

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

M. Sergio Campobasso*

School of Engineering, Cranfield University, MK43 0AL Cranfield, United Kingdom

Michael B. Giles†

Computing Laboratory, Oxford University, OX1 3QD Oxford, United Kingdom

The linear flow analysis of turbomachinery aeroelasticity views the unsteady flow as the sum of a background nonlinear flow field 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 which 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 which 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. This paper summarizes the implementation of

*Senior Research Fellow, AIAA member

†Professor of Scientific Computation, AIAA Member

the complex GMRES and RPM algorithms which 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 paper also reports the results of a nonlinear unsteady calculation which further emphasizes the physical origin of the numerical instabilities.

Nomenclature

A	coefficient matrix of complex system
\mathbf{b}	right-hand-side of complex linear system
\mathbf{F}	right-hand-side of Picard iteration
$F_{\mathbf{x}}$	Jacobian of \mathbf{F}
\mathbf{f}	projection of \mathbf{F} onto \mathcal{P}
\mathbf{g}	projection of \mathbf{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
\mathbf{p}	projection of \mathbf{x} onto \mathcal{P}
\mathcal{Q}	orthogonal complement of \mathcal{P}
\mathbf{q}	projection of \mathbf{x} onto \mathcal{Q}
V_m	matrix of m Krylov vectors
\mathbf{v}_m	m^{th} Krylov vector
\mathbf{v}	eigenvector of $M^{-1}A$
\mathbf{x}	vector unknown of complex linear equations
Z	orthonormal basis of \mathcal{P}
\mathbf{z}	vector of current coordinate variables of \mathbf{p} in Z

Symbols

λ	eigenvalue of $M^{-1}A$
ρ	spectral radius
σ	asymptotic convergence rate
ζ	vector of updated coordinate variables of \mathbf{p} in Z

I. Introduction

BLADE flutter and forced response of turbomachinery blade-rows¹ are aeroelastic phenomena which may both lead to dramatic mechanical failures if not properly accounted for in the design of an aero-engine. Flutter occurs if the working fluid feeds energy into the vibrating blades, while 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 flow field. The methods for the analysis of turbomachinery aeroelasticity vary from uncoupled linearized potential flow solvers² 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 methods³ 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) methods⁴ 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 reference⁵ and briefly sketched below, 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. Due to the cyclic periodicity of the steady flow, the nonlinear equations are solved on a computational domain consisting of a single blade passage. The unsteady flow field 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. Assuming 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 *Inter-Blade 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 pre-processing 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 reference⁵ 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 nonlinear⁶ and a linear harmonic⁷ Euler/NS solver, both using the one-equation Spalart-Allmaras turbulence model⁸ for closure in the case of turbulent flows (the validation of these codes is documented in references^{6,7,9,10,5}). 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 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,^{7,5} which also document the successful implementation of a restarted GMRES algorithm¹⁴ 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. In order 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.^{16,5} 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 *a)* summarize the implementation of the complex GMRES and RPM solvers in HYDLIN and *b)* demonstrate the significant improvement of the computational performance achieved by their use. Section II presents the real and complex FPI's in the framework of the linear computational fluid dynamics (CFD) solver HYDLIN, and addresses the mathematical and numerical implications of the real and complex viewpoints. In section 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 in section IV. The enhanced computational performance of the complex solvers is then demonstrated in sections 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. Section V also presents the results 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 section VII.

II. Linear equations and fixed-point iteration

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

$$A\mathbf{x} = \mathbf{b} . \tag{1}$$

The operator A depends on the Jacobian of the nonlinear flow equations, the constant vector \mathbf{b} depends on the source of unsteadiness (blade motion in flutter applications and incoming gust in forced response problems) and the complex unknown \mathbf{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 one is equal to 6 if a one-equation turbulence model is adopted. The construction of the linearized Euler and NS equations represented by equation (1) is thoroughly documented in reference.⁵ This linear system could be conveniently solved by means of the *complex* Picard

iteration:

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

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 part, 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:

$$A'\mathbf{x}' = \mathbf{b}', \quad (3)$$

with

$$A' = \begin{pmatrix} \Re(A) & -\Im(A) \\ \Im(A) & \Re(A) \end{pmatrix} \quad \mathbf{x}' = \begin{pmatrix} \Re(\mathbf{x}) \\ \Im(\mathbf{x}) \end{pmatrix} \quad \mathbf{b}' = \begin{pmatrix} \Re(\mathbf{b}) \\ \Im(\mathbf{b}) \end{pmatrix}.$$

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

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

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 \mathbf{x}_n of dimension k is stored in the real vector \mathbf{x}'_n of dimension $2k$. Furthermore the numerical execution of (4) requires fewer floating point operations than that of (2) would do 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, as the code determines the matrix-vector products $M'^{-1}A'\mathbf{x}'$ 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 pseudo-code of the preconditioned multigrid iteration looks like:

$$\begin{aligned}
\mathbf{x}' &= \mathbf{x}'_{start} \\
\mathbf{x}' &= \text{MG}(A', \mathbf{x}', \mathbf{b}', n_{cl}) \\
\mathbf{x}'_{finish} &= \mathbf{x}'
\end{aligned}$$

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

The preconditioner M'^{-1} consists of a 5-stage Runge-Kutta algorithm, a Jacobi preconditioner and a geometric multigrid scheme. Since 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 references¹⁰ and⁵.

