subsonic cases on the fine mesh. The residual convergence with respect to the nonlinear steps, the normalized CPU time, and the evolution of the Courant number and the Krylov vectors used during each nonlinear iteration are shown. It can be seen that for condition 'a' and 'b', the solver converges within

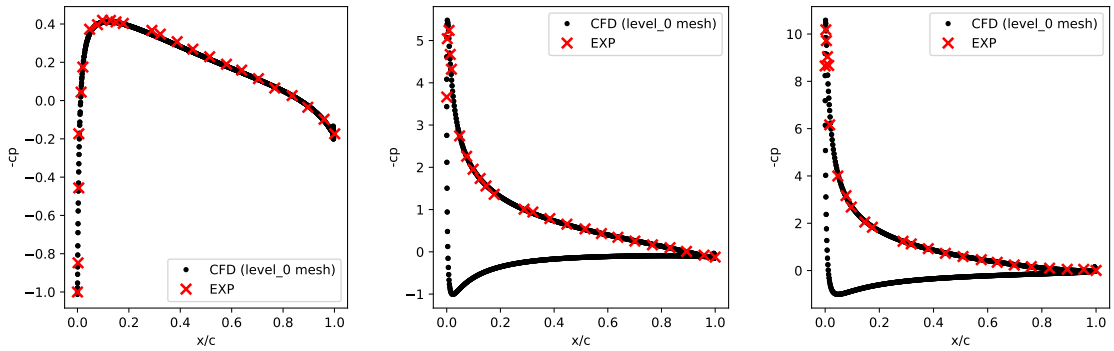

Fig. 5 Pressure coefficient distribution comparison between NutsCFD (using the fine mesh) and experimental results [47], cases a-c: $M = 0.15$, and $\alpha = 0°, 10°$ and $15°$ (left to right).

100 iterations, and the Courant number steadily ramps up to nearly 10^6 from unity. For condition 'c', more iterations are needed for convergence, and from the residual convergence history, it can be seen that the steady residual experiences some transient growth (from iterations 75 to 130) before finally rapidly dropping to 10^{-10}. The slow convergence in the transient process corresponds to the slow build-up of the turbulence in the boundary layer. Note that at every nonlinear step the norm of the unsteady residual is reduced but this does not constrain the steady residual from growing. This shows the importance of using the unsteady formulation. For case 'c', the increased numerical stiffness is also apparent from the Courant number evolution, which underwent a stagnation period before finally growing to 10^6. For this case, the Krylov vectors required to achieve the inner linear solution tolerance of $tol = 0.1$ is also larger, reaching about 200 towards the final convergence.

Also shown in Fig. 6 is the convergence history plotted against the normalized CPU time called work units. One work unit is defined as the CPU time required for a single residual evaluation. On average, a hundred thousand work units are required to fully converge the solution, which is comparable to the reported performances by others [41].

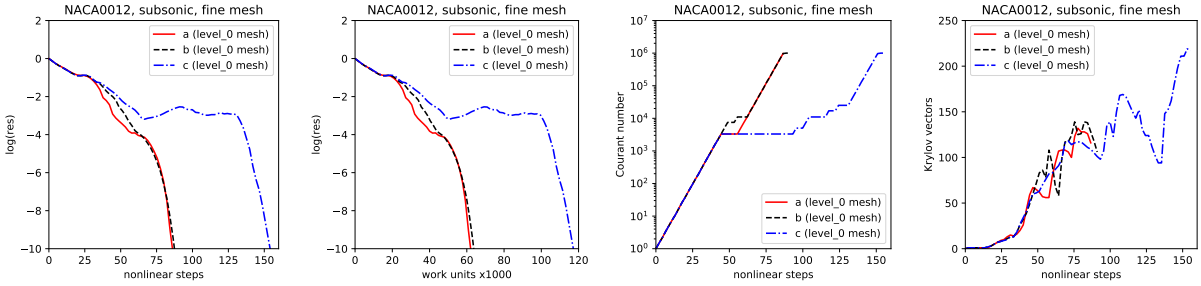

Fig. 6 Left to right: convergence history against iteration number and CPU time work units, Courant number evolution and number of Krylov vectors used in each step for computing the steady subsonic flow around the NACA0012 airfoil, cases a-c.

4. Accuracy validation and convergence behavior, transonic cases

The Mach number contours for the transonic flows around the airfoil are shown in Fig. 7, where a shock just starts to appear for case 'd' and becomes quite strong for cases 'e' and 'f'. For case 'f', the strong interaction between the shock and the boundary layer could lead to the onset of the transonic buffet, which is itself a topic of active research [48]. Relevant to this work is the increased numerical stiffness for case 'f'. As can be seen from Fig. 8, the solver converges fully in 200 iterations for case 'd' where the shock is very weak and the shock/boundary layer interaction is small. For case 'e', the solver experiences some difficulty around iteration 120. Due to the failure to find a valid β, the Courant number has to reduce to a smaller value and then start ramping up again. The convergence difficulty is most severe for case 'f', for which the CFL ramping struggles first at iteration 120 and then for iterations 250–400, during which the residual also experienced violent oscillations, before eventually starting to rapidly converge.

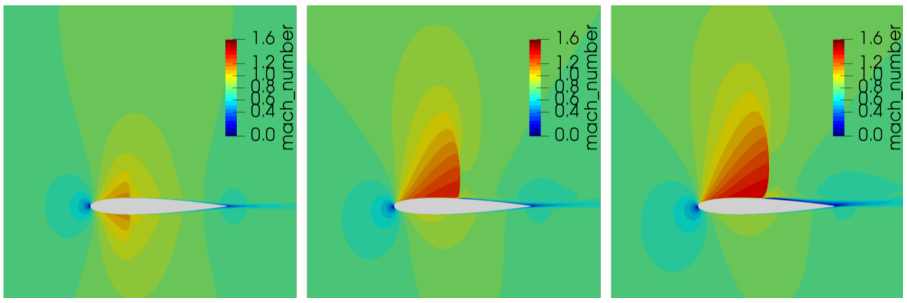

Fig. 7 Mach number contours for the steady transonic flow around the NACA0012 airfoil using the fine mesh, cases d-f: $M = 0.76$, and $\alpha = 0°, 2°$ and $3°$ (left to right).

