An Efficient Flexible GMRES Solver for the Fully-coupled Time-spectral Aeroelastic System

Nathan L. Mundis [*] Dimitri J. Mavriplis [†]

Department of Mechanical Engineering, University of Wyoming, Laramie, Wyoming 82071-3295

The time-spectral (TS) method applied to the coupled aeroelastic equations theoretically offers significant computational savings for purely periodic problems when compared to standard time-implicit methods. However, attaining superior efficiency with TS methods over traditional time-implicit methods hinges on the ability to rapidly solve the large non-linear system resulting from TS discretizations which become larger and stiffer as more time instances are employed. An ideal time-spectral solver would scale linearly, such that a doubling of the number of time-instances used would double the wall-clock time needed to converge the solution on the same computational hardware. The present work is focused on achieving an optimal solver for large numbers of time instances. In order to increase the efficiency of the solver, and to improve robustness particularly for large numbers of time instances and fluid/structure coupling, TS methods are reworked such that the Generalized Minimal Residual Method (GMRES) is used to solve the implicitly coupled, time-spectral, fluid/structure linear system over all time instances at each non-linear iteration. The use of GMRES as the linear solver makes these methods more robust, allows them to be applied to a far greater subset of time-accurate problems, including those with a broad range of harmonic content, and vastly improves the efficiency of time-spectral methods.

I. Introduction

For problems with strong periodic content, such as turbomachinery flows or rotorcraft aerodynamics, time-spectral methods can be used to substantially reduce the cost of computing the full, time-dependent solution for a given level of accuracy. In many cases, time-spectral methods using only a small number of time instances per period can provide equivalent or superior accuracy, at substantially reduced cost, compared to traditional time-implicit solutions using hundreds of time steps per period. In other cases, many time instances per period may be needed to resolve both high and low frequency periodic content simultaneously.

Both the time-spectral and harmonic-balance methods are derived using discrete Fourier analysis. These methods, developed by Hall,[1] McMullen,[2,3] and Gopinath,[4,5] transform the unsteady equations in the physical domain to a set of steady equations in the frequency domain and then use the time-discretization operator to transform the frequency content back into a discrete number of time instances that reside in the time domain. Each of these time instances is coupled to all other time instances through the time-discretization operator, and the entire system is solved as a single, large, steady-state problem. In the literature, the time-spectral method has been shown to be faster than the dual-time stepping implicit methods using backwards difference time formulae for time periodic computations, such as turbomachinery flows,[2,5] oscillatory pitching airfoil/wing cases,[4,6] flapping wing,[7] helicopter rotor[8,9] and vortex shedding problems,[3] which all use small numbers of time instances. However, for problems with large numbers of time instances, it is not yet clear if the theoretical efficiency gains of TS methods over time-implicit methods can be realized.

Furthermore, the solution of aeroelastic problems introduces yet another layer of complexity and frequently require many time instances to solve. In order to converge the coupled aeroelastic time-spectral system with large numbers of time instances efficiently and robustly, the Krylov subspace, Generalized Minimal Residual method[10] is utilized. By using GMRES, the disparate time instances are much more strongly coupled than they would be if a stationary iterative method such as the Jacobi method were used. Additionally, use of GMRES allows the fluid equations to be coupled to the structural equations and the structural equations to be coupled to the fluid equations implicitly. This implicit fluid/structure coupling has proven key in the development of a robust aeroelastic, time-spectral solver.[11,12]

[*]Graduate Student, AIAA Member; email: nmundis@uwyo.edu.

[†]Professor, AIAA Associate Fellow; email: mavripl@uwyo.edu.

In the following sections we present the necessary components of the aeroelastic, time-spectral time discretization and the application of GMRES to this aeroelastic system. We first outline the governing equations and the base solver for the Euler equations. We then discuss additions to the flow solver required to implement the time-spectral method. Next, we discuss the extension of the the time-spectral method to quasi-periodic flows. Subsequently, the structural equations and how they are transformed for solution as part of the implicit aeroelastic system is presented. Next, we discuss the linearization of the time-spectral Euler equations and the solution of the resultant linear system using GMRES. Following, we present our results: first, for a problem treating the flow alone, and then a fully coupled aeroelastic problem. Next, a brief inquiry into the effects of different Mach numbers and reduced frequencies on the convergence of the time-spectral, aeroelastic system is presented.

II. Governing Equations

A. Base Solver

The Euler equations in conservative form can be written as:

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{F}(\mathbf{U})) = 0 \tag{1}$$

where $\mathbf{U}$ represents the vector of conserved quantities (mass, momentum, and energy) and $\mathbf{F}(\mathbf{U})$ represents the convective fluxes. Integrating over a (moving) control volume $\Omega(t)$, we obtain:

$$\int_{\Omega(t)} \frac{\partial \mathbf{U}}{\partial t} dV + \int_{\partial\Omega(t)} (\mathbf{F}(\mathbf{U}) \cdot \tilde{\mathbf{n}}) dS = 0 \tag{2}$$

Using the differential identity

$$\frac{\partial}{\partial t} \int_{\Omega(t)} \mathbf{U} dV = \int_{\Omega(t)} \frac{\partial \mathbf{U}}{\partial t} dV + \int_{\partial\Omega(t)} \mathbf{U}(\dot{\mathbf{x}} \cdot \tilde{\mathbf{n}}) dS \tag{3}$$

where $\dot{\mathbf{x}}$ and $\tilde{\mathbf{n}}$ are the velocity and normal of the interface $\partial\Omega(t)$, respectively, equation (2) becomes:

$$\frac{\partial}{\partial t} \int_{\Omega(t)} \mathbf{U} dV + \int_{\partial\Omega(t)} (\mathbf{F}(\mathbf{U}) - \mathbf{U}\dot{\mathbf{x}}) \cdot \tilde{\mathbf{n}} dS = 0 \tag{4}$$

Considering $\mathbf{U}$ as cell averaged quantities, these equations are discretized in space as:

$$\frac{\partial}{\partial t}(V\mathbf{U}) + \mathbf{R}(\mathbf{U}, \dot{\mathbf{x}}(t), \tilde{\mathbf{n}}(t)) = 0 \tag{5}$$

where $\mathbf{R}(\mathbf{U}, \dot{\mathbf{x}}, \tilde{\mathbf{n}}) = \int_{\partial\Omega(t)} (\mathbf{F}(\mathbf{U}) - \dot{\mathbf{x}}\mathbf{U}) \cdot \tilde{\mathbf{n}} dS$ represents the discrete convective fluxes in ALE form and V denotes the control volume. In the discrete form, $\dot{\mathbf{x}}(t)$ and $\tilde{\mathbf{n}}(t)$ now represent the time varying velocities and surface normals of the control-volume boundary faces.

The Euler equations are discretized by a central difference finite-volume scheme with additional matrix-based artificial dissipation on hybrid meshes which may include triangles and quadrilaterals in two dimensions. Second-order accuracy is achieved using a two-pass construction of the artificial dissipation operator, which corresponds to an undivided biharmonic operator. A single unifying face-based data-structure is used in the flow solver for all types of elements. For a given face, the residual contribution of that face can be written as:

$$\mathbf{R}_{1stO,ik}(\mathbf{U}, \dot{\mathbf{x}}(t), \tilde{\mathbf{n}}(t)) = (\mathbf{F}_i(\mathbf{U}_i) + \mathbf{F}_k(\mathbf{U}_k) - \mathbf{U}_{ik}\dot{\mathbf{x}}_{ik}) \cdot \tilde{\mathbf{n}}\Delta S + \kappa \underline{\mathbf{T}} |\underline{\Lambda}| \underline{\mathbf{T}}^{-1} (\mathbf{U}_i - \mathbf{U}_k) \tag{6}$$

for first-order matrix dissipation, where $\underline{T}$ is the left-eigenvector matrix, $\underline{\Lambda}$ is the eigenvalue matrix, and $\underline{T}^{-1}$ is the right-eigenvector matrix of the convective fluxes. For second-order matrix dissipation, the residual on a face can be written as follows:

$$\mathbf{R}_{2ndO,ik}(\mathbf{U}, \dot{\mathbf{x}}(t), \tilde{\mathbf{n}}(t)) = (\mathbf{F}_i(\mathbf{U}_i) + \mathbf{F}_k(\mathbf{U}_k) - \mathbf{U}_{ik}\dot{\mathbf{x}}_{ik}) \cdot \tilde{\mathbf{n}}\Delta S + \kappa \underline{\mathbf{T}} |\underline{\Lambda}| \underline{\mathbf{T}}^{-1} (\mathbf{L}_i(\mathbf{U}) - \mathbf{L}_k(\mathbf{U})) \tag{7}$$

where $\mathbf{L}(\mathbf{U})$ is the undivided Laplacian operator, taken as:

$$\mathbf{L}_p(\mathbf{U}) = \sum_{q=1}^{neighbors} (\mathbf{U}_q - \mathbf{U}_p) \tag{8}$$

In both cases, κ is an empirical constant with a typical value of $1/2$ for first-order matrix dissipation and $1/8$ for second-order matrix dissipation.

B. Time-spectral Method

If the flow is periodic in time, the variables $\mathbf{U}$ can be represented by a discrete Fourier series. The discrete Fourier transform of $\mathbf{U}$ in a period of T is given by[4]

$$\widehat{\mathbf{U}}_k = \frac{1}{N}\sum_{n=0}^{N-1} \mathbf{U}^n e^{-ik\frac{2\pi}{T}n\Delta t} \tag{9}$$

where N is the number of time intervals and $\Delta t = T/N$. The Fourier inverse transform is then given as

$$\mathbf{U}^n = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \widehat{\mathbf{U}}_k e^{ik\frac{2\pi}{T}n\Delta t} \tag{10}$$

It should be noted that $\frac{N}{2}$ is an integer division operation. Also note that this corresponds to a collocation approximation, i.e. the function $\mathbf{U}(t)$ is projected into the space spanned by the truncated set of complex exponential (spectral) functions, and the expansion coefficients (in this case the $\widehat{\mathbf{U}}_k$) are determined by requiring $\mathbf{U}(t)$ to be equal to its projection at N discrete locations in time, as given by equations (9) and (10).
Differentiating equation (10) in time, we obtain:

$$\frac{\partial}{\partial t}(\mathbf{U}^n) = \frac{2\pi}{T}\sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} ik\widehat{\mathbf{U}}_k e^{ik\frac{2\pi}{T}n\Delta t} \tag{11}$$

Substituting equation (9) into equation (11), we get[13,14]

$$\frac{\partial}{\partial t}(\mathbf{U}^n) = \sum_{j=0}^{N-1} d_n^j \mathbf{U}^j \tag{12}$$

where

$$d_n^j = \begin{cases} \frac{2\pi}{T}\frac{1}{2}(-1)^{n-j}\cot\left(\frac{\pi(n-j)}{N}\right) & n \neq j \\ 0 & n = j \end{cases} \tag{13}$$

for an even number of time instances and

$$d_n^j = \begin{cases} \frac{2\pi}{T}\frac{1}{2}(-1)^{n-j}\csc\left(\frac{\pi(n-j)}{N}\right) & n \neq j \\ 0 & n = j \end{cases} \tag{14}$$

for an odd number of time instances. Next, substitute equation (10) into equation (5), and require equation (5) to hold exactly at the same N discrete locations in time (i.e. multiply (5) by the dirac delta test function $\delta(t-t^n)$ and integrate over all time), which yields the following time-spectral governing equation:

$$\sum_{j=0}^{N-1} d_n^j V^j \mathbf{U}^j + \mathbf{R}(\mathbf{U}^n, \dot{\mathbf{x}}^n, \tilde{\mathbf{n}}^n) = 0 \qquad n = 0,1,2,\ldots,N-1 \tag{15}$$

This results in a system of N equations for the N time instances $\mathbf{U}^n$ which are all coupled through the summation over the time instances in the time derivative term. The spatial discretization operators remain unchanged in the time-spectral approach, with only the requirement that they be evaluated at the appropriate temporal location. Thus, the time-spectral method may be implemented without any modifications to an existing spatial discretization, requiring only the addition of the temporal discretization coupling term, although the multiple time instances must be solved simultaneously due to this coupling.

C. Fully Implicit Method

A common approach for solving the system of equations resulting from the time-spectral method (c.f. equation (15)) consists of adding a pseudo-time term as:

$$\frac{\partial}{\partial \tau}(V^n \mathbf{U}^n) + \sum_{j=0}^{N-1} d_n^j V^j \mathbf{U}^j + \mathbf{R}(\mathbf{U}^n, \dot{\mathbf{x}}^n, \tilde{\mathbf{n}}^n) = 0 \tag{16}$$

and time-stepping these equations until a pseudo-time steady state is achieved. However, for explicit pseudo-time stepping approaches, it has been shown that the pseudo-time step is limited by stability considerations as:[5]

$$\Delta\tau_n = CFL \frac{V^n}{\parallel \lambda \parallel + V^n k'} \tag{17}$$

where λ is the spectral radius of the spatial discretization operator $\mathbf{R}(\mathbf{U}^n, \dot{\mathbf{x}}^n, \tilde{\mathbf{n}}^n)$ and k' represents the largest wave-number that can be resolved by the specified N time instances:

$$k' = \begin{cases} \frac{\pi N}{T} & \text{if } N \text{ is even} \\ \frac{\pi(N-1)}{T} & \text{if } N \text{ is odd} \end{cases} \tag{18}$$

This restriction on $\Delta\tau_n$ results in convergence degradation as the number of time instances or harmonics is increased. The impact of this restriction can be greatly lessened by resorting to an implicit approach in pseudo-time. Such an approach has been derived in reference[15] using a first-order backwards difference scheme in pseudo-time.

