AIAA JOURNAL
Vol. 44, No. 5, May 2006

Stabilizing Linear Harmonic Flow Solvers for Turbomachinery Aeroelasicity with Complex Iterative Algorithms

M. Sergio Campobasso*
Cranfield University, Cranfield, England MK43 0AL, United Kingdom
and
Michael B. Giles†
Oxford University, Oxford, England OX1 3QD, United Kingdom

The linear flow analysis of turbomachinery aeroelasticity views the unsteady flow as the sum of a background nonlinear flowfield and a linear harmonic perturbation. The background state is usually determined by solving the nonlinear steady-flow equations. The flow solution representing the amplitude and phase of the unsteady perturbation is given by the solution of a large complex linear system that results from the linearization of the time-dependent nonlinear equations about the background state. The solution procedure of the linear harmonic Euler/Navier–Stokes solver of the HYDRA suite of parallel FORTRAN codes consists of a preconditioned multigrid iteration that in some circumstances becomes numerically unstable. The results of previous investigations have already pointed at the physical origin of these numerical instabilities and have also demonstrated the code stabilization achieved by using the real GMRES and RPM algorithms to stabilize the existing preconditioned multigrid iteration. This approach considered an equivalent augmented real form of the original complex system of equations. We summarize the implementation of the complex GMRES and RPM algorithms that are applied directly to the solution of the complex harmonic equations. Results show that the complex solvers not only stabilize the code, but also lead to a substantial enhancement of the computational performance with respect to their real counterparts. The results of a nonlinear unsteady calculation that further emphasize the physical origin of the numerical instabilities are also reported.

Nomenclature

A	=	coefficient matrix of complex system
$\boldsymbol{b}$	=	right-hand side of complex linear system
$\boldsymbol{F}$	=	right-hand side of Picard iteration
F_x	=	Jacobian of $\boldsymbol{F}$
$\boldsymbol{f}$	=	projection of $\boldsymbol{F}$ onto $\mathcal{P}$
$\boldsymbol{g}$	=	projection of $\boldsymbol{F}$ onto $\mathcal{Q}$
$\tilde{H}_m$	=	Hessenberg matrix of size $((m+1)\times m)$
M	=	complex preconditioner
$\mathcal{P}$	=	subspace associated with dominant modes
$\boldsymbol{p}$	=	projection of $\boldsymbol{x}$ onto $\mathcal{P}$
$\mathcal{Q}$	=	orthogonal complement of $\mathcal{P}$
$\boldsymbol{q}$	=	projection of $\boldsymbol{x}$ onto $\mathcal{Q}$
V_m	=	matrix of m Krylov vectors
$\boldsymbol{v}$	=	eigenvector of $M^{-1}A$
$\boldsymbol{v}_m$	=	mth Krylov vector
$\boldsymbol{x}$	=	vector unknown of complex linear equations
Z	=	orthonormal basis of $\mathcal{P}$
$\boldsymbol{z}$	=	vector of current coordinate variables of $\boldsymbol{p}$ in Z
$\boldsymbol{\zeta}$	=	vector of updated coordinate variables of $\boldsymbol{p}$ in Z
λ	=	eigenvalue of $M^{-1}A$
ρ	=	spectral radius
σ	=	asymptotic convergence rate

Received 8 April 2005; presented as Paper 2005-4705 at the AIAA 17th Computational Fluid Dynamics Conference, Toronto, ON, Canada, 6–9 June 2005; revision received 4 November 2005; accepted for publication 17 November 2005.

*Senior Research Fellow, School of Engineering. Member AIAA.
†Professor of Scientific Computation, Computing Laboratory. Member AIAA.

I. Introduction

BLADE flutter and forced response of turbomachinery blade rows[1] are aeroelastic phenomena that may both lead to dramatic mechanical failures if not properly accounted for in the design of an aeroengine. Flutter occurs if the working fluid feeds energy into the vibrating blades, whereas forced response vibrations are due to an external source of excitation such as the wakes shed by an upstream blade row. The estimate of both the mean energy flux between fluid and structure in the flutter case and the unsteady forces acting on the blades in the forced response problem requires knowledge of the unsteady flowfield. The methods for the analysis of turbomachinery aeroelasticity vary from uncoupled linearized potential flow solvers[2] in which the structure and the aerodynamics are decoupled and a relatively simple flow model is used, to fully coupled nonlinear three-dimensional unsteady viscous methods[3] in which the structural and aerodynamic time-dependent equations are solved simultaneously and the aerodynamics is modeled using the Reynolds-averaged Navier–Stokes equations. Within this range, the uncoupled linear harmonic Euler and Navier–Stokes (NS) methods[4] have proved to be a successful compromise between accuracy and cost. This approach relies on the fact that the level of unsteadiness in turbomachinery flows is usually small and, hence, views it as a small perturbation of a space-periodic base or background flow. This latter is typically determined by solving the nonlinear steady flow equations, and for this reason it is often referred to as steady flow. For reasons thoroughly discussed in Ref. 5 and briefly described hereafter, however, there are circumstances in which the use of the adjective steady is inappropriate, and, hence, we will refer to the state used for the linearization as base or background flow in the rest of this paper. Because of the cyclic periodicity of the steady flow, the nonlinear equations are solved on a computational domain consisting of a single blade passage. The unsteady flowfield can then be linearized about the background state and due to linearity can be decomposed into a sum of harmonic terms, each of which can be computed independently. When it is assumed that all blades of the blade row have identical structural and geometric properties, the harmonic flow analysis can also focus on a single

blade passage rather than the whole blade row, leading to a great reduction of computational costs. This is achieved by introducing a complex periodic boundary condition that is a phase-shift between the two periodic boundaries known as the interblade phase angle (IBPA). The small amplitude of the unsteady aerodynamic forces with respect to the mechanical forces also allows one to neglect the aerodynamic coupling of structural modes, and the investigation can be carried out considering one mode at a time. In the flutter case the main source of unsteadiness is the blade vibration associated with the structural mode under investigation, and this latter is determined in a preprocessing phase by means of a finite element structural program. In the forced response problem the flow unsteadiness originates instead from a known incoming gust, such as the wakes shed by an upstream blade row. The interested reader is referred to Ref. 5 for the Lagrangian formulation of turbomachinery aeroelasticity and detailed algebraic models of the linear harmonic Euler and NS equations in the context of blade flutter and forced response.

The HYDRA suite of parallel FORTRAN codes includes a nonlinear[6] and a linear harmonic[7] Euler/NS solver, both using the one-equation Spalart–Allmaras turbulence model[8] for closure in the case of turbulent flows. (The validation of these codes is documented in Refs. 5–7, 9, and 10.) The solution procedure of both solvers is iterative and is based on Runge–Kutta time-marching accelerated by a Jacobi preconditioner and a multigrid algorithm. This preconditioned multigrid iteration can be viewed as a Picard or fixed-point iteration (FPI). Usually, the linear code converges without difficulty, but problems are encountered in situations in which the base flow calculation itself fails to converge to a steady state but instead finishes in a low-level limit cycle, often related to some physical phenomenon such as vortex shedding at a blunt trailing edge,[11,12] tip leakage flows,[13] and unsteady shock/boundary-layer or shock/wake interaction. In these circumstances the linear FPI on which the harmonic code is based becomes unstable, leading to an exponential growth of the residuals. The relationship between these numerical instabilities and the physical features of the underlying base flow is discussed in Refs. 5 and 7, which also document the successful implementation of a restarted GMRES algorithm[14] aimed at retrieving the numerical stability of the linear code. For large three-dimensional problems, however, the restarted GMRES solver may become computationally too expensive because the overall memory required by the GMRES code for an acceptable convergence rate can be twice as much that used by the standard FPI-based code. To overcome this problem, an alternative algorithm had been implemented in the linear code HYDLIN, namely, the recursive projection method[15] (RPM). The use of RPM led to both the code stabilization and a reduction of the memory usage with respect to the GMRES code.[5,16] As explained in the following section, we initially considered an augmented real form of the (complex) harmonic equations, and thus we implemented the real GMRES and RPM algorithms in the linear flow solver. However, it was later thought that the use of the complex counterparts of these two algorithms for the solution of the original complex system could result in a significant enhancement of the computational performance. The two complex solvers have been implemented and this has turned out to be true.

The main objectives of this paper are as follows: 1) Summarize the implementation of the complex GMRES and RPM solvers in HYDLIN and 2) Demonstrate the significant improvement of the computational performance achieved by their use. In Sec. II the real and complex FPIs are presented in the framework of the linear computational fluid dynamics (CFD) solver HYDLIN and the mathematical and numerical implications of the real and complex viewpoints are addressed. In Sec. III we explain why we expect the complex GMRES algorithm to have a better performance than its real counterpart and document the implementation of the complex solver. The complex RPM algorithm is reported on in Sec. IV. The enhanced computational performance of the complex solvers is then demonstrated in Secs. V and VI, which consider, respectively, the flutter analysis of a two-dimensional turbine section for a transonic flow regime and that of a civil engine fan rotor for a near-stall operating condition. In Sec. V the results are presented of a nonlinear time-accurate simulation providing further evidence of the relationship between the physical unsteadiness of the background flow and the numerical instabilities of the FPI-based linear solver. The conclusions of this work are finally summarized in Sec. VII.

II. Linear Equations and Fixed-Point Iteration

The discrete linear harmonic equations can be viewed as the complex non-Hermitian linear system

$$A\boldsymbol{x} = \boldsymbol{b} \tag{1}$$

The operator A depends on the Jacobian of the nonlinear flow equations, the constant vector $\boldsymbol{b}$ depends on the source of unsteadiness (blade motion in flutter applications and incoming gust in forced response problems), and the complex unknown $\boldsymbol{x}$ yields the amplitude and phase of the sought unsteady flow. The dimension k of system (1) is given by the product of the number of grid nodes and flow variables. For typical three-dimensional single-blade viscous aeroelastic analyses, the former parameter varies between 10^5 and 10^7, whereas the latter is equal to 6 if a one-equation turbulence model is adopted. The construction of the linearized Euler and NS equations represented by Eq. (1) is thoroughly documented in Ref. 5. This linear system could be solved conveniently by means of the complex Picard iteration

$$\boldsymbol{x}_{n+1} = (I - M^{-1}A)\boldsymbol{x}_n + M^{-1}\boldsymbol{b} \tag{2}$$

in which M^{-1} is a preconditioning operator. The linear CFD code HYDLIN, however, has been written using real arithmetic, that is, considering real vectors of size $2k$ with the factor 2 accounting for real and imaginary parts, rather than complex vectors of size k. This choice has been made for reasons of computational efficiency and because of errors sometimes introduced by highly optimizing FORTRAN compilers when dealing with complex arithmetic. In other words, the code solves the augmented real system

$$A'\boldsymbol{x}' = \boldsymbol{b}' \tag{3}$$

with

$$A' = \begin{pmatrix} \Re(A) & -\Im(A) \\ \Im(A) & \Re(A) \end{pmatrix}, \qquad \boldsymbol{x}' = \begin{pmatrix} \Re(\boldsymbol{x}) \\ \Im(\boldsymbol{x}) \end{pmatrix}, \qquad \boldsymbol{b}' = \begin{pmatrix} \Re(\boldsymbol{b}) \\ \Im(\boldsymbol{b}) \end{pmatrix}$$

The systems (1) and (3) have the same solution because if $\boldsymbol{x}^*$ satisfies Eq. (1), then $\boldsymbol{x}'^* = [\Re(\boldsymbol{x}^*)\ \Im(\boldsymbol{x}^*)]^T$ is the solution of system (3). Therefore, the real Picard iteration on which the linear code is based is

$$\boldsymbol{x}'_{n+1} = (I - M'^{-1}A')\boldsymbol{x}'_n + M'^{-1}\boldsymbol{b}' \tag{4}$$

in which M'^{-1} is a (2×2) block matrix with the same block structure as A'.