III. GMRES stabilization

Linear stability analysis of the iteration (4) shows that a necessary condition for its convergence is that all the eigenvalues of $(I - M'^{-1}A')$ lie within the unit circle centered at the origin in the complex plane or equivalently that all the eigenvalues of $M'^{-1}A'$ lie in the unit disc 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,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 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 (λ, \mathbf{v}) be an

eigenpair of $M^{-1}A$. Then

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

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

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

or

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

Since $M'^{-1}A'$ is real, taking the complex conjugate of the equation above yields:

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

Thus we see that if (λ, \mathbf{v}) is an eigenpair of $M^{-1}A$, then $\left(\lambda, \begin{bmatrix} \mathbf{v} \\ -i\mathbf{v} \end{bmatrix}\right)$ is the corresponding eigenpair of $M'^{-1}A'$. However, $(\bar{\lambda}, \bar{\mathbf{v}})$ is *not* necessarily an eigenpair of $M^{-1}A$, though $\left(\bar{\lambda}, \begin{bmatrix} \bar{\mathbf{v}} \\ i\bar{\mathbf{v}} \end{bmatrix}\right)$ is an eigenpair of $M'^{-1}A'$.

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

$$M_G^{-1}AV_m = V_{m+1}\tilde{H}_m, \quad (6)$$

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 \mathbf{v}_j ($j = 1, \dots, m$) form an orthonormal basis for the Krylov subspace K_m , V_{m+1} is V_m augmented with a new Krylov vector \mathbf{v}_{m+1} and M_G^{-1} is a suitable preconditioner whose dependence on M^{-1} is defined below. Denoting by $h_{j,m}$ ($j = 1, \dots, m$) the elements of the m^{th} column of \tilde{H}_m , the m^{th} column of equation (6) can be written as:

$$M_G^{-1}A\mathbf{v}_m = h_{1,m}\mathbf{v}_1 + \dots + h_{m+1,m}\mathbf{v}_{m+1}, \quad (7)$$

which shows that \mathbf{v}_{m+1} satisfies an $(m+1)$ -term recursive relation involving itself and the previous m Krylov vectors. At the m^{th} GMRES iteration the solution of (1) is approximated by the linear combination of the m \mathbf{v}_j 's which minimizes the 2-norm of the residual $\mathbf{r}_m = M_G^{-1}(\mathbf{b} - A\mathbf{x}_m)$ and is thus given by $\mathbf{x}_m = \mathbf{x}_{start} + V_m \mathbf{t}_m$, in which \mathbf{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 $M_G^{-1}A\mathbf{v}_j$. The computationally cheap minimization is also performed at the top routine level and the pseudo-code of the main HYDLIN using GMRES is:

```

 $\mathbf{v}_1 = \text{MG}(A, \mathbf{x}_{start}, \mathbf{b}, n_{cl}) - \mathbf{x}_{start}; \quad \mathbf{v}_1 = \mathbf{v}_1 / \|\mathbf{v}_1\|$ 
for  $m = 1 : n_{Kr}$ 
   $M_G^{-1}A\mathbf{v}_m = \mathbf{v}_m - \text{MG}(A, \mathbf{v}_m, \mathbf{0}, n_{cl})$ 
   $\mathbf{v}_{m+1}$  from equation (7);  $\mathbf{v}_{m+1} = \mathbf{v}_{m+1} / \|\mathbf{v}_{m+1}\|$ 
  determine  $\mathbf{t}_m$  which minimizes  $\|\mathbf{r}_m\|$ 
end
 $\mathbf{x}_{finish} = \mathbf{x}_{start} + V_{n_{Kr}} \mathbf{t}_{n_{Kr}}$ 

```

The first Krylov vector \mathbf{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. Since the iteration (2) is applied n_{cl} times at each GMRES iteration, the relationship between M_G^{-1} and M^{-1} is

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

The choice $n_{cl}=1$ 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.^{7,5} Note that the right-hand-side of (1) is set to zero before using MG to determine $M_G^{-1}A\mathbf{v}_m$. The value of n_{Kr} required for full convergence is much smaller than k , but nevertheless very big with respect to the computing resources usually available. This problem is overcome using the *restart* option, that is performing n_{Kr} iterations and re-starting GMRES from the updated solution re-computing from there a new set of n_{Kr} Krylov vectors. This is achieved by wrapping the inner loop described above with an outer one which 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 $n_{cl} = 3$.

In section II, it has been 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 $M_G^{-1}A\mathbf{v}_m$ 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 $M_G^{-1}A\mathbf{v}'_m$ 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 one. 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 $M_G^{-1}A\mathbf{v}_m$. Including the extra CPU-time required for the Arnoldi orthogonalizations and the solution of the optimization problem in the cost of one multigrid cycle, the CPU-time for executing a given number of multigrid cycles using complex GMRES with $n_{cl} = 3$ and $10 \leq n_{Kr} \leq 30$ 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 the test cases analyzed so far have highlighted that the complex solver has a substantially better numerical performance than its real counterpart, as 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. In order to stabilize the linear code reducing the additional memory requirement, the complex RPM solver has also been implemented in HYDLIN. Its real counterpart had already been implemented and it had 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 below.

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 equation (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. Denoting by Z 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. Denoting by \mathbf{F} 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:

$$\mathbf{f} = P\mathbf{F} = P[(I - M^{-1}A)\mathbf{x} + M^{-1}\mathbf{b}], \quad \mathbf{g} = Q\mathbf{F} = Q[(I - M^{-1}A)\mathbf{x} + M^{-1}\mathbf{b}],$$

and the stabilized iteration can be written as:

$$\mathbf{p}_{\nu+1} = \mathbf{p}_{\nu} + (I - f_{\mathbf{p}})^{-1}(\mathbf{f}(\mathbf{p}_{\nu}, \mathbf{q}_{\nu}) - \mathbf{p}_{\nu}) \quad (9)$$

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

where

$$\mathbf{p} = P\mathbf{x}, \quad \mathbf{q} = Q\mathbf{x}, \quad f_{\mathbf{p}} \equiv PF_{\mathbf{x}}P, \quad F_{\mathbf{x}} = I - M^{-1}A.$$

Note that equation (9) requires the inversion of $(I - f_{\mathbf{p}}) : \mathcal{P} \rightarrow \mathcal{P}$, which is the restriction of $(I - F_{\mathbf{x}})$ to the subspace \mathcal{P} , and is therefore equivalent to $(I - f_{\mathbf{p}})P$. It is easily verified that

$$(I - f_{\mathbf{p}})^{-1}P = Z[I - Z^H(I - M^{-1}A)Z]^{-1}Z^H = Z[I - H]^{-1}Z^H \quad (11)$$