A more general strategy consists of devising a Newton approach for solving the fully coupled non-linear equations at all time instances given by equation (15) or (16). The Newton scheme takes the form:

$$[\mathbf{A}]\,\Delta\mathbf{U} = -\sum_{j=0}^{N-1} d_n^j V^j \mathbf{U}^j - \mathbf{R}(\mathbf{U}^n, \dot{\mathbf{x}}^n, \tilde{\mathbf{n}}^n) \tag{19}$$

with the resulting Jacobian matrix given by:[15]

$$[\mathbf{A}] = \begin{bmatrix} \frac{V^0}{\Delta\tau_0}\underline{\mathbf{I}} + \underline{\mathbf{J}}_0 & V^1 d_0^1 \underline{\mathbf{I}} & \cdots & V^{N-1} d_0^{N-1} \underline{\mathbf{I}} \\ V^0 d_1^0 \underline{\mathbf{I}} & \frac{V^1}{\Delta\tau_1}\underline{\mathbf{I}} + \underline{\mathbf{J}}_1 & \cdots & V^{N-1} d_1^{N-1} \underline{\mathbf{I}} \\ \vdots & \vdots & \cdots & \vdots \\ V^0 d_{N-1}^0 \underline{\mathbf{I}} & V^1 d_{N-1}^1 \underline{\mathbf{I}} & \cdots & \frac{V^{N-1}}{\Delta\tau_{N-1}}\underline{\mathbf{I}} + \underline{\mathbf{J}}_{N-1} \end{bmatrix} \tag{20}$$

where a diagonal pseudo-time term can be included as shown for enhanced diagonal dominance of the Jacobian matrix. In the above matrix, $\underline{\mathbf{J}}_j$ corresponds to the Jacobian of the spatial discretization operator evaluated at time instance j. For a first-order spatial discretization, $\underline{\mathbf{J}}_{j,1stO}$ is as follows:

$$\underline{\mathbf{J}}_{j,1stO} = \frac{\partial \mathbf{R}_{1stO}}{\partial \mathbf{U}} \tag{21}$$

For a second-order spatial discretization, $\underline{\mathbf{J}}_{j,2ndO}$ contains many more entries, as each element of the mesh is not only influenced by its nearest neighbors, but also by the neighbors of its neighbors. The second-order Jacobian is derived as follows:

$$\underline{\mathbf{J}}_{j,2ndO} = \left.\frac{\partial \mathbf{R}_{2ndO}}{\partial \mathbf{U}}\right|_{L=constant} + \left.\frac{\partial \mathbf{R}_{2ndO}}{\partial \mathbf{L}}\right|_{U=constant} \cdot \frac{\partial \mathbf{L}}{\partial \mathbf{U}} \tag{22}$$

Returning to equation (19), each non-linear Newton iteration requires solving the linear system

$$[\mathbf{A}]\,\Delta\mathbf{U} = -\mathbf{R_{TS}}(\mathbf{U}) \tag{23}$$

where $\mathbf{R_{TS}}(\mathbf{U})$ represents the residual of the complete time-spectral system (i.e. right hand side of equation (19), and $[\mathbf{A}]$ is a large matrix spanning all spatial and temporal degrees of freedom. Since direct inversion of $[\mathbf{A}]$ is generally intractable, an inexact Newton scheme can be formulated using an approximate representation of $[\mathbf{A}]$ which is simpler to invert. One possible simplification is to replace the exact spatial Jacobian in each diagonal block $\underline{\mathbf{J}}_i$ by the corresponding first-order Jacobian $\underline{\mathbf{J}}_{i,1stO}$ as is typically done for steady-state solvers. Another simplification consists of dropping all the off-diagonal terms representing the coupling between different time instances. When this is done, the linear system becomes decoupled between time instances and can be written as:

$$\left[\frac{V^i}{\Delta\tau_i}\underline{\mathbf{I}} + \underline{\mathbf{J}}_{i,1stO}\right]\Delta\mathbf{U}_i = -\mathbf{R}_{\mathbf{TS}}(\mathbf{U}) \tag{24}$$

for each time instance $i = 0,1,2,...,N-1$. Alternatively, the diagonal blocks in the $[\mathbf{A}]$ matrix may be retained and the system solved in a block Jacobi fashion following:

$$\left[\frac{V^i}{\Delta\tau_i}\underline{\mathbf{I}} + \underline{\mathbf{J}}_{i,1stO}\right]\Delta\mathbf{U}_i^{l+1} = -\mathbf{R}_{\mathbf{TS}}(\mathbf{U}) - \sum_{j\neq i}\left[V^j d_i^j \underline{\mathbf{I}}\right]\Delta\mathbf{U}_j^l \tag{25}$$

where the block size corresponds to the entire spatial domain or each time instance and where l denotes the block Jacobi iteration index.

D. Block-colored Gauss-Seidel Linear Solver

As can be seen, both approaches require the inversion the first-order spatial Jacobian (augmented with a pseudo-time term) at each iteration. This may be accomplished using a suitable iterative solver such as block colored Gauss-Seidel (BCGS). In this case, the block now corresponds to the 4×4 block diagonal matrix at each cell of the mesh, and the iterative scheme can be written as:

$$\left[\frac{V^i}{\Delta\tau_i}\underline{\mathbf{I}} + \left[\underline{\mathbf{D}}_{i,1stO}\right]\right]\Delta\mathbf{U}_i^{l+1} = -\mathbf{R}_{\mathbf{TS}}(\mathbf{U}) - \sum_{j\neq i}\left[V^j d_i^j \underline{\mathbf{I}}\right]\Delta\mathbf{U}_j^l - \left[\underline{\mathbf{O}}_{i,1stO}\right]\Delta\mathbf{U}_i^l \tag{26}$$

where $\underline{\mathbf{D}}_{i,1stO}$ denotes the 4×4 block matrix for the current cell, and $\underline{\mathbf{O}}_{i,1stO}$ refers to the off-diagonal blocks for neighboring mesh cells. Although this equation describes a block Jacobi iteration (at the mesh cell level), a block-colored Gauss-Seidel scheme can be recovered with a few simple modifications. First, the computational elements are divided into computational "colors" such that no two adjacent elements are the same color. This coloring allows the Gauss-Seidel method to be run in parallel. The BCGS method then updates (in parallel) all elements of each individual color (with the different colors updated in sequence), such that each update uses the newest information available for all other colors. The BCGS algorithm, written in residual form for a generic linear system is given in algorithm (1).

Algorithm 1 : Block-colored Gauss-Seidel

```
1:  Given $\underline{\mathbf{A}}\mathbf{x} = \mathbf{b}$
2:  for i=1,...,$\zeta$ do
3:      for j=1,...,$n_{colors}$ do
4:          for k=1,...,$n_{e_j}$ do
5:              Compute $\mathbf{r}_{i,k} = \mathbf{b} - \underline{\mathbf{A}}\mathbf{x}_{current}$
6:              Compute $\Delta\mathbf{x}_{i,k} = \underline{\mathbf{D}}_k^{-1}\mathbf{r}_{i,k}$
7:              Update $\mathbf{x}_{i+1,k} = \mathbf{x}_{i,k} + \Delta\mathbf{x}_{i,k}$
8:          end for
9:      end for
10:     Compute $R_{L,i} = \|\mathbf{r}_i\|_2$
11:     If satisfied Stop.
12: end for
```

In algorithm (1), ζ is the maximum number of BCGS iterations allowed, n_{colors} is the number of colors into which the elements have been divided, and n_{e_j} is the number of elements having the jth color. Additionally, $\underline{\mathbf{D}}_k$ is the diagonal block of the Jacobian for element k, and is inverted directly using LU-decomposition. The specification $\mathbf{x}_{current}$ is used to indicate that, some colors will have information from the previous iteration $\mathbf{x}_i$ while other colors might have already been updated during the current iteration $\mathbf{x}_{i+1}$; in other words, whether from the previous or current iteration, the most up-to-date information at the time of the evaluation of line 5 is used.

As noted earlier, the $\underline{\mathbf{A}}$ matrix given in lines 1 and 5 of the algorithm can either include or exclude the off-diagonal terms given in equation (20). When these time-coupling terms are excluded, this corresponds to a BCGS solver with explicit time treatment, which is abbreviated "BCGS-EX." When the off-diagonal terms are included, implicit time treatment is used in the BCGS solver, and this method is abbreviated "BCGS-IM."

E. Generalized Minimal Residual Method

Despite the additional stability the above, implicit method affords, the time-spectral Euler equations still become difficult to solve as the number of time instances increases. With an increasing number of time instances, the Jacobian given in equation (20) proceeds farther and farther from diagonal dominance. To restore diagonal dominance, a decreasingly small pseudo-time step must be used. Thus, the potential efficiency gains of time-spectral methods over time-implicit methods begin to evaporate.

To regain the efficiency improvements afforded by time-spectral methods, a linear solver that does not require diagonal dominance of the Jacobian should be used. The Generalized Minimum Residual method is such a solver. A welcome byproduct of this solver choice is that GMRES also more fully couples the various time instances. This additional coupling arises because the Hessenberg matrix that is directly inverted as part of GMRES is constructed using the full Jacobian of the linear system coupled over all time instances; whereas, the matrix that is directly inverted during each iteration of a stationary iterative method is some easily invertible part of the full Jacobian (usually the diagonal blocks). In other words, stationary iterative methods ignore some information (and thereby some coupling), but GMRES uses this information and preserves the full coupling.

A flexible variant of the GMRES algorithm as described by Saad[10] is used. This flexible variant allows the use of an iterative method as preconditioner. The flexible GMRES (FGMRES) algorithm proceeds as given in Algorithm (2).

Algorithm 2 : Flexible GMRES

```
 1: Given $\underline{\mathbf{A}}\mathbf{x} = \mathbf{b}$
 2: Compute $\mathbf{r}_0 = \mathbf{b} - \underline{\mathbf{A}}\mathbf{x}_0$, $\beta = \|\mathbf{r}_0\|_2$, and $\mathbf{v}_1 = \mathbf{r}_0/\beta$
 3: for j=1,...,n do
 4:   Compute $\mathbf{z}_j := \underline{\mathbf{P}}^{-1}\mathbf{v}_j$
 5:   Compute $\mathbf{w} := \underline{\mathbf{A}}\mathbf{z}_j$
 6:   for i=1,...,j do
 7:     $h_{i,j} := (\mathbf{w}, \mathbf{v}_i)$
 8:     $\mathbf{w} := \mathbf{w} - h_{i,j}\mathbf{v}_j$
 9:   end for
10:   Compute $h_{j+1,j} = \|\mathbf{w}\|_2$ and $\mathbf{v}_{j+1} = \mathbf{w}/h_{j+1,j}$
11:   Define $\underline{\mathbf{Z}}_m := [\mathbf{z}_1, \ldots, \mathbf{z}_m]$, $\underline{\bar{\mathbf{H}}}_m = \{h_{i,j}\}_{1\le i\le j+1; 1\le j\le m}$
12: end for
13: Compute $\mathbf{y}_m = argmin_y\|\beta\mathbf{e}_1 - \underline{\bar{\mathbf{H}}}_m\mathbf{y}\|_2$ and $\mathbf{x}_m = \mathbf{x}_0 + \underline{\mathbf{Z}}_m\mathbf{y}_m$
14: If satisfied Stop, else set $\mathbf{x}_0 \leftarrow \mathbf{x}_m$ and GoTo 1.
```

In this description, $\underline{\mathbf{A}}$ corresponds to the full time-spectral Jacobian matrix $[\mathbf{A}]$ defined in equation (20), which may or may not be augmented with a pseudo time step, $\mathbf{b}$ corresponds to the negative of the non-linear time-spectral residual $-\mathbf{R_{TS}}(\mathbf{U})$, and x is the non-linear update $\Delta\mathbf{U}$ to be computed.

Preconditioning is applied in line 4 of the algorithm. The BCGS linear solver is used for preconditioning in several different configurations as covered in the next subsection. For efficiency, each Krylov vector is preconditioned with the corresponding first-order accurate flow Jacobian. A pseudo-time step must be applied to the BCGS system (i.e. added to $\underline{\mathbf{P}}$) to ensure diagonal dominance and convergence. To solve the minimization problem in line 13 of the algorithm, QR-factorization by means of Givens rotations is utilized.[10] Finally, line 14 of the algorithm indicates that this algorithm is, in fact, truncated, restarting GMRES using n Krylov vectors per restart.

Although algorithm (2) shows the minimization problem outside the loop over Krylov vectors, in fact, we update this minimization as each additional Krylov vector is added. This is done so that the current value of the linear residual is known for each iteration j. The FGMRES algorithm exits whenever this residual has either converged a specified amount or has converged to machine zero.

A pseudo-time step is used in the matrix $\underline{\mathbf{A}}$ in the FGMRES algorithm as well. By using this pseudo-time step within FGMRES itself the routine is modified positively as follows: an update that will avoid over-correction issues is used; the pseudo-time step is allowed to grow as the residual decreases so quadratic convergence can be retained; and the convergence rate of the FGMRES linear solver itself is increased (i.e. fewer Krylov vectors and/or restarts are required to converge the linear system) because the pseudo-time step term makes the Jacobian better conditioned for Krylov subspace methods. Thus, two different pseudo-time steps are used: one in the second-order Jacobian $\underline{\mathbf{A}}$ used by the FGMRES algorithm itself and another in the first-order Jacobian $\underline{\mathbf{P}}$ used in the BCGS portion of the preconditioner. Because the Jacobian used in the BCGS preconditioner must be diagonally dominant, the preconditioner pseudo-time step is almost always smaller and grows more slowly than the FGMRES pseudo-time step. Conversely, the pseudo-time step used in the FGMRES algorithm grows rapidly such that the diagonal term becomes vanishingly small and an exact Newton method is recovered after several orders of magnitude decrease in the non-linear residual.

