Journal of Computational Physics 299 (2015) 175–195

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Stabilisation of discrete steady adjoint solvers

Shenren Xu [a], David Radford [b], Marcus Meyer [c], Jens-Dominik Müller [a,*]

[a] *School of Engineering and Materials Science, Queen Mary, University of London, Mile End Road, London E1 4NS, United Kingdom*
[b] *Rolls-Royce plc, Derby DE24 8BJ, United Kingdom*
[c] *Rolls-Royce Deutschland Ltd & Co KG, D-15827 Blankenfelde-Mahlow, Germany*

A R T I C L E   I N F O

Article history:
Received 9 June 2014
Received in revised form 10 April 2015
Accepted 30 June 2015
Available online 7 July 2015

Keywords:
Reynolds-averaged Navier–Stokes
Discrete adjoint
Implicit
GMRES
Multigrid
FAS-cycle

A B S T R A C T

A new implicit time-stepping scheme which uses Runge–Kutta time-stepping and Krylov methods as a smoother inside FAS-cycle multigrid acceleration is proposed to stabilise the flow solver and its discrete adjoint counterpart. The algorithm can fully converge the discrete adjoint solver in a wide range of cases where conventional point-implicit methods fail due to either physical or numerical instability. This enables the discrete adjoint to be applied to a much wider range of flow regimes. In addition, the new algorithm offers improved efficiency when applied to stable cases for which the conventional Block–Jacobi solver can fully converge. Both stable and unstable cases are presented to demonstrate the improved robustness and performance of the new scheme. Eigen-analysis is presented to outline the mechanism of the adjoint stabilisation effect.

1. Introduction

The adjoint method is an essential ingredient of gradient-based steady-state CFD shape optimisation as it allows the computation of the gradient of an objective function with respect to a large number of design variables at near constant computational cost comparable to that of the flow solution. Two approaches are most prominently used to develop adjoint codes, the continuous and the discrete adjoint approach [1–4]. The continuous adjoint approach re-discretises the adjoint PDE, which offers the possibility to optimise and/or stabilise the adjoint solver by tuning the discretisation. The steady-state discrete adjoint starts from the discretised equations which are then linearised around the converged steady state flow field and transposed. This ensures that the computed gradients are the exact gradients of the discrete model, which is a very desirable property: the discrete gradient is then exactly zero where the flow solution has an unconstrained minimum. This also means that the properties of the flow discretisation, including any preconditioners, are inherited by the adjoint and govern its spectral behaviour. This ensures that if the primal solution (flow) converges (i.e. the system Jacobian is contractive with the magnitude of all eigenvalues $|\lambda| \leq 1$), so will the discrete adjoint as the exact transposition of the system Jacobian preserves the eigenvalues [5].

However, in many industrial cases the flow does not converge in this sense, but enters limit cycle oscillations (LCO). This can be caused for example by the mesh being fine enough in some area to resolve certain localised flow unsteadiness due to vortex shedding, or for example by separation bubbles in the flow which have only very loose physical coupling with the bulk flow. This is typically not perceived as a problem for the analysis of the flow field as the value of the objective function is often steady enough and considered accurate enough to be used. On the other hand, applying the 'steady-state' discrete

* Corresponding author.
E-mail address: j.mueller@qmul.ac.uk (J.-D. Müller).

http://dx.doi.org/10.1016/j.jcp.2015.06.036
0021-9991/© 2015 Elsevier Inc. All rights reserved.

approach in this situation often fails, as a Jacobian taken from a snapshot during the LCO is likely to be not contractive but to exhibit a number of eigenvalues with magnitude larger than one, and hence the iterative scheme for the adjoint will fail once the error modes associated with those eigenvalues have grown sufficiently [6].

To stabilise the linear solver, the Recursive Projection Method (RPM) and the Generalised Minimal Residual Method (GMRES) have been proposed [7,8]. GMRES [9] is an iterative method that is guaranteed not to diverge even in the presence of eigenvalue outliers. Therefore the existing linear fixed-point iteration (FPI) can be used as a preconditioner for the GMRES solver, and the adjoint GMRES solver, provided with enough Krylov vectors, can converge. This approach can be implemented with minimal amount of change to the existing code and should be capable of stabilising the adjoint. The main drawback is that for large industrial cases, a large number of Krylov vectors needs to be used in order to prevent convergence from stalling, e.g., 100 Krylov vectors with no restarts were used in [10].

RPM stabilises the adjoint with a different mechanism. Before RPM is switched on, the unstable FPI algorithm is performed for many iterations to allow the dominant error mode to grow and to be identified. Then a direct solver is applied to that particular mode and the existing FPI is applied to the remaining modes. If the remaining modes still contain unstable modes, then the unstable FPI will identify again the most unstable mode and the procedure above is repeated until all the unstable modes are picked out and the FPI is stable for all the remaining modes. The RPM method needs to run the FPI adjoint solver for k iterations in order to identify the dominant unstable modes, and a typical value can be $k = 100$–$10\,000$, or even larger for practical problems. Unfortunately, k cannot be determined *a priori* [7].

An additional problem with both RPM and GMRES is that the linearisation is then based on an unstable saddle-point of the flow solution which is arbitrarily picked from the LCO, hence the adjoint sensitivity may not be inconsistent with the average of the primal over the LCO [11]. Although stabilised and fully converged, the adjoint solution and hence the sensitivities will depend on which solution snapshot is chosen for the linearisation.

The objective function the designer is typically interested in is actually not the value of the function at an arbitrary snapshot but the average objective function over a long time-interval. To approximate this average, one could take the average of the flow field in the time interval and base the Jacobian on this averaged state. Irrespective of the question whether the objective function is linear enough with respect to the state to justify this approximation, in general the Jacobian will fail to be contractive as this averaged solution does not satisfy the steady state flow equations. Still, this approach has shown to be practicable for some limited cases of industrial relevance when used with continuous adjoint methods, where the re-discretisation of the adjoint equation can provide additional stabilisation terms [12]. For a rigorous application of this approach, regularisation in time would be required that gives rise to stabilising contributions to the Jacobian, but the knowledge in the field on this is in its early stages [13].

An approach guaranteed to work would be to treat the limit cycles as an unsteady flow and trace the adjoint characteristics backward in time [14]. Storing checkpoints and recomputing intermediate solutions would result in a significant increase in runtime and memory requirements, which would only be warranted if the time-averaged objective function is non-linearly affected by the instability.

So far we have not distinguished whether the instability of the iterative method is due to the flow physics, such as vortex shedding and separation bubbles decoupled from the mean flow, or whether it is due to the discretisation. There cannot be a clear distinction between the two as for example vortex shedding behind rounded trailing edges of turbine blades can be suppressed with coarse meshing and large time steps. Putting numerical stability into focus suggests another approach to achieve convergence of the adjoint solver, namely to improve the stability of the flow discretisation such that the flow can be converged to a level that the Jacobian is contractive. Clearly, such an approach may fail for flows with strong physical unsteadiness or will be inaccurate where the unsteady phenomena have a significant effect on the average of the objective function. But for many cases such an approach will be stable and sufficiently accurate, as well as being significantly less expensive than the computation of the unsteady flow and adjoint. In the authors' view, apart from [11], the approach of obtaining stable discrete adjoints through focusing on the primal stability has not been explored adequately in the literature.

Before presenting the proposed iterative scheme, a brief overview of the various techniques for accelerating convergence of nonlinear flow solvers, with particular focus on RANS, is given in the next section.

1.1. Convergence acceleration techniques for nonlinear flow solvers

A fixed-point iteration can generally be represented as

$$\mathbf{P}\delta U^n = -R(U^n) \tag{1}$$

where the right hand side represents the residual of the discretisation which determines the accuracy of the converged solution, while the left hand side matrix $\mathbf{P}$ is a non-singular preconditioning matrix that controls the transient behaviour of the intermediate solution U^n over the iterations. At each iteration, the flow update is computed by inverting $\mathbf{P}$ directly or approximately.

A hierarchy of time-marching methods can be derived by using for the preconditioning matrix $\mathbf{P}$ different approximations of the flow Jacobian matrix $\partial R/\partial U$ [15], as illustrated in Fig. 1. Scalar time stepping, with either a uniform or a spatially varying time step, is too inefficient for practical steady RANS cases due to the highly stretched mesh in the boundary layer regions and the numerical decoupling between the equations. The physical coupling can be included by retaining the diagonal blocks of the Jacobian in the preconditioning matrix, thus termed a point-implicit or Block–Jacobi (B–J) solver.

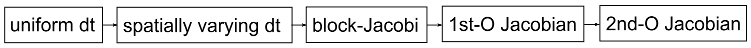

Fig. 1. A spectrum of various time-stepping methods based on different approximations of the Jacobian.

A B–J solver accelerated with multigrid [16] is in general efficient enough to compute viscous flows when combined with semi-coarsened geometric multigrid [17]. In addition, the B–J solver is easy to implement, to parallelise and has very low memory requirements. The drawback is that the convergence tends to degenerate for large industrial cases, especially for cases where viscous effects dominate. Furthermore, as stated in the introduction, for large industrial cases with complex geometries, the B–J flow solver may converge only to LCO after an initial residual drop, corresponding to pseudo-unsteadiness of either numerical or physical origins.

A better approximation of the Jacobian can be constructed by including the off-diagonal blocks in the Jacobian matrix, coupling the neighbouring nodes. This leads to an implicit solver with faster convergence by damping the transient modes more effectively. The highest level of approximation is obviously the exact second-order Jacobian, i.e., the exact linearisation of the residual with respect to the flow variables, which could for example be used with Newton's method. Such a Newton solver is applied to a 2-D viscous RAE2822 case in [18]. In addition to a start-up difficulty, which requires such stringent control of the step-width that it renders the Newton solver practically useless for this application, it is reported that the resulting linear system at each nonlinear iteration is very stiff and the GMRES(50) solver needs to be preconditioned by ILU(4) in order to prevent stalling. The combined memory requirements of the exact Jacobian and ILU(4) combined are reported to be roughly 15 times that of B–J in 2-D. The memory requirements will be even more substantial in 3-D due to the increased number of neighbouring nodes.

Jacobian–Free Newton–Krylov (JFNK) methods [19] can avoid storing the Jacobian but instead work out the Jacobian-vector product needed in the Krylov solvers on the fly, typically using finite differences,

$$\frac{\partial R}{\partial U}\cdot\delta U=\frac{R(U+\epsilon\delta U)-R(U-\epsilon\delta U)}{2\epsilon}. \tag{2}$$

However, for the Krylov solver to converge, effective preconditioners are needed, with ILU factorisation being recognised as one of the most effective. In order to limit memory overhead, the Jacobian that is ILU-factorised could be approximated [19], but practical preconditioners require storage equivalent to the Jacobian.

Major storage savings arise from dropping second-order neighbours and computing the Jacobian on nearest-neighbour contributions only. For a standard MUSCL scheme [20] this would correspond to frozen spatial gradients. In our implementation we use a zero spatial gradient for the MUSCL scheme when computing the inviscid flux and a non-zero but frozen gradient for computing the viscous flux. The resulting memory savings of course are paid for with an impaired convergence rate. While an exact second-order Jacobian typically does not lead to a robust scheme [18], requiring at the least a sophisticated solver steering strategy [21], a heuristic blend of first and second-order Jacobians can provide very good results [22], but again at the cost of very large memory use.

Multigrid (MG) is a very effective preconditioner, especially in its full approximation storage form (FAS) [23] which includes non-linear effects on the coarser grids. On each grid level FAS-MG needs to solve the non-linear equations using an h-elliptic discretisation [23] that has good high-frequency smoothing. Unfortunately Krylov solvers such as GMRES on their own do not provide adequate high-frequency damping to be used as a smoother within MG, but it is possible to use MG as a preconditioner for JNFK [19], which however then reduces to the linear Correction Scheme (CS) MG which is much less effective [24].