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 \mathbf{q}_{ν} of the modified iteration (10), without computing Jacobians. This is done by monitoring the rate of convergence of the iterates \mathbf{q}_{ν} . If the residual starts growing, it is argued that some of the eigenvalues of $g_{\mathbf{q}} = QF_{\mathbf{x}}Q$ 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 references^{15,16} 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 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, as 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, as 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 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 \mathbf{z} of length l whose elements are the coordinate variables for the representation of \mathbf{p} in the basis Z :

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

from which it follows that

$$\mathbf{p} = Z\mathbf{z} \quad \text{and} \quad \mathbf{x} = Z\mathbf{z} + \mathbf{q}.$$

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

$$\mathbf{z}_{\nu+1} = \mathbf{z}_{\nu} + (I - H)^{-1} (Z^H \mathbf{F}(\mathbf{x}_{\nu}) - \mathbf{z}_{\nu}).$$

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 which is used to determine the projection \mathbf{q} of the solution \mathbf{x} onto \mathcal{Q} (equation (10)). The two subvectors of length k associated with the first k elements of $M_G^{-1}A'\mathbf{x}'_\nu$ 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 pseudo-code of the main HYDLIN using complex RPM is:

```

Z = [ ];  l = 0;  ν = 0;  x_ν = x_start
while || b - Ax_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} - Zζ_ν
    x_{ν+1} = Zz_{ν+1} + q_{ν+1}
  endif
  ν = ν + 1
  % increase basis size
  if not converging
    Z = [Z d];  l = l + 1;;  ν = 0
    for j = 1 : l
      H(:, j) = Z^H MG(A, Z(:, j), 0, n_cl)
    end
  endif
end
x_finish = q_{ν+1} + Zz_{ν+1}

```

The section of the pseudo-code labeled 'stabilized iteration' corresponds to the implementation of equations (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 \mathbf{d} containing the eigenmode associated with the current outlier is

appended to Z . Thereafter the subroutine MG is used to determine the columns of $F_{\mathbf{x}}Z$ by setting $\mathbf{b}=0$ and performing n_{cl} multigrid cycles on each column of Z . Then the matrix H is updated according to equation (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 versus number of multigrid cycles' and its mathematical definition is:

$$\sigma = \frac{\Delta(\log(rms))}{N_{MG}},$$

where rms is the root-mean-square of the nodal residuals of the linearized flow equations and Δ denotes its variation over N_{MG} multigrid cycles. The paper by Shroff and Keller¹⁵ 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 of reference⁵ 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 $n_{cl} \leq 10$ and increases for $n_{cl} > 10$ 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, as 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 above 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 $n_{cl} = 1$, 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 $2k \times (l + 1) \times v_m$, in which the factor 1 accounts for a new work array and v_m is the memory required for storing a single scalar. By comparison, the extra memory allocation required by GMRES is $2k \times (n_{Kr} + 1) \times v_m$ and is independent of the number of outliers. All codes of the HYDRA suite have been implemented using double-precision, and in this circumstance $v_m = 8$ Bytes.

V. Two-dimensional turbine section

A. real and complex GMRES

The two-dimensional turbine section of the 11th Standard Configuration is the mid-span blade-to-blade section of an annular turbine cascade with 20 blades. The test rig and cascade geometry are described in reference,¹⁸ which also provides experimental measurements and various computed results of the steady and unsteady flow field 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 *hz* and 212 *hz* respectively. This test case had already been used to demonstrate the predictive capabilities of HYDLIN and test the real GMRES solver,⁷ and also 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 direction, for a total of 17745 grid nodes. A preliminary mesh-refinement analysis carried out using a coarser 7869-node (183×43) and finer 39673-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 figure 1-a, whereas figure 1-b depicts 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 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 iteration⁶ 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 *IBPA*'s. This is emphasized by the convergence history labeled 'FPI' in figure 2, which refers to the calculation of the harmonic flow field with $IBPA = 180^\circ$. 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 *rms*, the root-mean-square 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 root-mean-square of the unpreconditioned residuals). The exponential growth of the residuals could be removed neither by lowering the CFL number nor by changing the grid topology and refinement. Other attempts to suppress the numerical instabilities included: *a*) varying the multigrid control parameters and *b*) 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 presents the logarithm of *rms* using both the complex and the real implementation with various values of n_{Kr} and $n_{cl} = 3$. This plot 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 this figure 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 2 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 one results by 'doubling' the former through the 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 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 reference,⁵ whereas the GMRES convergence histories of figure 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 require-

ments of the FPI and complex GMRES codes are compared in table 1. The second column reports the memory used by the linear code in Mbytes. The first entry of the column is the memory used by the FPI-based code, whereas the following 5 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. It should be noted 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 it 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 (using 40 Krylov vectors, 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 figure 2 can also be viewed as a scaled CPU-time. All 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 8 processors of the OCCF computer cluster consisting of 24 four-processor Sun Ultra-80 nodes, with a Sun Blade-1000 front-end. Using this set-up, the execution of 1000 multigrid cycles with $n_{Kr}=20$ is about 38 minutes.

B. spectral analysis

In order 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 references⁷ and,⁵ and this required performing 150 GMRES iterations ($n_{Kr}=150$) with $n_{cl} =$

n_{Kr}	mem. (Mb)	ex. mem. (%)	CPU t. (s)	ex. CPU t. (%)
–	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

Table 1. Flutter analysis of the 2D turbine: memory and CPU time required by the complex GMRES solver for various n_{Kr} and $n_{cl}=3$.