1. *Preconditioning Methods for FGMRES*

As mentioned above, all preconditioning methods make use of the block-colored Gauss-Siedel solver in one form or another. The first preconditioning method uses BCGS with explicit treatment of time. In other words, we use algorithm (1), with a specified number of iterations ζ on each Krylov vector. Here the preconditioner matrix $\underline{\mathbf{P}}$ corresponds to the $[\mathbf{A}]$ matrix, given by equation (20), using the first-order spatial Jacobian with a pseudo-time step that is different, and smaller, than the pseudo-time step used in FGMRES. Additionally, this preconditioner matrix does not include the off-diagonal, time-spectral terms. We abbreviate GMRES using this preconditioner as "GMRES-EX" in the remainder of this work.

The next preconditioner used is BCGS with implicit treatment of time. This preconditioner uses Algorithm (1) with a specified number of iterations ζ on each Krylov vector. The full $[\mathbf{A}]$ matrix, including the time-spectral coupling terms, is used, but again, the first-order spatial Jacobian is used with a smaller pseudo-time step. This solver/preconditioner combination is abbreviated "GMRES-IM".

Both of the preconditioners presented thus far have a serious limitation: the pseudo-time step size needed to preserve diagonal-dominance of the Jacobian and an appropriately scaled update remains relatively small. In order to bypass this limitation, we form a preconditioner for FGMRES as a defect correction method applied to the residual of equation (25) as:

$$\left[\frac{V^i}{\Delta\tau_{BCGS}}\underline{\mathbf{I}}+\underline{\mathbf{J}}_{i,1stO}\right](\Delta\mathbf{U}_i^{l+1}-\Delta\mathbf{U}_i^l) = -\mathbf{R_{TS}}(\mathbf{U})-\sum_{j\neq i}\left[V^j d_i^j \underline{\mathbf{I}}\right]\Delta\mathbf{U}_j^l-\left[\frac{V^i}{\Delta\tau_{FGMRES}}\underline{\mathbf{I}}+\underline{\mathbf{J}}_{i,1stO}\right]\Delta\mathbf{U}_i^l \tag{27}$$

where the right-hand-side corresponds to the residual of equation (25). A dual iteration strategy is required for this preconditioner (thus the term defect-correction). Inner BCGS iterations are used to invert the left-hand side matrix providing an updated value for $\Delta\mathbf{U}^{l_{i+1}}$, which is then substituted into the right-hand side terms, and the process is repeated, effectively driving the right-hand-side residual to zero after a number of outer iterations. The advantage of this approach comes from the fact that the pseudo-time step on the right-hand side residual is generally much larger than that required for stability of the BCGS iterative scheme. This solver/preconditioner combination is abbreviated "GMRES-DC".

Table 1 summarizes the various solvers and solver/preconditioner combinations presented in the current work and the abbreviations used to describe them.

Table 1: Summary of Solvers and Preconditioners

Linear Solver	**Preconditioner**	**Abbreviation**
Time-explicit BCGS	N/A	BCGS-EX
Time-implicit BCGS	N/A	BCGS-IM
FGMRES	Time-explicit BCGS	GMRES-EX
FGMRES	Time-implicit BCGS	GMRES-IM
FGMRES	Defect Correction with Time-implicit BCGS	GMRES-DC

F. Structural Equations and Solver

The aeroelastic model is based on the response of an airfoil with two degrees of freedom, namely pitch and plunge as shown in Figure (1). The equations of motion for such a system can be summarized as:

$$\begin{aligned} m\ddot{h}+S_\alpha\ddot{\alpha}+K_h h &= -L \\ S_\alpha\ddot{h}+I_\alpha\ddot{\alpha}+K_\alpha\alpha &= M_{ea} \end{aligned} \tag{28}$$

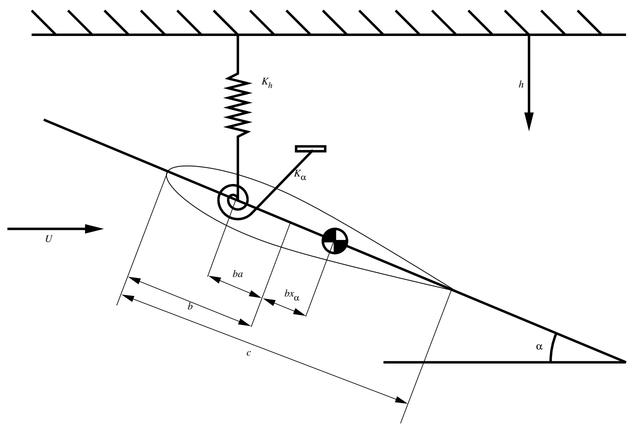


Figure 1: Two degree of freedom 2-D aeroelastic problem schematic

where the variable correspond to the following quantities:

m:	mass of airfoil
S_α:	static imbalance
I_α:	sectional moment of inertia of airfoil
K_h:	plunging spring coefficient
K_α:	pitching spring coefficient
h:	vertical displacement (positive downward)
α:	angle-of-attack
L:	sectional lift of airfoil
M_{ea}:	sectional moment of airfoil about elastic axis

Non-dimensionalizing time in equations (28) by the uncoupled natural frequency of pitch yields:

$$[M]\ddot{\breve{\mathbf{q}}} + [K]\mathbf{q} = \mathbf{F} \tag{29}$$

where the non-dimensional mass matrix $[M]$ and stiffness matrix $[K]$ are as follows:

$$[M] = \begin{bmatrix} 1 & x_\alpha \\ x_\alpha & r_\alpha^2 \end{bmatrix}, \quad [K] = \begin{bmatrix} \left(\frac{\omega_h}{\omega_\alpha}\right)^2 & 0 \\ 0 & r_\alpha^2 \end{bmatrix} \tag{30}$$

The corresponding non-dimensional load and displacement vectors are defined by the following relations:

$$\mathbf{F} = \frac{1}{\pi\mu k_c^2}\begin{bmatrix} -C_l \\ 2C_m \end{bmatrix}, \quad \mathbf{q} = \begin{bmatrix} \frac{h}{b} \\ \alpha \end{bmatrix}, \quad \ddot{\breve{\mathbf{q}}} = \frac{\partial^2 \mathbf{q}}{\partial \breve{t}^2} \tag{31}$$

The newly introduced quantities are give in the following table:

b:	semichord of airfoil
k_c:	reduced frequency of pitch, $k_c = \frac{\omega_\alpha c}{2U_\infty}$
μ:	airfoil mass ratio, $\mu = \frac{m}{\pi\rho b^2}$
ω_h, ω_α:	uncoupled natural frequencies of plunge and pitch
x_α:	structural parameter defined as $\frac{S_\alpha}{mb}$
r_α^2:	structural parameter defined as $\frac{I_\alpha}{mb}$
C_l, C_m:	section lift coefficient and section moment coefficient about the elastic axis
$\breve{t}$:	structural time, $\breve{t} = \omega_\alpha t$

The reduced frequency k_c is typically written in terms of the flutter velocity V_f as

$$k_c = \frac{\omega_\alpha c}{2U_\infty} = \frac{1}{V_f\sqrt{\mu}} \tag{32}$$

where c is the chord length of the airfoil, U_∞ is the freestream velocity, and the flutter velocity V_f is defined as

$$V_f = \frac{U_\infty}{\omega_\alpha b\sqrt{\mu}} \tag{33}$$

The natural pitch frequency is found by solving equation (33) for ω_α in terms of the prescribed flutter velocity V_f.

1. *Transformation of Structural Equations into First Order Form:*

The aeroelastic equations as shown in Equation (29) are second-order partial differential equations. A transformation to first-order equations is used in order to facilitate the solution procedure. The transformation used is:

$$\begin{aligned} \mathbf{r_1} &= \mathbf{q} \\ \mathbf{r_2} &= \mathring{\mathbf{r}}_\mathbf{1} \end{aligned} \tag{34}$$

The resulting first-order equations are then:

$$\begin{aligned} \mathring{\mathbf{r}}_\mathbf{1} &= \mathbf{r_2} \\ \mathring{\mathbf{r}}_\mathbf{2} &= -[M]^{-1}[K]\mathbf{r_1} + [M]^{-1}\mathbf{F} \end{aligned} \tag{35}$$

and in matrix notation:

$$\begin{bmatrix} \mathring{\mathbf{r}}_\mathbf{1} \\ \mathring{\mathbf{r}}_\mathbf{2} \end{bmatrix} + \begin{bmatrix} 0 & -[I] \\ [M]^{-1}[K] & 0 \end{bmatrix} \begin{bmatrix} \mathbf{r_1} \\ \mathbf{r_2} \end{bmatrix} = \begin{bmatrix} 0 \\ [M]^{-1}\mathbf{F} \end{bmatrix} \tag{36}$$

$$\mathring{\mathbf{r}} \quad + \quad [\Psi]\mathbf{r} = \Phi \tag{37}$$

The matrix $[\Psi]$ is a constant and can be precomputed and stored for use during the time integration process. The time derivative term $\mathring{\mathbf{r}}$ can be discretized using second-order (or first-order) accurate backward difference formulae similar to the time derivative term in the flow equations, as follows:

$$\frac{3\mathbf{r}^{n+1} - 4\mathbf{r}^n + \mathbf{r}^{n-1}}{2\Delta\breve{t}} \quad + \quad [\Psi]\mathbf{r}^{n+1} = \Phi \tag{38}$$

Or the time derivative can be discretized using time-spectral similar to equation (15) for purely periodic time-spectral:

$$\sum_{j=0}^{N-1} \breve{d}_n^j \mathbf{r}^j + [\Psi]\mathbf{r}^n = \Phi \qquad n = 0, 1, 2, \ldots, N-1 \tag{39}$$

It should be noted that the time-step $\Delta\breve{t}$ appearing in the denominator of the discretized version of the structural equations is different from the time-step of the flow equations, and their relation is $\Delta\breve{t} = \omega_\alpha \Delta t$, where Δt is the non-dimensional time-step used for the flow equations. Similarly, the period used to calculate the $\breve{d}_n^j$ and $\breve{\phi}$ in time-spectral and BDFTS is the period in the structural time frame. A breve ($\breve{}$) is placed above all quantities that use structural time for clarity. If one wanted to assume that the torsional stiffness were infinite, the torsional equation would need to be removed from the system and the plunge equation non-dimensionalized in a different way.

2. *FGMRES Aeroelastic Solver:*

The fluid and structure equations are coupled together implicitly; thus, for each iteration, fluid and structure residuals are found using either the BDF2 or the TS/BDFTS equations as discussed above. Then a Jacobian is found for the entire fluid/structure system. This Jacobian not only couples the fluid residuals to the flow variables and the structural residuals to the structural variables, but also the fluid residuals to the structural variables and the structural variables to the flow variables. Starting with the flow Jacobian (of a single time instance in the case of the time-spectral method), the coupled Jacobian will have as many additional rows and columns as there are structural variables. Thus, a flow Jacobian that is of size $[[4\times N_c]\times[4\times N_c]]$ where N_c is the number of elements or cells in the computational domain of the flow, would become a $[[4\times N_c+N_s]\times[4\times N_c+N_s]]$ matrix for the implicitly coupled fluid/structure system, where N_s is the number of structural variables.

While the above represents the size of the Jacobian if all entries were stored, it should be readily apparent that a matrix of this size could not be stored on modern computers for the sizes of computational meshes that are typically used. Instead, only the non-zero blocks of the Jacobian are stored. The first-order Jacobian is stored as an array of its diagonal blocks, which is $[4\times 4\times N_c]$, and an array of its off-diagonal blocks, which are stored on the faces and is $[2\times 4\times 4\times N_f]$ where the 2 represents the opposite sides of each face and N_f is the number of faces. The full second-order Jacobian is not stored explicitly; rather, we enable the evaluation of exact Jacobian-vector products as required at each linear solver iteration by storing three sets of diagonal and off-diagonal blocks as described by equation (22), which are then used to assemble Jacobian-vector products.

The changes in the structural residuals with respect to the flow variables occur entirely as a function of the lift and moment coefficient variations with the flow variables, thus:

$$\frac{\partial \mathbf{R_s}}{\partial \mathbf{U}} = f\left(\frac{\partial C_L}{\partial \mathbf{U}}, \frac{\partial C_m}{\partial \mathbf{U}}\right) \tag{40}$$

This portion of the Jacobian $\frac{\partial \mathbf{R_s}}{\partial \mathbf{U}}$, above, only has non-zero values in cells which include a face that lies on the surface of the airfoil and only these non-zero values are stored.