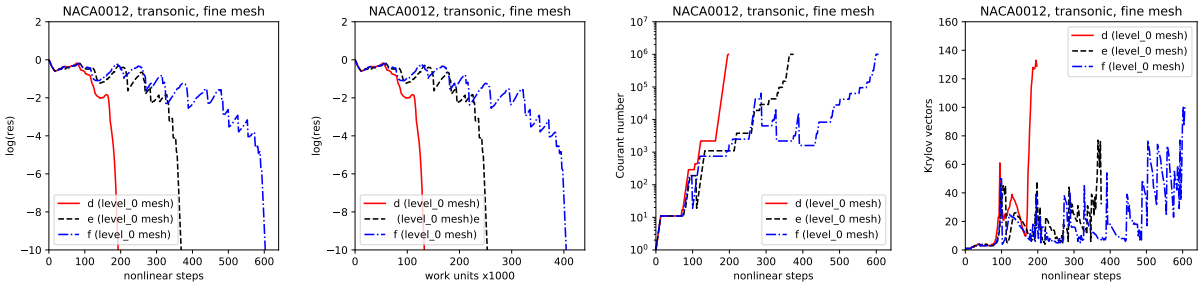

Fig. 8 Left to right: convergence history against iteration number and CPU time work units, Courant number evolution and numbers of Krylov vectors used in each step for computing the steady transonic flow around the NACA0012 airfoil, cases d-f.

5. Influence of mesh density

To investigate the effect of mesh density on convergence, all cases, 'a' to 'f', are re-computed for the level_1 and level_2 meshes, both initialized with uniform flow field based on the farfield condition. The convergence histories are shown in Fig. 9. It can be seen that for subsonic cases 'a', 'b' and 'c', the number of nonlinear steps to reach convergence is roughly the same for all three meshes. However, for transonic cases, the nonlinear steps required for full convergence

for the level_0 mesh increases by a factor of 2 to 4, compared with the level_1 and level_2 meshes. This to some extent shows the elevated numerical stiffness for challenging flow conditions, which is usually not observed for benign conditions, and emphasizes the necessity for a very strong and robust solver.

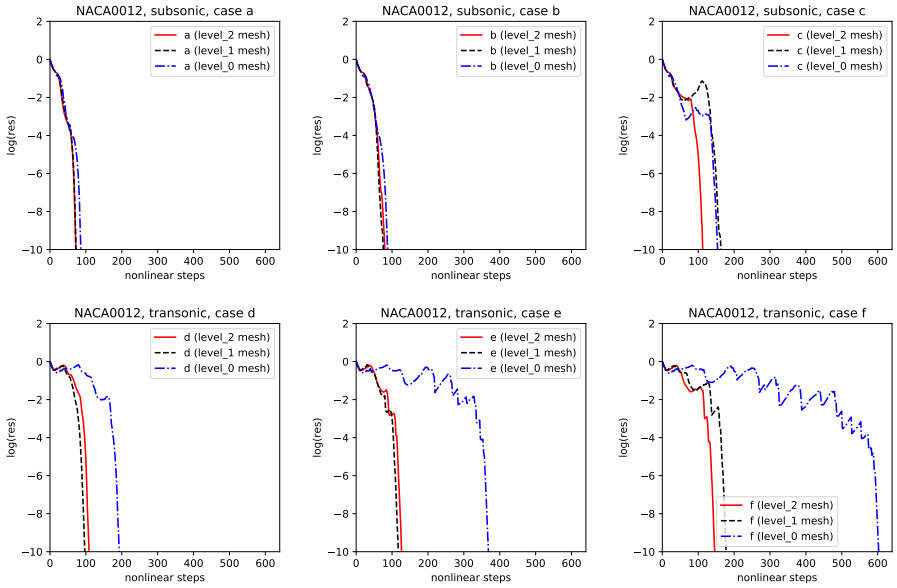

Fig. 9 Convergence history for the three different meshes. Upper: subsonic cases; lower: transonic cases.

B. Case 2: LS89 turbine cascade

1. Case overview

The second case, LS89, is a highly loaded turbine cascade designed, analyzed, and tested at the von Kármán Institute for Fluid Dynamics [49]. The steady flow for the test condition numbered 'MUR43' is computed using NutsCFD. A total temperature of 420 K and total pressure p_0 =143.5 kPa are imposed as the inlet condition. The inlet velocity is in the axial direction. The outlet condition is a static back pressure p_s, corresponding to an isentropic Mach number of $M_{isen} = 0.84$. The value of p_s is the solution to the following equation

$$M_{isen} = \sqrt{(2/(\gamma-1)) \cdot ((p_0/p_s)^{(\gamma-1)/\gamma} - 1)}. \tag{9}$$

The Reynolds number for the flow simulated is 10^6. The flow is initialized using a uniform flow, based on the static back pressure p_s and a Mach number of $M = 0.84$. Six levels of systematically coarsened quadrilateral meshes are used with the mesh coarsening mainly taking place in the boundary layer region in the direction normal to the wall. The statistics of the six meshes are listed in Table 3 and the meshes are shown in Fig. 10. For the finest level_0 mesh, the first layer cell height satisfies $y^+ \approx 1$. All flow calculations for this case are performed using one processor.

Table 3 Statistics of the six meshes used for LS89

Mesh information	level_0	level_1	level_2	level_3	level_4	level_5
Number of grid points	90,192	57,936	39,078	27,654	19,118	14,520
First layer cell height [m]	10^{-6}	$0.8 \cdot 10^{-5}$	$1.0 \cdot 10^{-5}$	$1.2 \cdot 10^{-5}$	$1.4 \cdot 10^{-5}$	$1.7 \cdot 10^{-5}$

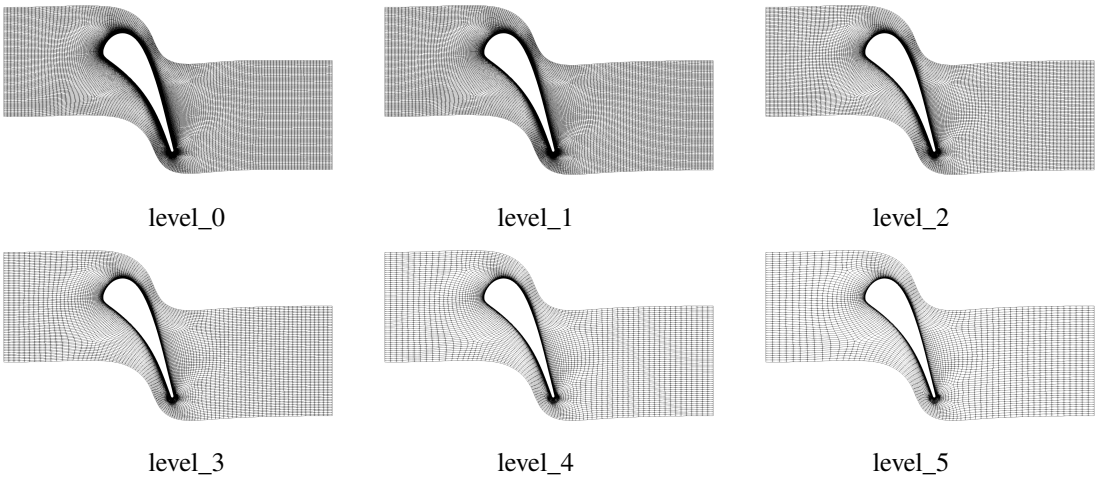

Fig. 10 Meshes used for the LS89 turbine cascade case.