1. This choice of n_{cl} allows one to determine the eigenmodes of $M^{-1}A$ since it leads to the equality $M_G^{-1}A = M^{-1}A$, which follows from equation (8). The first 150 dominant eigenvalues are plotted in the complex plane of figure 3-a and the first four dominant eigenvalues are labeled from 1 to 4 in order of decreasing distance from the center of the unit disc in the three enlarged views of figure 3-b. Note that the first two eigenvalues are complex outliers and they 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 (figure 2) and the spectral radius of $(I - M^{-1}A)$ (radius of the outlier 1) into equation (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 reported in figures 4-a and 4-b respectively. More precisely, these two figures depict 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 non-zero amplitudes only in a tiny neighborhood about it (figure 4-a). This points to the fact 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 4-b 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 non-zero 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 disc. 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 *IBPA*'s and it has been found that the first few dominant eigenmodes do not vary with the *IBPA* despite the fact that $M^{-1}A$ depends on it.^{7,5} This observation is in line with the fact that the standard FPI is unstable for all *IBPA*'s. 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 ($n_{cl} = 1$) are provided in figures 5-a and 5-b respectively. The residual curves of the FPI-based code and the GMRES solver using $n_{Kr} = 20$ and $n_{cl} = 3$ are also reported for completeness. The discontinuities of the RPM convergence curves at the iterations labeled 1 and 2 in both figures occur when the first two dominant modes (i.e. the two outliers with the same labels in figure 3-b) 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 disc in figure 3-a. 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 disc. Inserting in equation (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 outliers are appended to \mathcal{P} . The substantial difference between the real and complex solvers is that the additional memory required by the complex solver is half that of

the real one. This is due to the fact that the real system has twice as many unstable modes as its complex counterpart, as explained in section 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 Mbytes of the complex solver and that of 66 Mbytes of the real solver is quite small because the unstable space \mathcal{P} contains only 3 dominant modes. On the other hand, figure 5-b 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, as the memory used by HYDLIN in the latter case is 87 Mbytes. 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 references¹⁶ and⁵). These estimates are reported with an empty circle in figure 3-b. 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 figure 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 flow field. Since 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 in order to achieve a good level of convergence. As explained in⁵, 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 motionless grid. This is the strategy we have adopted for the investigation, and despite the fact 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 flow field. This choice has been made to overcome possible mismatches between the linear and nonlinear codes arising from the lack of non-reflecting boundary conditions¹⁹ in the latter one. This issue is carefully examined in the thesis of Campobasso⁵ making use of the dispersive relation of the Euler equations. This reference also summarizes the implementation of the dual time-stepping algorithm²⁰ used by the nonlinear flow solver HYD for the integration of the time-dependent flow equations.

Figure 6-a depicts the computational domain for the calculation of the nonlinear time-accurate flow field associated with the transonic flow regime along with the positions of 16 points at which the unsteady flow field is sampled. Note that the first 8 points all belong to the first passage. The positions labeled 1 and 8 are close to the inflow and outflow boundaries respectively, where the flow field 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 it could be found by superimposing this map on the Mach contour plot of figure 1-b, points 2 and 3 are at the front edge of the separation bubble and 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 8 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 8 points.

The flow field determined by solving the nonlinear steady equations has been used as starting solution of the time-accurate simulation with motionless grid. The unsteady equations have been solved on an overall time-interval of $2.0e-2$ seconds using 1024 intervals for the time-integration. This integration time is that taken by 8 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 6-b shows the time-dependent variation of the static pressure with respect to its mean value at the 8 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 Pascal and thus substantially smaller than the mean pressure field that is of the order of $1.0e5$ Pascal. The static pressure histories at the 8 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, 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 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.

It should be noted, finally, that these unsteady analyses are quite lengthy, despite the fact that they are two-dimensional. The time-dependent calculation reported in this section has required nearly 3 days of CPU-time using 8 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 analysis of the first flap mode for four points of a constant-speed working line is reported in the article,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 flow field of this rotor linearized about any of the four working conditions failed to converge for all *IBPA*'s 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 figure 7. This grid has only 157441 nodes and is quite coarse, but similar observations to those reported in this section have also been made adopting finer computational meshes and analyzing other

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 figure 7-b refer to the calculation of the linear harmonic flow field with $IBPA=180^\circ$ based on a near-stall flow regime (working condition labeled 'D' in figure 9-a of reference⁷). 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) show 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 figure 7-b, 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 figure 7-b, in fact, have been obtained by using either implementation for various values of n_{Kr} and $n_{cl}=3$. 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 2 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 Mbytes 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 $n_{cl}=3$ and $10 \leq n_{Kr} \leq 30$ is only from 0.82 to 1.37% higher than using the FPI (4). All the linear calculations of this test case have been run using 8 processors of the OCCF computer cluster and the execution of 100 multigrid cycles with $n_{Kr}=30$ has required about 1.5 hours.

n_{Kr}	mem. (Mb)	ex. mem. (%)
8	571	29
15	672	52
30	888	101
60	1320	199

Table 2. Flutter analysis of the 3D fan rotor: memory required by GMRES solver for various n_{Kr} .

B. spectral analysis

The first 150 dominant eigenmodes of $M^{-1}A$ determined using Arnoldi’s method with $n_{cl} = 1$ are plotted in the complex plane of figure 8-a. The first six eigenvalues are labeled from 1 to 6 in order of decreasing distance from the center of the unit disc in the four enlarged views of figure 8-b. 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 residual curve (figure 7-b) and the spectral radius of $(I - M^{-1}A)$ (radius of outlier 1) into equation (12) yields $39.60e-3 \approx 40.18e-3$, which confirms the correctness of the mathematical analysis.