The iteration (4) is mathematically equivalent to the iteration (2) because it leads to the same real update of the solution when the initial complex state $\boldsymbol{x}_n$ of dimension k is stored in the real vector $\boldsymbol{x}'_n$ of dimension $2k$. Furthermore, the numerical execution of Eq. (4) requires fewer floating point operations than that of Eq. (2) would because almost all elements of A are either purely real or purely imaginary. The matrices associated with the operators M'^{-1} and A' are not built explicitly in HYDLIN because the code determines the matrix–vector products $M'^{-1}A'\boldsymbol{x}'$ directly from the numerical fluxes. This matrix-free approach substantially reduces the required memory allocation. Note also that the latter quantity would not change if the complex FPI (2) were used, due to matrix-free architecture.

The pseudocode of the preconditioned multigrid iteration looks like

$$\boldsymbol{x}' = \boldsymbol{x}'_{\text{start}}$$

$$\boldsymbol{x}' = MG(A', \boldsymbol{x}', \boldsymbol{b}', n_{\text{cl}})$$

$$\boldsymbol{x}'_{\text{finish}} = \boldsymbol{x}'$$

where MG is the core routine that performs n_{cl} multigrid cycles of the preconditioned FPI (4), $\boldsymbol{x}'_{\text{start}}$ is the initial state, and $\boldsymbol{x}'_{\text{finish}}$ is the solution after the n_{cl} iterations.

The preconditioner M'^{-1} consists of a five-stage Runge–Kutta algorithm, a Jacobi preconditioner and a geometric multigrid scheme. Because the Jacobi preconditioner is based on the solution of the nonlinear steady equations, M^{-1} depends on the particular problem under investigation and also on the choice of several numerical parameters, such as the number of iterations on each grid level. More details on the preconditioned solution strategy of the linearized flow equations can be found in Refs. 5 and 10.

III. GMRES Stabilization

Linear stability analysis of iteration (4) shows that a necessary condition for its convergence is that all of the eigenvalues of $(I - M'^{-1}A')$ lie within the unit circle centered at the origin in the complex plane or, equivalently, that all of the eigenvalues of $M'^{-1}A'$ lie in the unit disk centered at (1, 0). For most aeroelastic problems of practical interest, this condition is fulfilled and the linear code converges without difficulty. In some cases, however, the FPI fails to converge because of a few complex eigenvalues lying outside the unit disc and causing the exponential growth of the residual. The physical origin of these outliers has already been highlighted in Ref. 7 by means of an Arnoldi-based eigenmode analysis of the operator $M'^{-1}A'$. That paper also reports the successful application of a preconditioned restarted real GMRES solver for the solution of system (3) in the presence of outliers. However, it was later supposed that the use of the complex GMRES solver for the solution of system (1) could result in a reduction of the computational work required for achieving a given level of convergence. This was motivated by the fact that the eigenspace of the complex system is a subset of that of its real augmented counterpart. More precisely, all of the eigenvalues of $M'^{-1}A'$ are complex conjugate pairs, whereas only one eigenvalue of each pair is also an eigenvalue of $M^{-1}A$. Hence, the convergence rate of complex GMRES may be higher than that of real GMRES for a given number of Krylov vectors because the former solver captures more distinct (that is, not conjugate) eigenmodes performing the same computational work. The relationship between the real and complex spectrum can be established as follows. Let $(\lambda, \boldsymbol{v})$ be an eigenpair of $M^{-1}A$. Then

$$\begin{pmatrix} \Re(M^{-1}A) & -\Im(M^{-1}A) \\ \Im(M^{-1}A) & \Re(M^{-1}A) \end{pmatrix} \begin{pmatrix} \Re(\boldsymbol{v}) \\ \Im(\boldsymbol{v}) \end{pmatrix} = \begin{pmatrix} \Re(\lambda\boldsymbol{v}) \\ \Im(\lambda\boldsymbol{v}) \end{pmatrix} \tag{5}$$

A suitable linear combination of the rows of Eq. (5) gives

$$\begin{pmatrix} \Re(M^{-1}A) & -\Im(M^{-1}A) \\ \Im(M^{-1}A) & \Re(M^{-1}A) \end{pmatrix} \begin{pmatrix} \Re(\boldsymbol{v}) + i\Im(\boldsymbol{v}) \\ \Im(\boldsymbol{v}) - i\Re(\boldsymbol{v}) \end{pmatrix} = \begin{pmatrix} \Re(\lambda\boldsymbol{v}) + i\Im(\lambda\boldsymbol{v}) \\ \Im(\lambda\boldsymbol{v}) - i\Re(\lambda\boldsymbol{v}) \end{pmatrix}$$

or

$$(M'^{-1}A') \begin{pmatrix} \boldsymbol{v} \\ -i\boldsymbol{v} \end{pmatrix} = \lambda \begin{pmatrix} \boldsymbol{v} \\ -i\boldsymbol{v} \end{pmatrix}$$

Because $M'^{-1}A'$ is real, taking the complex conjugate of the foregoing equation yields

$$(M'^{-1}A') \begin{pmatrix} \bar{\boldsymbol{v}} \\ i\bar{\boldsymbol{v}} \end{pmatrix} = \bar{\lambda} \begin{pmatrix} \bar{\boldsymbol{v}} \\ i\bar{\boldsymbol{v}} \end{pmatrix}$$

Thus we see that if $(\lambda, \boldsymbol{v})$ is an eigenpair of $M^{-1}A$, then

$$\left(\lambda, \begin{bmatrix} \boldsymbol{v} \\ -i\boldsymbol{v} \end{bmatrix} \right)$$

is the corresponding eigenpair of $M'^{-1}A'$. However, $(\bar{\lambda}, \bar{\boldsymbol{v}})$ is not necessarily an eigenpair of $M^{-1}A$, though

$$\left(\bar{\lambda}, \begin{bmatrix} \bar{\boldsymbol{v}} \\ i\bar{\boldsymbol{v}} \end{bmatrix} \right)$$

is an eigenpair of $M'^{-1}A'$.

The complex GMRES algorithm is based on the progressive reduced Arnoldi factorization of $M_G^{-1}A$,

$$M_G^{-1}AV_m = V_{m+1}\tilde{H}_m \tag{6}$$

where m is the current iteration; $\tilde{H}_m$ is a Hessenberg matrix of size $((m+1) \times m)$; V_m is a matrix of size $(k \times m)$ whose m columns $\boldsymbol{v}_j$, $j = 1, \ldots, m$, form an orthonormal basis for the Krylov subspace K_m; V_{m+1} is V_m augmented with a new Krylov vector $\boldsymbol{v}_{m+1}$, and M_G^{-1} is a suitable preconditioner whose dependence on M^{-1} is defined hereafter. When $h_{j,m}$, $j = 1, \ldots, m$, denotes the elements of the mth column of $\tilde{H}_m$, the mth column of Eq. (6) can be written as

$$M_G^{-1}A\boldsymbol{v}_m = h_{1,m}\boldsymbol{v}_1 + \cdots + h_{m+1,m}\boldsymbol{v}_{m+1} \tag{7}$$

which shows that $\boldsymbol{v}_{m+1}$ satisfies an $(m+1)$-term recursive relation involving itself and the preceding m Krylov vectors. At the mth GMRES iteration, the solution of Eq. (1) is approximated by the linear combination of the m $\boldsymbol{v}_j$ that minimizes the 2-norm of the residual $\boldsymbol{r}_m = M_G^{-1}(\boldsymbol{b} - A\boldsymbol{x}_m)$ and is thus given by $\boldsymbol{x}_m = \boldsymbol{x}_{\text{start}} + V_m\boldsymbol{t}_m$, in which $\boldsymbol{t}_m$ is the column vector containing the m coefficients of the linear combination.

The implementation of GMRES in HYDLIN has been carried out at the top routine level, and it has not required any change to the routine MG, which is used as a black-box to determine the preconditioned Krylov vectors $M_G^{-1}A\boldsymbol{v}_j$. The computationally cheap minimization is also performed at the top routine level, and the pseudocode of the main HYDLIN using GMRES is

$\boldsymbol{v}_1 = \text{MG}(A, \boldsymbol{x}_{\text{start}}, \boldsymbol{b}, n_{\text{cl}}) - \boldsymbol{x}_{\text{start}}$; $\quad \boldsymbol{v}_1 = \boldsymbol{v}_1/\|\boldsymbol{v}_1\|$

for $m = 1 : n_{Kr}$

$M_G^{-1}A\boldsymbol{v}_m = \boldsymbol{v}_m - \text{MG}(A, \boldsymbol{v}_m, \boldsymbol{0}, n_{\text{cl}})$

$\boldsymbol{v}_{m+1}$ from equation (7); $\quad \boldsymbol{v}_{m+1} = \boldsymbol{v}_{m+1}/\|\boldsymbol{v}_{m+1}\|$

determine $\boldsymbol{t}_m$ which minimizes $\|\boldsymbol{r}_m\|$

end

$\boldsymbol{x}_{\text{finish}} = \boldsymbol{x}_{\text{start}} + V_{n_{Kr}}\boldsymbol{t}_{n_{Kr}}$

The first Krylov vector $\boldsymbol{v}_1$ is the normalized residual of the preconditioned system after n_{cl} multigrid cycles and n_{Kr} is the overall number of GMRES iterations. Because the iteration (2) is applied n_{cl} times at each GMRES iteration, the relationship between M_G^{-1} and M^{-1} is

$$M_G^{-1}A = I - (I - M^{-1}A)^{n_{\text{cl}}} \tag{8}$$

The choice $n_{\text{cl}} = 1$ results in equal preconditioning of the Picard and the GMRES iterations, but previous work has shown that a suitable (case-dependent) choice of n_{cl} can reduce the overall number of multigrid cycles needed to achieve a given level of convergence.[5,7] Note that the right-hand side of Eq. (1) is set to zero before using MG to determine $M_G^{-1}A\boldsymbol{v}_m$. The value of n_{Kr} required for full convergence is much smaller than k, but nevertheless very large with respect to the computing resources usually available. This problem is overcome using the restart option, that is, performing n_{Kr} iterations and restarting GMRES from the updated solution, recomputing from there a new set of n_{Kr} Krylov vectors. This is achieved by wrapping the inner loop described earlier with an outer one that restarts GMRES each time. Values of n_{Kr} between 10 and 30 make the computation affordable even for large problems, and a good convergence level is usually achieved within 20 restarted cycles using $n_{\text{cl}} = 3$.