The changes in the flow residuals with respect to the structural variables are a function only of the change in the grid motion in response to variations in the structural variables. That is, the structural variables influence both the grid speeds and the orientation of the face normal vectors:

$$\frac{\partial \mathbf{R_f}}{\partial \mathbf{r}} = f\left(\frac{\partial \dot{\mathbf{x}}}{\partial \mathbf{r}}, \frac{\partial \tilde{\mathbf{n}}}{\partial \mathbf{r}}\right) \tag{41}$$

In the present work, mesh motion is achieved by translating and rotating the mesh rigidly. Thus, every face in the mesh changes speed and orientation when the mesh is rotated and translated. Because every face is affected, $\frac{\partial \mathbf{R_f}}{\partial \mathbf{r}}$ has the same size as the diagonal blocks of the first-order Jacobian, $[4\times 4\times N_c]$. For three-dimensional, fixed-wing or deforming-blade aeroelastic analysis, where rigid mesh motion is not feasible, a deforming mesh capability would be required. This deforming mesh motion will add additional dependencies on cell volume and face area in the above portion of the coupled Jacobian. Additionally, since all face-normals change orientation when a rigid mesh rotates, the above component of the Jacobian has non-zero values for all cells. This would not necessarily be the case for a deforming mesh. It should also be noted that an additional dependence of the structural residuals on the structural variables is included in this framework. This dependence arises as a result of the grid motion since the coefficient of lift is itself dependent on the orientation of the face normals on the airfoil surface:

$$\frac{\partial \mathbf{R_s}}{\partial \mathbf{r}} = f\left([\Psi], \frac{\partial C_L}{\partial \tilde{\mathbf{n}}}\frac{\partial \tilde{\mathbf{n}}}{\partial \mathbf{r}}\right) \tag{42}$$

The preceding portion of the full fluid/structure Jacobian retains its $[4\times 4]$ size.

The fully-coupled linear system described in the preceding paragraphs and equations is solved for each non-linear iteration using the GMRES-DC solver, as described above,but with the following modifications. First, both the first-order Jacobian used in the BCGS preconditioner and the second-order Jacobian used in both the Defect Correction step and in the FGMRES algorithm contain the full fluid/structure coupling given by $\partial\mathbf{R_f}/\partial\mathbf{U}$, $\partial\mathbf{R_s}/\partial\mathbf{U}$, $\partial\mathbf{R_f}/\partial\mathbf{r}$, and $\partial\mathbf{R_s}/\partial\mathbf{r}$ as described above. These additions are included in both levels of preconditioning. It has been found that for more difficult aeroelastic problems, convergence could stagnate if structural coupling was neglected from the preconditioning, as had been done in previous work.[12] It is hypothesized that this convergence stagnation occurs primarily because the GMRES solver will exit when only the flow portion of the residual has been reduced (because of preconditioning) and the structural residual has either increased or not decreased at all. This will continue until the structural residual becomes significantly large relative to the flow residual, that it dominates the system, and is itself updated. When this structural update occurs, however, the new orientation of the airfoil leads to an increased flow residual. This pattern repeats as the flow and structural residuals alternately increase and decrease. By including the fluid/structure coupling in the preconditioner, this problem has been eliminated.

Second, the defect-correction step described for the flow-alone solver uses the first-order flow Jacobian; however, for aeroelastic cases, it was found that use of the second-order Jacobian in the defect correction step proved slightly more efficient. As such, equation (27) has been modified to use the 2nd order flow Jacobian, as follows:

$$\left[\frac{V^i}{\Delta\tau_{BCGS}}\underline{\mathbf{I}}+\underline{\mathbf{J}}_{i,1stO}\right](\Delta\mathbf{U}_i^{l+1}-\Delta\mathbf{U}_i^l)=-\mathbf{R}_{\mathbf{TS}}(\mathbf{U})-\sum_{j\neq i}\left[V^j d_i^j\underline{\mathbf{I}}\right]\Delta\mathbf{U}_j^l-\left[\frac{V^i}{\Delta\tau_{FGMRES}}\underline{\mathbf{I}}+\underline{\mathbf{J}}_{i,2ndO}\right]\Delta\mathbf{U}_i^l \tag{43}$$

Finally, just as is the case with the flow Jacobian, a pseudo-time term is added to the diagonal-block of the structural contribution to the Jacobian. This pseudo-time term has the same form as that of the flow, given by equation (17); however, the value for the spectral radius λ that couples the fluid and structure together is not found analytically. Through trial and error, a value of $\lambda = 6.0$ is used for all cases presented in this paper. This value was found to allow for the efficient solution of all the cases considered herein, but is most likely specific to the structural parameters used for these test cases. Future work should include the derivation of an analytic value for the spectral radius of the structural contribution.

G. Implementation

The various time instances in the time-spectral approach are coupled and must be solved simultaneously. This coupling can be implemented serially, whereby a single time-instance is solved at any given moment, and then transmits its update to the next time instance, which is then solved and the process repeated sequentially until all time instances have been updated. However, since the coupling occurs as a source term in the residual, each individual time instance may be solved in parallel with the other time instances. This introduces an additional dimension for achieving parallelism compared to time-implicit computations, where progress in the time dimension is necessarily sequential. In our implementation, two levels of parallelism are introduced, the first in the spatial dimension, and the second in the time dimension where the various time instances are solved by spawning multiple instances of the spatial solver on a parallel computing cluster. The implementation uses MPI for parallelism in the time dimension and OpenMP for additional parallelism in the space dimension. This additional parallelism in time may prove to be particularly enabling with the advent of rapidly expanding massively parallel computing clusters, particularly for cases where parallelism in the spatial dimension has been exhausted (perhaps due to the adequacy of moderate grid sizes).

One of the drawbacks of time-spectral methods is that each time instance must broadcast its entire solution field to all other time instances, which can result in a significant amount of communication. Various strategies for communicating the different time instances to all processors have been investigated. Currently, a Round-Robin approach is implemented, where each processor sends its time instance to a single neighboring processor. The received time instance is added to the time derivative source term on the local processor, and then passed on to the next processor. By repeating this procedure $N-1$ times, where N is the number or time instances, the complete time derivative involving summations from all time instances is accumulated without the requirement of creating a local temporary copy of all the additional time-instance solution vectors or performing any communication intensive broadcast operations.

In the time-spectral method, each time-spectral instance is converged such that the the root-mean-square of its non-linear residual vector is less than 1.0×10^{-11} for the flow alone cases and 1.0×10^{-9} for the aeroelastic cases. Similarly, for the time-implicit results that are presented, at each time step, the root-mean-square of the non-linear residual vector is driven to those same 1.0×10^{-11} and 1.0×10^{-9} levels of precision at each physical time step. While these convergence tolerances would be considered to be overly tight by most industry standards, these levels of precision are necessary for the accuracy studies presented in the results section.

III. Results

A. Fluid-only Test Case

A two-dimensional inviscid flow test case is constructed with the forced, pitching oscillation of a NACA-0012 airfoil at a Mach number of 0.755 and a mean incidence α_0 of 0.016 degrees. The pitching motion is prescribed about the quarter chord of the airfoil as follows:

$$\alpha(t)=\alpha_0+\alpha_A\sin(\omega t). \tag{44}$$

The reduced frequency k_c is equal to 0.0814 and the amplitude α_A is equal to 2.51 degrees. The unstructured, computational mesh consists of 4471 nodes and 8747 triangles. Figure (2) shows the near field mesh. This test case corresponds to the AGARD test case No. 5.[16] Figure (3a) shows the comparison of the lift coefficient versus non-dimensional time between a reference solution obtained using a second-order accurate time-implicit solver with $\Delta t = T/4096$ (where T

denotes the period of airfoil motion) and the time-spectral method with $N = 3, 7, 15$, and 47 time instances. For this case, the lift computed by the time-spectral method with even 1 harmonic or 3 time instances shows reasonable agreement with the reference solution. Figure (3b) shows the comparison of the moment coefficient versus non-dimensional time for the same reference and time-spectral solutions. The moment history contains multiple harmonics and thus is not captured accurately with $N = 3$ in the TS method; in fact, a rather large number, $N = 47$, of time instances is needed to produce near-exact agreement. Figure (3c) shows the comparison of the drag coefficient versus non-dimensional time for those same solutions. It should be noted that, since the results use the Euler equations, the drag shown is pressure drag and does not contain viscous effects. The drag coefficient shows good agreement for the $N = 47$ solution. The most difficult areas of the pressure drag curve to resolve occur around $t = 7$s and $t = 28$s, where the $N = 47$ and reference solutions show a slight reversal. It can be seen that the results of the TS method converge to the reference solution as the number of time instances increases.

1. *Convergence of the Fluid-only Test Case*

While the accuracy of time-spectral method is tantamount, equally important is the efficiency of the solver. The present work is primarily intended to discuss advancements made in solver efficiency and capability as our work with time-spectral methods has proceeded. Figure (4) plots the convergence of the time-spectral method for the AGARD 5 test case using $N = 3$ time instances for the five different solver configurations summarized in Table (1). Similarly, in order to resolve any details for the convergence of the GMRES based solvers, it was necessary to scale the x-axis such that the full convergence history of both BCGS-EX and BCGS-IM are not visible. Table (2) summarizes the convergence of the various solvers that were shown in Figure (4).

Table 2: Convergence of the AGARD 5 Test Case with $N = 3$ Time Instances

Solver	**Non-linear Iter.**	**BCGS Iter.**	**Krylov Vectors**	**Wall-clock Time (s)**
BCGS-EX	106,844	213,687	N/A	2,317
BCGS-IM	8,744	17,487	N/A	214
GMRES-EX	151	9,205	4,564	72
GMRES-IM	25	6,101	2,061	37
GMRES-DC	21	6,800	540	30

It can be seen that the GMRES-DC solver is the most computationally efficient overall. When comparing the BCGS-EX solver to the BCGS-IM solver, the latter reduces the number of non-linear iterations, the number of BCGS iterations, and the wall-clock time all by more than a factor of 10. These gains are accomplished almost exclusively by the increased CFL number allowed by the BCGS-IM. A CFL number of only 0.25 can be used for BCGS-EX, while a CFL number more than $10\times$ higher, i.e. 2.6, can be used for the BCGS-IM solver.

The GMRES-EX method gains over the BCGS-IM by halving the number of BCGS iterations used, while decreasing the number of non-linear iterations by a factor of 58 and the wall-clock time by a factor of almost 3. The gains of GMRES-EX are summarized as follows: each BCGS iteration is more efficient when used as a preconditioner rather than as the main linear solver and GMRES allows the use of more BCGS iterations for every non-linear iteration, reducing overhead costs. The further efficiency increases provided by the GMRES-IM solver over the GMRES-EX solver occur because of the implicit coupling among time instances, in the same way that BCGS-IM gains over BCGS-EX. Emphasis should be placed on just how much the use of GMRES improves the efficiency of the solution: the GMRES-IM solver cuts the wall-clock time by over 5.7 times and the BCGS iterations by over 2.8 times when compared to the BCGS-IM solver; the GMRES-DC solver saves a further 7 seconds or 19% of the wall-clock time over the GMRES-IM solver.

The gains of adding a defect-correction step to the GMRES preconditioner accrue mainly because of the decreased numbers of non-linear iterations and Krylov vectors, which decreases the total overhead cost of convergence to the final solution. This is evidenced by the number of BCGS iterations increasing slightly as compared to the GMRES-IM method while the number of non-linear iterations and the number of Krylov vectors used decreases.

Results for the various solvers when $N = 15$ are presented in Table (3) except BCGS-EX is omitted as it is so clearly non-competitive. As can be seen, the same trends hold for the $N = 15$ solutions as were discussed for the $N = 3$ solutions. Although not presented herein, the same trends continue to hold for increasing numbers of time

Table 3: Convergence of the AGARD 5 Test Case with $N = 15$ Time Instances

Solver	Non-linear Iter.	BCGS Iter.	Krylov Vectors	Wall-clock Time (s)
BCGS-IM	12,992	27,653	N/A	2,430
GMRES-EX	125	22,197	11,098	873
GMRES-IM	124	11,551	3,922	440
GMRES-DC	22	14,525	383	423

instances. Specifically, the best GMRES solver, GMRES-DC takes one-fifth or less time than BCGS-IM for greater numbers of time instances. In fact, it has been found that the BCGS-IM solver cannot efficiently converge solutions with more than fifty time instances, as the CFL number needed to maintain diagonal dominance in the BCGS solver becomes prohibitively small.

It should be reiterated that all cases for both $N = 3$ and $N = 15$ have been run on a single 16-core cluster node. This identical hardware means that each time instance of the $N = 3$ solutions is run on 5 cores, whereas each time instance of the $N = 15$ solution is run on a single core. The wall-clock time to convergence of the $N = 15$ solution can be greatly decreased if each time instance is run on 5 cores, as was done in the $N = 3$ case. This will have required a total of 75 cores for the $N = 15$ case.

With the GMRES-DC solver established as the most computationally efficient, we now compare its wall-clock time required to complete a converged solution for different numbers of time instances in Figure (5) with different numbers of time steps per period for the time-implicit method. All time-implicit solutions presented in this plot have been run for 5 complete periods of airfoil motion in order to obtain a purely periodic solution. Although, not included in the present work, examination of the accuracy of the time-implicit solutions show that at least 256 time steps per per period are needed to replicate the reference solution visually. As can be seen from this figure, time-spectral solutions using up to 27 time instances can be converged in about the same or less time than the 256 time steps per period time-implicit solution. The result that 47 time instances are needed to reproduce the reference solution and that the $N = 47$ time-spectral solution requires $2.4\times$ the wall-clock time of the 256 time steps per period time-implicit solution to converge, however, indicates that future work remains to make the time-spectral method more efficient than the time-implicit method for high-harmonic content test cases.

It should be noted that all time-implicit solutions are run on a single 16-core cluster node. Time-spectral solutions for $N = 3, .., 15$ are run on a single 16-core cluster node. The time-spectral solutions for 17 or more time instances are run on as many nodes as are necessary such that each time instance uses one CPU core. For example, the 39 time-instance solution is run on $3\times$ 16-core cluster nodes with each time instance using one core; thus 9 cores among the three nodes are left idle. It should also be noted that the wall-clock time of the time-spectral solutions can be decreased by utilizing additional computational resources. In this way, even the $N = 47$ time instances, time-spectral solution can still be completed in less wall-clock time than the time-implicit solution.