Algebraic Multigrid (AMG) does not require to build a coarse grid, but the coarsening is applied to the system matrix of the linear problem, rather than the grid [25]. It is hence a linear CS MG scheme, which avoids re-discretisation on the coarse grid, but hence loses the convergence advantage of the non-linear coarse-grid re-discretisation. This may be compensated by efficient directional coarsening in all areas of the flowfield, not just the viscous layer.

Swanson et al. [26] showed that an implicit discretisation with symmetric Gauss–Seidel (SGS), preconditioned with the first-order Jacobian is a good smoother for multigrid [27] if the linear system is wrapped inside a standard Runge–Kutta (RK) multistage scheme, with RK providing the desired damping for the high-frequency error modes. Furthermore, the RK coefficients could be fine-tuned for better robustness and performance [28]. This approach of solving the first-order linear system using SGS at each RK stage, accelerated by MG at the outer iteration, has proven to be robust and is now generally accepted as the benchmark for solver performance. SGS is susceptible to a lack of diagonal dominance, and thus when SGS is used to solve the linear system, the CFL number is lowered not for the nonlinear instability of the outer iteration, but for the linear instability of the inner SGS iteration, reducing the convergence rate of the scheme.

The performance of the MG method does strongly depend on the method that is used to generate the coarse grids and how the problem is discretised on them. Agglomeration MG fuses fine grid cells into coarse grid cells which then are no longer convex [29]. The scheme can be implemented as an efficient non-linear FAS scheme if the equations are re-discretised on the coarser meshes, however an accurate discretisation of the viscous operator is not straightforward on the non-convex cells. Directional agglomeration with additional line-relaxation is needed for high-Reynolds number flows [24].

Geometric coarsening through edge-collapsing is very effective for simplex grids, but much more difficult to achieve for hybrid grids. The element-collapsing method [17,16] collapses sets of edges to remove elements and produces coarse grids with the standard element types, possibly with degenerate edges. Semi-coarsened grids for high-Reynolds flows can be produced. However the mesh quality and coarsening ratio degenerate with repeated application on the coarser levels.

The authors acknowledge that the performance of a particular smoother is very closely linked to the way the coarse grids are generated, the results presented here use the geometric element-collapsing scheme of Müller [17]. Assessing the performance of the proposed iterative method in the context of for example agglomeration multigrid or AMG will be reported in future work.

1.2. Proposed new algorithm

The survey of the existing approaches in Section 1.1 shows that a robust RANS method should make use of a) non-linear FAS multigrid which offers improved convergence, b) robust GMRES linear solvers which allow large CFL numbers, and c) strong preconditioners.

At the time of writing, to the authors' knowledge there was no published algorithm using multigrid as a RANS solver ('outside') and GMRES as part of the smoother ('inside'), due to the poor smoothing properties of the Krylov solvers. Developing a solution to this problem and successful application of such an algorithm is the main objective of this paper. To achieve adequate high-frequency damping of the GMRES linear solver, in the proposed scheme GMRES is embedded in a Runge–Kutta multistage time-stepping scheme, preconditioned by an ILU(0)-factorisation of the first-order Jacobian. The proposed method is hence referred to as the Jacobian-Trained Krylov–Implicit–Runge–Kutta or JT–KIRK algorithm.

Since our original work, an independently developed approach by Langer et al. [30] has been presented which also uses GMRES inside Runge–Kutta timestepping. Their work focuses on accelerating convergence of the flow solver rather than robust convergence as required for discrete adjoints. A further difference is that their work uses a matrix-free approach, while our work computes a Jacobian which in turn permits to build strong preconditioners.

Our paper first explains in Section 2 the mathematical background of the nonlinear flow solver and the development of the JT–KIRK algorithm with emphasis on the temporal rather than spatial discretisation. Section 3 describes the discrete adjoint equation and how its time-marching relates to the one of the flow solver. Section 4 compares the ILU preconditioned GMRES and SGS solution strategies for the linear system arising from the implicit scheme and presents parameter studies on how to optimise the solver efficiency. Section 5 shows the results for four test cases, two stable cases which the B–J solver can fully converge, and two unstable cases that cannot be converged by the B–J solver, resulting in divergence of the discrete B–J adjoint. Eigenvalue analysis is shown for the second unstable testcase, Case 4, to highlight the eigenvalue clustering using the implicit algorithm. Conclusions are presented in Section 6.

2. Flow solver

2.1. Typical temporal discretisation for RANS flow solvers

The flow solver used here is an industrial compressible RANS flow solver using a vertex-centred finite volume method on unstructured grids. A multistage Runge–Kutta time integration scheme is used to time-march the solution to steady state, and geometric multigrid with semi-coarsening as well as B–J preconditioning are used to accelerate the convergence [31,32,16]. The steady state solution is reached when the residual of each control volume reaches zero, i.e.,

$$R(U) = 0,$$

where the flow variables U and residual R are both column vectors of dimension $5 \times N$ for 3-D Euler and laminar Navier–Stokes and $6 \times N$ when a one-equation turbulence model is used, on a mesh with N nodes.

A generic semi-discrete time-marching scheme for a conservation equation can be written as

$$\frac{U^{n+1} - U^n}{\sigma \Delta t} V + (1-\beta) R(U^n) + \beta R(U^{n+1}) = 0 \tag{3}$$

where Δt is the time-step, σ denotes the CFL number for the linear solver of the implicit scheme and V is the size of the control volume. The coefficient β provides a blending between explicit and implicit residuals. The residual at time level $n+1$ can be linearised using the Jacobian at time level n to be

$$R(U^{n+1}) \approx R(U^n) + \left.\frac{\partial R}{\partial U}\right|_{U=U^n} (U^{n+1} - U^n)$$

and Eq. (3) becomes

$$\left(\frac{V}{\sigma \beta \Delta t} + \left.\frac{\partial R}{\partial U}\right|_{U=U^n} \right) \Delta U^n = \frac{-1}{\beta} R(U^n). \tag{4}$$

For an implicit scheme with $\beta > 0$, the CFL number usually can be very large which is important for fast convergence to the steady state solution. Note that although β is usually chosen between 0 and 1, it is possible to allow β to be greater than 1, equivalent to under-relaxation.

2.2. Block–Jacobi solver

To simplify solving the linear system at each time step and to reduce the memory overhead, the Jacobian matrix $\frac{\partial R}{\partial U}$ on the left hand side of Eq. (4) can be approximated by the Block–Jacobian for node i independently [32]. All off-diagonal terms are then neglected, and hence the diagonal-enhancing term can be set to zero with $\sigma = \infty$. The higher-order accurate residual operator $R(U^n)$ is used on the right hand side to maintain the second order spatial accuracy of the converged steady state solution. The B–J matrix can be computed using hand differentiation [32] or using AD tools such as Tapenade [33]. The resulting B–J or point-implicit scheme is

$$\left(\beta \left. \frac{\partial R_i^{(I)}}{\partial U_i} \right|_{U=U^n} \right) \Delta U_i^n = -R_i^{(II)}(U^n), \tag{5}$$

where the Roman superscript for the residual denotes that the spatial operator is of either first-order (I) or second-order (II) accuracy, and the subscript i denotes the i-th node.

To generalise the notation, the left hand side matrix of Eq. (5) can be denoted by a generic preconditioning matrix $\mathbf{P}$, with the i-th block on the diagonal defined as

$$\mathbf{P}_i = \beta \frac{\partial R_i^{(I)}}{\partial U_i}$$

and the discrete governing equation can be simplified as

$$\mathbf{P} \Delta U^n = -R^{(II)}(U^n). \tag{6}$$

The B–J time-stepping is combined with RK to provide additional damping for high-frequency error modes which makes the smoother suitable for MG [34].

Including the derivatives of the one-equation Spalart–Allmaras turbulence model variable [35] in the Block–Jacobian increases the computational cost, but in our experience does not improve convergence rate and in some cases adversely affects the robustness. The turbulence model is hence treated as a passive scalar. The resulting structure of each block in the Block–Jacobian in 3-D is hence a full 5×5 block for the conservative variables and a diagonal entry in the $(6, 6)$ position for the Spalart–Allmaras variable. To keep the implementation simple we retain the zeros in the 6th row and column in the Block–Jacobian. The non-zero elements in the B–J matrices can be formed on the fly during the update step as they depend only on node i. Alternatively, the associated memory cost for storing the B–J matrices for all control volumes in the case of a one-equation turbulence model is 6 times the storage of the flow solution, which is typically deemed affordable.

The solution is updated using an m-stage RK scheme,

$$\begin{aligned}
U^{(0)} &= U^n, \\
U^{(1)} &= U^{(0)} - \alpha_1 \mathbf{P}^{-1} R^{(II)}(U^{(0)}), \\
&\vdots \\
U^{(m)} &= U^{(0)} - \alpha_m \mathbf{P}^{-1} R^{(II)}(U^{(m-1)}), \\
U^{n+1} &= U^{(m)}.
\end{aligned}$$

At each RK stage the preconditioner $\mathbf{P}^{-1}$ is computed by directly inverting each of the diagonal block matrices to update the flow solution. The B–J solver combined with MG can produce grid-independent convergence for inviscid flows and works reasonably well for viscous cases if semi-coarsening is used [16]. Grid-independent convergence is usually no longer observed for general 3-D RANS simulations.

2.3. Implicit solver with first-order Jacobian

To obtain a stronger coupling between neighbouring nodes and to further accelerate convergence, the first-order Jacobian can be used instead of the B–J matrix. The preconditioner $\mathbf{P}$ based on the first-order Jacobian can be written as

$$\mathbf{P}_{i,j} = \frac{V_i}{\sigma \Delta t_i} \delta_{ij} + \beta \frac{\partial R_i^{(I)}}{\partial U_j},$$

where nodes i and j either are immediate neighbours of each other or $i = j$. Each node i and each edge ij then give rise to a 6×6 block (in the case of the Spalart one equ. model) with the same structure as discussed for B–J in Section 2.2. To save on storage, the full 5×5 blocks for the Navier–Stokes variables are stored in a block-wise compressed sparse row format (BCSR) separately from the SA-variable which is stored in point-wise compressed sparse row (CSR) format. Both matrices have the same sparsity pattern which allows to share row and column index vectors. This approximation to the Jacobian implies coupling between first-order neighbours of the Navier–Stokes variables, and an implicit treatment of a passive scalar similar to B–J for the turbulence variable.

In this case each $\mathbf{P}_{i,j}$ block has the same 6×6 structure as for the B–J matrix, but is located at (i, j) of the matrix $\mathbf{P}$. It can be computed by differentiating the residual subroutine either by hand or by using Automatic Differentiation. For the presented results it was computed in vector-forward mode with the AD-tool Tapenade [33].

The restriction to first-order neighbour contributions in the Jacobian precludes a differentiation of the gradients. As typical in literature, the inviscid flux contributions are evaluated with zero gradients (first-order accuracy), which will impair convergence rate, but increase robustness. The viscous terms are evaluated here using a correction for the normal component of the Green–Gauss face gradient $\overline{\nabla U}_{ij}$ [32] to improve high frequency damping formulated as

$$\nabla U_{ij} = \overline{\nabla U}_{ij} - \left(\overline{\nabla U}_{ij} \delta s_{ij} - \frac{U_i - U_j}{|x_i - x_j|} \right) \delta s_{ij}$$

where

$$\delta s_{ij} = \frac{x_i - x_j}{|x_i - x_j|}$$

with the nodal coordinates x_i for node i. The Jacobian of the viscous flux then differentiates the viscous flux with respect to the term $U_i - U_j$ but considers $\overline{\nabla U}_{ij}$ fixed.

The cost of computing the approximate Jacobian with AD is equivalent to evaluating the nonlinear residual 5 times (or 6 times for SA turbulence), but is not substantially increased compared to evaluating the B–J matrix using AD. The cost of computing the approximate Jacobian can be further reduced by optimising the implementation of the nonlinear flux subroutine [36] and by selectively eliminating parts of the flux calculation from the AD tool using scripts or pragmas. This is currently under investigation and will be reported in our future work.