Both the Arnoldi^{5,7} 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. The numerical instabilities of the standard linear iteration are therefore due to the fact that the linearization of the unsteady equations is performed about of a base flow containing traces of these two unsteady phenomena. The eigenmode corresponding to the complex conjugate pair 5 takes non-zero values both at the same locations as the first 4 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 described above 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 ($n_{cl} = 1$) are provided in figures 9-a and 9-b respectively. The residual curves of the FPI-based code and the GMRES solver using $n_{Kr} = 30$ and $n_{cl} = 3$ are also reported for completeness. The discontinuities of the RPM convergence curves at the iterations labeled from 1 to 4 in both figures occur when the first four dominant modes (i.e. the four outliers with the same labels in figures 8-b) are included in 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 disc (figure 8-a). 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 disc. Inserting in equation (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 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 Mbytes respectively. But more importantly, figure 9-b 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, as the GMRES code required 888 Mbytes. The 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 figure 8-b 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 flow field due to the blade vibration of a two-dimensional turbine section and a three-dimensional fan rotor. A time-accurate nonlinear analysis of the flow field 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 forms is mathematically equivalent. Numerically, the real iteration needs fewer floating point operations due to the fact that all 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 vector 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 due to the fact that 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 as the opposite trend has also been observed by other researchers in some special problems.²¹

The use of the complex RPM solver also leads to a halving of the additional memory usage, since 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 5 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 vector 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 which 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

¹Campobasso, 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, December 2001, pp. 885–896.

²Hall, K., “Deforming grid variational principle for unsteady small disturbance flows in cascades,” *AIAA Journal*, Vol. 31, No. 5, May 1993, pp. 891–900.

³Breard, C., Vahdati, M., Sayma, A., and Imregun, M., “An integrated time-domain aeroelasticity model for the prediction of fan forced response due to inlet distortion,” *Journal of Engineering for Gas Turbines and Power*, Vol. 124, January 2002, pp. 196–208.

⁴Hall, K. and Crawley, E., “Calculation of unsteady flows in turbomachinery using the linearized Euler equations,” *AIAA Journal*, Vol. 27, No. 6, May 1989, pp. 777–787.

⁵Campobasso, M. S., *Effects of Flow Instabilities on the Linear Harmonic Analysis of Unsteady Flow in Turbomachinery*, Ph.D. thesis, University of Oxford, United Kingdom, Aug. 2004, see <http://web.comlab.ox.ac.uk/oucl/work/mike.giles/theses.html>.

⁶Moinier, P., Müller, J., and Giles, M., “Edge-based multigrid and preconditioning for hybrid grids,” *AIAA Journal*, Vol. 40, No. 10, October 2002, pp. 1954–1960.

⁷Campobasso, M. and Giles, M., “Effects of Flow Instabilities on the Linear Analysis of Turbomachinery Aeroelasticity,” *Journal of Propulsion and Power*, Vol. 19, No. 2, March–April 2003, pp. 250–259.

⁸Spalart, P. and Allmaras, S., “A one-equation turbulence model for aerodynamic flows,” *La Recherche Aérospatiale*, Vol. 1, 1994, pp. 5–21.

⁹Moinier, P., *Algorithm developments for an unstructured viscous flow solver viscous flow solver*, Ph.D. thesis, University of Oxford, United Kingdom, Dec. 1999.

¹⁰Duta, M. C., *The use of the adjoint method for the minimization of forced response*, Ph.D. thesis, University of Oxford, United Kingdom, March 2002.

¹¹Ning, W. and He, L., “Some Modelling Issues on Trailing-Edge Vortex Shedding,” *AIAA Journal*, Vol. 39, No. 5, May 2001, pp. 787–793.

¹²Bassi, F., Crivellini, A., Rebay, S., and Savini, M., “Discontinuous Galerkin solution of the Reynolds averaged Navier-Stokes and $k - \omega$ turbulence model equations,” *Computers and Fluids*, Vol. 34, No. 4-5, 2005, pp. 507–540.

¹³März, J., Hah, C., and Neise, W., “An experimental and numerical investigation into the mechanisms of rotating instability,” *Journal of Turbomachinery*, Vol. 124, No. 3, July 2002, pp. 367–375.

¹⁴Saad, 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.

¹⁵Shroff, 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.

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

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

¹⁸Fransson, 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, October 1999, pp. 717–725.

¹⁹Giles, M., “Nonreflecting boundary conditions for Euler equation calculations,” *AIAA Journal*, Vol. 28, No. 12, December 1990, pp. 2050–2058.

²⁰Jameson, A., “Time-Dependent Calculations Using Multi-Grid, With Applications to Unsteady Flows Past Airfoil and Wings,” AIAA Paper 91-1596, 1991.

²¹Embree, M., “The tortoise and the hare restart GMRES,” *SIAM Review*, Vol. 45, No. 2, 2003, pp. 259–266.