In Sec. II, it was observed that the real FPI (4) and the complex FPI (2) are mathematically equivalent, and the former one has been implemented in HYDLIN because it is computationally more efficient due to the particular structure of the matrices in the problem at hand. For the same reasons, the complex GMRES solver calculates the preconditioned complex Krylov vector $M_G^{-1}A\boldsymbol{v}_m$ using the existing real routine MG, as the real GMRES solver does. As far as the use of this routine is concerned, the difference between the real and the complex case is only theoretical: In the former case the subvector of length k associated with the first k elements of $M_G'^{-1}A'\boldsymbol{v}'_m$

and that containing the remaining k elements shall be seen as a subdivision of a real vector of length $2k$, whereas such subvectors truly are the real and complex parts of a k-dimensional complex vector in the latter case. The only practical differences between the real and complex GMRES solvers are the orthogonalization (7) and the solution of the least-square problem, which are real or complex in either case. The complex orthogonalization and minimization require twice as many operations as their real counterparts, but their cost is still very small with respect to the calculation of $M_G^{-1}A\boldsymbol{v}_m$. When the extra CPU time required for the Arnoldi orthogonalizations and the solution of the optimization problem is included in the cost of one multigrid cycle, the CPU time for executing a given number of multigrid cycles using complex GMRES with $n_{\mathrm{cl}} = 3$ and $10 \le n_{Kr} \le 30$ is only from 1 to 2% higher than using the FPI (4). For a given n_{Kr}, the extra memory allocation of the complex and real solvers are the same. However, all of the test cases analyzed so far have highlighted that the complex solver has a substantially better numerical performance than its real counterpart because it requires significantly fewer multigrid iterations to achieve a given convergence level.

IV. RPM Stabilization

The price to be paid for the GMRES stabilization is the burden associated with the extra memory allocation for the Krylov vectors. To stabilize the linear code reducing the additional memory requirement, the complex RPM solver had also been implemented in HYDLIN. Its real counterpart had already been implemented, and it has led to the code stabilization with the desired memory reduction.[17] The benefit of using the complex version is a further reduction of the required memory, as shown hereafter.

The complex RPM solver is based on the projection of the Picard iteration (2) onto two orthogonal subspaces $\mathcal{P}$ and $\mathcal{Q}$ of $\mathcal{C}^k$. The former is the maximal invariant subspace associated with the subset of l outliers and the latter is the orthogonal complement of the former subspace. At each RPM iteration, only the projection of Eq. (2) onto the subspace $\mathcal{Q}$ is solved with the Picard iteration; the projection onto the typically low-dimensional subspace $\mathcal{P}$ is instead solved with Newton's method. When Z denotes an orthonormal basis of $\mathcal{P}$, the orthogonal projectors P and Q of the subspaces $\mathcal{P}$ and $\mathcal{Q}$ are defined, respectively, as $P = ZZ^H$ and $Q = I - P$. Each time the calculation is diverging, the basis Z is augmented with the current dominant eigenmode, and the projectors P and Q are updated accordingly. When $\boldsymbol{F}$ denotes the right-hand side of the Picard iteration (2), the projections f and g of this equation onto $\mathcal{P}$ and $\mathcal{Q}$ are defined, respectively, as

$$\boldsymbol{f} = P\boldsymbol{F} = P[(I - M^{-1}A)\boldsymbol{x} + M^{-1}\boldsymbol{b}]$$

$$\boldsymbol{g} = Q\boldsymbol{F} = Q[(I - M^{-1}A)\boldsymbol{x} + M^{-1}\boldsymbol{b}]$$

and the stabilized iteration can be written as

$$\boldsymbol{p}_{\nu+1} = \boldsymbol{p}_\nu + (I - f_p)^{-1}[\boldsymbol{f}(\boldsymbol{p}_\nu, \boldsymbol{q}_\nu) - \boldsymbol{p}_\nu] \tag{9}$$

$$\boldsymbol{q}_{\nu+1} = \boldsymbol{g}(\boldsymbol{p}_\nu, \boldsymbol{q}_\nu) \tag{10}$$

where

$$\boldsymbol{p} = P\boldsymbol{x}, \qquad \boldsymbol{q} = Q\boldsymbol{x}, \qquad f_p \equiv PF_xP, \qquad F_x = I - M^{-1}A$$

Note that Eq. (9) requires the inversion of $(I - f_p) : \mathcal{P} \to \mathcal{P}$, which is the restriction of $(I - F_x)$ to the subspace $\mathcal{P}$, and is therefore equivalent to $(I - f_p)P$. It is easily verified that

$$(I - f_p)^{-1}P = Z[I - Z^H(I - M^{-1}A)Z]^{-1}Z^H = Z[I - H]^{-1}Z^H \tag{11}$$

where $(I - H)$ is a small matrix of size l, whose inversion requires minimum computational effort. The stability analysis of this algorithm shows that its spectral radius is smaller than 1, that is, the stabilized RPM iteration is stable.[15]

The basis Z is updated directly from the iterates $\boldsymbol{q}_\nu$ of the modified iteration (10), without computing Jacobians. This is done by monitoring the rate of convergence of the iterates $\boldsymbol{q}_\nu$. If the residual starts growing, it is argued that some of the eigenvalues of $g_q = QF_xQ$ lie outside the unit circle. When the real Picard iteration (4) is considered, the instability is usually caused by a complex conjugate eigenpair, and two real vectors of dimension $2k$ have to be appended to Z. When the complex system is considered, however, the instability is caused by a complex eigenvalue and, consequently, only one complex vector of dimension k has to be included in Z, thus halving the memory allocation. This procedure can be used to append to Z not only the unstable eigenmodes, but also the stable ones whose eigenvalues are very close to the unit circle, allowing one to speed up the convergence rate even in the absence of outliers. The interested reader is referred to Refs. 15 and 16 for the description of the procedure to augment Z. In the current implementation, we work recursively, adding one eigenmode at a time to Z until a satisfactory convergence rate is obtained. However, one could also look for more than one dominant mode at a time and include in Z all of the modes needed for stabilizing the initial FPI. The experience gained so far makes us believe that this approach would require fewer iterations to achieve the desired level of convergence. Looking for one dominant mode at a time, in fact, may result in a longer numerical transient because the RPM iteration continues to diverge until all outliers have been included in $\mathcal{P}$. On the other hand, the single-mode search is likely to yield a reduced memory usage because the memory for storing a new vector of Z can be allocated dynamically only when the convergence rate needs to be improved. Looking for all unstable modes at a time, conversely, would require assuming the number of unstable modes and allocating all of the corresponding memory as soon as the first instability is detected. Hence, computer memory is wasted each time the actual number of unstable modes of the problem at hand is less than the assumed value.

In the actual computation, one introduces a vector $\boldsymbol{z}$ of length l whose elements are the coordinate variables for the representation of $\boldsymbol{p}$ in the basis Z,

$$z \equiv Z^H\boldsymbol{p} = Z^H\boldsymbol{x}$$

from which it follows that

$$\boldsymbol{p} = Zz, \qquad \boldsymbol{x} = Zz + \boldsymbol{q}$$

The iteration (9) in the subspace $\mathcal{P}$ can be written in these variables using Eq. (11) and observing that $Z^HZ = I$,

$$z_{\nu+1} = z_\nu + (I - H)^{-1}\left[Z^H\boldsymbol{F}(\boldsymbol{x}_\nu) - z_\nu\right]$$

The implementation of the complex RPM solver in HYDLIN has been carried out at the top routine level. Similarly to the GMRES case, the implementation of RPM does not require any change to the existing real routine MG that is used to determine the projection $\boldsymbol{q}$ of the solution $\boldsymbol{x}$ onto $\mathcal{Q}$ [Eq. (10)]. The two subvectors of length k associated with the first k elements of $M_G'^{-1}A'\boldsymbol{x}'_\nu$ and the remaining k elements shall be viewed as the real and complex parts of a k-dimensional complex vector. The computationally cheap inversion of the complex matrix $(I - H)$ along with the other Hermitian vector products is carried out at the top routine level, and the pseudocode of the main HYDLIN using complex RPM is

```
Z = [ ];   l = 0;   ν = 0;   x_ν = x_start
while ‖ b − A x_ν ‖ > tolerance
  x_{ν+1} = MG(A, x_ν, b, n_cl)
  % stabilized iteration
  if l > 0
    z_ν = Z^H x_ν;   ζ_ν = Z^H x_{ν+1};
    z_{ν+1} = z_ν + (I − H)^{-1}(ζ_ν − z_ν)
    q_{ν+1} = x_{ν+1} − Z ζ_ν
    x_{ν+1} = Z z_{ν+1} + q_{ν+1}
  endif
  ν = ν + 1
  % increase basis size
  if not converging
```

```
        Z = [Z d];     l = l + 1; ;     v = 0
        for j = 1 : l
           H(:, j) = Z^H MG(A, Z(:, j), 0, n_cl)
        end
      endif
    end
    x_finish = q_v+1 + Z z_v+1
```

The section of the pseudocode labeled stabilized iteration corresponds to the implementation of Eqs. (9) and (10). At each step of the stabilized iteration n_{cl} multigrid cycles are performed, as set by the last argument of MG. In the section labeled "increase basis size," the column vector $\boldsymbol{d}$ containing the eigenmode associated with the current outlier is appended to Z. Thereafter, the subroutine MG is used to determine the columns of $F_x Z$ by setting $\boldsymbol{b} = 0$ and performing n_{cl} multigrid cycles on each column of Z. Then the matrix H is updated according to Eq. (11).