Strategies for efficiently solving the linear system for both the nonlinear flow and the corresponding adjoint solver are explained in detail in Section 4.

3. Adjoint solver

To develop a discrete adjoint solver using an existing flow solver, one could explicitly compute and store the exact Jacobian corresponding to the second-order accurate discretisation and solve the resulting linear system. However, for practical cases, both the memory and runtime for this approach could be prohibitively expensive [18]. Moreover, efficient preconditioners would need to be implemented. Alternatively, one can solve the adjoint equations using the same time-marching scheme as the nonlinear flow solver [37,38,5,39]. The discrete adjoint equation [1] uses the transposed exact Jacobian matrix of the nonlinear flow equation

$$\mathbf{L}^T v = g$$

with

$$\mathbf{L} = \frac{\partial R}{\partial U}^T \quad \text{and} \quad g = \frac{\partial J}{\partial U}^T,$$

where v is the adjoint variable and J is the objective function, which is a function of both the mesh and the flow field. The residual of the adjoint equations

$$R_v(v) = \mathbf{L}^T v - g$$

can computed by application of AD tools [33,39] and then be used to update the adjoint solution,

$$\mathbf{P}^T \delta v^n = -R_v(v^n), \tag{7}$$

where the preconditioner $\mathbf{P}^T$ is the transpose of the preconditioner for the flow solver. Assume $\tilde{v}$ is the exact solution of the adjoint equation, then for an error $e^{n+1} = v^{n+1} - \tilde{v}$ at the $n+1$-th iteration, we have

$$e^{n+1} = (\mathbf{I} - \mathbf{P}^{-T}\mathbf{L}^T)e^n.$$

The asymptotic convergence rate of the flow and the adjoint are guaranteed to be the same [40] since

$$\rho(\mathbf{I} - \mathbf{P}^{-1}\mathbf{L}) = \rho(\mathbf{I} - \mathbf{P}^{-T}\mathbf{L}^T),$$

where the operator $\rho(\cdot)$ means the spectral radius of a matrix, which determines the asymptotic convergence rate of a FPI. The conclusion on the identity of the asymptotic convergence rates of flow and adjoint can be straightforwardly extended to include Runge–Kutta multistage integration as well as multigrid. If all operators contributing to $\mathbf{P}$ are exactly transposed for $\mathbf{P}^T$ in (7), not only is full convergence of the adjoint solver guaranteed as long as the nonlinear flow solver asymptotically converges, but also the transient of the sensitivity is exactly the same when comparing tangent-linear and adjoint approaches [40], which is a very powerful validation tool. This is the approach taken for the results presented in this paper. However to achieve stability typically this condition can be relaxed, with only transposing non-symmetric parts of $\mathbf{P}$ such as low-Mach preconditioners and not reversing the sequence of the operators [39].

4. Solving the linear system

For both the flow and adjoint equations, a linear system needs to be solved at each RK stage on each level of MG. Two approaches to solve this linear equation are considered here, Symmetric Gauss–Seidel (SGS) and ILU(0)-preconditioned GMRES.

4.1. SGS as linear solver

A Symmetric Gauss–Seidel (SGS) iteration is a forward sweep followed by a backward sweep of Gauss–Seidel (GS). The system matrix $\mathbf{A}$ is decomposed as $\mathbf{A} = \mathbf{D} + \mathbf{L} + \mathbf{U}$, where $\mathbf{D}$, $\mathbf{L}$ and $\mathbf{U}$ are the block-wise diagonal, lower and upper triangular matrices. The forward and backward SGS sweeps then compute:

$$\begin{aligned} (\mathbf{D}+\mathbf{L})x^* &= -\mathbf{U}x^n - R \\ (\mathbf{D}+\mathbf{U})x^{n+1} &= -\mathbf{L}x^* - R. \end{aligned}$$

Since $\mathbf{D} + \mathbf{L}$ and $\mathbf{D} + \mathbf{U}$ are both block-wise triangular matrices, both equations can be solved with block-wise backward/forward substitutions. In order for the SGS to converge, diagonal dominance is required which can be achieved by decreasing the CFL number σ from infinity to a smaller finite value.

It is not necessary to fully converge the linear system at each RK stage as the system matrix is only a first-order accurate approximation. In practice, 3 sweeps of SGS are used for each linear system solve, hence the label SGS(3). Using more than 3 iterations of SGS will increase the overall runtime which is proportional to the number of SGS sweeps, but does not further improve convergence rate or stability as reported by Swanson et al. [41] and also independently verified in our numerical experiment.

The presented results with SGS use the standard implementation as described here. Other implementation choices may produce slightly different figures, however the presented SGS results by Swanson et al. [41], as well as Langer et al. [15,30] all demonstrate a similar CFL number limitation as observed with our implementation.

4.2. ILU-preconditioned GMRES as linear solver

The proposed JT–KIRK scheme uses GMRES [9] to solve the linear system for its proven robustness for non-symmetric systems. GMRES stores m Krylov vectors and hence requires additional memory equivalent to that of m nonlinear flow solutions. The memory requirement becomes prohibitive for large m, hence in practice, a limited m is used. To then avoid stalling convergence one uses restarted GMRES: whenever the number of basis vectors reaches m, they are discarded and GMRES is restarted using the partly converged solution. For all the cases used in this paper, the stopping criterion for the GMRES solver is either using three GMRES vectors with zero restart or a one order of magnitude drop of the relative residual of the linear system, whichever criteria is met first, denoted by GMRES$(1, 3, 0.1)$.

GMRES still needs an appropriate preconditioner for good convergence, most widely used are Jacobi, Lower–Upper–Symmetric Gauss–Seidel (LU–SGS) and Incomplete LU factorisation (ILU) of the approximate Jacobian [9]. The first two are easy to calculate but our results show that ILU(0), i.e. ILU with 0 level of fill-in, is more robust and efficient. ILU(0) is thus used for all the results in this paper and for simplicity will be denoted as ILU hereafter. The preconditioned linear system (1) then becomes

$$\mathbf{U}^{-1}\mathbf{L}^{-1}\mathbf{P}\Delta U^n = -\mathbf{U}^{-1}\mathbf{L}^{-1}R, \tag{8}$$

where the matrices $\mathbf{U}$ and $\mathbf{L}$ are from the ILU factorisation.

$$\mathbf{P} \approx \mathbf{L} \cdot \mathbf{U}. \tag{9}$$

GMRES uses the linear combination of orthogonal basis vectors of the Krylov subspace to approximate the exact solution. The Krylov subspace is constructed for the system matrix $\mathbf{A}$ of a linear system $\mathbf{A}x = b$ as follows,

$$\mathbf{K}_m(\mathbf{A}, b) = \text{span}\{b, \mathbf{A}b, \mathbf{A}^2 b, \mathbf{A}^3 b, \ldots, \mathbf{A}^{m-1} b\}, \tag{10}$$

where $\mathbf{A} = \mathbf{U}^{-1}\mathbf{L}^{-1}\mathbf{P}$ and $b = -\mathbf{U}^{-1}\mathbf{L}^{-1}R$. In (8), both the RHS and LHS are computed via several sweeps of matrix–vector multiplication and no matrix–matrix multiplication is done even though all the matrices are stored.

In the JT–KIRK scheme, the first-order approximate Jacobian and its ILU factorisation are computed only at the first RK stage at each nonlinear iteration of the flow solver. In the adjoint JT–KIRK solver, the approximate Jacobian and its ILU are constant, and hence only computed once at the first iteration for all grid levels. In the parallel implementation, ILU is performed for each sub-domain, and due to this decoupled implementation, the convergence deteriorates with increased number of partitions, as explained in Section 5.1.4.

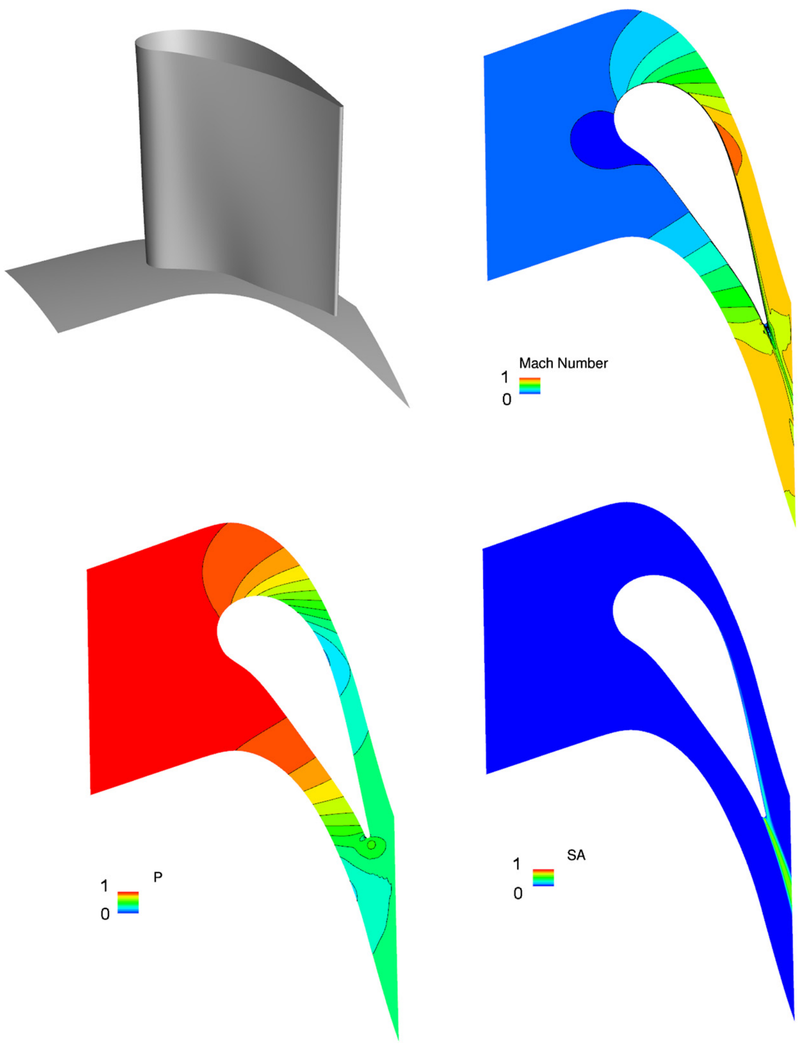

Fig. 2. Case 1, Nozzle Guide Vane (NGV). Upper left: NGV geometry; upper right: Mach number contours at midspan; lower left: static pressure contours at midspan; lower right: Spalart–Allmaras turbulence variable contours at midspan. All legends are non-dimensionalised using their corresponding minimum and maximum values.

5. Results

In this section, the proposed JT–KIRK flow solver and its discrete adjoint are applied to four different test cases with different flow features to assess both the solver efficiency and robustness improvement. The four test cases from turbomachinery applications are first briefly explained in terms their geometries, meshes and the flow characteristics, followed by a detailed analysis of the solver performance on each case.

The Block–Jacobi (B–J) discretisation manages to converge cases 1 and 2 fully, hence labelled as 'stable cases'. These cases demonstrate the solver efficiency improvement of the JT–KIRK algorithm. Cases 3 and 4 can only be converged by B–J to LCO with the adjoint diverging, hence these are labelled as 'unstable cases'. Case 3 exhibits a mild instability with insignificant effect on the objective function, an example of a case where a steady-state stabilisation is entirely justified. Case 4 is an artificially created case with severe instability where objective functions obtained by steady-state simulation will markedly deviate from the full unsteady ones. The unstable cases demonstrate the stabilisation of the adjoint solver when JT–KIRK is used to converge fully both the flow and adjoint solutions.

5.1. Case 1, nozzle guide vane (NGV)

Case 1 is a nozzle guide vane (NGV) with subsonic inlet and outlet. The domain is meshed with 0.5 million hexahedral elements. The B–J solver converges fully with the residual dropping by 12 orders of magnitude. The NGV geometry and various contour plots (Mach number, static pressure and SA variable) are shown in Fig. 2 for the surface at midspan. The flow is fully attached to the blade surface.