2. Accuracy validation

The Mach number contour plots as well as the isentropic Mach number along the blade surface, according to Eq. (9), for the level_0 mesh, are shown in Fig. 11. Good agreement is found between numerical and experimental results.

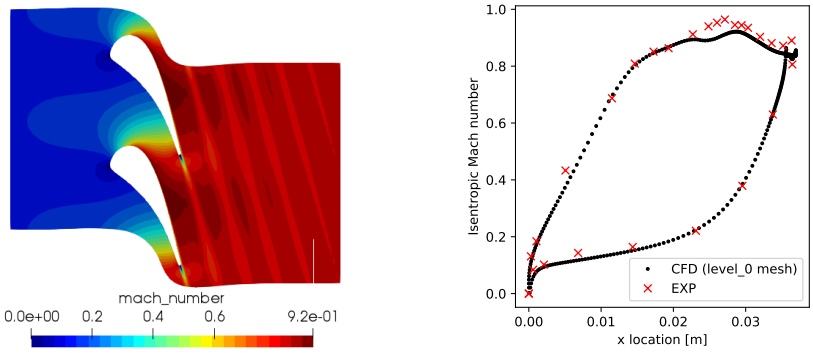

Fig. 11 Left: computed Mach number contours of L89; right: isentropic Mach number distribution along the blade surface computed by NutsCFD compared against experimental results [49].

3. Convergence behavior

For flow calculations on all six meshes, the flow is initialized with a uniform flow of Mach 0.84 and the corresponding static pressure and density are computed based on the inlet total pressure and total temperature. With an initial Courant number of $\sigma^0 = 1$, the solver stably reaches full convergence within a few hundred of nonlinear iterations for all six

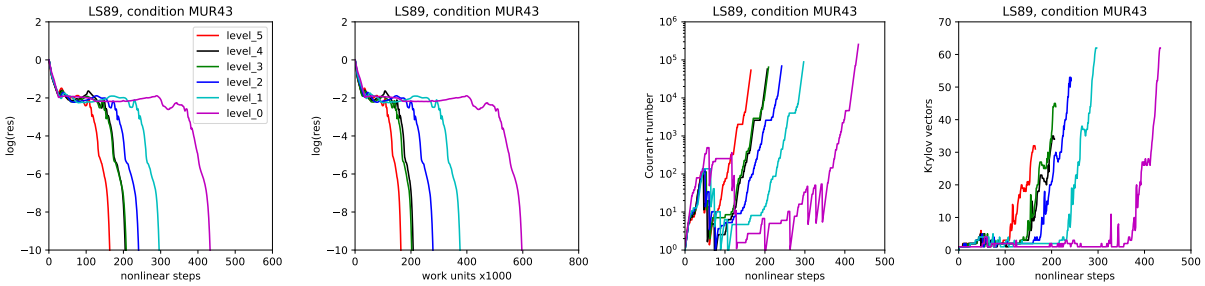

Fig. 12 Left to right: convergence history against iteration number and normalized CPU time in terms of work units, Courant number evolution and Krylov vector used in each step for computing the steady flow for LS89.

meshes. The residual convergence history against the nonlinear iterations and against work units, as well as the evolution of Courant number and number of Krylov vectors at each nonlinear iteration are plotted in Fig. 12. It can be seen that as the mesh density increases, the number of iterations also increases. In all cases, the solver struggles to converge during the initial transient phase, which is evident in the Courant number evolution, especially for the finest level_0 mesh: the Courant number stays below 10 for the first 400 iterations, before it starts to grow to large values and reaches steady state during the last few iterations. For full convergence it takes a few hundred thousands of work units, which is much larger than the NACA0012 near-buffet case (condition 'f'). Note that the LS89 is widely accepted to be a numerically challenging case due to the highly loaded design of the turbine cascade. Most in-house or commercial solvers are known to face difficulty to compute the steady solution for this case.

4. Analysis of computational cost

Finally, we investigate the proportion of the overall CPU time of each component of the NK algorithm to identify the solver performance bottlenecks. For each mesh, the percentage of the CPU time spent in (i) forming the second- and first-order Jacobian matrices, (ii) forming the ILU(0) preconditioner, and (iii) the GMRES solver is shown in Table 4. The Jacobian construction dominates the overall cost, accounting for almost 70–90% of the total solver runtime. As the mesh size increases the time spent in computing the ILU(0) preconditioner steadily increases. Overall the proportion of CPU time for the GMRES orthogonalization is between 1% to 3% for all cases, as the Krylov vector numbers used for the LS89 case are quite small (<70).

Table 4 CPU time [%] of each element of the NK algorithm for six LS89 mesh levels.

Alg. element	level_0	level_1	level_2	level_3	level_4	level_5
Forming 1st/2nd-order Jacobian	35.3	38.4	40.7	42.2	44.2	44.9
Computing ILU(0)	28.3	21.1	16.3	13.1	9.8	8.1
GMRES	1.0	2.0	2.2	2.5	1.8	2.1

C. Case 3: "Radiver" centrifugal compressor

1. Case overview

The "Radiver" centrifugal compressor is an open test case with extensive experimental results available for validation [50]. The case is a highly-loaded centrifugal compressor with an impeller provided by MTU Aero Engines.The key parameters of the compressor/impeller are tabulated in Table 5. In this work, we validate the NutsCFD solver by computing the 80% speedline using the simplified configuration with a vaneless diffuser. All flow calculations for computing the speedpline are performed using 72 processors.

Table 5 Technical data for the centrifugal compressor

Impeller tip speed	498 m/s
Shaft speed (100%)	35200 rev/min
Impeller tip radius	135 mm
Impeller tip gap	0.7 mm
Number of blades	15
Blade back sweep angle at impeller exit	38°
Diffuser channel height	11.1 mm
Diffuser exit radius	335 mm

2. Accuracy validation

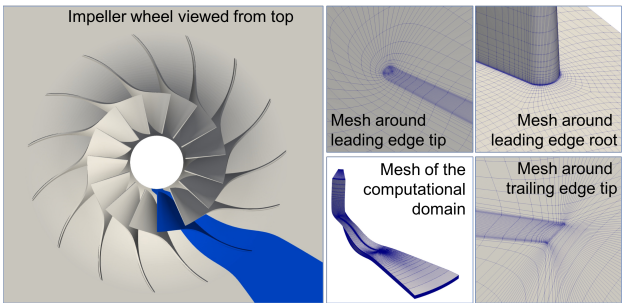

Fig. 13 The impeller wheel viewed from the top with one blade passage highlighted in blue (left) and the overview and detailed views of the computational mesh (right).