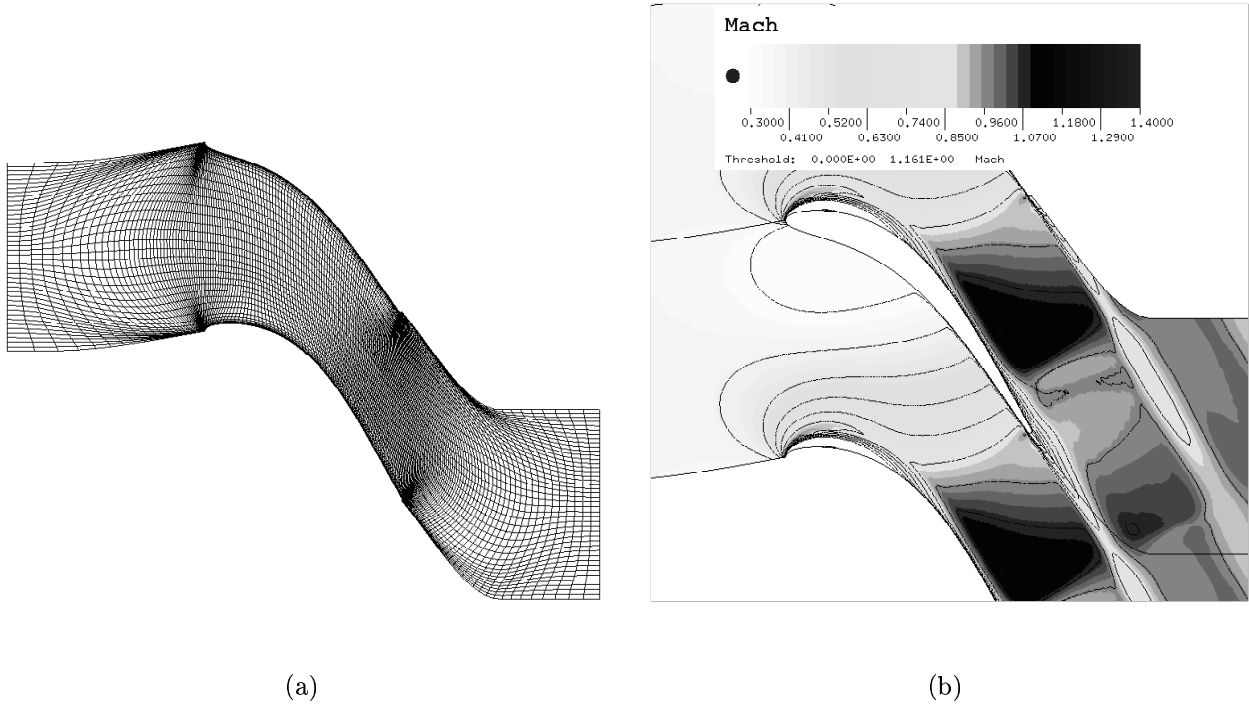


Figure 1. (a) Mesh for the two-dimensional turbine section and (b) Mach contours for transonic conditions.

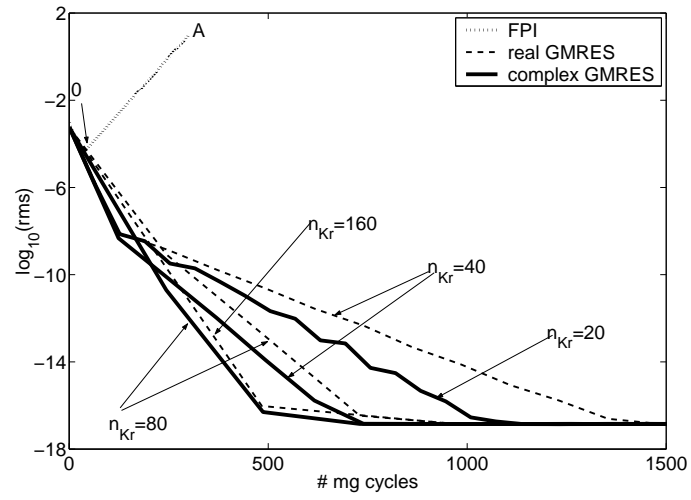
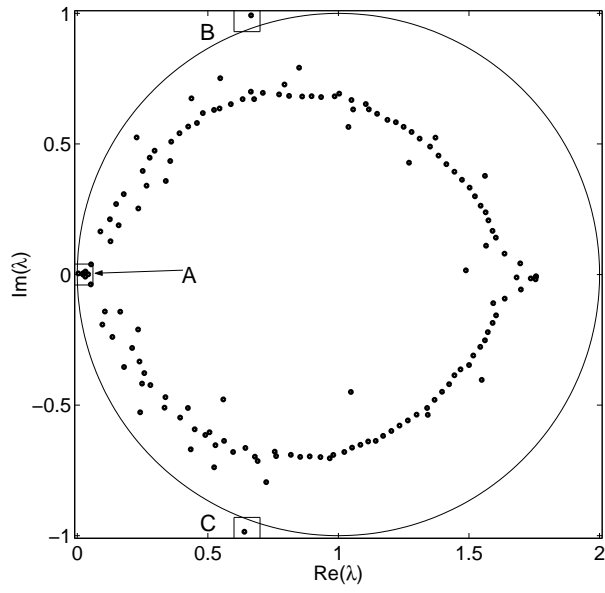
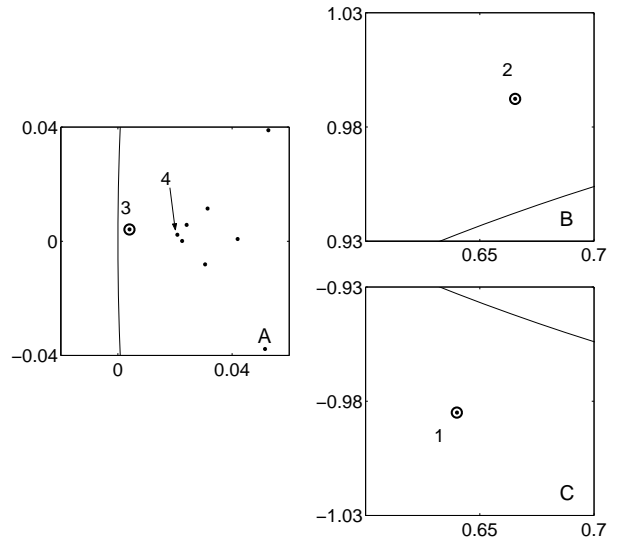


Figure 2. Flutter analysis of the 2D turbine ($IBPA = 180^\circ$): convergence histories of complex and real GMRES solvers ($n_{cl} = 3$).