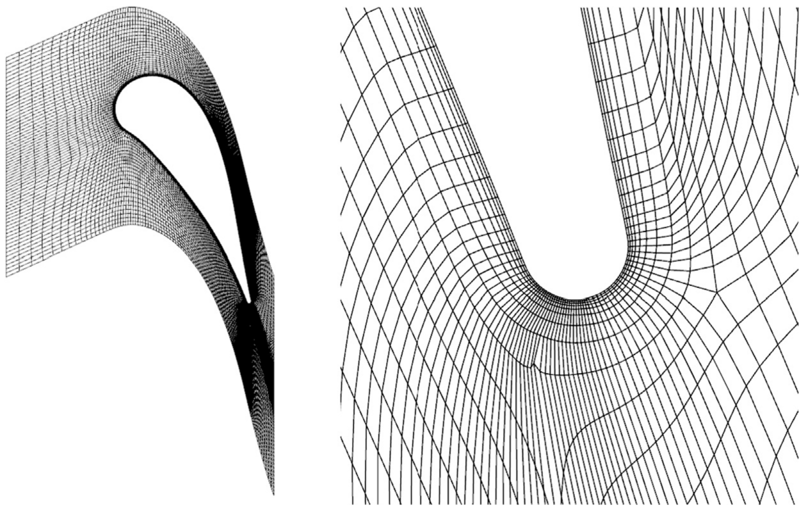

Fig. 3. Case 1, NGV. Left: mesh for the whole domain, right: close-up view at the trailing edge. The quasi-2-D mesh is taken at midspan.

Table 1
Multigrid mesh statistics for Case 1.

Level	# of nodes	Node ratio	# of edges	Edge ratio
1	499 150		1 471 161	
2	276 565	1.80	894 771	1.64
3	162 065	1.71	565 082	1.58
4	126 743	1.28	451 674	1.25
5	115 514	1.10	414 550	1.09

Table 2
Summary of different solvers.

Solver	Preconditioner	Linear equation solve
B–J	Block–Jacobian	Block-wise direct inversion
SGS	First-order Jacobian	SGS(3)
JT–KIRK	First-order Jacobian	ILU(0) + GMRES(1, 3, 0.1)

As typical for this type of RANS computation, the mesh around the trailing edge shown in Fig. 3 is chosen coarse enough not to resolve the vortex shedding, but fine enough to limit the truncation error. The maximum cell aspect ratio for the boundary layer cells is 170 and y^+ for the first node off the viscous wall is around unity.

Geometric multigrid is used to accelerate the convergence. The hierarchy of multigrid meshes has been generated using an element-collapsing algorithm with semi-coarsening [16], with a maximum allowable isotropic coarsening factor of 2 per dimension and an element aspect-ratio threshold of 3 for applying semi-coarsening. This set of parameters has been optimised to provide the best convergence rate with the B–J scheme, which has inferior smoothing characteristics compared to the implicit schemes. The sizes of the mesh levels and their effective coarsening ratios are listed in Table 1. It can be seen that the overall coarsening ratio remains below 2 and gradually deteriorates with coarser meshes.

5.1.1. Parameter study

Both the original B–J solver and the implicit SGS and JT–KIRK solvers are applied to Case 1 to explore the effect of the parameter choices on the stability and efficiency. The three solvers are labelled B–J, SGS and JT–KIRK in the following, their adjoint variants use exactly the same time-stepping algorithm as the flow solver. The nomenclature used for all the solvers is explained in Table 2.

5.1.2. Number of RK stages

Numerical experiments were performed for the B–J and JT–KIRK schemes to find the optimal number of Runge–Kutta (RK) stages. The coefficients for each stage are used as optimised for viscous flow [42] and also proposed for B–J [32], both authors advocating the use of RK5. The linear solver settings for both B–J and JT–KIRK are based on the optimal parameters found in Section 5.1.3. Convergence results for a drop in residual of 12 orders of magnitude are shown in Fig. 4.

In all our numerical experiments RK5 outperforms RK with fewer stages in terms of iterations. B–J shows an optimum in runtime for RK4, with a slight but insignificant increase for RK5. The optimum between RK4 and RK5 for B–J is as expected, since the coefficients used have been optimised to achieve optimal high-frequency damping. For JT–KIRK, smoothing

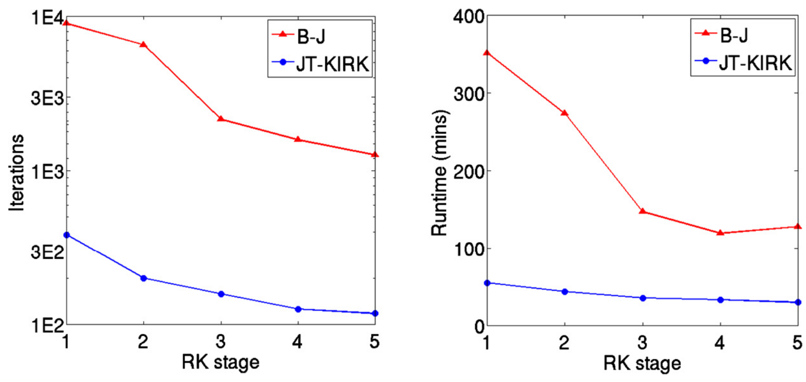

Fig. 4. Effect of RK stage number on iteration number (left) and runtime (right) for full convergence of Case 1.

Table 3
Case 1: parameter study for β, σ and multigrid level.

B–J					SGS					JT–KIRK				
$1/\beta$	σ	MG level	# of iter.	Time (min)	$1/\beta$	σ	MG level	# of iter.	Time (min)	$1/\beta$	σ	MG level	# of iter.	Time (min)
1.00	∞	5	1927	158	1.00	1k	5	212	94	1.00	∞	5	183	98
1.11	∞	5	1789	142	1.11	∞	5	197	88	1.11	∞	5	170	91
1.25	∞	5	1648	135	1.25	1k	5	182	79	1.25	1k	5	167	89
1.43	∞	5	1503	122	1.25	1k	4	182	65	1.43	∞	5	143	76
1.67	∞	5	1369	109	1.25	1k	3	184	50	1.67	∞	5	129	69
1.67	∞	4	1362	**109**	1.25	1k	2	228	**38**	1.67	∞	4	192	57
1.67	∞	1–3	>10k	>1k	1.25	1k	1	735	46	1.67	∞	3	129	44
2.00	∞	5	1580	128	1.43	10	5	2262	977	1.67	∞	2	147	**30**
										1.67	∞	1	443	34
										2.00	1k	5	144	79
										2.50	30	5	1132	584

also improves as shown by the decrease in number of iterations needed. In this case runtime further decreases with the number of RK stages since the most expensive part of the non-linear step is the computation of the Jacobian and its ILU decomposition, which is only performed once for all RK stages.

A further improvement might be achieved for the JT–KIRK scheme by adding more RK stages, but as Fig. 4 suggests, this gain will be slight. A more significant effect may be the optimisation of the stage coefficients, which however is difficult to analyse due to the nonlinear nature of the GMRES solver. As a base for comparison all results in the paper are produced using a 5-stage Runge–Kutta multi-stage scheme with standard coefficients.

5.1.3. *Number of multigrid levels, β, and σ*

The two parameters in Eq. (4), β and σ, are important for stability and efficiency. To determine best values we perform a parameter study for all three solvers on Case 1.

Table 3 varies β from 0 to 1 for each scheme. All the data corresponds to converging Case 1 fully with a residual drop of 12 orders of magnitude, cases are run on an HPC cluster with 24 cores. For each tested value of β, initially using 5 levels of MG, σ is varied as $10 \le \sigma \le \infty$. Table 3 lists the σ for each β that achieves the lowest runtime. For the best-performing 5-level MG version of each of the three schemes, we then decrease the levels of MG (data highlighted in grey) to determine the optimal number of levels. Best overall performance is reported in bold font. The values β and σ have then been varied for these optimal combinations (results not reported here) to confirm that the combination of all three parameters is indeed optimal.

In summary, we find that

- For B–J, the best run-time performance is achieved with $\beta = 0.6$, and 4 or 5 levels of MG. Reducing the MG levels below 4 prevents convergence.
- For SGS, $\beta = 0.8$ with $\sigma = 1000$ and 2 levels of MG is most efficient.
- For JT–KIRK, $\beta = 0.6$ with $\sigma = \infty$ and 2 levels of MG is most efficient.
- For both SGS and JT–KIRK, the smallest acceptable β is around 0.4–0.7, below which the scheme quickly becomes either too slow or unstable. A similar conclusion has been reported in [26] and [43].
- Overall, the best runtime is achieved with the JT–KIRK solver, which is 20% more efficient than SGS and 70% more efficient than B–J.

Table 4
Case 1: runtime of different adjoint solvers, 12 cores.

Solver	Parameter setting			Runtime (min)			
	MG lvl	β	σ	Flow	Norm.	Adjoint	Norm.
B–J	5	0.6	1.67	109	1.00	355	1.00
SGS	2	0.8	1000	38	0.35	31	0.09
JT–KIRK	2	0.6	∞	30	0.28	22	0.06

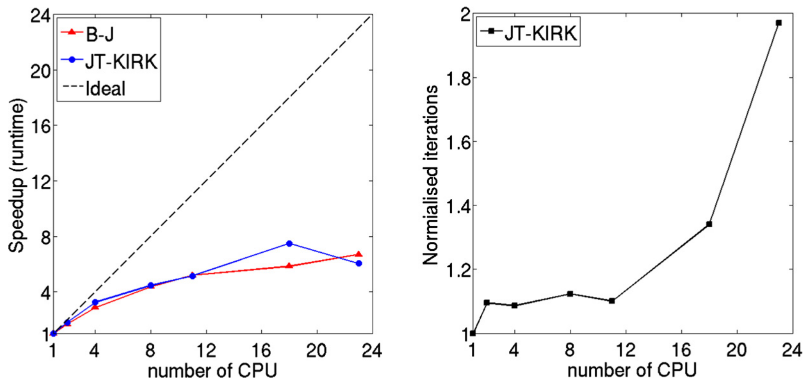

Fig. 5. Scalability of Block–Jacobi (B–J) and JT–KIRK solvers on test Case 1 (left) and the iteration number with respect to the number of processors for JT–KIRK solver showing the degeneration of ILU+GMRES per subdomain (right).

Both implicit smoothers are shown to work best in a 2-level multigrid when using the B–J-optimised coarsening parameters (cf. Table 1). The implicit solvers offer much better smoothing than the point-implicit B–J, which should allow more aggressive coarsening. In particular the ILU preconditioner, although very effective, is expensive to compute. This cost does not reduce adequately on the coarser levels which are quite dense due to the need for semi-coarsening and the requirement of adequate mesh quality.

The B–J scheme requires semi-coarsened grids. An alternative method to alleviate the stiffness that arises from the boundary layer mesh is to construct linelets and apply a line-implicit smoother to couple the nodes in the direction normal to the wall [44]. The implicit scheme using a first-order Jacobian, either SGS or JT–KIRK, may provide sufficient coupling in the boundary layer to either allow the threshold for semi-coarsening to be raised significantly, or to even allow isotropic coarsening, which will increase the coarsening ration significantly. As a basis for comparison, the B–J coarsening parameters are used in all cases.

In the steady-state discrete adjoint approach the computation of the approximate Jacobian matrix needs to be performed only once for each MG level throughout the entire adjoint run. Thus the ratio of runtime of both SGS and JT–KIRK compared to B–J will be even lower for the adjoint solver than for the flow solver. We use the optimal flow solver parameters of Table 3 also for each adjoint solver. The runtime of the flow and adjoint are shown in Table 4, runtimes are presented for a residual convergence of 12 orders of magnitude using 24 cores, same as for the flow calculation. For B–J, the adjoint solver requires around 3 times the runtime of the flow solver, which is typical for a discrete adjoint solver using source-transformation AD. As expected, the SGS and JT–KIRK adjoint solvers require less runtime than their respective flow solver. For JT–KIRK, the adjoint needs only 10% of the runtime compared to B–J.