The steady state CFD calculation is performed for one blade passage only, which is meshed with hexahedral elements with a total of 905,953 grid points. The height of the first layer cell off the viscous wall is 10^{-6} m, satisfying $y^+ \approx 1$. The geometry of the impeller and the computational mesh are shown in Fig. 13. The total pressure ratio of the 80% speedline (28,160 rpm) for the vaneless configuration obtained from NutsCFD and the experimental measurements [50] are shown in Fig. 14. Overall the CFD results and the experimental data agree well with each other. The discrepancy in the choked mass flow is less than 0.3%. The pressure ratio near stall is over-predicted by about 4%. We attribute this difference to the way the total-pressure is measured between CFD and experiment: in the CFD results, the total

pressure ratio is calculated between the impeller inlet and the exit of the vaneless diffuser, while in the experiment, it was calculated between the settling chamber (upstream the impeller inlet) and the exit pressure pipe (downstream the vaneless diffuser channel exit).

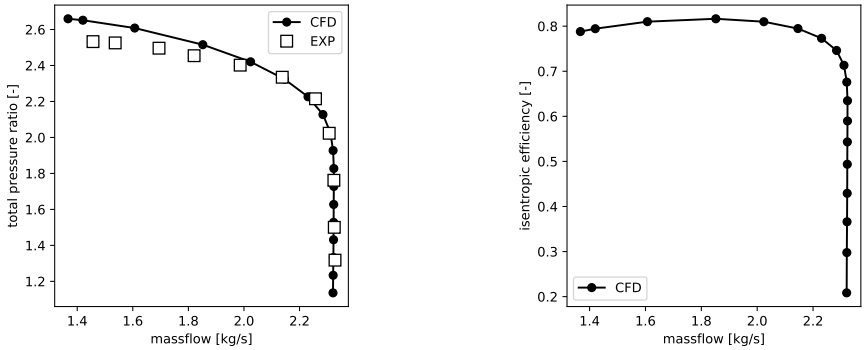

Fig. 14 CFD calculations v.s. measurements [50] for Radiver at 80% design rotational speed.

3. Automated speedline generation

To produce the speedline, an automated procedure is adopted using the back-pressure outlet boundary condition. A uniform flow initialization is used for the first data point, corresponding to the choked condition, with a zero gauge back pressure (0 Pa gauge pressure = 101,325 Pa absolute pressure). Subsequent calculations have an increment of 10 kPa in back pressure and are initialized using the converged solution from the previous lower back pressure. Towards the stall boundary, an increment of 10 kPa for the back pressure might cause the solution to become unstable. In such an event, the back pressure increment is reduced to 5 kPa, failing that it is further reduced to 1 kPa. When the solver does not converge even with a 1 kPa increment then the solver is terminated. The procedure is illustrated in Fig. 15.

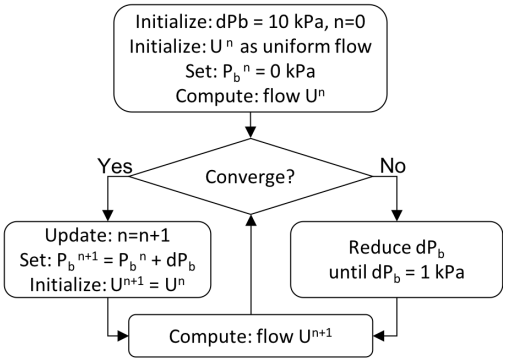

Fig. 15 Solution procedure for automatically generating the speedline. The subscript n denotes the n-th data point on the speedline.

The convergence history of residual and the exit mass flow for all the converged data points are shown in Fig. 16. The first run with zero back pressure initialized with uniform flow takes about 130 nonlinear iterations to converge.

Subsequent runs converge in fewer than 80 iterations. For a back pressure of 157 kPa even after initialization with the converged solution at a lower back pressure we could not converge the NK solver within 250 iterations and the solver is terminated. From the mass flow convergence history we find that for this back pressure the mass flow attains a negative value, indicating the appearance of reverse flow. Therefore, we mark the last converged condition with a back pressure of 156 kPa as the numerical stall boundary. Compared to the two-dimensional test cases (in sec III.A and sec III.B) the three-dimensional Radiver case requires a significantly larger number of GMRES vectors, reaching a terminal value of 400. As will be shown in detail in Sec. III.C.4, for such cases the computational cost of the GMRES dominates the overall cost. The cost of Jacobian forming and the ILU preconditioner becomes secondary. This is contrary to the two-dimensional case (see table 4) where the numerical stiffness is less severe.

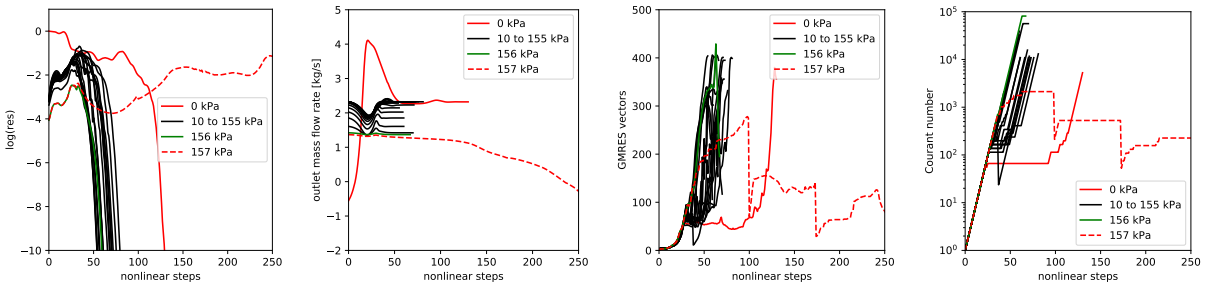

Fig. 16 Left to right: convergence history of residual, exit mass flow, number of Krylov vectors and the Courant number at each nonlinear iteration for automated runs generating the speedline, using 72 processors.