B. Aeroelastic Results

To examine the convergence of the aeroelastic solver, the AGARD5 test case presented above is modified so that it responds aeroelastically. The same NACA-0012 unstructured, computational mesh of 4471 nodes and 8747 triangles is used. The freestream Mach number of 0.755 and mean incidence α_0 of 0.016 degrees are retained. The airfoil again undergoes a forced pitching oscillation of $\alpha_A = 2.51^o$, however, this oscillation occurs about the airfoil elastic axis, instead of the quarter-chord as was true for the AGARD5 test case. In addition to the forced pitching, the airfoil is allowed to respond aeroelastically in both pitch and plunge such that the true pitch angle of the airfoil at a given time is the combination of the prescribed and aeroelastic contributions to pitch. The reduced frequency of the forced pitching is $k_c = 0.0814$ just as in the AGARD5 test case.

The structural parameters for this and all other aeroelastic cases considered are as follows:

V_f	$= 0.50$
x_α	$= 1.8$
r_α^2	$= 3.48$
$\frac{\omega_h}{\omega_\alpha}$	$= 1.0$
μ	$= 64.0$
a	$= -2.0$

The quantity a is the non-dimensional elastic axis location along the chord of the airfoil measured from the mid-chord of the airfoil when it is in the neutral position. Since it is non-dimensionalized by the semi-chord of the airfoil, the elastic axis is located half a chord length ahead of the leading edge of the airfoil in this particular case. As was noted above, this is the axis about which the forced pitching occurs.

Figure (6) compares the lift, moment, and pressure drag coefficients of a single period of motion of this test case for different numbers of time-spectral time instances to a reference time-implicit solution with $4,096$ time steps per period. It appears that all the time-spectral solutions show good agreement in lift and moment, while only the 63 time instance solution shows visually exact agreement in pressure drag. However, visual inspection alone is not always adequate to assess accuracy. Figure (7) plots the RMS error of the solutions in lift, moment, and drag coefficients as well as pitch and plunge for different number of time steps per period and time instances. Since peak lift and moment coefficients are on the order of 0.1, lift and moment are considered accurate when RMS error in these quantities is less than 0.0002 or 0.2% which occurs with 47 time instances and 256 time steps per period. Peak drag, pitch, and plunge are all on the order of 0.01 so the solution is precise when the error in these quantities is also less than 0.2% or 2×10^{-5}. An accurate solution in drag occurs for 512 time steps per period and 63 time instances. An accurate solution in pitch occurs for 128 time steps per period and 23 time instances, while a precise solution in plunge occurs using 256 time steps per period and 47 time instances. For this case, the drag coefficient is the most limiting quantity.

Another trend that should be noted on Figure (7) is the exponential convergence of the accuracy of the time-spectral method as time instances are added. Specifically, this trend is noted above 31 time instances. Exponential convergence is the expected trend for spectral methods and means that the rate of error decrease of the method accelerates as more time instances are added. This contrasts with the time-implicit method which has linear accuracy convergence. For a second-order accurate method in time, the slope of the error convergence is expected to be 2.0. For this test case, the slope from 512 to 1024 time steps per period for all five errors ranges between 1.946 for plunge and 2.025 for drag, while the greatest slope for the time spectral method is 6.907 and occurs for C_D between 87 and 95 time instances.

Figure (8) compares the wall-clock time required to complete a converged solution for different numbers of time instances for the time-spectral method and for different numbers of time steps per period for the time-implicit method. Two series are included for each solution type. The time implicit data are all run on one cluster node with 16 CPU cores. The two time-implicit series plot the time needed for a purely-periodic solution (solid line, squares) and 4 periods of the solution (dashed line, diamonds). It should be noted that 19 periods are needed before the 64 time step per period solution attains pure periodicity while 33 periods are needed before the 1024 time steps per period solution becomes purely periodic. The determination of periodicity is made by comparing the successive periods of the time-implicit solution to the reference solution; the first period where the first four non-zero digits of the error in all the three force coefficients and both structural variables match the next period is used as the period at which the solution attains a purely periodic state. The two time-spectral series plot the time needed for convergence when one CPU core is used per time instance (solid, triangles) and when the solution is calculated on a single, 16 CPU core node (assuming linear scaling, dashed, inverted triangles). Figures (9) and (10) combine the data contained in Figures (7) and (8) in a way that might be more accessible. Specifically, wall-clock time is plotted as a function of the error in drag with the points labeled with their numbers of time steps per period or time instances. Drag error was chosen since it is the most limiting quantity, as noted above. For solutions that are precise in the limiting drag coefficient, we compare the 512 time step per period time-implicit solution to the 63 time instance time-spectral solution. To run enough periods to reach a purely periodic time-implicit solution (27 periods) $9,680$ seconds are required. If, however, only 4 periods are used, only $1,434$ seconds are needed. To converge a time-spectral solution with 63 time instances, $5,665$ seconds are needed when using 1 time instance per CPU core. If, however, the same 16 computational cores as were used for the time-implicit solution were used, $22,306$ seconds would be required.

In previous work,[17] it has been shown that both the freestream Mach number and reduced frequency of motion have a significant impact on the convergence of time-spectral methods for the flow alone cases. As such, it only follows to examine the impacts that both these variable have on the convergence time of aeroelastic, time-spectral methods. For all of the cases presented in the next two subsections, the same precision limits that are used in the baseline case are used, i.e. less than 2×10^{-4} for lift and moment coefficients and less than 2×10^{-5} for drag coefficient, pitch angle, and plunge height. Additionally, all the wall-clock times given in the subsequent plots use 1 computational core per time instance. Finally, for each of the alternate Mach number and reduced frequency cases, a reference time-implicit solution with $4,096$ time steps per period and run for 100 periods is used to compute the temporal error in the time-spectral solutions.

1. Convergence Variation with freestream Mach number

The aeroelastic test case run above is re-run with all of the same input parameters except that the freestream Mach number is varied. The new cases use freestream Mach numbers of 0.555, 0.705, and 0.805, or -0.20. -0.05, and +0.05 added to the baseline Mach number, respectively.

Figure (11) plots the error and the wall-clock time for $M_\infty = 0.555$. A precise solution in all measures can be found using only 7 time instances for this case and requires only 276 seconds to compute. Exponential convergence is seen in plunge and pitch between 3 and 9 time instances before the error convergence flattens. It is believed this error flattening occurs either because the 1×10^{-9} convergence criteria limits accuracy or the accuracy of the time-spectral solutions has exceeded the accuracy of the reference, time-implicit solution. Further investigation is needed to make this determination.

Figures (12) and (13) are analogous to Figure (11) but for $M_\infty = 0.705$ and $M_\infty = 0.805$, respectively. For $M_\infty = 0.705$, drag error limits the accuracy of the solution. A precise solution occurs for 47 time instances which takes $2,263$ seconds to compute on 47 cores. Exponential error convergence occurs for greater than 31 time instances. Error in pitch and plunge flatten for more than 79 time instances, for one of the two reasons noted above. $M_\infty = 0.805$ is a more difficult case with strong shock waves; however, a precise solution can still be found using 47 time instances which requires $4,349$ seconds to converge. For this test case, exponential convergence is not seen; further, error flattens while it is still quite large, possibly because of insufficient accuracy in the reference solution used to compute the error. It should be noted that the although both of these solutions require 47 time instances, almost twice as much wall-clock time is needed for the $M_\infty = 0.805$ solution.

The manner in which the freestream Mach number affects convergence efficiency in now examined. Figure (14) plots convergence time versus Mach number for the three cases presented here and the baseline case. As can be seen, for a given number of time instances, convergence time grows at an accelerating rate as Mach number increases. Although more data points between Mach 0.555 and 0.705 are needed to confirm the following hypothesis, it appears that once shock waves begin occurring in the flow, which usually happens around $M_\infty = 0.7$, convergence time greatly increases.

2. Convergence Variation with Reduced Frequency

To examine the effect of reduced frequency on convergence, the baseline, aeroelastic test case examined above is recomputed for $\frac{1}{2}\times$ and $2\times$ the baseline reduced frequency, i.e. $k_c = 0.0407$ and $k_c = 0.1628$, respectively. Error plots for these two frequencies are given in Figures (15) and (16). In the $k_c = 0.0407$ case, plunge limits the precision of the solution and 79 times instances, which run in $4,598$ seconds, are needed. As in previous cases, exponential convergence begins to occur after 31 time instances.

When $k_c = 0.1628$, even 95 time instances do not produce a precise solution in the lift coefficient, C_L. Because this case also uses, by far, the most wall-clock time to converge, solutions with even higher numbers of time instances are not attempted. It is noteworthy that exponential error convergence is not yet clearly present for this case; whether this is because of imprecision in the reference solution or because even more time instances are required is not known and is a subject for future examination.

As was the case with Mach number, reduced frequency has a great impact on convergence time and solver efficiency. Figure (17) plots the convergence time versus the reduced frequency for the three reduced frequencies presented in the present work. Unlike with Mach number, there is no clear frequency after which convergence time gets noticeably worse; however, increasing reduced frequency has a much more noticeable impact when the number of time instances is high, as evidence by the slope of the $N = 79$ being so much greater than the slope of the $N = 47$ line. As with Mach number, further investigation is necessary to understand how and why reduced frequency impacts convergence time in this way.

IV. Conclusions and Future Work

In the present work, a robust and efficient aeroelastic time-spectral solver has been further developed and refined as compared to that presented in our previous work.[12,17] The flexible variant of the Krylov subspace, generalized minimal residual method is used to solve the fully-coupled linear system of aeroelastic, time-spectral equations for each non-linear iteration. The strong coupling afforded by including the contribution of both the fluid and structural equations at all levels of solution allows for the efficient and robust solution of aeroelastic problems with large number of time instances. In the present work, solutions using as many as 127 time instances have been obtained. In future work, more difficult problems requiring even more time-instances will be investigated.

Also in the present work, the accuracy and efficiency of the aeroelastic time-spectral method has been demonstrated for problems with a variety of freestream Mach numbers and reduced frequencies. For many of the cases presented herein, the expected exponential error convergence of spectral methods is noted. For other cases, however, exponential error convergence is not apparent and further investigation is required to ascertain why the theoretically correct behavior is not obtained.

The aeroelastic, time-spectral solver presented herein is more than an order of magnitude more efficient than the solver that was presented at this same conference a year ago;[11] however, it is still far from optimal. For an optimal solver, the wall-clock time to convergence should remain constant for time-spectral solutions with any number of time instances using one CPU core per time instance. The current variant of the solver uses significantly more than the optimal amount of wall-clock time for large numbers of time instances when 1 core is used for each time instance. This trend worsens as the freestream Mach number and reduced frequency of forced pitching increase, especially for high transonic Mach numbers. However, despite the non-optimality of the solver, the time-spectral method can solve most aeroelastic problems in less wall-clock time than time-implicit method, both because the time-implicit method requires so many periods of solution to arrive at a purely periodic solution and because of the additional parallelism in time afforded by the time spectral method.

When comparing time-spectral and time-implicit methods, the following should be kept in mind. The time-spectral method becomes stiff at a) high freestream Mach number because of the presence of shock waves, b) high reduced frequency because the various time instances become more closely coupled, and c) large numbers of time instances because the frequency of the highest harmonic that can be resolved increases with the number of time instances and the thereby decreases the pseudo-time step size. Exacerbating this situation, it is usually these high Mach number, high reduced frequency cases that require large numbers of time instances to obtain an accurate solution. On the other hand, time-implicit methods scale more slowly than linearly when additional time steps are added per period because a smaller time step tends to make the Jacobian more diagonally dominant and thus each time step easier to solve. These two aspects combine to demonstrate how challenging it is to develop a time-spectral solver that can outperform a time-implicit solver for problems requiring very large numbers of time instances and time steps per period. It should also be noted that how the efficiency of the two solvers compare may well change when the solvers are converged to engineering accuracy instead of the restrictive tolerances used in the current work.

Despite the solver efficiency gains that have been made as compared to a year ago, solver efficiency must be improved further still. First and foremost, a more efficient preconditioner, linear, agglomerated multigrid will be implemented in the near future. While the expected efficiency gains from multigrid are large, other preconditioning techniques that completely eliminate the need for diagonal dominance, such as incomplete LU factorization, will also be tested. Finally, these methods need not be used in isolation; a preconditioner that combines the two might prove to offer the efficiency gains that are being sought. Exploration of better preconditioning methods is where the pursuit of an optimal time-spectral solver leads next.

References

[1]Hall, K. C., Thomas, J. P., and Clark, W. S., "Computation of Unsteady Nonlinear Flows in Cascades Using a Harmonic Balance Technique," *AIAA Journal*, Vol. 40, No. 5, 2002, pp. 879–886.

[2]McMullen, M., Jameson, A., and Alonso, J. J., "Acceleration of Convergence to a Periodic Steady State in Turbomachineary Flows," AIAA Paper 2001-0152, Jan. 2001.

[3]McMullen, M., Jameson, A., and Alonso, J. J., "Application of a Non-Linear Frequency Domain Solver to the Euler and Navier-Stokes Equations," AIAA Paper 2002-0120, Jan. 2002.

[4]Gopinath, A. K. and Jameson, A., "Time Spectral Method for Periodic Unsteady Computations over Two- and Three- Dimensional Bodies," AIAA Paper 2005-1220, Jan. 2005.

[5]van der Weide, E., Gopinath, A. K., and Jameson, A., "Turbomachineary Applications with the Time Spectral Method," AIAA Paper 2005-4905, June 2005.

[6]Lee, K.-H., Alonso, J. J., and van der Weide, E., "Mesh Adaptation Criteria for Unsteady Periodic Flows Using a Discrete Adjoint Time-Spectral Formulation," AIAA paper 2006-0692, Jan. 2006.