5.1.4. Parallelism

The baseline Block–Jacobi solver is straightforward to parallelise as communication is only needed for the residual calculation of the nodes shared by multiple partitions. For implicit solvers, the communication due to solving the resulting linear system using SGS or GMRES is also not substantial as both SGS and GMRES only need the matrix–vector products which can be computed for each sub-domain and assembled in a reduction step. To avoid this reduction step, we choose to compute the Jacobian and perform the SGS or GMRES for each partition. For JT–KIRK, the ILU preconditioner is also computed for each partition separately.

Fig. 5 shows how this simplification affects the convergence rate on Case 1 with 500k nodes. The results are produced with the optimal settings for each solver, i.e. 5 MG levels for B–J, 2-level MG for JT–KIRK. The number of iterations is barely affected by the partition size for JT–KIRK using up to 12 cores, or around 40k nodes per partition, but convergence does degenerate beyond 12 cores. While a global ILU factorisation would be prohibitively expensive to compute, using a global GMRES solve is possible and is likely to reduce the minimal partition size needed for effective convergence. Nevertheless, strong scaling in iteration numbers down to a partition size of 40k nodes will be acceptable for typical industrial applications.

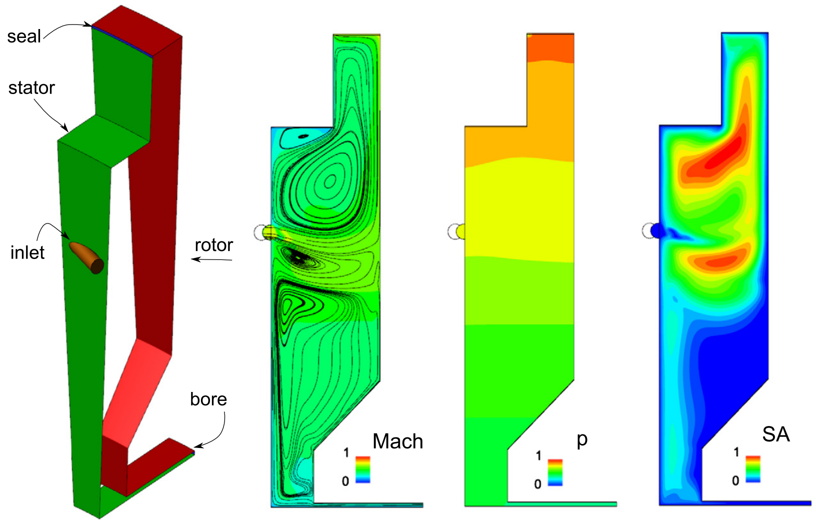

Fig. 6. Case 2, cavity. From left to right: (1) cavity geometry, (2) Mach contours with 2-D streamline, (3) static pressure contours and (4) SA variable contours. Contour plots are taken at the medium azimuthal angle. All legends are non-dimensionalised using their corresponding minimum and maximum values.

Table 5
Case 2: runtime of different adjoint solvers, 12 cores.

Solver	Parameter setting			Runtime (s)			
	MG levels	β	CFL	Flow	Normalised	Adjoint	Normalised
B–J	5	0.6	1.67	49 512	1.00	163 389	1.00
JT–KIRK	2	0.6	2000	22 330	0.45	19 933	0.12

5.2. Case 2: NGV/rotor cavity

Case 2 is a cavity between the Nozzle Guide Vane (NGV) and the rotor disks. The geometry with labelling for the various boundaries is shown in the leftmost panel of Fig. 6. The 'stator' boundary is attached to the NGV and thus stationary while the 'rotor' boundary is attached to the shaft and in spinning with the same speed as the rotor. A subsonic outflow boundary condition is applied to both 'seal' and 'bore' boundaries while a subsonic inflow boundary condition is applied to the inlet. The mesh has approximately 1 million hexahedral cells and is refined near the viscous wall with the maximum cell aspect ratio around 90 and a y^+ of the first interior node around unity.

Different from Case 1 where the main flow is driven by the axial flow, the flow in this case is dominated by swirling flow resulting from the wall rotation. In addition, the majority of the flow field is low speed, a further challenge to convergence.

The baseline B–J solver, although converging very slowly, does fully converge to steady state, as shown in Fig. 6. The streamline plot superimposed on the Mach number contour plot illustrates the complex flow pattern due to the strong shearing.

5.2.1. Convergence speedup

The purpose of presenting Case 2 is to assess the performance improvement of the proposed JT–KIRK algorithm for cases where viscous dissipation dominates and a stronger coupling is expected to have a significant impact on the convergence.

Using the same parameters as optimised for Case 1, JT–KIRK flow and adjoint solvers are run for Case 2, except that the CFL number has to be lowered to 2000 for this case for startup. To permit a fair comparison, possible improvements with more advanced initialisation or ramping techniques are not investigated here and the results for a constant CFL number for the entire run are reported in Table 5. Same as for Case 1, the residual is reduced by 12 orders of magnitude. All the computations for Case 2 are run on 12 processors.

The runtime for Case 2 is significantly longer than Case 1, which is due to the shearing-dominating nature of this type of flows. The JT–KIRK solvers speed up the flow computation by 50% and the adjoint by 88%.

5.3. Case 3: turbine stage with rotor tip gap

Having demonstrated the efficiency gains of the JT–KIRK scheme for stable flows, Case 3 presents a mild instability that has only small influence on the objective function.

Case 3 is a one-stage turbine, the rotor has a tip gap of constant clearance along the axial direction. The geometry of the stage is shown in the upper left of Fig. 7, periodic boundaries and the casing surfaces for both the NGV and the rotor

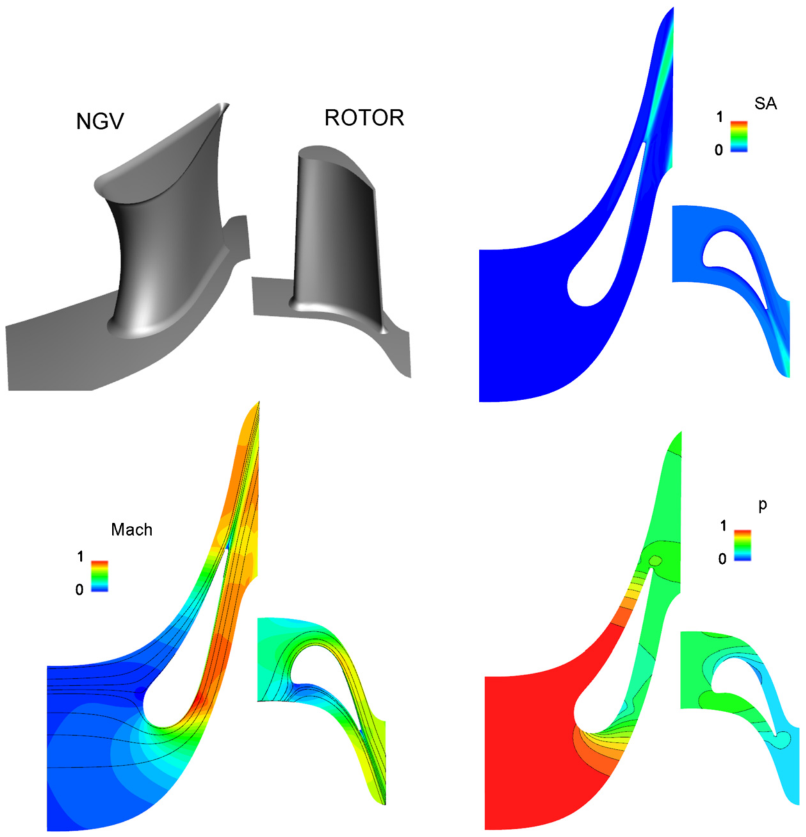

Fig. 7. Case 3, NGV + rotor with tip gap. Upper left: NGV and rotor geometry; upper right: SA variable contour; lower left: Mach contour with 2-D streamline; lower right: static pressure contour. The contour plots are taken at midspan. All legends are non-dimensionalised using their corresponding minima and maxima.

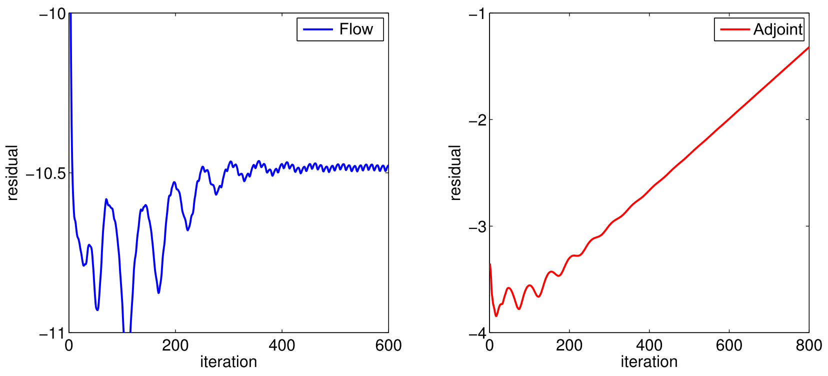

Fig. 8. Case 3: convergence history of both Block–Jacobi flow and adjoint solvers.

are not shown for better illustration. Total pressure and temperature are prescribed at the NGV inlet and subsonic outflow is applied at the rotor exit with prescribed static pressure. A mixing plane is used to model the NGV and rotor interaction for steady state flow. The case is meshed with 2.4 million hexahedral cells with a maximum cell aspect ratio of 230. The tip gap is discretised by 25 layers of cells with a growth ratio around 1.3.

The B–J flow solver is run first with a typical setting of $\beta = 1$ and $\sigma = +\infty$ and then with alternative settings with reduced σ and increased β, but the B–J solver is not able to fully converge with any setting, at best it converges to LCO, at worst diverges. Shown in Fig. 8 is the convergence history for both the flow and adjoint with the typical setting $\beta = 1$ and $\sigma = +\infty$.

Using any snapshot of the LCO-converged flow solution will lead to divergence of the discrete B–J adjoint solver shown on the right of Fig. 8.

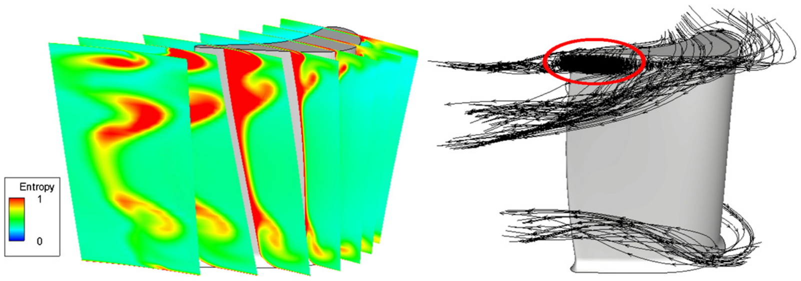

Fig. 9. Case 3, flow visualisation around the rotor. Left: entropy contour plot at various axial locations; right: streamline illustrating the tip vortex and the two passage vortices.

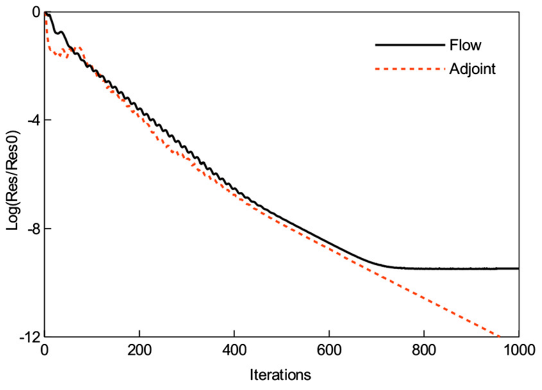

Fig. 10. Case 3: convergence history of flow and adjoint solvers using JT–KIRK algorithm.

Table 6
Case 3: deviation of objective functions of both fully converged steady solution using JT–KIRK and LCO-converged solution using Block–Jacobi, from the objective function of the time-averaged unsteady flow solution.