4. Parallel scalability

To demonstrate the scalability of the proposed parallel NK algorithm, the parallel speedup (and the corresponding parallel efficiency) of the solver when computing the flow for the Radiver case is shown in Fig. 17. The speedup of the NK solver is measured by the wall clock time to complete an additional nonlinear iteration for the choked condition (back pressure 0 kPa) after the residual has converged ten orders of magnitude. The parallel speedup of the residual evaluation is also plotted as a reference, which reflects the scalability of the underlying nonlinear residual subroutine (accounting for both the load imbalance of the partitioning and the MPI communication). It can be seen that the parallel scalability of the residual evaluation deteriorates steadily to 50% when the partition number increases from 1 to 72. In contrast to the monotonic decrease of the parallel efficiency for the residual evaluation, the parallel efficiency of the NK solver is superlinear for up to 24 partitions (over 80,000 grid points per domain), beyond which, the parallel efficiency reduces again.

To understand better how the parallel efficiency deteriorates for an inceased number of partitions, the total CPU time (wall clock time multiplied by the number of processors) taken by different major components of the NK algorithm is collected during the one nonlinear iteration and plotted in Fig. 18 (the values are scaled such that the largest number is 1).

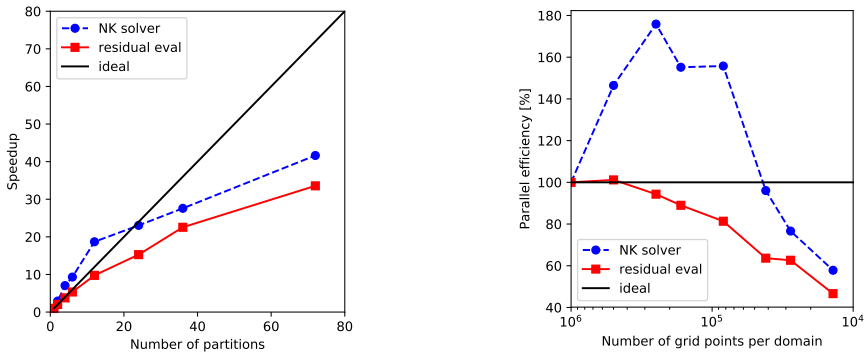

Fig. 17 The parallel speedup of the NK solver, compared with the residual evaluation as reference (left) and the corresponding parallel efficiency of both with respect to the number of grid points in each domain (right).

It can be seen that the cost for forming the Jacobian matrices remains nearly constant. The cost for computing ILU(0) almost linearly decreases as the partition number increases (more obvious in the log-log plot, not shown here). This is a consequence of the additive-Schwarz approach for computing ILU(0). The third major component is the GMRES solver whose cost increases overall with the partition number. This is closely related to an increase of GMRES vectors when partition number grows, from 166 vectors for 1 partition to 388 for 72 partitions, as shown in Fig. 18. This is also a consequence of the additive-Schwarz ILU(0) approach. As the preconditioning effect deteriorates with increasing partition numbers, more GMRES iterations are required to reach the same tolerance of $tol = 0.1$.

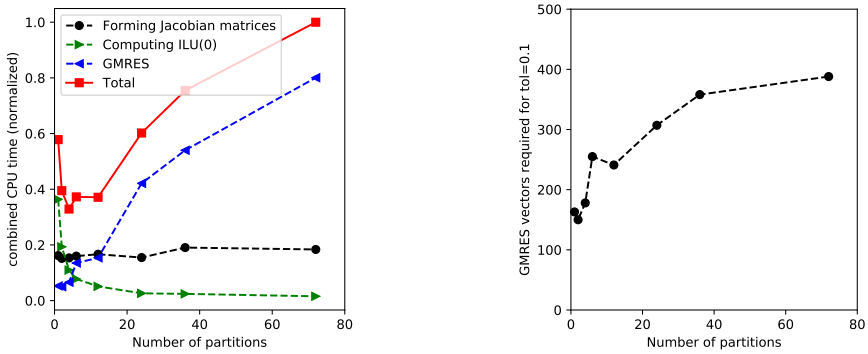

Fig. 18 Aggregated CPU time for different components of the NK solver (left) and the GMRES vectors required to achieve a fixed tolerance (right) with different partition numbers.

It can also be seen from Fig. 18 that the relative CPU time cost for different components of the NK solver varies for different partition numbers. It is already shown for the NACA0012 airfoil and LS89 cases that the cost obviously also depends on the particular flow condition investigated and the size of the mesh used (for example, see Table 4). The additional very complex behavior shown in Fig. 18 for the Radiver case further implies that in order to improve the efficiency of the parallel NK solver, various aspects of the solver need to be examined together and attention needs to be paid particularly to their strong interaction. Focusing on one single component and over-optimizing it may not improve

the overall performance. However, since the performance fine-tuning of the parallel NK solver is not the focus of this work and is thus not investigated further here.

D. Case 4: NASA Rotor 67 transonic fan

1. Case overview

The final case is the NASA Rotor 67 transonic fan, which is the first-stage rotor of a two-stage transonic fan. It has 22 low aspect-ratio blades and was designed for a rotational speed of 16,043 rpm, with a total pressure ratio of 1.63 and a mass flow of 33.25 kg/s [51]. The flow calculations for this case are performed using 72 processors.

2. Accuracy validation

The computational domain with a single blade passage is meshed with 973,065 grid points and 934,400 hexahedral elements. The height of the first layer cell off the viscous wall is 10^{-6} m, satisfying $y^+ \approx 1$. The geometry of the rotor blade as well as detailed views of the mesh are shown in Fig. 19.

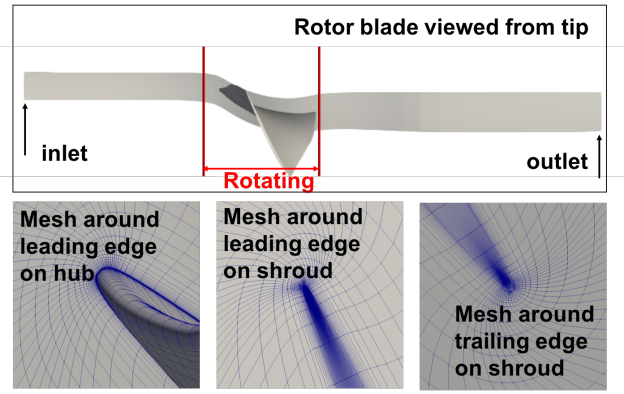

Fig. 19 NASA rotor 67 blade viewed from the tip (top) and detailed views of the computational mesh (bottom). The rotating part of the hub surface, from $x = -1.374$ cm to $x = 9.365$ cm, is marked with the two vertical lines.

At the inlet boundary, a total pressure of 101325 Pa and a total temperature of 288.15 K are imposed. The incoming flow is in the axial direction. Since the outlet boundary is relatively far from the rotor, i.e., the distance from the rotor trailing edge to the outlet boundary is over twice the chord length, a constant back pressure is imposed at the outlet as the boundary condition. Our numerical experiment shows that using a radial equilibrium boundary condition only changes the resulting speedline negligibly. Therefore, the constant back pressure boundary condition is used for simplicity. The back pressure is set to 0 kPa initially and then gradually raised to produce the speedline. The procedure illustrated in Fig. 15, is used to generate the speedline, except that for this case, pressure increments of 5 kPa, 1 kPa and 0.1 kPa are used.