Let us denote by σ the asymptotic convergence rate of the RPM iteration after all outliers have been appended to $\mathcal{P}$. This parameter is the slope of the curve "residual vs number of multigrid cycles," and its mathematical definition is

$$\sigma = \Delta[\log(\mathrm{rms})]/N_{\mathrm{MG}}$$

where rms is the rms of the nodal residuals of the linearized flow equations and Δ denotes its variation over N_{MG} multigrid cycles. The paper by Shroff and Keller[15] shows that the spectral radius ρ of the RPM-stabilized iteration is that of the operator obtained by projecting $M^{-1}A$ onto the (final) stable subspace $\mathcal{Q}$. Hence, it follows that

$$\sigma \approx \log \rho \tag{12}$$

The theoretical analysis and the numerical results in Ref. 5 show that σ is independent of n_{cl}. However, the overall number of multigrid cycles needed to achieve a given level of convergence is constant only for $n_{\mathrm{cl}} \leq 10$ and increases for $n_{\mathrm{cl}} > 10$ due to the higher level of the residual when the last outlier is appended to $\mathcal{P}$ and RPM starts converging. Indeed, the relationship (12) also holds for the standard FPI (2), namely, when the invariant space $\mathcal{P}$ is kept empty. In this circumstance σ will clearly be positive if $M^{-1}A$ has one or more outliers. Note also that σ has the same value in both the real and complex implementation of RPM because the real operator $M'^{-1}A'$ and the complex operator $M^{-1}A$ have the same spectral radius.

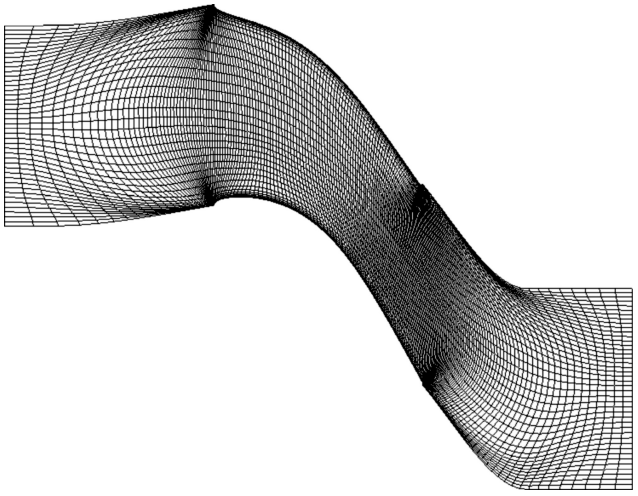
The additional CPU time used by the complex RPM solver with respect to the standard FPI code is that required for the Hermitian vector products of the stabilized iteration, the calls of MG to update H, and the inversion of H. This computational cost grows with l. The order of magnitude of this parameter based on the problems investigated so far varies between 1 and 10 and the operations listed earlier are fairly inexpensive for values of l in this range. When this additional CPU time is included in that required for performing a multigrid cycle and MG is used with $n_{\mathrm{cl}} = 1$, the CPU time for executing a given number of multigrid cycles using complex RPM in a problem with $l \leq 5$ is less than 1% higher than using the standard FPI. The additional memory required by the implementation of the complex RPM algorithm in HYDLIN is that needed for the vectors of the basis Z and is given by $2k \times (l+1) \times v_s$, in which the factor 1 accounts for a new work array and v_s is the memory required for storing a single scalar. By comparison, the extra memory allocation required by GMRES is $2k \times (n_{Kr}+1) \times v_s$ and is independent of the number of outliers. All codes of the HYDRA suite have been implemented using double precision, and in this circumstance $v_s = 8$ bytes.

V. Two-Dimensional Turbine Section

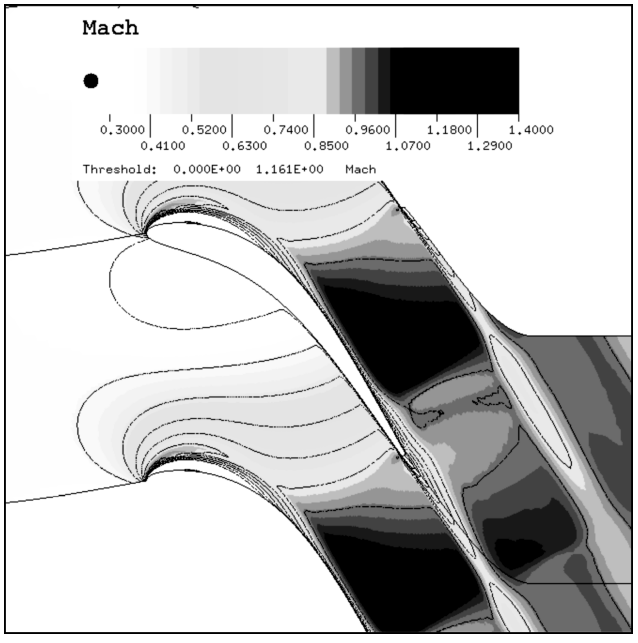
A. Real and Complex GMRES

The two-dimensional turbine section of the 11th standard configuration is the midspan blade-to-blade section of an annular turbine cascade with 20 blades. The test rig and cascade geometry are described in Ref. 18, which also provides experimental measurements and various computed results of the steady and unsteady flowfield due to blade plunging with prescribed IBPA. Two steady working conditions are considered: a subsonic one with exit Mach number of 0.68 and a transonic one with exit Mach number of 0.96. The frequencies of the imposed blade vibration at these two flow conditions are 209 and 212 Hz, respectively. This test case had already been used to demonstrate the predictive capabilities of HYDLIN and to test the real GMRES solver,[7] as well as to test the real RPM solver.[16] In this paper, it will be used to demonstrate the effectiveness of the complex GMRES and RPM algorithms.

The computational grid used for the investigation is a quasi-orthogonal H-type mesh with medium refinement: It has 273 nodes in the streamwise and 65 nodes in the pitchwise directions, for a total of 17,745 grid nodes. A preliminary mesh-refinement analysis carried out using a coarser 7869-node (183×43) and finer 39,673-node (409×97) mesh has shown no difference of practical interest between the results obtained with the medium and finer grids. The coarse mesh is shown in Fig. 1a, whereas Fig. 1b shows the Mach contours associated with the transonic flow regime. They reveal the existence of a separation bubble on the suction side close to the leading edge and a shock impinging on the suction side close



a)



b)

Fig. 1 Two-dimensional turbine section: a) computational mesh and b) Mach contours for transonic conditions.

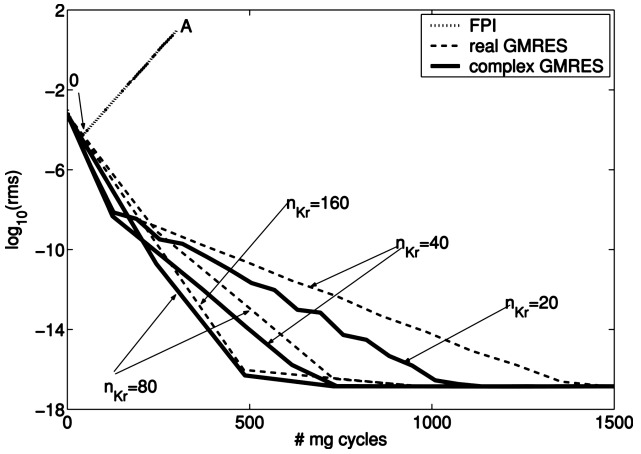


Fig. 2 Flutter analysis of two-dimensional turbine, IBPA = 180 deg: convergence histories of complex and real GMRES solvers ($n_{cl} = 3$).

to the trailing edge and crossing the wake shed by the upper blade. Note also that both the wake and the boundary layer on the suction side thicken remarkably after passing through the shock.

The calculation of this transonic nonlinear base flow has been carried out using the nonlinear flow solver HYD, whose iterative solution strategy is based on a preconditioned multigrid iteration[6] similar to that used by HYDLIN. The nonlinear code has been used with three grid levels for the multigrid solver, and its residuals have converged to within machine accuracy. On the other hand, the linear calculations based on the transonic base flow and using the standard FPI (4) with the same numerical control parameters of the nonlinear calculation diverge for all IBPAs. This is emphasized by the convergence history labeled FPI in Fig. 2, which refers to the calculation of the harmonic flowfield with IBPA = 180 deg. The variable on the x axis is the number of multigrid cycles, and that on the y axis is the logarithm in base 10 of the rms of the nodal residuals of the continuity, momentum, energy, and turbulence equations evaluated at the end of each multigrid cycle. (In practice, this variable is the rms of the unpreconditioned residuals.) The exponential growth of the residuals could be removed neither by lowering the Courant–Friedrichs–Lewy number nor by changing the grid topology and refinement. Other attempts to suppress the numerical instabilities included the following: 1) varying the multigrid control parameters and 2) performing single-grid calculations. In none of these cases could a converged solution be determined.

Conversely, the use of GMRES allows one to retrieve the convergence of the linear code. Figure 2 also shows the logarithm of the rms using both the complex and the real implementation with various values of n_{Kr} and $n_{cl} = 3$. Figure 2 shows that, for a given n_{Kr}, the number of multigrid cycles required to obtain a converged solution is always significantly smaller when using the complex solver. An equivalent interpretation of the results of Fig. 2 is that a given asymptotic convergence rate σ can be achieved with fewer Krylov vectors when the complex rather than the real solver is used. More precisely, the ratio between the number of Krylov vectors needed to obtain the same σ using the real and the complex solvers tends to two as n_{Kr} increases. This behavior can be explained by the fact that the eigenspace of the complex system is a subset of that associated with its real augmented counterpart. The latter results by doubling the former through complex conjugation. For a given n_{Kr}, the complex solver looks for more distinct dominant eigenmodes converging faster to the solution, whereas the real solver has to spend one-half of its resources identifying the fictitious conjugate modes.

The number of Krylov vectors per restarted GMRES cycle n_{Kr} and the number of preconditioned multigrid cycles per GMRES iteration n_{cl} significantly affect the convergence rate σ of the linear calculations. The latter dependence is analyzed in Ref. 5, whereas the GMRES convergence histories of Fig. 2 show that σ increases monotonically with n_{Kr} for both the real and the complex algorithm. The price to be paid for this convergence speed up is the additional memory needed for the Krylov vectors. The memory and CPU-time requirements of the FPI and complex GMRES codes are compared in Table 1. The second column reports the memory used by the linear code in megabytes. The first entry of the column is the memory used by the FPI-based code, whereas the following five elements provide the memory allocated by the GMRES code for the values of n_{Kr} in the first column. The third column reports the additional memory of the GMRES code with respect to that used by the Picard iteration as a percentage increment of this latter value. Note that the use of GMRES with 40 Krylov vectors already requires more than twice as much memory as the standard FPI. The averaged cost of a multigrid cycle performed by the GMRES code is also higher than that of the standard FPI due to the Arnoldi factorization and the solution of the optimization problem performed at each GMRES step. This extra cost, however, is very small. The fourth column of Table 1 provides the CPU time required for the execution of one multigrid cycle using either solver. The first entry refers to the FPI, whereas the remaining five elements refer to the GMRES code run with the value of n_{Kr} given in the first column. These five numbers have been obtained by dividing the overall CPU time taken to run the GMRES solver by the overall number of multigrid cycles executed by HYDLIN. Note that this is an averaged cost of the multigrid cycle because the amount of operations of the Arnoldi factorization is not constant at each GMRES step, but rather increases with the current number of Krylov vectors involved in the orthogonalization. These results demonstrate that the additional CPU time of the GMRES solver is very small, and this conclusion is further emphasized in the fifth column of Table 1, which presents this additional cost as a percentage increment with respect to the reference CPU time of one multigrid cycle of the FPI code. (When 40 Krylov vectors are used, for instance, the additional CPU time is only 1.6% higher.) For this reason, the variable on the x axis of the convergence plot in Fig. 2 can also be viewed as a scaled CPU time. All of the data of Table 1 have been obtained benchmarking HYDLIN on a Sun Blade 1000 workstation. These calculations have also been carried out using the parallel code on eight processors of the OCCF computer cluster consisting of 24 four-processor Sun Ultra-80 nodes, with a Sun Blade-1000 front end. With this setup, the execution of 1000 multigrid cycles with $n_{Kr} = 20$ is about 38 min.