Quantity	JT–KIRK error %	B–J error %
Efficiency	+0.00246	−0.01208
Capacity	+0.00352	+0.00092
Reaction	−0.00385	−0.03651

To investigate the flow physics, Fig. 7 shows various contour plots at midspan for an arbitrary snapshot within the LCO. The streamline plot in Fig. 9 shows the flow to be mainly attached, except near the tip where the flow around the rotor reveals the complex vortex structure due to the tip gap. Although not shown here, it is confirmed that the LCO convergence corresponds to a small-scale oscillation of the stagnation zone near the rotor tip on the suction side towards the trailing edge (highlighted with the ellipse in Fig. 9).

5.3.1. Stabilisation with the JT–KIRK scheme

Using JT–KIRK the flow solver fully converges to steady state with a residual reduction of 10 orders of magnitude with $\beta = 1$ and $\sigma = 1000$ on 12 processors. The fully converged flow solution is then linearised for the adjoint which also fully converges using the JT–KIRK scheme with the same parameters. The convergence history of both the flow and adjoint are shown in Fig. 10

In the presence of physical unsteadiness both the B–J and JT–KIRK results may be questioned. While B–J does demonstrate some unsteady behaviour, it is not a time-accurate scheme and the LCO are likely to be governed by the numerical scheme rather than the flow physics. On the other hand, JT–KIRK adds a large artificial viscosity first-order in time to stabilise the discretisation. If the flow does contain unstable modes that have a nonlinear effect, this may result in a seemingly steady solution that is not an accurate reflection of the time-averaged solution [11].

To assess the accuracy of both the B–J and JT–KIRK results, an unsteady analysis is performed for this case using dual time-stepping with a 2-step backward differentiation formula (BDF2). The Block–Jacobi solver is used for fully converging the inner system at each physical time step. Convergence studies on both the convergence level of the inner system and time step size of the outer system have been performed to ensure that the unsteady flow simulation has converged for this particular mesh. For this comparison the focus is on resolving the unsteadiness of the blade flow and comparing that to a fully steady treatment. The mixing-plane treatment between rotor and stator has hence been maintained as in the steady case.

Table 7
Case 3: effect of flow convergence level and adjoint scheme on the adjoint convergence. The solver setting for all the JT–KIRK flow and adjoint runs are the same.

Set	Flow solver	Flow convergence	Adj. solver	Adj. convergence
A	JT–KIRK	Full convergence	JT–KIRK	Full convergence
B	JT–KIRK	Full convergence	B–J	Divergence
C	B–J	LCO	JT–KIRK	Full convergence
D	B–J	LCO	B–J	Divergence

Table 6 shows the errors of the objective functions for the flow solutions using the B–J and JT–KIRK solvers relative to the time-averaged objective functions of the unsteady flow simulation. The following objective functions are considered:

$$\text{Efficiency:}\qquad \eta = \frac{(\dot m_1 h_{T,1} - \dot m_3 h_{T,3})}{(\dot m_1 h_{T,1} - \dot m_3 h_{isen,T,3})},$$

$$\text{Capacity:}\qquad \phi = \frac{\dot m_1 \sqrt{T_{T,1}}}{p_{T,1}},$$

$$\text{Reaction:}\qquad \chi = \frac{h_2 - h_3}{h_{T,1} - h_{T,3}},$$

with mass flow rate $\dot m$, entropy h, temperature T and pressure p. The subscripts refer as 1 to the NGV inlet, 2 to the rotor inlet, 3 to the rotor outlet, T to the total quantity, '*isen*' to the isentropic state.

The relative deviations for all the three objective functions are very small (all below 0.05%), justifying in this case the use of the JT–KIRK solver to suppress the instability and converge the flow to steady state. However, the deviations in sensitivities may be more significant than for the values of the objective functions.

5.3.2. *Effect of flow convergence level on adjoint stability*

In the stabilisation results shown in the previous subsection the JT–KIRK adjoint solver was applied to the fully converged solution while the B–J adjoint solver was applied to the LCO-converged flow. The stabilisation could be due to the more contractive Jacobian of JT–KIRK or the better convergence level of the flow that the Jacobian is based on. To decouple the effect of flow convergence level and adjoint time-stepping algorithm, two additional runs are performed: (i) application of the JT–KIRK adjoint solver to LCO-converged flow and (ii) application of the B–J adjoint solver to fully-converged flow. The results are summarised in Table 7.

The B–J adjoint based on the fully converged flow diverges, while on the other hand, for this flow JT–KIRK is able to converge the adjoint solution based on the LCO-converged flow solution. The latter is not guaranteed as the preconditioned Jacobian $\mathbf{A} := \mathbf{I} - \mathbf{P}^{-T}(U)\mathbf{L}^T(U)$ depends on the nonlinear flow solution U about which the adjoint is linearised. The fact that the JT–KIRK adjoint is also stable for the LCO-converged flow in this case can probably be attributed to the fact the flow deviation is small, implied by the small deviation of the objective functions shown in Table 6. It will be shown in Section 5.4 that this is not hold for cases with stronger unsteadiness.

5.4. *Case 4: turbine stage with skewed rotor*

Case 4 is a one-stage turbine with a NGV and a rotor. Different from Case 3, the rotor does not have a tip gap (geometry shown in Fig. 11 upper left). Instead, in order to investigate the effect of large unsteadiness on the convergence of both the flow and adjoint solver, the angle of attack of the rotor blade has been deliberately changed significantly away from the design condition to create a massive flow separation on the rotor suction side. The case consists of 0.6 million hexahedral cells with maximal cell aspect ratio of 75.

Similar to Case 3, the baseline B–J solver only converges to LCO, stalling after an initial drop of the residual of 2 orders of magnitude. The convergence curve in Fig. 12 is presented for $\beta = 0.6$ and $\sigma = \infty$. The discrete adjoint based on the flow solution at the 400-th iteration diverges after around 100 adjoint iterations. Using a different snapshot of the flow solution, a smaller σ and/or different β does not achieve convergence, the lack of convergence is due to outlying eigenvalues as demonstrated by the eigen-analysis in Section 5.4.2.

An arbitrary snapshot of the flow solution taken during the LCO is shown in Fig. 11. The 2-D streamlines around the rotor at midspan clearly show the separation zone with a separation point close to the leading edge (marked with arrow in lower left of Fig. 11).

5.4.1. *Stabilisation using JT–KIRK solvers*

Case 4 has a much stronger instability, hence for this case we assess the robustness of the stabilisation effect of the JT–KIRK scheme. Differently from Case 3, the strong instability may result in significant differences of objective functions between an unsteady simulation and the steady-state B–J or JT–KIRK approaches.

When applying JT–KIRK to Case 1, fastest convergence was obtained with $\beta = 0.6$. This choice is actually not stable for Case 4 when initialising with freestream flow and keeping σ and β constant for all iterations. A more accurate solve of the

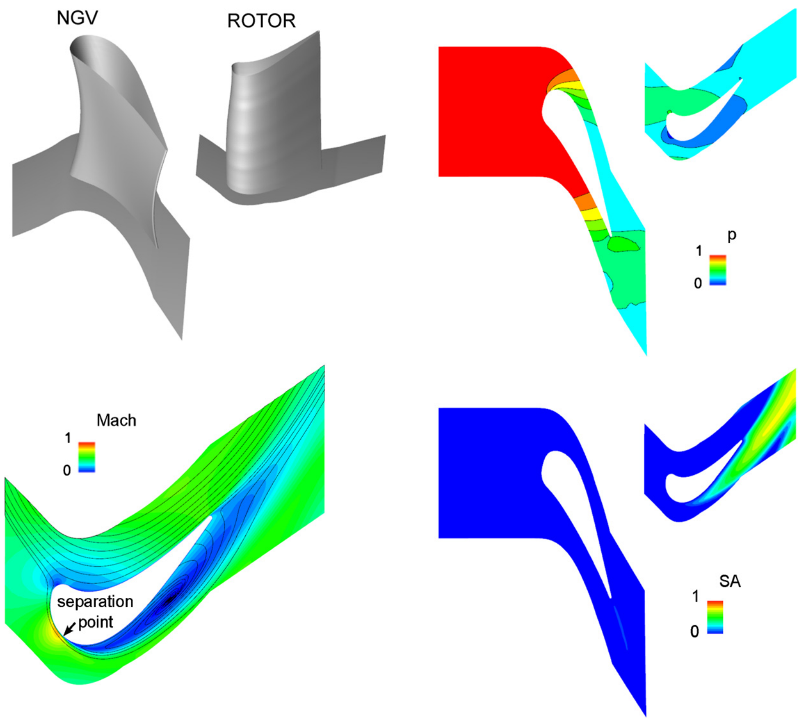

Fig. 11. Case 4, NGV and skewed rotor. Upper left: NGV and rotor geometry; upper right: static pressure contour; lower left: Mach contour with 2-D streamline for the rotor only; lower right: SA variable contour. The contour plots are taken at midspan. All legends are non-dimensionalised using their corresponding minima and maxima.

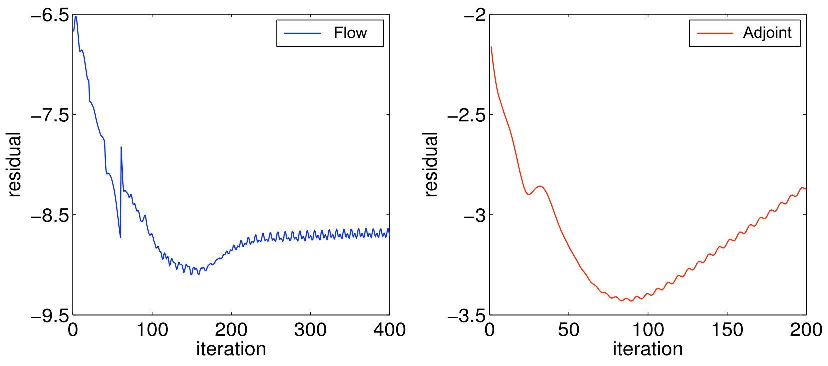

Fig. 12. Case 4, LCO convergence of the flow solver and the divergence of the adjoint solver.

Table 8
Case 4: Convergence of B–J and JT–KIRK with various solver settings, (DNC = Did Not Converge). The B–J solver result highlighted in bold is shown in Fig. 11. Highlighted B–J and JT–KIRK results are compared with the unsteady results in Table 9.

Solver	β	σ	MG level	Iteration	Runtime (min)
Flow (B–J)	0.6	$+\infty$	4	DNC	DNC
Adjoint (B–J)	0.6	$+\infty$	4	DNC	DNC
Flow (JT–KIRK)	0.6	$+\infty$	4	DNC	DNC
Flow (JT–KIRK)	1.0	$+\infty$	4	DNC	DNC
Flow (JT–KIRK)	1.5	$+\infty$	4	DNC	DNC
Flow (JT–KIRK)	2.0	$+\infty$	4	1042	1750
Flow (JT–KIRK)	2.0	$+\infty$	2	1058	801
Adjoint (JT–KIRK)	2.0	$+\infty$	2	1096	507

linear system using more Krylov vectors or a lower convergence threshold do not help to achieve full convergence, however increasing the under-relaxation to $\beta = 2.0$ does, while the CFL number can remain at $\sigma = \infty$. Optimal runtime for this combination is achieved with 2 levels of MG, solver settings and convergence results are shown in Table 8 and Fig. 13.

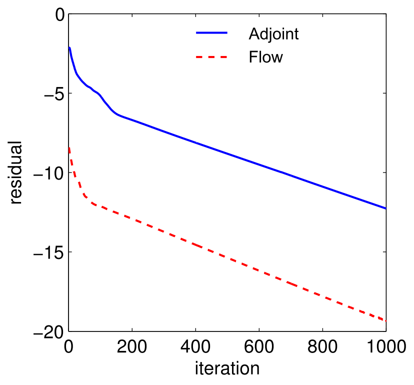

Fig. 13. Case 4: convergence history of both JT–KIRK flow and adjoint solvers.