The experimental and numerical speedlines are shown in Fig. 20 for comparison, both plotted against the massflow rate normalized at choked condition. It should be mentioned that the choked massflow rate predicted by CFD is 34.62kg/s,

about 1.0% lower than measurement result. This under-prediction is similar to other numerical investigations [52]. The slight under-prediction of the total pressure ratio is suspected to be caused by the fact that SA turbulence model is not fully appropriate for this case. However, as this work focuses on the solver convergence and robustness, this is not further investigated here.

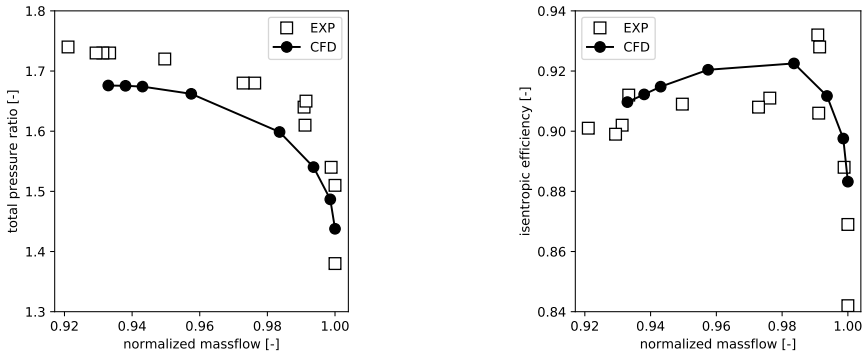

Fig. 20 CFD calculations compared with measurements [51] for NASA Rotor 67.

3. Convergence behaviour

The convergence history of the residual and the exit mass flow for the various runs to generate the speedline are shown in Fig. 21. The last fully-converged point is for a back pressure of 21.3 kPa, for which the converged solution has a mass flow rate of 9.53 kg/s and a total pressure ratio of 1.418, significantly lower than other points. This point is believed to be in the post-stall regime and thus the second last stable point (back pressure 21.2 kPa) is taken as the numerical stall boundary. The evolution of the Courant number and the Krylov vectors used at each nonlinear iteration are also shown in Fig. 21. It can be seen that, except for the first (choke) and the last (stall) runs, all other runs (except one outlier) converge in less than 100 iterations, due to the initialization using the converged flow from a lower back pressure.

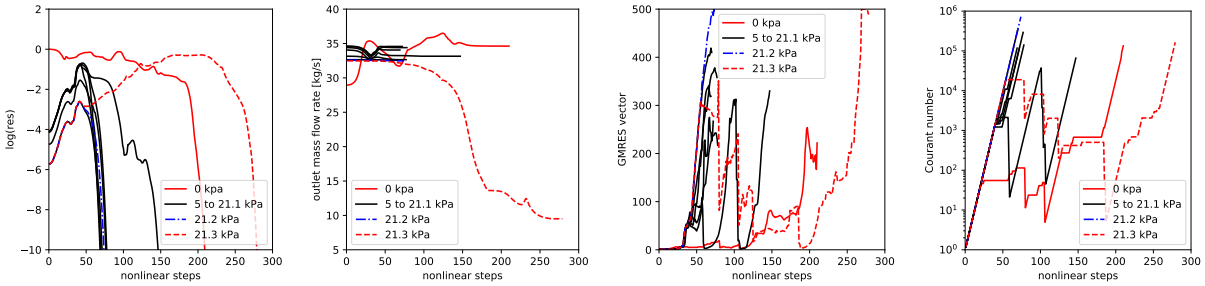

Fig. 21 NASA rotor 67. Left to right: convergence history of residual, exit mass flow, number of Krylov vectors and the Courant number at each nonlinear iteration for runs generating the speedline, using 72 processors.

E. Memory overhead

For typical calculations of each of the four cases, the memory required for storing the Jacobian matrix, the ILU(0) preconditioner, and the maximum number of Krylov vectors used during the solution process (not the default number of 500) are listed in Table 6. The statistics for the Radiver and Rotor 67 cases is for the total memory used by all 72 processors. The number in the parenthesis, m, is the maximum number of Krylov vectors reached during the nonlinear solution process. It can be seen that the Jacobian-forming NK approach adopted in this work is indeed quite memory heavy, requiring up to 35 GB for calculating a case with 1,000,000 grid points.

Table 6 Memory overhead for all cases at different conditions.

Case	Condition	Grid points	Jacobian or ILU(0) [GB]	Krylov vectors (m) [GB]
NACA0012	'c', level_0	134,976	1.28	1.43 (219)
	'f', level_0	134,976	1.28	0.71 (101)
LS89	MUR43, level_0	90,192	0.86	0.26 (60)
	MUR43, level_1	57,936	0.55	0.17 (62)
	MUR43, level_2	39,078	0.37	0.11 (62)
	MUR43, level_3	27,654	0.26	0.08 (53)
	MUR43, level_4	19,118	0.18	0.04 (35)
	MUR43, level_5	14,520	0.14	0.02 (32)
Radiver	choked condition	905,953	8.60	17.38 (388)
Rotor 67	choked condition	973,065	9.25	12.24 (245)

IV. Conclusion

This paper presents the algorithmic development and detailed implementation of a Reynolds-averaged Navier–Stokes solver that specifically aims at the robust solution of steady state turbomachinery flow problems. The proposed algorithm features the use of Newton's method for the nonlinear iteration. During each outer iteration, the exact Jacobian matrix is formed using algorithmic differentiation and graph coloring, and the resulting large sparse linear system of equations is solved using preconditioned Krylov methods. The robustness of the solver is enhanced using a globalization strategy, which adaptively updates the time step to overcome the convergence difficulties during strong nonlinear transients. Parallelization is achieved via a two-halo domain-decomposition approach, which not only simplifies the MPI communication pattern and the forming of the Jacobian matrix, but also makes the extension to linearized analysis tools extremely straightforward. This work is the first attempt reported in literature on the use of the globalized Newton–Krylov method for turbomachinery aerodynamic analysis.