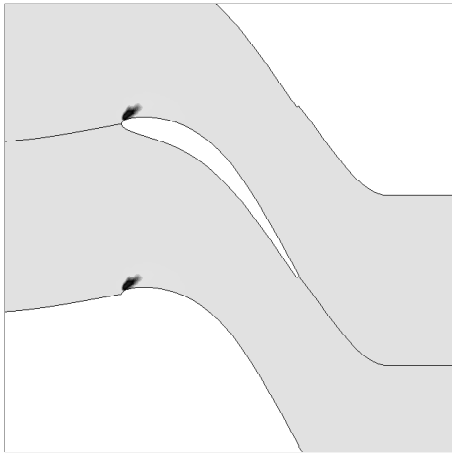


(a)

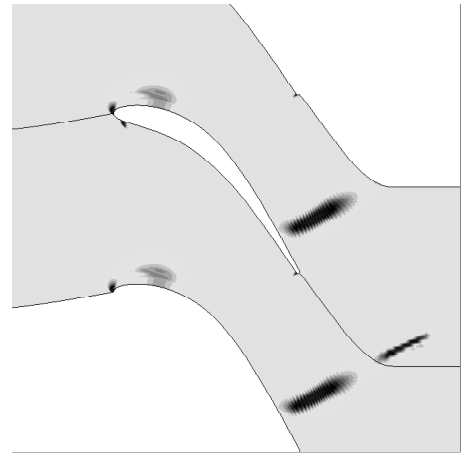


(b)

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

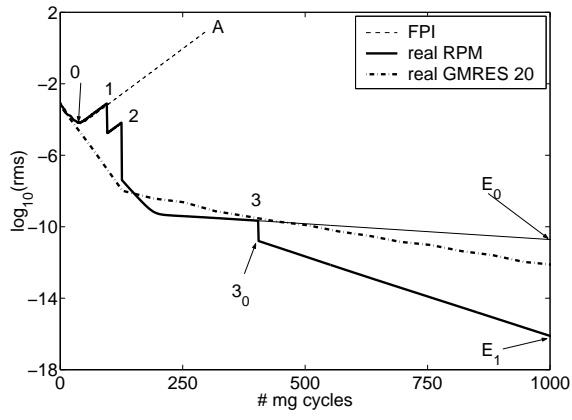


(a)

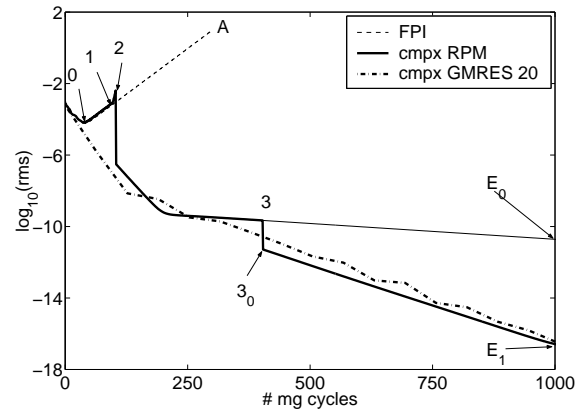


(b)

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

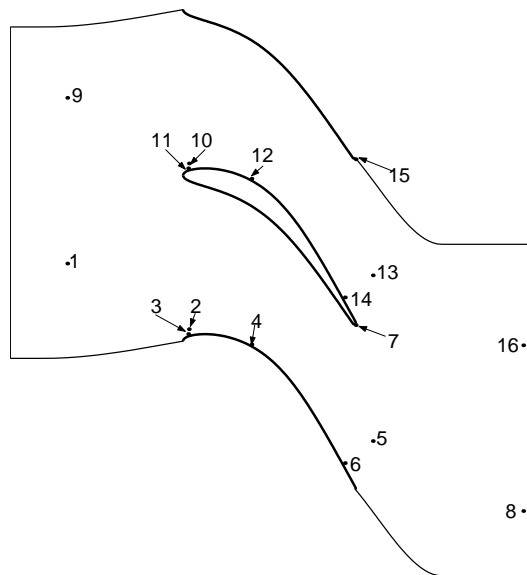


(a)

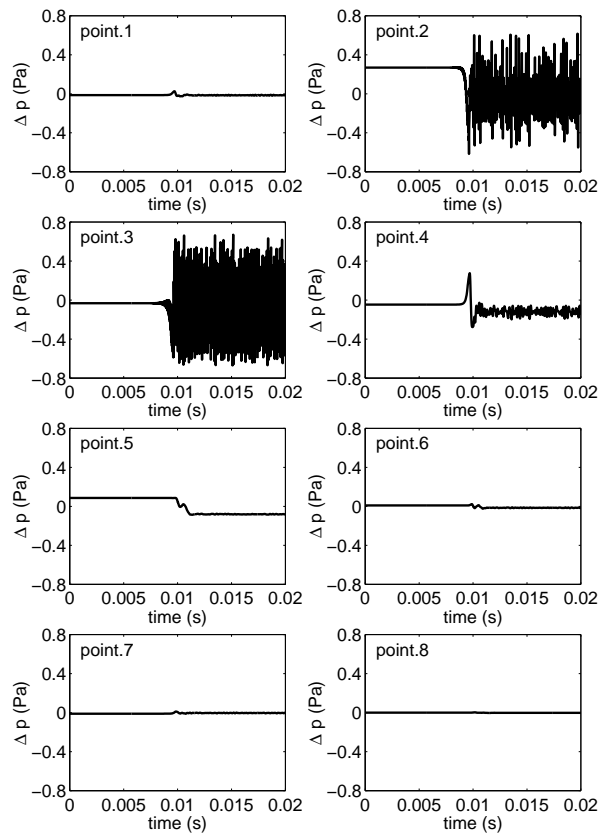


(b)

Figure 5. Flutter analysis of the 2D turbine ($IBPA=180^\circ$): convergence history of the (a) real and (b) complex RPM iterations.