Table 9
Case 4: relative flow output deviation between fully and LCO-converged flow solution.

Quantity	JT–KIRK error %	B–J error %
Efficiency	−0.03124	+0.10064
Capacity	+0.00052	−0.00035
Reaction	−17.24848	+23.12120

Similar to Case 3, a time-accurate unsteady simulation is performed to assess the accuracy of the values of the objective functions of the LCO-converged flow solution from B–J and the fully converged flow solution from JT–KIRK. Convergence studies have been performed for the unsteady solver to ensure that the prescribed level of convergence for solving the inner system is low enough and that the time step is chosen small enough. The errors of the objective functions from LCO-converged B–J and fully-converged JT–KIRK relative to the unsteady simulation are summarised in Table 9. The deviations of both efficiency and capacity of the steady-state approaches are small (below 0.1%), but the reaction ratios of both the B–J and the JT–KIRK solutions differ significantly from the unsteady average, with the deviation having comparable magnitude for both schemes at around 20%, but with opposite signs.

On one hand, the stabilisation effect of the JT–KIRK solver is very strong thus managing to fully converge a case with such strong unsteadiness. Such strong flow separations or other severe instabilities may occur for initial or intermediate designs, and it is important that the sequence of design iterations is not broken by a diverging adjoint. Reduced accuracy of objective functions and sensitivities may be perfectly acceptable in this phase.

On the other hand, it implies that extra care needs to be taken when such a strong stabilisation is used. The user needs to carefully assess when objective functions obtained from steady approaches are acceptable, and when unsteady simulations are required. Providing guidance on this goes beyond the scope of this paper, however two observations can be made. The comparison of cases 3 and 4 suggests that if the JT–KIRK adjoint is able to converge based on the flow solution converged to LCO by the B–J flow solver, this may indicate small unsteady deviations resulting in negligible changes in objective function. However, this is a rather arbitrary threshold and the second observation is that the deviation of the objective function is strongly linked to its non-linearity: two of the objectives in Case 4 only show small errors, while reaction ratio does strongly deviate (Table 9).

5.4.2. *Eigen-analysis of the stabilisation mechanism*

To better understand the effect of the JT–KIRK scheme on the convergence of both the flow and the adjoint solvers, eigen-analysis is performed on the Jacobian of both the B–J and JT–KIRK adjoint solvers. For the B–J solver, we take as before the limit-cycle flow solution at the 400-th iteration of the B–J solver. For JT–KIRK, we take the fully converged flow solution.

To calculate the dominant eigenvalues we use Arnoldi iterations as described by Campobasso et al. [6]. Strictly speaking, the Arnoldi iteration can only be applied to a linear operator, such as linear multigrid smoothers. The smoother used in the JT–KIRK algorithm has a GMRES solver wrapped inside an RK multi-stage scheme. Due to the nonlinear nature of the GMRES solver, the overall smoother is not a linear operator, and thus the Arnoldi iterations may fail to correctly compute the eigen-spectrum. However, the specific novelty of the JT–KIRK scheme is to wrap the GMRES into an RK multi-stage scheme to achieve high-frequency damping, it is this attribute that makes the solver suitable to be used as a smoother within multigrid. Similarly, this diminishes the non-linear character of GMRES sufficiently that the effects of the linear RK multi-stage scheme and the linear multigrid scheme dominate. As a confirmation, the magnitude of the largest eigenvalue is found to be in excellent agreement with the asymptotic convergence/divergence rate of the adjoint solver, demonstrating the validity of the eigen-analysis.

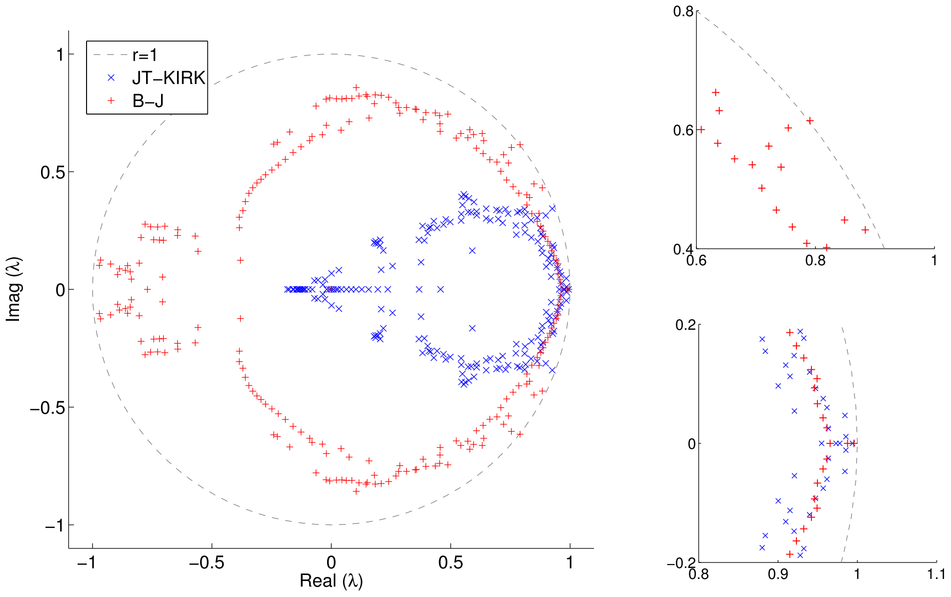

Fig. 14. The dominant eigenvalues values for both the B–J and JT–KIRK solvers produced using Arnoldi iterations. The zoomed region on the top right shows the unstable mode.

Table 10
Case 4: effect of flow convergence level and adjoint scheme on the adjoint convergence. The solver setting for all the JT–KIRK flow and adjoint runs are the same.

Set	Flow solver	Flow convergence	Adjoint solver	Adjoint convergence
A	JT–KIRK	Full convergence	JT–KIRK	Full convergence
B	JT–KIRK	Full convergence	B–J	Divergence
C	B–J	LCO	JT–KIRK	Divergence
D	B–J	LCO	B–J	Divergence

150 dominant eigenvalues are computed for both the B–J and JT–KIRK solvers, as shown in Fig. 14. The eigenvalues for B–J include two outliers which are responsible for the lack of full convergence, while the JT–KIRK solver shows all eigenvalues well within the linear stability boundary and thus the JT–KIRK adjoint can converge fully.

Similar to the experiment with Case 3, it is also attempted to converge the adjoint using another two time-stepping combinations for the adjoint: (i) JT–KIRK for the adjoint based on the LCO-converged flow solution using B–J and (ii) B–J adjoint solver on the fully converged JT–KIRK flow solution, results are summarised in Table 10.

Different from Case 3, the JT–KIRK adjoint solver using the same setting does not manage to stabilise the adjoint for LCO-converged flow for Case 4. Eigen-analysis is performed for all the four combinations in Table 10 and spectra are shown in Fig. 15. The magnitude of the largest eigenvalue in every plot is confirmed to be in agreement with the convergence/divergence rate of the respective adjoint solves. The spectra of the two JT–KIRK adjoint solutions (A and C in Fig. 15) are very similar, both showing significant clustering of the eigenvalues compared with their counterparts using the B–J adjoint solver on the right. However, the spectrum of the adjoint solver for LCO-converged flow (C in Fig. 15) has multiple outliers that cause the adjoint to diverge. This is different from the results for Case 3, where the JT–KIRK adjoint solver is stable for both the fully converged and the LCO-converged flows. As stated in the discussion of Case 3, there is no guarantee for discrete adjoint convergence based on a non-contractive Jacobian even when using a stronger solver for the adjoint. In Case 4 the instability of the flow is large enough to prevent convergence of discrete adjoints based on the LCO state, even with JT–KIRK. Some fine tuning of the JT–KIRK adjoint solver parameters might make the adjoint solver stable also for the B–J LCO linearisation, but using JT–KIRK for both flow and adjoint is guaranteed to converge since JT–KIRK does converge the flow.

6. Conclusions

A robust implicit time-stepping scheme for the steady flow and adjoint RANS equations has been presented to address the lack of robustness of the typical Block–Jacobi (point-implicit) multigrid solvers when calculating steady state discrete adjoint solutions for cases with mild physical or numerical instabilities.

The JT–KIRK algorithm has two main features. First, as opposed to Newton–Krylov methods that use exact Jacobian-vector products, a first-order approximate Jacobian is formed, stored and approximately inverted using ILU factorisation. In our implementation, the Jacobian is computed using the Automatic Differentiation tool Tapenade [33] in vector-forward mode.

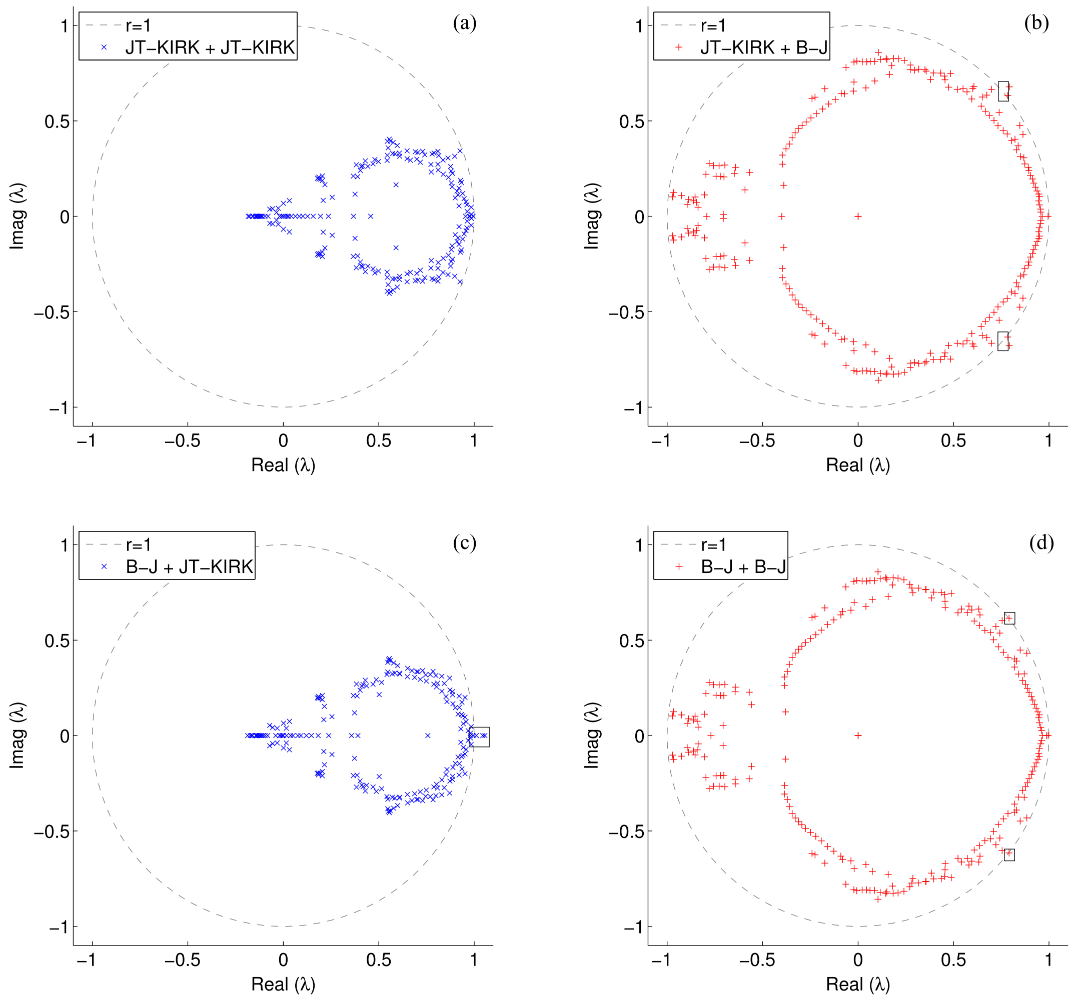

Fig. 15. The eigenvalues for the solver combinations of Table 10. Outlying eigenvalues with $|\lambda| > 1$ are highlighted in boxes.