[7]Sankaran, S., Gopinath, A., Weide, E. V. D., Tomlin, C., and Jameson, A., "Aerodynamics and Flight Control of Flapping Wing Flight Vehicles: A Preliminary Computational Study," AIAA 2005-0841, Jan. 2005.

[8]Choi, S., Potsdam, M., Lee, K., Iaccarino, G., and Alonso, J. J., "Helicopter Rotor Design Using a Time-Spectral and Adjoint-Based Method," AIAA 2008-5810, Sep. 2008.

[9]Choi, S. and Datta, A., "CFD Prediction of Rotor Loads using Time-Spectral Method and Exact Fluid-Structure Interface," AIAA 2008-7325, Aug. 2008.

[10]Saad, Y., *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, 1996.

[11]Mundis, N. L. and Mavriplis, D. J., "Quasi-Periodic Time Spectral Method for Aeroelastic Flutter Analysis," AIAA paper 2013-0638, Jan. 2013.

[12]Mundis, N. L., Mavriplis, D. J., and Sitaraman, J., "Quasi-Periodic Time-spectral Methods for Flutter and Gust Response," 69th Forum of the American Helicopter Society, AHS International, Alexandria, VA, May 2013.

[13]Canuto, C., Hussaini, M. Y., Quarteroni, A., and Zang, T. A., *Spectral Methods in Fluid Dynamics*, Springer, 1987.

[14]Hesthaven, J., Gottlieb, S., and Gottlieb, D., *Spectral Methods for Time-Dependent Problems*, Cambridge Monographs on Applied and Computational Mathematics, 2007.

[15]Sicot, F., Puigt, G., and Montagnac, M., "Block-Jacobi Implicit Algorithm for the Time Spectral Method," *AIAA Journal*, Vol. 46, No. 12, 2008, pp. 3080–3089.

[16]AGARD, "Compendium of Unsteady Aerodynamic Measurements," Tech. Rep. No. 702, 1982.

[17]Mundis, N. L. and Mavriplis, D. J., "GMRES applied to the Time Spectral and Quasi-periodic Time Spectral Methods," AIAA paper 2013-3084, June 2013.

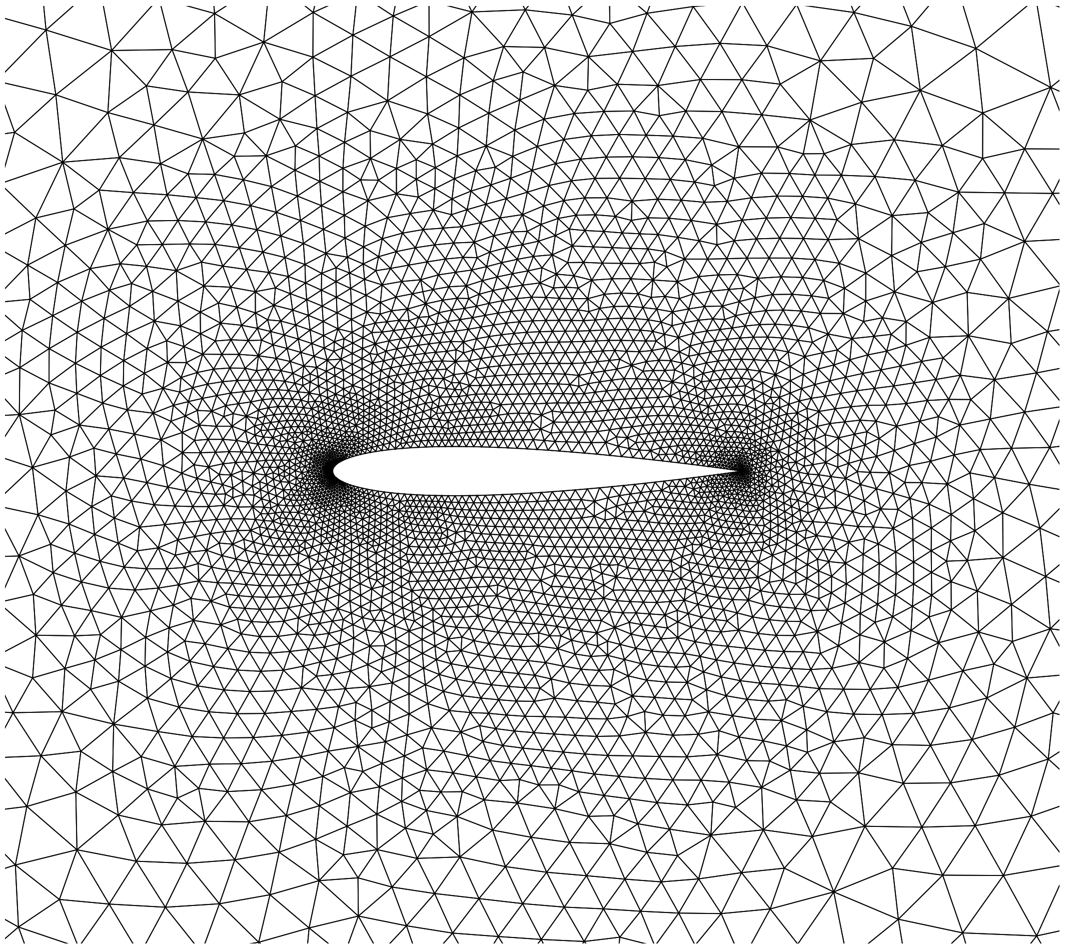
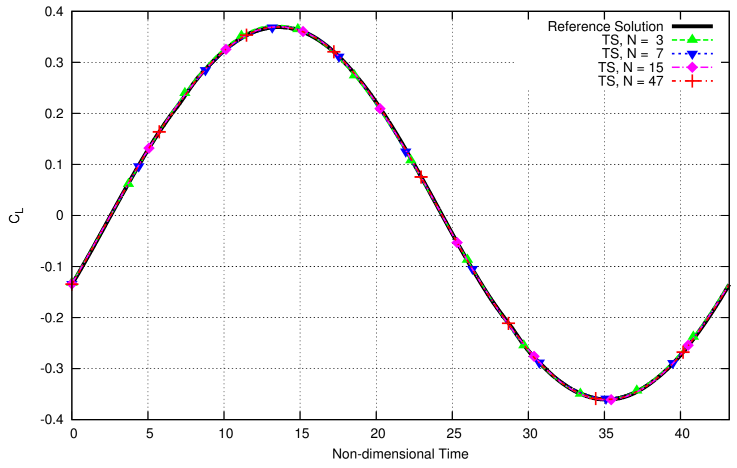
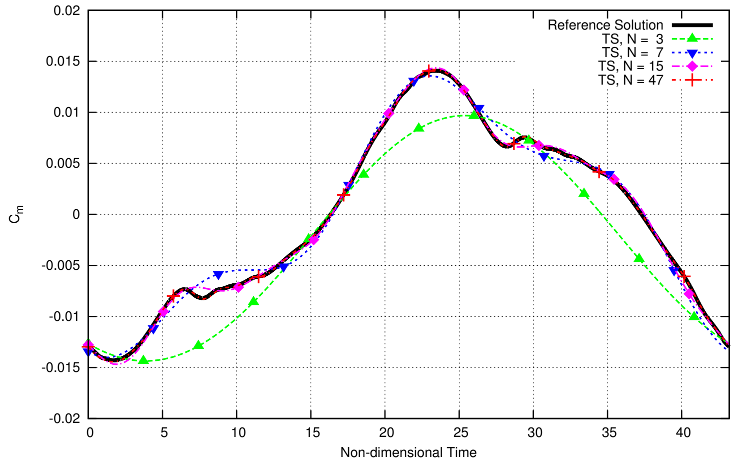


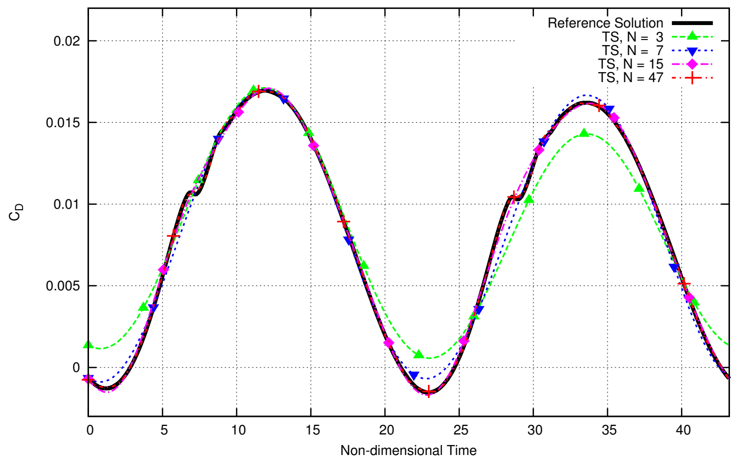
Figure 2: Near field mesh for the NACA-0012 airfoil



(a)



(b)



(c)

Figure 3: Comparison of computed lift coefficient (a), moment coefficient (b), and pressure drag coefficient (c) using the time-spectral method to a reference, time-implicit solution for the AGARD5 test case

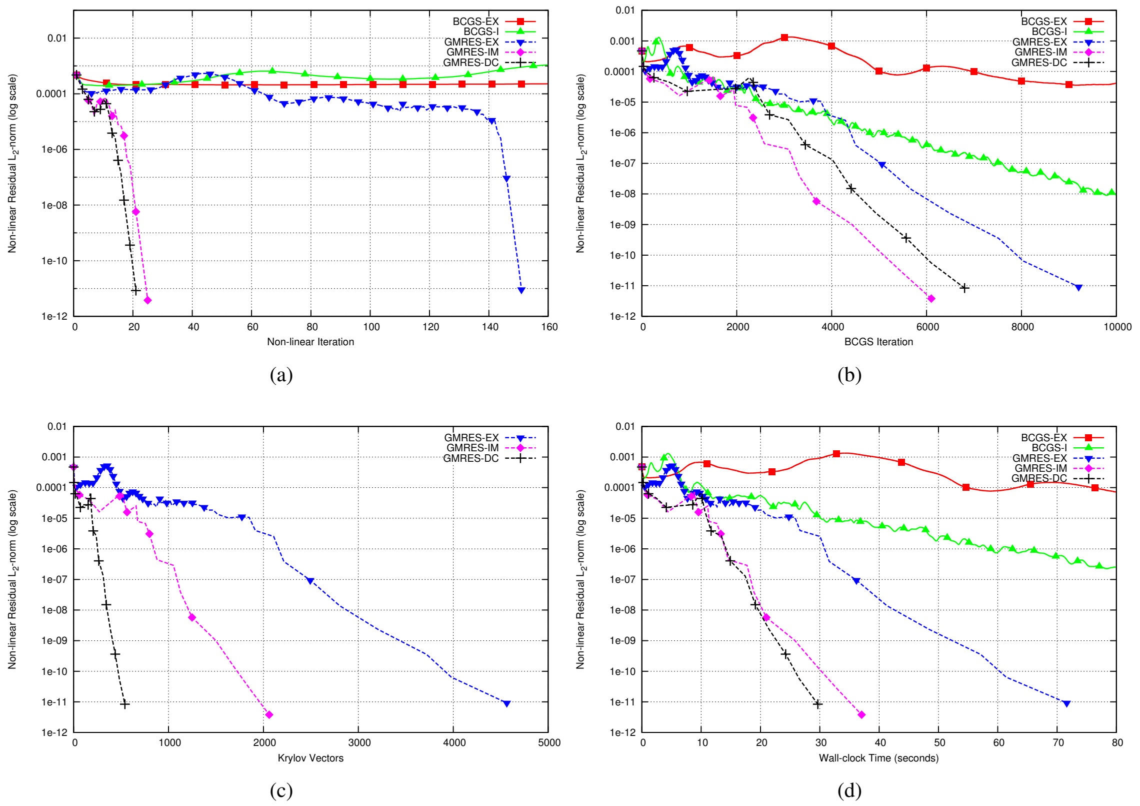


Figure 4: Comparison of convergence for Test Case 2 with 3 time-spectral time instances using the 5 different solvers discussed with Non-linear Iterations (a), BCGS Iterations (b), Krylov vectors (c), and wall-clock time (d) on the x-axis