The solver developed is applied to the steady state flow computation for a two-dimensional airfoil, a two-dimensional turbine cascade, a three-dimensional centrifugal compressor, and a three-dimensional axial compressor. Good solution accuracy is demonstrated by comparing the results against available experimental data. Satisfactory parallel efficiency

is achieved down to 80,000 grid points per partition. Whether starting from scratch or a converged solution for a similar condition, the solver always stably reaches machine-zero converged solutions, showing the superior robustness of the solver. This is an enabling feature if reliable automated analysis of turbomachinery flows over a wide operating range were desired. The main bottleneck limiting the performance of the current solver is the expensive forming of the Jacobian matrix (costing approximately 400 residual evaluations) using the algorithmic differentiation tool. Analytical approach to derive the Jacobian matrix, although tedious and error-prone, could be explored in order to accelerate this.

Author contributions

Shenren Xu and Pavanakumar Mohanamuraly made equal contributions to this work.

Acknowledgements

We thank Dr Tom Verstraete of the von Kármán Institute for Fluid Dynamics for the invaluable help on the implementation of the rotating frame of reference. We thank the Institute of Jet Propulsion and Turbomachinery at RWTH Aachen, Germany, for providing the geometry and test results of the Radiver case (part of their investigations was funded by the Deutsche Forschungsgemeinschaft) and the permission to use the data in this paper. Part of the work in this paper was carried out at Queen Mary University of London, funded by Mitsubishi Heavy Industries and by the European Commission's Horizon 2020 programme under Grant Agreement No. 642959, the IODA project. Shenren Xu and Dingxi Wang were also supported by the National Natural Science Foundation of China (Grant No. 51790512).

References

[1] Denton, J., and Dawes, W., "Computational fluid dynamics for turbomachinery design," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, Vol. 213, No. 2, 1998, pp. 107–124.

[2] Yang, H., Nuernberger, D., and Kersken, H.-P., "Toward excellence in turbomachinery computational fluid dynamics: A hybrid structured-unstructured Reynolds-averaged Navier–Stokes solver," *Journal of Turbomachinery*, Vol. 128, No. 2, 2006, pp. 390–402.

[3] Lapworth, L., "Hydra-CFD: a framework for collaborative CFD development," *International Conference on Scientific and Engineering Computation (IC-SEC), Singapore, June*, Vol. 30, 2004.

[4] Pinto, R., Afzal, A., D'Souza, L., Ansari, Z., and Samee, A., "Computational fluid dynamics in turbomachinery: a review of state of the art," *Archives of Computational Methods in Engineering*, Vol. 24, No. 3, 2017, pp. 467–479.

[5] Hills, N., "Achieving high parallel performance for an unstructured unsteady turbomachinery CFD code," *The Aeronautical Journal*, Vol. 111, No. 1117, 2007, pp. 185–193.

[6] Tucker, P., "Computation of unsteady turbomachinery flows: Part 2–LES and hybrids," *Progress in Aerospace Sciences*, Vol. 47, No. 7, 2011, pp. 546–569.

[7] Denton, J., "Some limitations of turbomachinery CFD," *ASME Turbo Expo 2010: Power for Land, Sea, and Air*, American Society of Mechanical Engineers, 2010, pp. 735–745.

[8] Moinier, P., Müller, J.-D., and Giles, M., "Edge-based multigrid and preconditioning for hybrid grids," *AIAA Journal*, Vol. 40, No. 10, 2002, pp. 1954–1960.

[9] Langer, S., "Agglomeration multigrid methods with implicit Runge–Kutta smoothers applied to aerodynamic simulations on unstructured grids," *Journal of Computational Physics*, Vol. 277, 2014, pp. 72–100.

[10] Campobasso, M., and Giles, M., "Effects of flow instabilities on the linear analysis of turbomachinery aeroelasticity," *Journal of Propulsion and Power*, Vol. 19, No. 2, 2003, pp. 250–259.

[11] Xu, S., Radford, D., Meyer, M., and Müller, J.-D., "Stabilisation of discrete steady adjoint solvers," *Journal of Computational Physics*, Vol. 299, 2015, pp. 175–195.

[12] Kamenetskiy, D., Bussoletti, J., Hilmes, C., Venkatakrishnan, V., Wigton, L., and Johnson, F., "Numerical evidence of multiple solutions for the Reynolds-averaged Navier–Stokes equations," *AIAA Journal*, Vol. 52, No. 8, 2014, pp. 1686–1698.

[13] Johnson, F., Kamenetskiy, D., Melvin, R., Venkatakrishnan, V., Wigton, L., Young, D., Allmaras, S., Bussoletti, J., and Hilmes, C., "Observations regarding algorithms required for robust CFD codes," *Mathematical Modelling of Natural Phenomena*, Vol. 6, No. 3, 2011, pp. 2–27.

[14] Giles, M., Duta, M., Müller, J.-D., and Pierce, N., "Algorithm developments for discrete adjoint methods," *AIAA Journal*, Vol. 41, No. 2, 2003, pp. 198–205.

[15] Badcock, K. J., Timme, S., Marques, S., Khodaparast, H., Prandina, M., Mottershead, J., Swift, A., Da Ronch, A., and Woodgate, M., "Transonic aeroelastic simulation for instability searches and uncertainty analysis," *Progress in Aerospace Sciences*, Vol. 47, No. 5, 2011, pp. 392–423.

[16] Campobasso, M., and Giles, M., "Stabilization of a linear flow solver for turbomachinery aeroelasticity using Recursive Projection Method," *AIAA Journal*, Vol. 42, No. 9, 2004, pp. 1765–1774.

[17] Richez, F., Leguille, M., and Marquet, O., "Selective frequency damping method for steady RANS solutions of turbulent separated flows around an airfoil at stall," *Computers & Fluids*, Vol. 132, 2016, pp. 51–61.

[18] Langer, S., "Hierarchy of Preconditioning Techniques for the Solution of the Navier-Stokes equations discretized by 2nd order finite volume methods," *6th European Congress on Computational Methods in Applied Sciences and Engineering*, 2012.

[19] Langer, S., "Preconditioned Newton methods to approximate solutions of the Reynolds averaged Navier-Stokes equations," Tech. rep., Institut für Aerodynamik und Strömungstechnik, Juni 2018. URL `https://elib.dlr.de/121515/`.

[20] Knoll, D. A., and Keyes, D. E., "Jacobian-free Newton–Krylov methods: a survey of approaches and applications," *Journal of Computational Physics*, Vol. 193, No. 2, 2004, pp. 357–397.

[21] Barth, T., and Linton, S., "An unstructured mesh Newton solver for compressible fluid flow and its parallel implementation," *AIAA Paper 95-0221*, 1995.

[22] Pueyo, A., and Zingg, D. W., "Efficient Newton–Krylov solver for aerodynamic computations," *AIAA Journal*, Vol. 36, No. 11, 1998, pp. 1991–1997.

[23] Bramkamp, F. D., Bücker, H. M., and Rasch, A., "Using Exact Jacobians in an Implicit Newton–Krylov Method," *Computers & Fluids*, Vol. 35, No. 10, 2006, pp. 1063–1073.