Secondly, the algorithm uses GMRES as a smoother within a FAS-cycle multigrid method. This is achieved by embedding the linear solver inside a Runge–Kutta multi-stage time-stepping scheme, which provides high-frequency damping similar to explicit smoothers.

For a typical explicit or point-implicit compressible RANS solver using Runge–Kutta time-stepping and multigrid, only moderate change to the iterative scheme is needed to upgrade existing flow and adjoint solvers to JT–KIRK. The memory required by the implicit solver to store the approximate Jacobian and its ILU decomposition is approximately 7 times larger than the total memory required by the reference Block–Jacobi solver. The use of GMRES inside multigrid needs minimal runtime using only three Krylov vectors which does not affect the overall memory requirement significantly.

The implicit solver is applied to four representative test cases from turbo-machinery applications. For cases 1 and 2 where the baseline Block–Jacobi solver can fully converge, the JT–KIRK solver is much more efficient than the Block–Jacobi solver with 28% of the runtime for the flow solver, and slightly more efficient than the implicit solver using Symmetric Gauss–Seidel (SGS) with 79% of the runtime. The performance advantage is much more significant for the adjoint, since the expensive preconditioners have to be built only once. The adjoint JT–KIRK uses only 6% of the runtime of the adjoint Block–Jacobi, and 70% of that of SGS. Further improvements in runtime can be expected by adapting the multigrid coarsening strategy inherited from the Block–Jacobi scheme to the improved smoothing of the implicit solver, and by selectively applying Automatic Differentiation to the fluxes omitting costly but less relevant elements of the approximate Jacobian.

For the unstable test cases the Block–Jacobi flow solver converges only to limit cycle oscillations and consequently the discrete adjoint solver diverges due to a non-contractive system Jacobian. The JT–KIRK algorithm fully converges flow and adjoint, even if the unsteadiness is severe. The improvement in robustness is analysed by examining the eigenvalues of the JT–KIRK and Block–Jacobi schemes. The spectral analysis conducted for a strongly unstable case confirms that the stabilisation is brought about by JT–KIRK moving the outliers of the Block–Jacobi time-stepping inside the stability boundary.

Numerical experiments were conducted with different solver combinations for flow and adjoint to determine whether the instability is transient, and a strong and expensive solver is only needed for the primal, or whether it is still present in the fully converged solution. It is found that, as the theory suggests, the instability is persistent, i.e. if the flow requires a strong solver to converge, convergence of its discrete adjoint is only guaranteed if the same solver is used for the adjoint. However, the results also show that in the case of mild unsteadiness a stronger adjoint solver may be able to converge the adjoint based on a flow converged only to limit cycle oscillations.

For the unstable cases the choice of iterative solver does affect the values of typical objective functions for turbomachinery applications. In the case of mild unsteadiness, even though the Block–Jacobi solver does not converge and its discrete adjoint diverges, both steady-state approaches can provide objective functions with good accuracy compared to an objective function evaluated by time-averaging fully unsteady calculations. For cases with pronounced instability, the objective functions of both the Block–Jacobi and the JT–KIRK steady-state approaches can deviate strongly from the unsteady average. While stable with JT–KIRK, the use of the steady-state approach in such cases may not provide an accurate objective function value, moreover the gradient sensitivity may vary much more dramatically. Designs with destabilising separations are not untypical at intermediate steps in a design optimisation loop. Maintaining convergence of the adjoint solver also in these situations is important to not break the optimisation loop.

The JT–KIRK algorithm therefore extends the applicability of the steady-state discrete adjoint approach to marginally stable flows without compromising the efficiency for stable cases.

Acknowledgements

This research has been supported jointly by Rolls-Royce Deutschland Ltd & Co KG under grant no. Q011.001.000 and Queen Mary University of London. The authors would like to thank Rolls-Royce for the permission to publish the results. This research utilised Queen Mary's MidPlus computational facilities supported by QMUL Research-IT and funded by EPSRC grant EP/K000128/1.

References

[1] M.B. Giles, N.A. Pierce, An introduction to the adjoint approach to design, Flow Turbul. Combust. 65 (3–4) (2000) 393–415.
[2] W.K. Anderson, V. Venkatakrishnan, Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation, Comput. Fluids 28 (4) (1999) 443–480.
[3] D. Papadimitriou, K. Giannakoglou, A continuous adjoint method with objective function derivatives based on boundary integrals, for inviscid and viscous flows, Comput. Fluids 36 (2) (2007) 325–341.
[4] C. Othmer, A continuous adjoint formulation for the computation of topological and surface sensitivities of ducted flows, Int. J. Numer. Methods Fluids 58 (8) (2008) 861–877.
[5] M.B. Giles, M.C. Duta, J.-D. Müller, N.A. Pierce, Algorithm developments for discrete adjoint methods, AIAA J. 41 (2) (2003) 198–205.
[6] M.S. Campobasso, M.B. Giles, Effects of flow instabilities on the linear analysis of turbomachinery aeroelasticity, J. Propuls. Power 19 (2) (2003) 250–259.
[7] M.S. Campobasso, M.B. Giles, Stabilization of a linear flow solver for turbomachinery aeroelasticity using recursive projection method, AIAA J. 42 (9) (2004) 1765–1774.
[8] R.P. Dwight, J. Brezillon, Efficient and robust algorithms for solution of the adjoint compressible Navier–Stokes equations with applications, Int. J. Numer. Methods Fluids 60 (4) (2009) 365–389.
[9] Y. Saad, Iterative Methods for Sparse Linear Systems, Society for Industrial and Applied Mathematics, 2003.
[10] K. Mani, D.J. Mavriplis, Geometry optimization in three-dimensional unsteady flow problems using the discrete adjoint, AIAA-CP 2013-0662, 2013.
[11] J.A. Krakos, D.L. Darmofal, Effect of small-scale unsteadiness on adjoint-based output sensitivity, AIAA-CP 2009-4274, 2009.
[12] C. Othmer, Adjoint methods for car aerodynamics, J. Math. Ind. 4 (1) (2014) 6.
[13] B. Protas, T.R. Bewley, G. Hagen, A computational framework for the regularization of adjoint analysis in multiscale PDE systems, J. Comput. Phys. 195 (1) (2004) 49–89.
[14] D.J. Mavriplis, Solution of the unsteady discrete adjoint for three-dimensional problems on dynamically deforming unstructured meshes, AIAA-CP 08-727, 2008.
[15] S. Langer, Agglomeration multigrid methods with implicit Runge–Kutta smoothers applied to aerodynamic simulations on unstructured grids, J. Comput. Phys. 277 (2014) 72–100.
[16] P. Moinier, J.-D. Müller, M.B. Giles, Edge-based multigrid and preconditioning for hybrid grids, AIAA J. 40 (10) (2002) 1954–1960.
[17] J.-D. Müller, Coarsening 3-D hybrid meshes for multigrid methods, in: Proceedings of the 9th Copper Mountain Multigrid Conference, Copper Mountain, Colorado, USA, 1999.
[18] R.P. Dwight, Efficiency improvements of RANS-based analysis and optimization using implicit and adjoint methods on unstructured grids, Ph.D. thesis, University of Manchester, 2006.
[19] D.A. Knoll, D.E. Keyes, Jacobian-free Newton–Krylov methods: a survey of approaches and applications, J. Comput. Phys. 193 (2) (2004) 357–397.
[20] B. Van Leer, Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method, J. Comput. Phys. 32 (1979) 101–136.
[21] T.M. Smith, R.W. Hooper, C.C. Ober, A.A. Lorber, Intelligent nonlinear solvers for computational fluid dynamics, AIAA-CP 2006-1483, 2006.
[22] A.J. McCracken, S. Timme, K.J. Badcock, J. Eberthardsteiner, Accelerating convergence of the CFD linear frequency domain method by a preconditioned linear solver, in: 6th European Congress on Computational Methods in Applied Sciences and Engineering, 2012.
[23] A. Brandt, Multigrid techniques: 1984 guide, in: 14th VKI Lecture Series on Computational Fluid Dynamics 1984-04, Von Karman Institute for Fluid Dynamics, Rhode-St.-Genèse, Belgium, 1984.
[24] D.J. Mavriplis, An assessment of linear versus nonlinear multigrid methods for unstructured mesh solvers, J. Comput. Phys. 175 (1) (2002) 302–325.
[25] A. Naumovich, M. Förster, R. Dwight, Algebraic multigrid within defect correction for the linearized Euler equations, Numer. Linear Algebra Appl. 17 (2–3) (2010) 307–324.
[26] R. Swanson, E. Turkel, C.-C. Rossow, Convergence acceleration of Runge–Kutta schemes for solving the Navier–Stokes equations, J. Comput. Phys. 224 (1) (2007) 365–388.
[27] T.W. Roberts, R. Swanson, A study of multigrid preconditioners using eigensystem analysis, AIAA-CP 05-5229, 2005.

[28] C.-C. Rossow, Efficient computation of compressible and incompressible flows, J. Comput. Phys. 220 (2) (2007) 879–899.
[29] M. Lallemand, H. Steve, A. Dervieux, Unstructured multigridding by volume agglomeration: current status, Comput. Fluids 21 (3) (1992) 397–433.
[30] S. Langer, A. Schwöppe, N. Kroll, The DLR flow solver TAU – status and recent algorithmic developments, AIAA-CP 14-0080, 2014.
[31] P. Crumpton, P. Moinier, M. Giles, An unstructured algorithm for high Reynolds number flows on highly stretched grids, in: Numerical Methods in Laminar and Turbulent Flow, 1997, pp. 561–572.
[32] P. Moinier, Algorithm developments for an unstructured viscous flow solver, Ph.D. thesis, Oxford University, 1999.
[33] L. Hascoët, V. Pascual, The Tapenade automatic differentiation tool: principles, model, and specification, ACM Transactions on Mathematical Software 39 (3) (2013).
[34] A. Jameson, W. Schmidt, E. Turkel, et al., Numerical solutions of the Euler equations by finite volume methods using Runge–Kutta time-stepping schemes, AIAA-CP 81-1259, 1981.
[35] P.R. Spalart, S.R. Allmaras, A one equation turbulence model for aerodinamic flows, AIAA-CP 92-439, 1992.
[36] J.-D. Müller, P. Cusdin, On the performance of discrete adjoint CFD codes using automatic differentiation, Int. J. Numer. Methods Fluids 47 (8–9) (2005) 939–945.
[37] B. Christianson, Reverse accumulation and implict functions, Optim. Methods Softw. 9 (4) (1998) 307–322.
[38] F. Courty, A. Dervieux, B. Koobus, L. Hascoët, Reverse automatic differentiation for optimum design: from adjoint state assembly to gradient computation, Optim. Methods Softw. 18 (5) (2003) 615–627.
[39] F. Christakopoulos, D. Jones, J.-D. Müller, Pseudo-timestepping and verification for automatic differentiation derived CFD codes, Comput. Fluids 46 (1) (2011) 174–179.
[40] M.B. Giles, On the iterative solution of adjoint equations, in: G. Corliss, C. Faure, A. Griewank, L. Hascoët, U. Naumann (Eds.), Automatic Differentiation of Algorithms, Springer, New York, 2002, pp. 145–151.
[41] R. Swanson, E. Turkel, S. Yaniv, Analysis of a RK/implicit smoother for multigrid, in: Computational Fluid Dynamics 2010, Springer, 2011, pp. 409–417.
[42] L. Martinelli, Calculations of viscous flows with a multigrid method, Ph.D. thesis, Dept. of Mech. and Aerospace Eng., Princeton University, 1987.
[43] R. Swanson, C.-C. Rossow, An efficient solver for the RANS equations and a one-equation turbulence model, Comput. Fluids 42 (1) (2011) 13–25.
[44] D.J. Mavriplis, Multigrid strategies for viscous flow solvers on anisotropic unstructured meshes, J. Comput. Phys. 145 (1) (1998) 141–165.