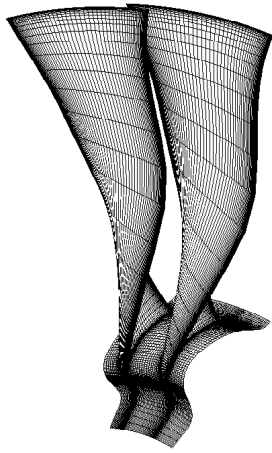


(a)

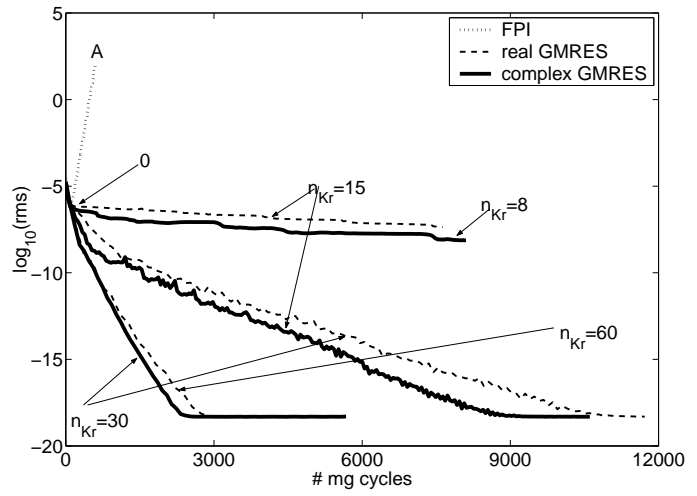


(b)

Figure 6. (a) Computational domain of the nonlinear time-accurate calculation and positions at which the unsteady flow is sampled , and (b) time-dependent pressure at the positions labeled from 1 to 8 (first passage).



(a)



(b)

Figure 7. (a) Fan geometry and surface mesh and (b) convergence histories of complex and real GMRES solvers ($IBPA=180^\circ, n_{cl}=3$).

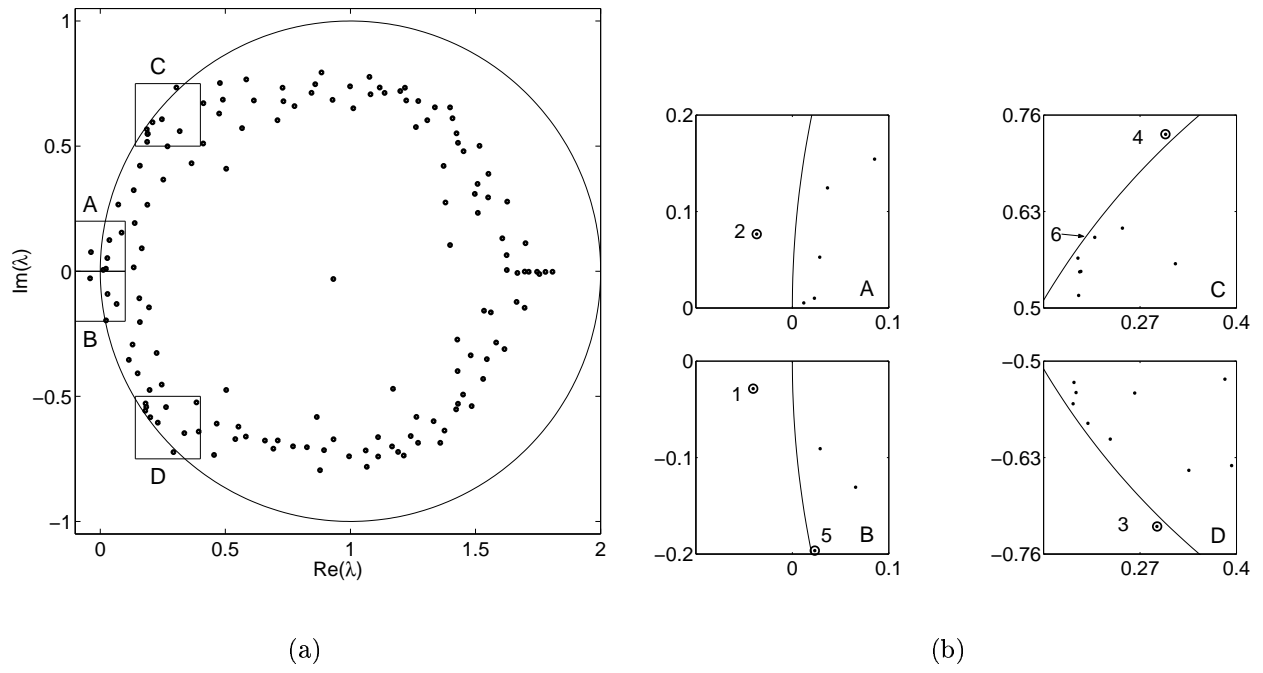
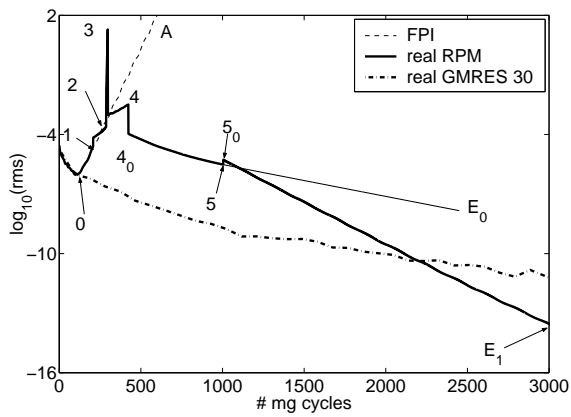
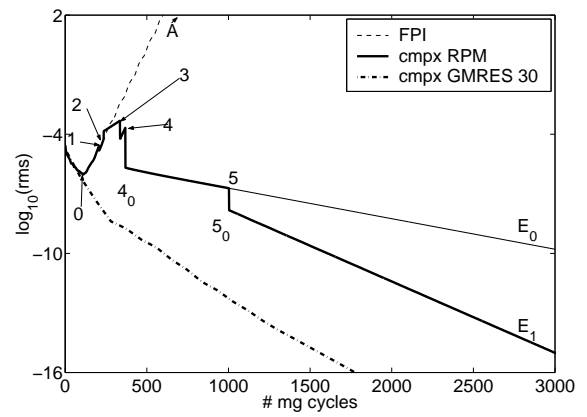


Figure 8. Flutter analysis of the three-dimensional fan rotor ($IBPA = 180^\circ$): (a) first 150 dominant eigenvalues of $M^{-1}A$ and (b) enlarged views with outliers.



(a)



(b)

Figure 9. Flutter analysis of the 3D fan rotor ($IBPA = 180^\circ$): convergence history of the (a) real and (b) ,complex RPM iterations.