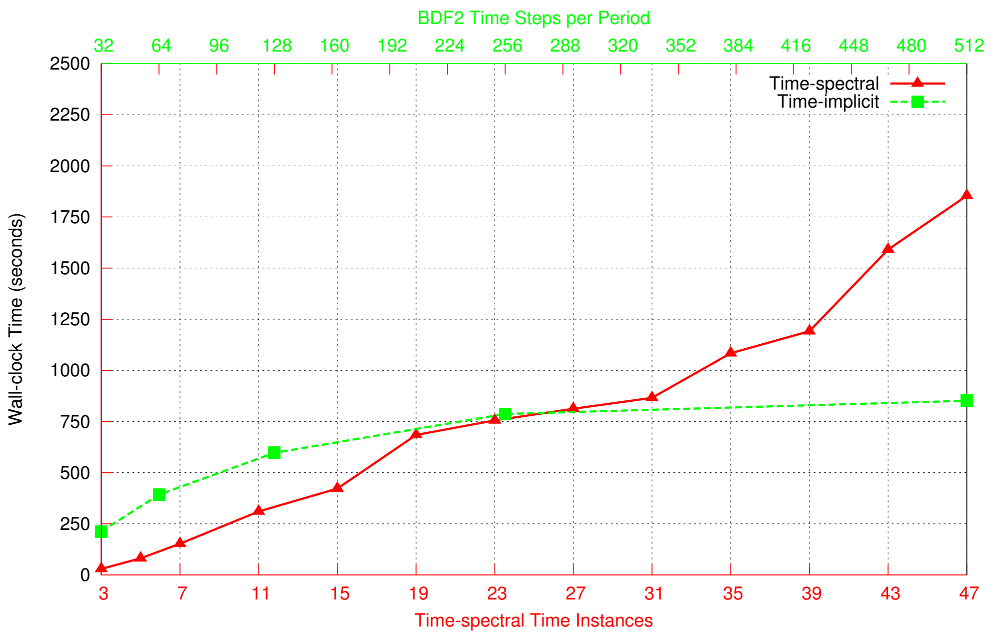
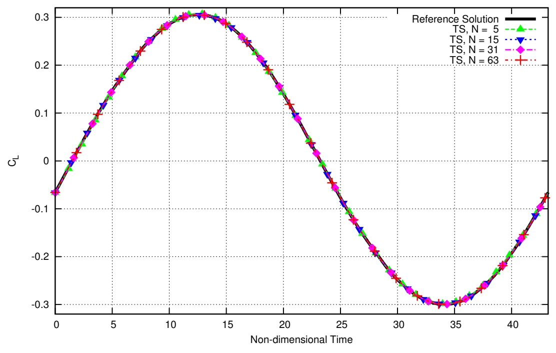
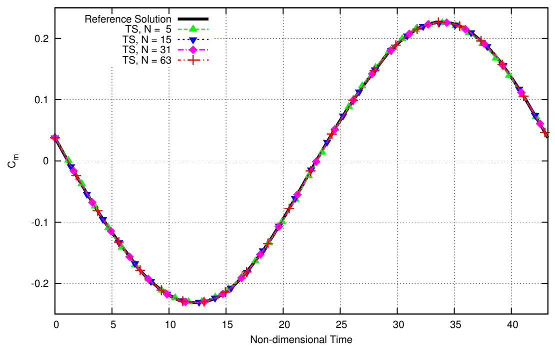


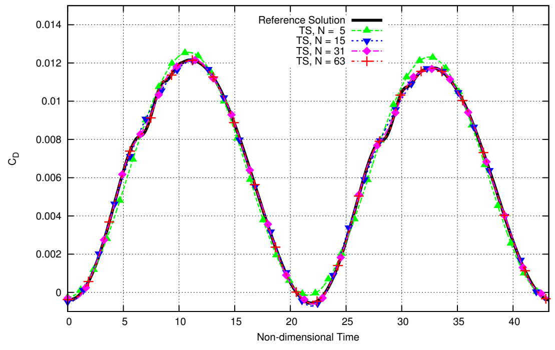
Figure 5: Comparison of wall-clock time to convergence for time-implicit and time-spectral methods with different numbers of time steps per period (top axis) and time instances (bottom axis) for the AGARD 5 test case



(a)



(b)



(c)

Figure 6: Comparison of computed lift coefficient (a), moment coefficient (b), and pressure drag coefficient (c) using the time-spectral method to a reference, time-implicit solution for first aeroelastic test case

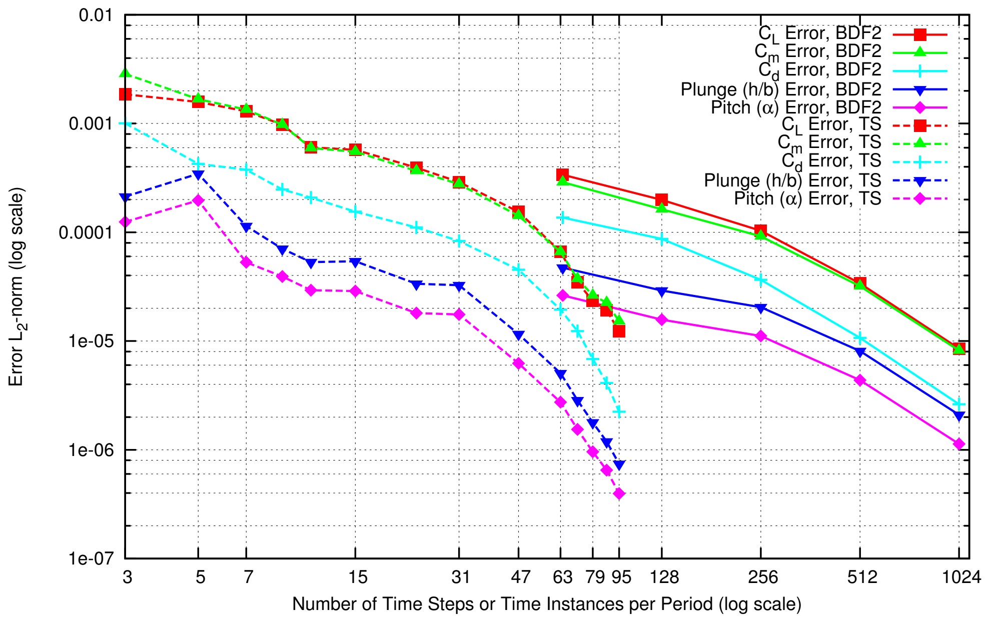


Figure 7: Aeroelastic test case 1 error for time-implicit and time-spectral methods with varying numbers of time steps and time instances per period ($M_\infty = 0.755, k_c = 0.0814$)

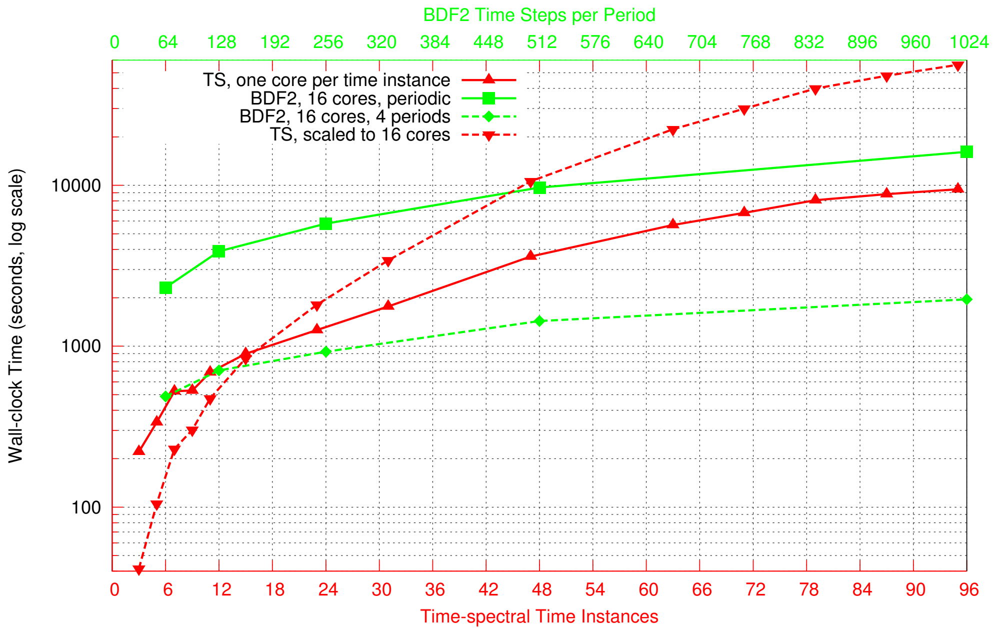


Figure 8: Comparison of wall-clock time to convergence for time-implicit and time-spectral methods with different numbers of time steps per period (top axis) and time instances (bottom axis) for the first aeroelastic test case

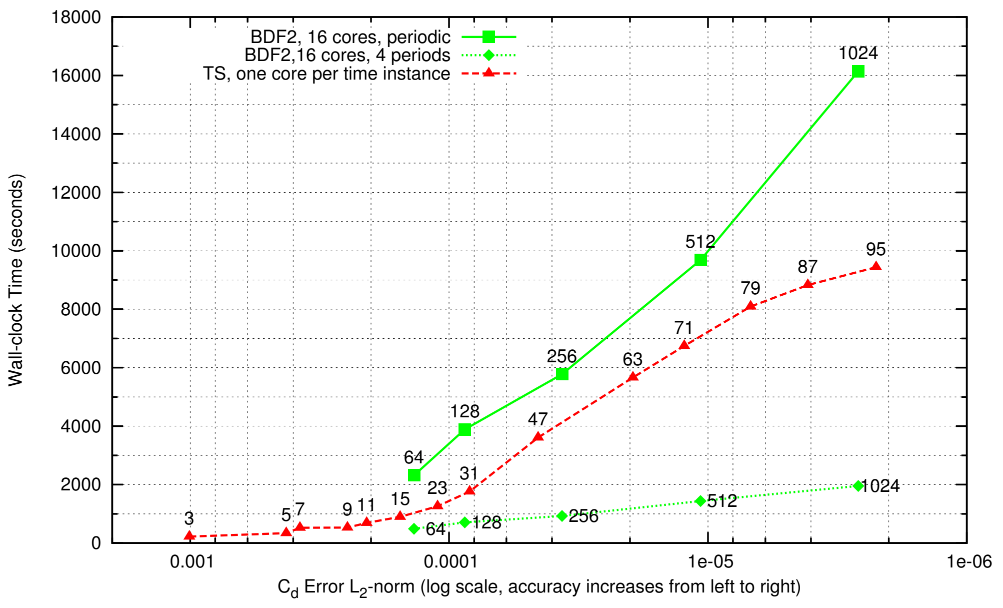


Figure 9: Comparison of wall-clock time to convergence for time-implicit and time-spectral methods as a function of the $L_2 - norm$ of the error in C_D

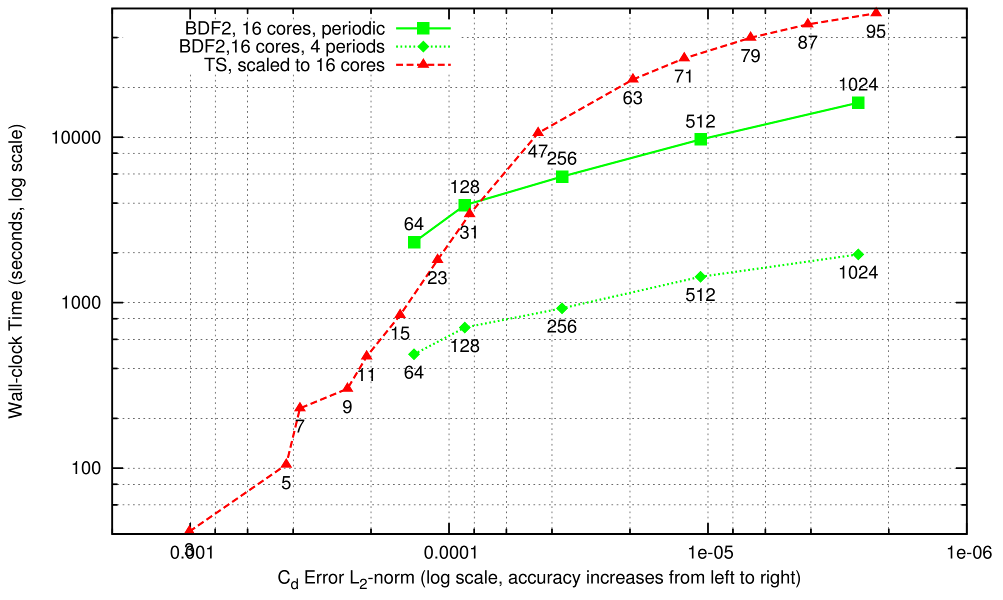


Figure 10: Comparison of wall-clock time to convergence for time-implicit and time-spectral methods as a function of the $L_2 - norm$ of the error in C_D with the time-spectral method linearly scaled to the same hardware as the time-implicit results

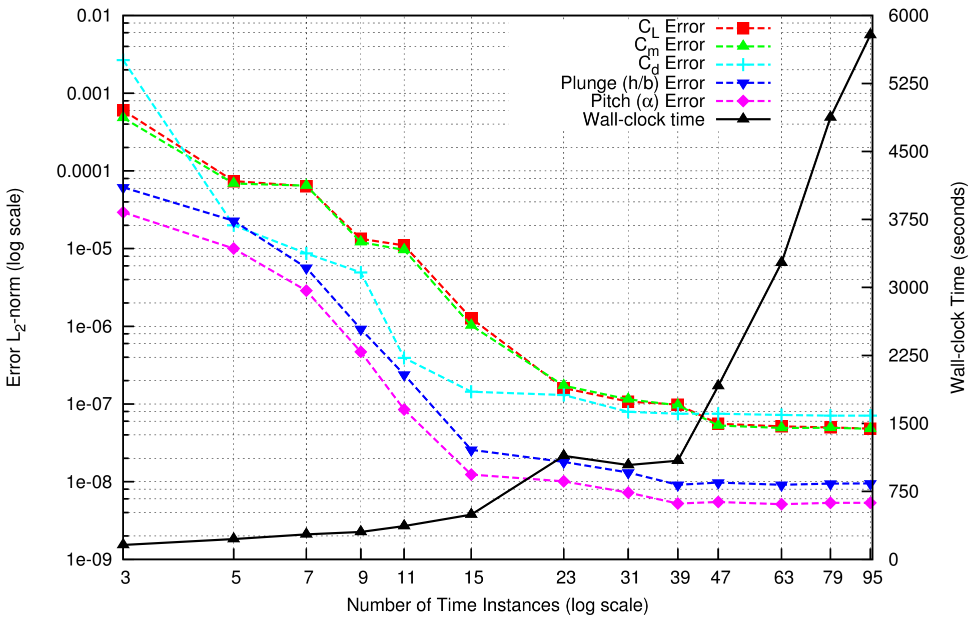


Figure 11: Error (compared to a reference time-implicit solution with 4,096 time steps per period, left axis) and wall-clock time to convergence (one core per time instance, right axis) for the aeroelastic time-spectral method as a function of the number of time instances ($M_\infty = 0.555, k_c = 0.0814$)