Table 1 Flutter analysis of two-dimensional turbine: Memory and CPU time required by complex GMRES solver for various n_{Kr} and $n_{cl} = 3$

n_{Kr}	Memory, MB	Additional memory, %	CPU time, s	Additional CPU time, %
——	55	——	23.01	——
10	71	34	23.22	0.91
20	87	64	23.25	1.04
40	119	124	23.37	1.56
80	184	247	23.63	2.69
160	314	492	24.09	4.69

B. Spectral Analysis

To investigate the origin of the numerical instability of the FPI, the first 150 dominant eigenmodes of $M^{-1}A$ have been determined using the Arnoldi-based procedure described in Refs. 7 and 5, and this required performing 150 GMRES iterations ($n_{Kr} = 150$), with $n_{cl} = 1$. This choice of n_{cl} allows one to determine the eigenmodes of $M^{-1}A$ because it leads to the equality $M_G^{-1}A = M^{-1}A$, which follows from Eq. (8). The first 150 dominant eigenvalues are plotted in the complex plane of Fig. 3a, and the first four dominant eigenvalues are labeled from 1 to 4 in order of decreasing distance from the center of the unit disk in the three enlarged views in Fig. 3b. Note that the first two eigenvalues are complex outliers, and these are responsible for the instability of the standard FPI (4). In fact, inserting the data relative to the slope σ_{OA} of the ascending branch OA of the FPI residual curve (Fig. 2) and the spectral radius of $(I - M^{-1}A)$ (radius of the outlier 1) into Eq. (12) yields $47.31e{-}3 \approx 47.53e{-}3$, which demonstrates the correctness of the mathematical analysis.

The static pressure contours of the eigenvectors associated with the eigenvalues 1 and 3 are shown in Figs. 4a and 4b, respectively.

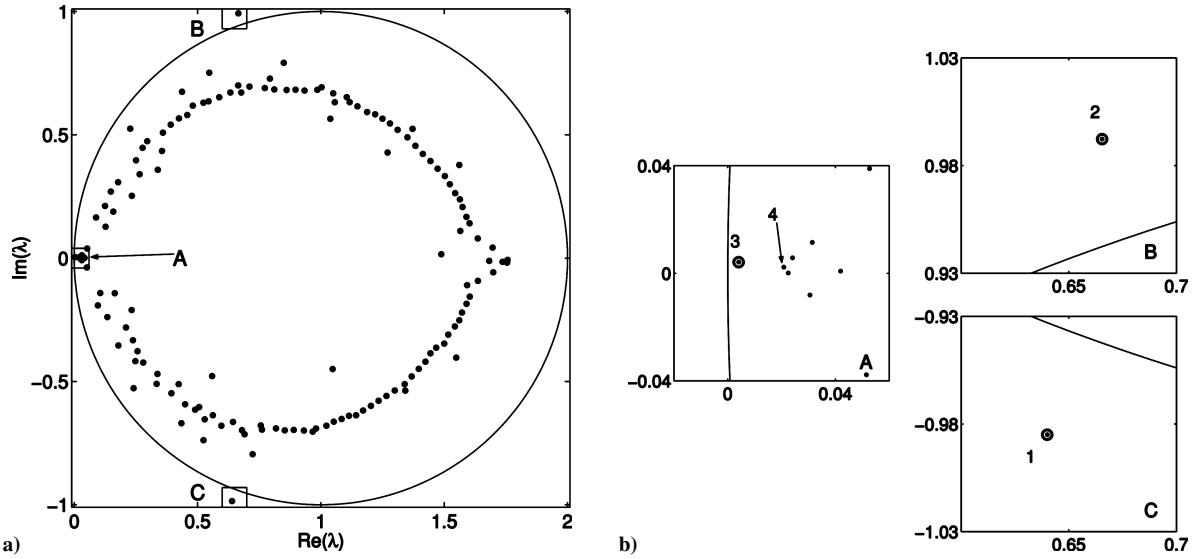


Fig. 3 Flutter analysis of the two-dimensional turbine, IBPA = 180 deg: a) first 150 dominant eigenvalues of $M^{-1}A$ and b) enlarged views with outliers.

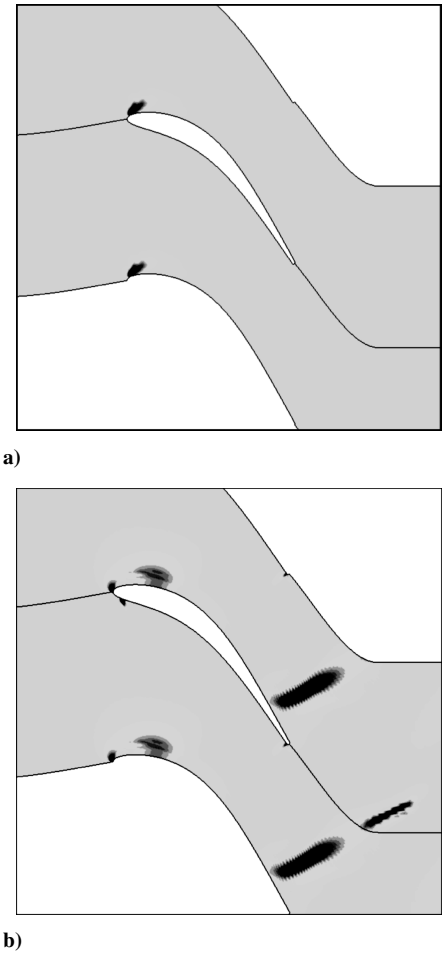


Fig. 4 Pressure amplitude of dominant eigenmodes: a) eigenvector associated with outlier 1 and b) eigenvector associated with eigenvalue 3.

More precisely, Figs. 4a and 4b show the contours of the amplitude of the static pressure nondimensionalized by the peak pressure amplitude. The maximum pressure amplitude of the eigenvector corresponding to the outlier 1 occurs at the edge of the separation bubble on the suction surface, and it has nonzero amplitudes only in a tiny neighborhood around it (Fig. 4a). This shows that the origin of the numerical instability is the limit cycle associated with the unsteady character of this flow separation. The eigenmode associated with the outlier 2 looks exactly like that just described and, therefore, is not reported here. Figure 4b shows that the pressure amplitude of the eigenvector associated with the eigenvalue 3 is slightly less localized than the former two. In fact, one can now observe patches of nonzero amplitudes in both the separation and the shock regions, but also at the stagnation point on the pressure side close to the leading edge, where the gas stream experiences a strong acceleration. This mode does not lead to any numerical instability because the eigenvalue 3 lies in the unit disk. In the absence of outliers, however, it would be responsible for a very low convergence rate of the standard FPI because of the proximity of its eigenvalue to the unit circle.

This modal analysis has been carried out for all other IBPAs, and it has been found that the first few dominant eigenmodes do not vary with the IBPA despite that $M^{-1}A$ depends on it.[5,7] This observation is in line with the fact that the standard FPI is unstable for all IBPAs. The independence of the unstable modes on the IBPA is presumably due to their high spatial localization.

C. Real and Complex RPM

The convergence histories of the real and complex RPM solvers performing one multigrid cycle per stabilized iteration ($n_{\mathrm{cl}} = 1$) are provided in Figs. 5a and 5b, respectively. The residual curves of the FPI-based code and the GMRES solver using $n_{Kr} = 20$ and $n_{\mathrm{cl}} = 3$ are also reported on for completeness. The discontinuities of the RPM convergence curves at the iterations labeled 1 and 2 in Figs. 5a and 5b occur when the first two dominant modes, that is, the two outliers with the same labels in Fig. 3b, are included in the unstable eigenspace $\mathcal{P}$. The slope of the branches $3E_0$ is determined by the spectral radius of the projection of $M^{-1}A$ onto the stable space $\mathcal{Q}$, which at this stage is the distance of the eigenvalue 3 from the center of the unit disk in Fig. 3a. Note that this value is the same for both the real and complex operators, and the only difference between the two cases is that the complex conjugate of the eigenvalue 3 is also an eigenvalue of the real operator. This explains why the slope of the branch $3E_0$ is the same in the real and complex convergence plots. The same observations apply to the branches 3_0E_1: The convergence speed up occurring at the iteration labeled 3 takes place when the dominant eigenvalue 3 is also appended to $\mathcal{P}$. The branches 3_0E_1 are steeper than the branches $3E_0$ because the spectral radius of the stabilized iteration is now determined by the eigenvalue 4, which is closer than the eigenvalue 3 to the center of the unit disk. Inserting in Eq. (12) the computed data relative to the slope of the branch 3_0E_1 and the radius of the eigenvalue 4 yields $-20.31e{-}3 \approx -21.04e{-}3$, which proves once again the correctness of the mathematical analysis and code implementation.