[24] Nemec, M., and Zingg, D. W., "Newton-Krylov algorithm for aerodynamic design using the Navier-Stokes equations," *AIAA Journal*, Vol. 40, No. 6, 2002, pp. 1146–1154.

[25] Wong, P., and Zingg, D. W., "Three-dimensional aerodynamic computations on unstructured grids using a Newton–Krylov approach," *Computers & Fluids*, Vol. 37, No. 2, 2008, pp. 107–120.

[26] Osusky, M., and Zingg, D. W., "Parallel Newton–Krylov–Schur flow solver for the Navier–Stokes equations," *AIAA Journal*, Vol. 51, No. 12, 2013, pp. 2833–2851.

[27] Allmaras, S., and Johnson, F., "Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model," *Seventh International Conference on Computational Fluid Dynamics (ICCFD7)*, 2012, pp. 1–11.

[28] Spalart, P. R., and Allmaras, S. R., "A One-Equation Turbulence Model for Aerodynamic Flows," *Recherche Aerospatiale*, Vol. 1, No. 1, 1994, pp. 5–21.

[29] Roe, P. L., "Approximate Riemann solvers, parameter vectors, and difference schemes," *Journal of Computational Physics*, Vol. 43, No. 2, 1981, pp. 357–372.

[30] Blazek, J., *Computational fluid dynamics: principles and applications*, Butterworth-Heinemann, 2015.

[31] Weiss, J. M., Maruszewski, J. P., and Smith, W. A., "Implicit solution of preconditioned Navier–Stokes equations using algebraic multigrid," *AIAA Journal*, Vol. 37, No. 1, 1999, pp. 29–36.

[32] Hascoët, L., and Pascual, V., "The Tapenade Automatic Differentiation tool: Principles, Model, and Specification," *ACM Transactions On Mathematical Software*, Vol. 39, No. 3, 2013.

[33] Gebremedhin, A. H., Manne, F., and Pothen, A., "What color is your Jacobian? Graph coloring for computing derivatives," *SIAM Review*, Vol. 47, No. 4, 2005, pp. 629–705.

[34] Gebremedhin, A. H., Nguyen, D., Patwary, M. M. A., and Pothen, A., "ColPack: Software for graph coloring and related problems in scientific computing," *ACM Transactions on Mathematical Software (TOMS)*, Vol. 40, No. 1, 2013, p. 1.

[35] Michalak, C., and Ollivier-Gooch, C., "Globalized matrix-explicit Newton–GMRES for the high-order accurate solution of the Euler equations," *Computers & Fluids*, Vol. 39, No. 7, 2010, pp. 1156–1167.

[36] Mccracken, A., Ronch, A. D., Timme, S., and Badcock, K. J., "Solution of linear systems in Fourier-based methods for aircraft applications," *International Journal of Computational Fluid Dynamics*, Vol. 27, No. 2, 2013, pp. 79–87.

[37] Xu, S., and Timme, S., "Robust and efficient adjoint solver for complex flow conditions," *Computers & Fluids*, Vol. 148, 2017, pp. 26–38.

[38] Chisholm, T. T., and Zingg, D. W., "A Jacobian-free Newton–Krylov algorithm for compressible turbulent fluid flows," *Journal of Computational Physics*, Vol. 228, No. 9, 2009, pp. 3490–3507.

[39] Mavriplis, D., "A residual smoothing strategy for accelerating Newton method continuation," *ArXiv e-prints*, 2018.

[40] Parks, M., de Sturler, E., Mackey, G., Johnson, D., and Maiti, S., "Recycling Krylov subspaces for sequences of linear systems," *SIAM Journal on Scientific Computing*, Vol. 28, No. 5, 2006, pp. 1651–1674.

[41] Jalali, A., and Ollivier-Gooch, C., "Higher-order unstructured finite volume RANS solution of turbulent compressible flows," *Computers & Fluids*, Vol. 143, 2017, pp. 32–47.

[42] Sarje, A., Song, S., Jacobsen, D., Huck, K., Hollingsworth, J., Malony, A., Williams, S., and Oliker, L., "Parallel Performance Optimizations on Unstructured Mesh-based Simulations," *Procedia Computer Science*, Vol. 51, 2015, pp. 2016–2025.

[43] Karypis, G., and Kumar, V., "METIS–unstructured graph partitioning and sparse matrix ordering system, version 2.0," Tech. rep., Dept of Computer Science, University of Minnesota, Minneapolis, 1995.

[44] Keyes, D. E., "Aerodynamic applications of Newton- Krylov-Schwarz solvers," *Lecture Notes in Physics*, Vol. 453, No. 453, 1995, pp. 1–20.

[45] Saad, Y., and Sosonkina, M., "Distributed Schur Complement Techniques for General Sparse Linear Systems," *SIAM Journal on Scientific Computing*, Vol. 21, No. 4, 1997, pp. 1337–1356.

[46] Schlichting, H., and Shapiro, A. H., "Boundary Layer Theory, Sixth Edition," *Journal of Applied Mechanics*, Vol. 35, No. 4, 1968, p. 569–588.

[47] Gregory, N., and O'Reilly, C. L., "Low-speed aerodynamic characteristics of naca0012 aerofoil section, including the effects of upper-surface roughness simulating hoar frost," *NASA R&M 3726*, 1970.

[48] Crouch, J. D., Garbaruk, A., and Magidov, D., "Predicting the onset of flow unsteadiness based on global instability," *Journal of Computational Physics*, Vol. 224, No. 2, 2007, pp. 924–940.

[49] Arts, T., and Lambert de Rouvroit, M., "Aero-Thermal Performance of a Two-Dimensional Highly Loaded Transonic Turbine Nozzle Guide Vane: A Test Case for Inviscid and Viscous Flow Computations," *Journal of Turbomachinery*, Vol. 114, No. 1, 1992, p. 147.

[50] Ziegler, K. U., Gallus, H. E., and Niehuis, R., "A study on impeller-diffuser interaction: part I—influence on the performance," *ASME Turbo Expo 2002: Power for Land, Sea, and Air*, American Society of Mechanical Engineers, 2002, pp. 545–556.

[51] Strazisar, A. J., Wood, J. R., Hathaway, M. D., and Suder, K. L., "Laser anemometer measurements in a transonic axial-flow fan rotor," Tech. Rep. NASA TP-2879, NASA, Nov 1989.

[52] Arnone, and A., "Viscous Analysis of Three-Dimensional Rotor Flow Using a Multigrid Method," *Journal of Turbomachinery*, Vol. 116, No. 3, 1994, p. V001T03A008.