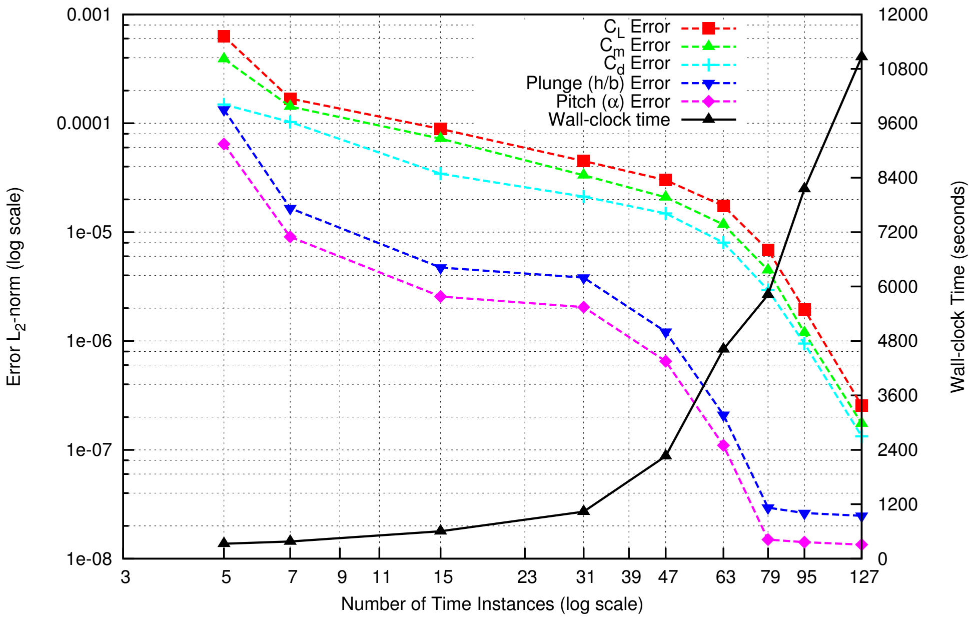


Figure 12: Error (compared to a reference time-implicit solution with $4,096$ time steps per period, left axis) and wall-clock time to convergence (one core per time instance, right axis) for the aeroelastic time-spectral method as a function of the number of time instances ($M_\infty = 0.705, k_c = 0.0814$)

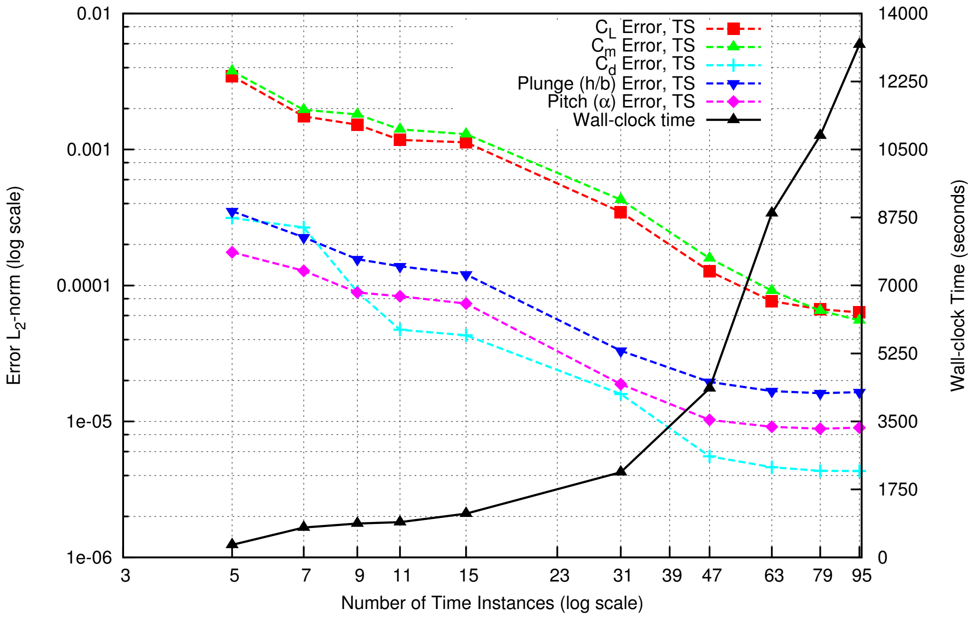


Figure 13: Error (compared to a reference time-implicit solution with $4,096$ time steps per period, left axis) and wall-clock time to convergence (one core per time instance, right axis) for the aeroelastic time-spectral method as a function of the number of time instances ($M_\infty = 0.805, k_c = 0.0814$)

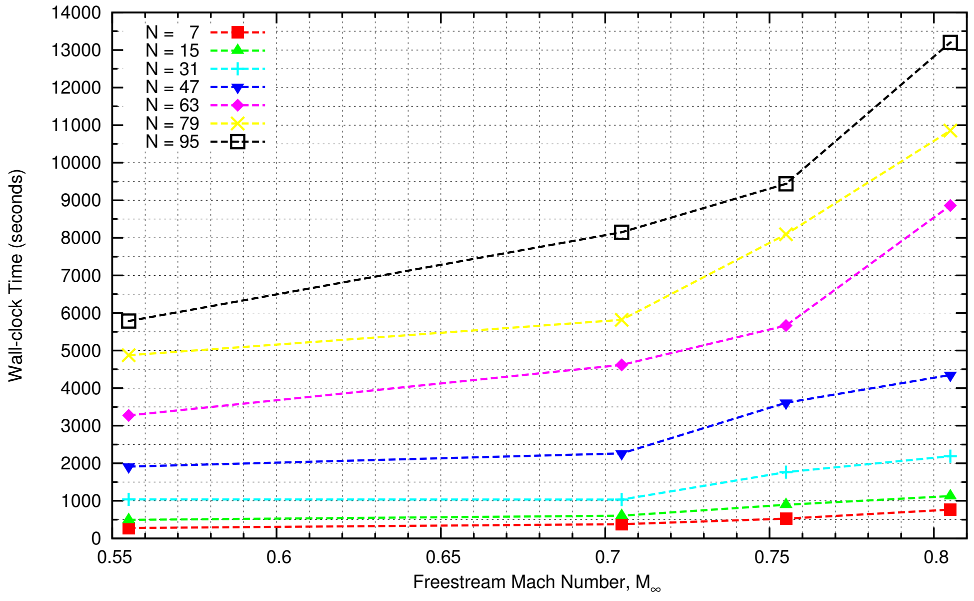


Figure 14: Wall-clock time to convergence (one core per time instance) versus freestream Mach number M_∞ for the aeroelastic time-spectral method with different numbers of time instances ($k_c = 0.0814$)

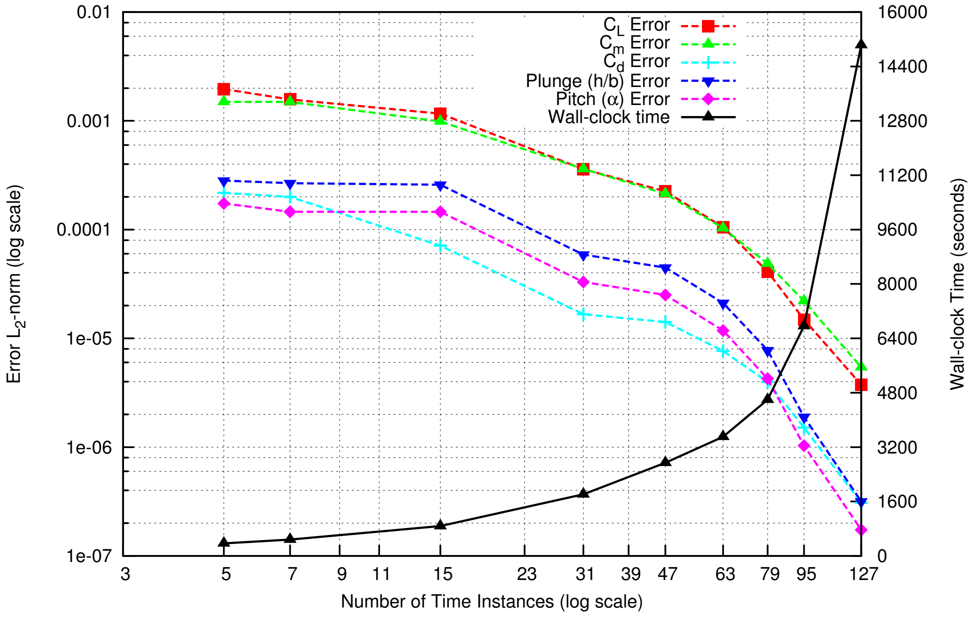


Figure 15: Error (compared to a reference time-implicit solution with 4,096 time steps per period, left axis) and wall-clock time to convergence (one core per time instance, right axis) for the aeroelastic time-spectral method as a function of the number of time instances ($M_\infty = 0.755, k_c = 0.0407$)

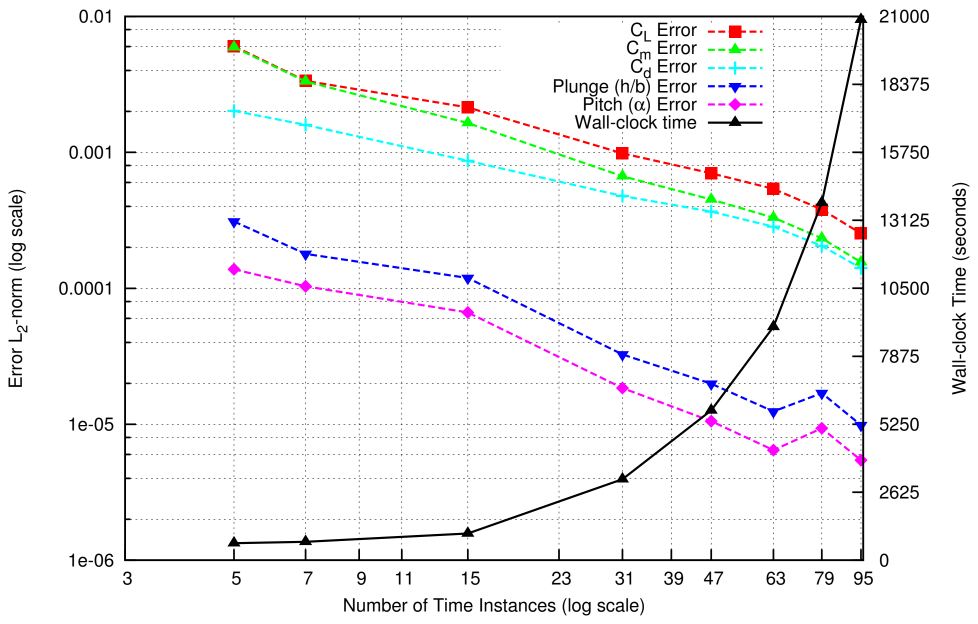


Figure 16: Error (compared to a reference time-implicit solution with 4,096 time steps per period, left axis) and wall-clock time to convergence (one core per time instance, right axis) for the aeroelastic time-spectral method as a function of the number of time instances ($M_\infty = 0.755, k_c = 0.1628$)

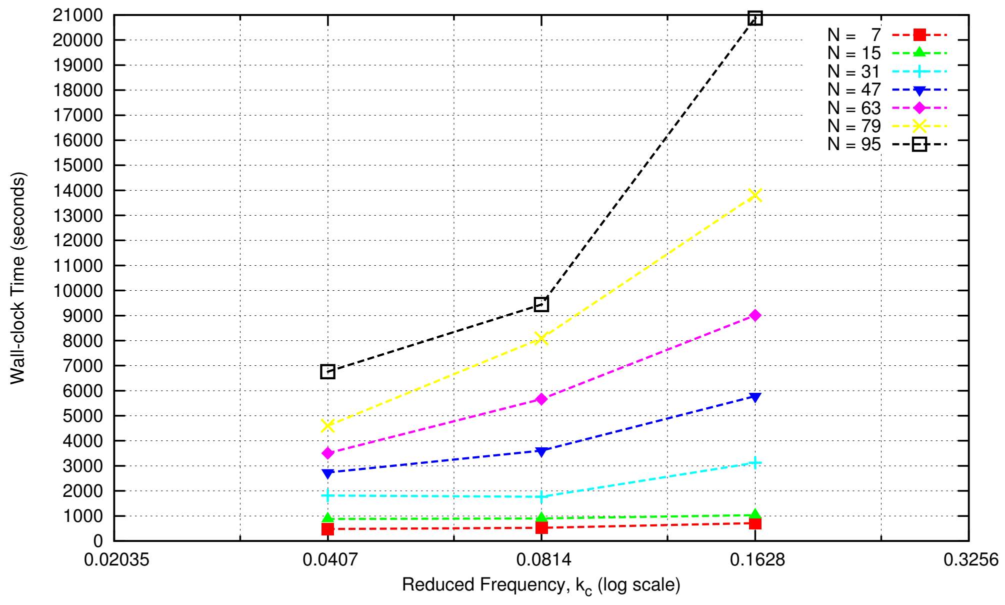


Figure 17: Wall-clock time to convergence (one core per time instance) versus reduced frequency k_c for the aeroelastic time-spectral method with different numbers of time instances ($M_\infty = 0.755$)