The overall number of iterations needed to achieve a given convergence level using either real or complex RPM is comparable, and it differs only because of the numerical transient during which all

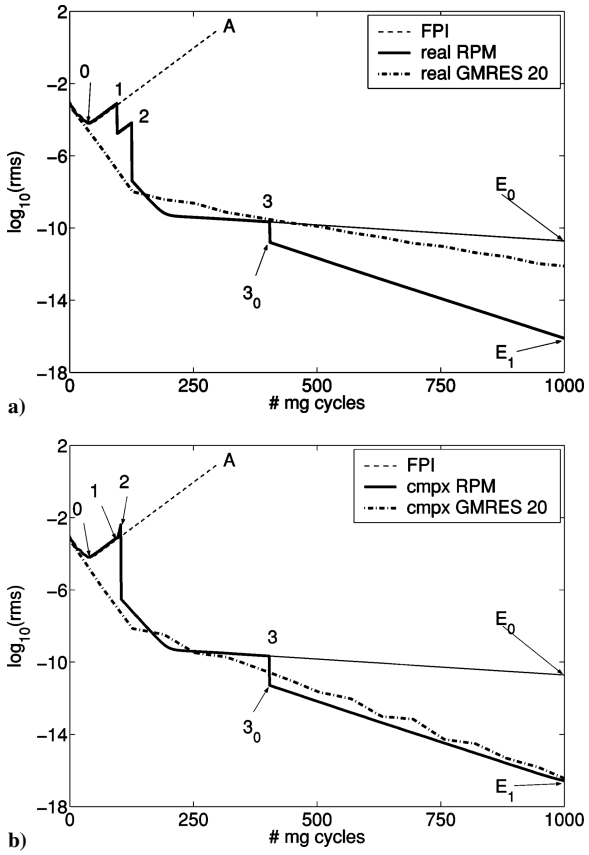


Fig. 5 Flutter analysis of two-dimensional turbine, IBPA = 180 deg: convergence history of a) real and b) complex RPM iterations.

outliers are appended to $\mathcal{P}$. The substantial difference between the real and complex solvers is one-that the additional memory required by the complex solver is one-half that of the real one. This is because the real system has twice as many unstable modes as its complex counterpart, as explained in Sec. IV. This memory saving can be crucially important for large three-dimensional problems when the computing resources are limited. In this test case, however, the difference between the memory of 62 MB of the complex solver and that of 66 MB of the real solver is quite small because the unstable space $\mathcal{P}$ contains only three dominant modes. On the other hand, Fig. 5b shows that the complex RPM solver allows one to achieve a convergence rate similar to that of complex GMRES 20 with a substantially lower memory allocation because the memory used by HYDLIN in the latter case is 87 MB. The cost of a multigrid cycle in the framework of the complex RPM code is about 0.3% higher than in the FPI-based code.

The first three dominant eigenvalues of $M^{-1}A$ have also been determined by examining the eigenmodes of the matrix H. (This technique is documented in Refs. 5 and 16.) These estimates are shown with an empty circle in Fig. 3b. The RPM eigenvalues are in very good agreement with those obtained with Arnoldi's method. The RPM estimates of the first three dominant eigenvectors of $M^{-1}A$ have also been found to be in excellent agreement with those provided in Fig. 4.

D. Nonlinear Unsteady Analysis

So far the hypothesis that the numerical instability of the harmonic solver is due to the limit cycle associated with the unstable separation bubble is only based on the inspection of the dominant eigenvectors of $M^{-1}A$. It has been proposed, however, that a more rigorous investigation of the relationship between the numerical instabilities and the small unsteadiness of the background flow should be based on the spectral analysis of the operator A, which does not include any preconditioning and, thus, corresponds exactly to the linearization of the nonlinear unsteady flow equations. Eventual unstable eigenmodes determined in this way would exactly correspond to physical instabilities of the nonlinear flowfield. Because these numerical instabilities appear to be independent of the IBPA, one could still use a single blade passage for the modal analysis, and the complex periodic boundary condition would thus reduce to the real one used for the nonlinear equations. The Arnoldi-based eigenmode analysis would be performed turning off all preconditioners, but unfortunately this would lead to an extremely poor convergence rate. Furthermore, the memory requirement would also become very high because of the large number of Krylov vectors one would have to use to achieve a good level of convergence. As explained in Ref. 5, in fact, this condition is required for an accurate estimate of the dominant eigenmodes.

An alternative and conceptually simpler approach is to investigate the purely aerodynamic unsteadiness by solving directly the nonlinear time-dependent flow equations with a motionless grid. This is the strategy we have adopted for the investigation, and despite that the numerical instabilities are independent of the IBPA of the blade vibration, two blade passages rather than a single one have been used for calculating the nonlinear time-dependent flowfield. This choice has been made to overcome possible mismatches between the linear and nonlinear codes arising from the lack of nonreflecting boundary conditions[19] in the latter one. This issue is carefully examined by Campobasso,[5] making use of the dispersive relation of the Euler equations. Reference 5 also summarizes the implementation of the dual time-stepping algorithm[20] used by the nonlinear flow solver HYD for the integration of the time-dependent flow equations.

Figure 6a shows the computational domain for the calculation of the nonlinear time-accurate flowfield associated with the transonic flow regime along with the positions of 16 points at which the unsteady flowfield is sampled. Note that the first eight points all belong to the first passage. The positions labeled 1 and 8 are close to the inflow and outflow boundaries, respectively, where the flowfield is expected to be fairly stable. On the other hand, the points labeled from 2 to 7 have been positioned in the regions where some unsteady flow phenomena could occur. As could be found by superimposing this map on the Mach contour plot of Fig. 1b, points 2 and 3 are at the front edge of the separation bubble, point 4 is at the rear edge, point 5 is on the shock, point 6 is at the intersection between the shock and the suction side boundary layer, and point 7 is behind the trailing edge. The other eight points labeled from 9 to 16 belong to the second passage, and their positions have been obtained by adding a blade pitch to the circumferential coordinate of the first eight points.

The flowfield determined by solving the nonlinear steady equations has been used as a starting solution of the time-accurate simulation with a motionless grid. The unsteady equations have been solved on an overall time interval of $2.0e{-}2$ s using 1024 intervals for the time integration. This integration time is that taken by eight cycles of the sought unsteady phenomena assuming a frequency of 400 Hz, which is nearly twice the vibration frequency of 212 Hz. This choice of the parameters for the unsteady calculation would give 128 intervals per period at the assumed frequency. Figure 6b shows the time-dependent variation of the static pressure with respect to its mean value at the eight probe locations in the first passage over the interval of integration. The only significant fluctuations occur at the front edge of the separation bubble (points 2 and 3), and some smaller instabilities are also visible at the rear edge (point 4). The maximum amplitude of these oscillations is of the order of 1 Pa and is thus substantially smaller than the mean pressure field that is of the order of $1.0e5$ Pa. The static pressure histories at the eight probe points in the second passage (not reported here for brevity) highlight the same pattern of unsteadiness observed in the first passage. Note also that the time step used for this calculation is not small enough to deliver a fully time-resolved solution, and this is a symptom of the fact that the frequency of the aerodynamic unsteadiness is higher than the assumed frequency of 400 Hz and, therefore, is even higher than the mechanical frequency of vibration.

These results confirm the physical origin of the numerical instabilities of the FPI-based linear code. Furthermore, the nonlinear analysis corroborates the assumption that the amplitude of the

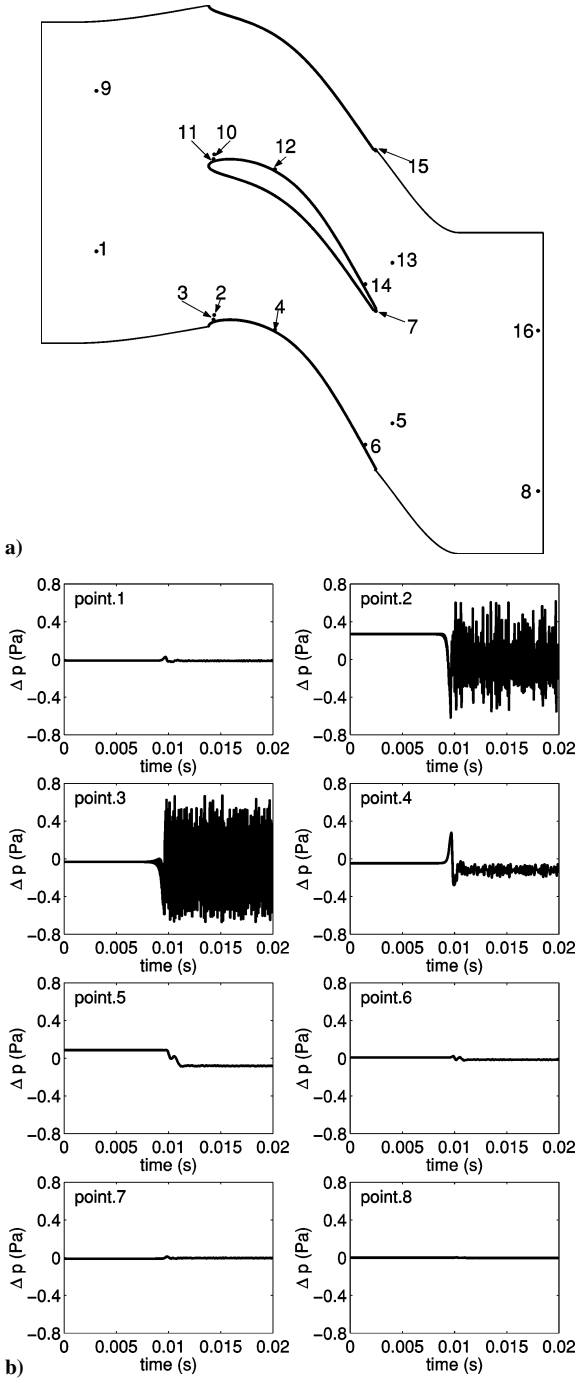


Fig. 6 Unsteady flow analysis of two-dimensional turbine section: a) computational domain of nonlinear time-accurate calculation and positions at which unsteady flow is sampled and b) time-dependent pressure at positions from 1 to 8 (first passage).

oscillatory phenomena causing the numerical instabilities is small, and it also reveals that their frequency is substantially different from that of the unsteady flow associated with the blade vibration. In these circumstances the effects of the background unsteadiness on the output of engineering interest can therefore be neglected.

Note, finally, that these unsteady analyses are quite lengthy, despite that they are two-dimensional. The time-dependent calculation reported in this section has required nearly 3 days of CPU time using eight processors of the OCCF computer cluster.

VI. Three-Dimensional Fan Rotor

A. Real and Complex GMRES

The second test case we consider is the three-dimensional fan rotor of a civil turbofan engine with 26 blades. The linear flutter

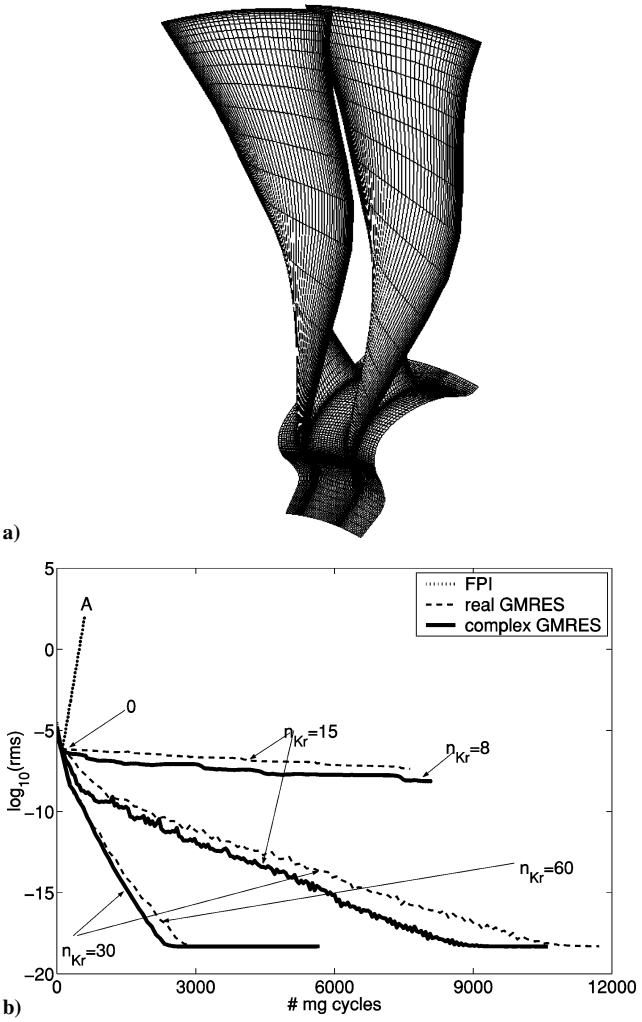


Fig. 7 Three-dimensional fan rotor: a) blade geometry and surface mesh and b) convergence histories of complex and real GMRES solvers (IBPA = 180 deg and $n_{cl} = 3$).

analysis of the first flap mode for four points of a constant-speed working line is reported in Ref. 7, in which this test case is used to demonstrate the effectiveness of the real GMRES solver for a typical industrial problem. The calculations of the unsteady flowfield of this rotor linearized about any of the four working conditions failed to converge for all IBPA when the FPI (4) was used, but convergence could be restored by using the real GMRES solver. Here we consider again this test case to test the complex GMRES algorithm.

The surface mesh of two blades and the hub end wall is shown in Fig. 7a. This grid has only 157,441 nodes and is quite coarse, but similar observations to those reported in this section have also been made adopting finer computational meshes and analyzing similar test cases. Because of the grid coarseness in the wall proximity, wall functions have been used to resolve the wall boundary layers.

The convergence histories of Fig. 7b refer to the calculation of the linear harmonic flowfield with IBPA = 180 deg based on a near-stall flow regime (working condition labeled D in Fig. 9a of Ref. 7). Four grid levels have been used for both the nonlinear and the linear calculations, and all other numerical control parameters are also the same in either case. The convergence history of HYD (not reported here) showed that the residuals of the nonlinear equations drop by more than three orders of magnitude and then end up in a low-amplitude limit cycle. The residuals of the linear FPI, however, grow exponentially. This is pointed out by the curve labeled FPI in Fig. 7b, which shows again that convergence is retrieved by using GMRES. The better numerical performance of the complex GMRES solver over that of its real counterpart is also observed in this test case. The GMRES residual histories of Fig. 7b, in fact, have been

Table 2 Flutter analysis of three-dimensional fan rotor: memory required by GMRES solver for various n_{Kr}

n_{Kr}	Memory, MB	Additional memory, %
8	571	29
15	672	52
30	888	101
60	1320	199

obtained by using either implementation for various values of n_{Kr} and $n_{\text{cl}} = 3$. As expected, the ratio between the number of Krylov vectors needed to achieve the same asymptotic convergence rate σ using either the real or the complex solver is about two due to the duplication of the eigenmodes. Thus, the complex algorithm always needs significantly fewer multigrid cycles to go to convergence for a given number of Krylov vectors.

Similarly to the turbine test case, the asymptotic convergence rate σ of both the real and complex GMRES algorithm increases monotonically with n_{Kr}. The price for the code stabilization and the convergence speed up is once more the additional memory burden. The memory requirement of the GMRES code for four values of n_{Kr} is reported in Table 2. The second column provides the overall memory allocation of HYDLIN corresponding to the values of n_{Kr} given in the first column. The additional memory with respect to the reference value of 441 MB that the FPI-based code would require are provided in the third column as percentage increments. Note that the use of GMRES with 30 Krylov vectors already requires twice as much memory as the FPI. The CPU time for performing a given number of multigrid cycles using complex GMRES with $n_{\text{cl}} = 3$ and $10 \leq n_{Kr} \leq 30$ is only from 0.82 to 1.37% higher than using the FPI (4). All of the linear calculations of this test case have been run using eight processors of the OCCF computer cluster, and the execution of 100 multigrid cycles with $n_{Kr} = 30$ required about 1.5 h.

B. Spectral Analysis

The first 150 dominant eigenmodes of $M^{-1}A$ determined using Arnoldi's method with $n_{\text{cl}} = 1$ are plotted in the complex plane of Fig. 8a. The first six eigenvalues are labeled from 1 to 6 in order of decreasing distance from the center of the unit disk in the four enlarged views of Fig. 8b. Note that the first four are outliers, and these are the modes causing the instability of the FPI (4). In fact, inserting the data relative to the slope σ_{OA} of the ascending branch OA of the FPI residual curve (Fig. 7b) and the spectral radius of $(I - M^{-1}A)$ (radius of outlier 1) into Eq. (12) yields $39.60e{-}3 \approx 40.18e{-}3$, which confirms the correctness of the mathematical analysis.

Both the Arnoldi (see Refs. 5 and 7) and the RPM[5,16] modal analysis show that the eigenvectors associated with the complex eigenvalues 1 and 2 correspond to a mild hub corner stall, whereas those associated with the pairs 3 and 4 correspond to a separation bubble on the suction side close to the leading edge in the hub region. Therefore the numerical instabilities of the standard linear iteration are due to the fact that the linearization of the unsteady equations is performed around a base flow containing traces of these two unsteady phenomena. The eigenmode corresponding to the complex conjugate pair 5 takes nonzero values both at the same locations as the first four and in the proximity of a shock on the suction side close to the blade tip. Similarly to the turbine test case, the dominant eigenmodes just described have been found to be independent of the IBPA, and this might be due again to their high spatial localization.

C. Real and Complex RPM

The convergence histories of the real and complex RPM solvers performing one multigrid iteration per stabilized iteration ($n_{\text{cl}} = 1$) are provided in Figs. 9a and 9b, respectively. The residual curves of the FPI-based code and the GMRES solver using $n_{Kr} = 30$ and $n_{\text{cl}} = 3$ are also shown for completeness. The discontinuities of the RPM convergence curves at the iterations labeled from 1 to 4 in Figs. 9a and 9b occur when the first four dominant modes, that is, the four outliers with the same labels in Figs. 8b, are included in

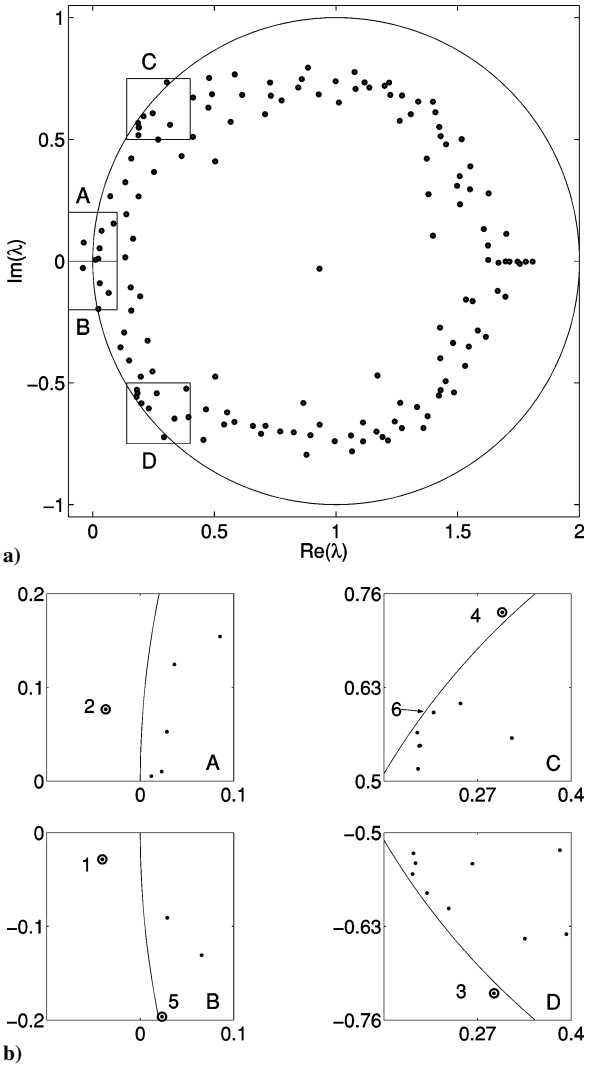


Fig. 8 Flutter analysis of three-dimensional fan rotor, IBPA = 180 deg: a) first 150 dominant eigenvalues of $M^{-1}A$ and b) enlarged views with outliers.

the unstable eigenspace $\mathcal{P}$. The slope of the branches $5E_0$ is determined by the spectral radius of the projection of the preconditioned operator onto the stable subspace $\mathcal{Q}$, which is the distance of the eigenvalue 5 from the center of the unit disk (Fig. 8a): This value is the same for both the real and complex operators and, for this reason, the slope of the branch $5E_0$ is the same in the real and complex convergence plots. The same remarks apply to the branches 5_0E_1: The convergence speed up occurring at the iteration labeled 5 takes place when the dominant eigenvalue 5 is also appended to $\mathcal{P}$. The branches 5_0E_1 are steeper than the branches $5E_0$ because the spectral radius of the RPM iteration is now determined by the eigenvalue 6, which is closer than the eigenvalue 5 to the center of the unit disk. Inserting in Eq. (12) the computed data relative to the slope of the branch 5_0E_1 and the radius of the eigenvalue 6 yields $-8.28e{-}3 \approx -8.15e{-}3$, which demonstrates once again the correctness of the mathematical models presented in the preceding sections. The overall number of iterations needed to achieve a given convergence level using either real or complex RPM is comparable, and it differs only because of the numerical transient during which all outliers are appended to $\mathcal{P}$. The only difference between the real and complex solvers is again that the additional memory required by the complex solver is one-half that needed by the real solver. In this case, the total memory used by the real and complex RPM-based HYDLIN are 615 and 528 MB, respectively. However, more importantly, Fig. 9b shows that the complex RPM solver allows one to achieve a convergence rate similar to that of complex GMRES 30 with a substantially lower memory allocation because the GMRES code required 888 MB. The

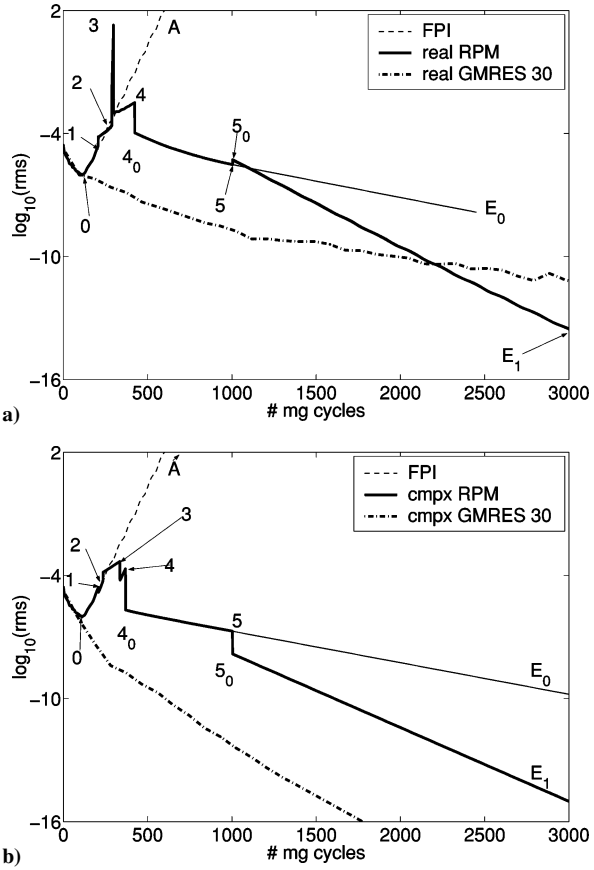


Fig. 9 Flutter analysis of three-dimensional fan rotor, IBPA = 180 deg: convergence history of a) real and b) complex RPM iterations.

cost of a multigrid cycle in the framework of the complex RPM code is about 1% higher than in the FPI-based code.

The first five dominant eigenvalues in Fig. 8b are also marked with an empty circle. This symbol denotes the estimates of the least stable modes of $M^{-1}A$ based on the eigenvalues of the matrix H. The RPM eigenvalues are in very good agreement with the first five eigenvalues determined with Arnoldi's method. Similarly, the first five dominant eigenvectors of $M^{-1}A$ have been computed using the eigenvectors of H, and they have been found to be in excellent agreement with those determined using Arnoldi's method.

VII. Conclusions

The stabilization of an existing linear flow solver based on a preconditioned multigrid iteration had been achieved by using two alternative methods, GMRES and RPM. This paper has presented a comparative analysis of the real and complex implementations of the GMRES and RPM algorithms. The GMRES- and RPM-stabilized harmonic codes have been used to compute and analyze the unsteady flowfield due to the blade vibration of a two-dimensional turbine section and a three-dimensional fan rotor. A time-accurate nonlinear analysis of the flowfield of the former test case has also been carried out to provide further evidence of the relationship between the unsteady flow features of the background state and the numerical instabilities of the FPI.

The linear harmonic flow equations can be viewed either as a complex or a real augmented system. The use of the preconditioned multigrid iteration for solving either form is mathematically equivalent. Numerically, the real iteration needs fewer floating point operations because most elements of the linear operator arising from the linearization of the nonlinear unsteady equations are either purely real or purely imaginary. Therefore, the choice of real arithmetic for the implementation of the core part of the linear code performing the preconditioned FPI was made on the basis of the reduced number of operations and also the better code optimization achieved by FORTRAN compilers with real, rather complex arithmetic.

The same conclusions, however, do not hold when the restarted GMRES algorithm is used to solve either the complex system or its augmented real counterpart: For a given number of Krylov vectors per restarted cycle the number of preconditioned multigrid cycles required to drop the residuals by a given amount is substantially lower when GMRES is applied to the complex equations. In particular, the numerical investigations carried out so far show that the same convergence rate of the residuals can be obtained by using either the complex GMRES solver with a given number of Krylov vectors or the real solver with twice as many vectors. This is possibly because the eigenmodes of the complex system are duplicated through the complex conjugation when the real augmented system is considered. This feature plays a significant role for the application of the linear harmonic flow analysis to the daily engineering practice when the available computing resources are limited: The use of the complex solver allows one to halve the additional memory requirement for the Krylov vectors with negligible enhancement of the CPU time with respect to the real solver. In all test cases we have considered, the convergence rate increases with the number of Krylov vectors per restarted cycle for both the real and the complex GMRES algorithm. This observation, however, cannot be generalized because the opposite trend has also been observed by other researchers in some special problems.

The use of the complex RPM solver also leads to a halving of the additional memory usage because the real system has twice as many outliers as the complex counterpart.

For a given set of multigrid parameters, that is, for a given preconditioner, the asymptotic convergence rate of RPM depends on the spectral radius of the projection of the preconditioned linear operator onto the stable subspace $\mathcal{Q}$. Hence, a significant convergence speed up can be achieved by appending to the unstable subspace $\mathcal{P}$ not only the outliers but also the first few dominant modes in the unit circle. The extra memory allocation depends on the overall number of eigenmodes included in $\mathcal{P}$, which is greater or equal to the number of outliers. The additional memory required by a run in which five modes are included in $\mathcal{P}$ is about 20% that of the standard code. The convergence rate of the GMRES solver depends on the spectrum of $M_G^{-1}A$ (and, hence, on the number of multigrid cycles per GMRES step), but also on the number of Krylov vectors per restarted GMRES cycle. The extra memory allocation depends on the latter parameter, but not on the number of outliers. The use of GMRES 30 for a typical three-dimensional viscous problem yields an additional memory usage that is about twice that used by the standard multigrid iteration. Hence, the more convenient choice of the stabilizing solver in the presence of outliers depends on the number of unstable modes: The complex RPM solver should be used in problems with a small number of outliers (lying between 10 and 20), whereas the adoption of the complex GMRES solver is advisable for test cases with more unstable modes.

Acknowledgments

This research has been carried out in the framework of the GEODISE project supported by the Engineering and Physical Sciences Research Council under Grant GR/R67705/01. The permission of Rolls–Royce, plc, to publish results from the HYDRA codes is gratefully acknowledged. We also acknowledge the contributions of M. C. Duta, P. Moinier, J. D. Müller, L. Lapworth, and M. West to the development of the HYDRA codes and the very useful discussions with A. Wathen and M. Embree on the properties of RPM and GMRES.

References

[1]Campobasso, M., and Giles, M., "Analysis of the Effect of Mistuning on Turbomachinery Aeroelasticity," *Proceedings of the 9th International Symposium on Unsteady Aerodynamics, Aeroacoustics and Aeroelasticity in Turbomachines*, Presses Universitaires de Grenoble, Grenoble, France, 2001, pp. 885–896.

[2]Hall, K., "Deforming Grid Variational Principle for Unsteady Small Disturbance Flows in Cascades," *AIAA Journal*, Vol. 31, No. 5, 1993, pp. 891–900.

[3]Breard, C., Vahdati, M., Sayma, A., and Imregun, M., "An Integrated Time-Domain Aeroelasticity Model for the Prediction of Fan Forced Respomse due to Inlet Distorsion," *Journal of Engineering for Gas Turbines and Power*, Vol. 124, No. 1, 2002, pp. 196–208.

[4]Hall, K., and Crawley, E., "Calculation of Unsteady Flows in Turbomachinery Using the Linearized Euler Equations," *AIAA Journal*, Vol. 27, No. 6, 1989, pp. 777–787.

[5]Campobasso, M. S., "Effects of Flow Instabilities on the Linear Harmonic Analysis of Unsteady Flow in Turbomachinery," Ph.D. Dissertation, Computing Lab., Univ. of Oxford, Oxford, Aug. 2004; URL: http://web.comlab.ox.ac.uk/oucl/work/mike.giles/theses.html [cited 15 Jan. 2005].

[6]Moinier, P., Müller, J., and Giles, M., "Edge-Based Multigrid and Preconditioning for Hybrid Grids," *AIAA Journal*, Vol. 40, No. 10, 2002, pp. 1954–1960.

[7]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.

[8]Spalart, P., and Allmaras, S., "A One-Equation Turbulence Model for Aerodynamic Flows," *La Recherche Aerospatiale*, Vol. 1, 1994, pp. 5–21.

[9]Moinier, P., "Algorithm Developments for an Unstructured Viscous Flow Solver," Ph.D. Dissertation, Computing Lab., Univ. of Oxford, Oxford, Dec. 1999.

[10]Duta, M. C., "The Use of the Adjoint Method for the Minimization of Forced Response," Ph.D. Dissertation, Computing Lab., Univ. of Oxford, Oxford, March 2002.

[11]Ning, W., and He, L., "Some Modeling Issues on Trailing-Edge Vortex Shedding," *AIAA Journal*, Vol. 39, No. 5, 2001, pp. 787–793.

[12]Bassi, F., Crivellini, A., Rebay, S., and Savini, M., "Discontinuous Galerkin Solution of the Reynolds Averaged Navier–Stokes and k–ω Turbulence Model Equations," *Computers and Fluids*, Vol. 34, No. 4–5, 2005, pp. 507–540.

[13]März, J., Hah, C., and Neise, W., "An Experimental and Numerical Investigation into the Mechanisms of Rotating Instability," *Journal of Turbomachinery*, Vol. 124, No. 3, 2002, pp. 367–375.

[14]Saad, Y., and Schultz, M., "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM Journal on Scientific and Statistical Computing*, Vol. 7, No. 3, 1986, pp. 856–869.

[15]Shroff, G., and Keller, H., "Stabilization of Unstable Procedures: The Recursive Projection Method," *SIAM Journal of Numerical Analysis*, Vol. 30, No. 4, 1993, pp. 1099–1120.

[16]Campobasso, M. S., and Giles, M. B., "Stabilization of Linear Flow Solver for Turbomachinery Aeroelasticity Using Recursive Projection Method," *AIAA Journal*, Vol. 42, No. 9, 2004, pp. 1765–1774.

[17]Campobasso, M., and Giles, M., "Stabilization of a Linearized Navier–Stokes Solver for Turbomachinery Aeroelasticity," *Computational Fluid Dynamics 2002*, Springer-Verlag, Berlin, 2003, pp. 343–348.

[18]Fransson, T., Joeker, M., Boelcs, A., and Ott, P., "Viscous and Inviscid Linear/Nonlinear Calculations versus Quasi Three-Dimensional Experimental Cascade Data for a New Aeroelastic Turbine Standard Configuration," *Journal of Turbomachinery*, Vol. 121, No. 4, 1999, pp. 717–725.

[19]Giles, M., "Nonreflecting Boundary Conditions for Euler Equation Calculations," *AIAA Journal*, Vol. 28, No. 12, 1990, pp. 2050–2058.

[20]Jameson, A., "Time-Dependent Calculations Using Multi-Grid, with Applications to Unsteady Flows Past Airfoil and Wings," AIAA Paper 91-1596, 1991.

K. Ghia
Associate Editor