GMRES applied to the Time-spectral and Quasi-periodic Time-spectral Methods

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

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

The time-spectral method applied to the Euler equations theoretically offers significant computational savings for purely periodic problems when compared to standard time-implicit methods. A recently developed quasi-periodic time-spectral (BDFTS) method extends the time-spectral method to problems with fast periodic content and slow mean flow transients, which should lead to faster solution of these types of problems as well. However, attaining superior efficiency with TS or BDFTS 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. In order to increase the efficiency of these solvers, and to improve robustness, particularly for large numbers of time instances, the TS and BDFTS methods are reworked such that the Generalized Minimal Residual Method (GMRES) is used to solve the implicit linear system over all coupled time instances. 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. The present work primarily targets cases that require many time instances to demonstrate the capabilities of the solver.

Both the time-spectral and harmonic-balance methods are based on the use of 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 problem. The time-spectral method was 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]

In previous work, we have introduced a hybrid BDF/time-spectral approach (BDFTS) which aims to simulate quasi-periodic flows with slow transients combined with relatively fast periodic content using global BDF time step sizes of the order of the period length, while making use of the properties of the time-spectral approach to capture accurate details of the periodic flow components.[10,11,12,13] Both the pure TS and the BDFTS methods have previously been applied to problems with spectral content contained only in the first few harmonics,[10,11,12,13] because the number of time instances that could be solved simultaneously has been limited in the past. Previously, a stationary iterative method, such as Jacobi or Gauss-Seidel, has been used to solve the implicit, linear system. Because these methods require the Jacobian to be diagonally dominant, they rapidly become inefficient as the number of time-spectral instances increases. To alleviate this problem, a method which does not require diagonal dominance must instead be used to solve the implicit, linear system. The Krylov subspace, Generalized Minimal Residual method has been

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

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

chosen for this purpose. By using GMRES, the disparate time instances are much more strongly coupled than in previous approaches. To accelerate the convergence of GMRES, a stationary iterative method, specifically block-colored Gauss-Seidel (BCGS), is used as a preconditioner. This BCGS preconditioned GMRES represents an efficient linear solver that allows time-spectral based methods to be applied to a much wider variety of problems than was previously possible, including those problems that combine both fast and slow periodic content.

In the following sections we present the necessary components of the time-spectral discretization and the application of GMRES to this discretization. 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 linearization of the time-spectral Euler equations and the solution of the resultant linear system using BCGS and GMRES. Following, we present results for two test cases, both of which contain the influence of higher harmonic content. We examine the efficiency of the different linear solvers in depth to ascertain how efficiency increases are obtained and how the time-spectral solver might be made even more efficient in the future. Finally, we discuss our future work to improve the efficiencies and capabilities of the TS/BDFTS methods further.

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 $\underline{\mathbf{T}}$ is the left-eigenvector matrix, $\underline{\Lambda}$ is the eigenvalue matrix, and $\underline{\mathbf{T}}^{\mathbf{-1}}$ 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 instances 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[14, 15]

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

where

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

for an even number of time instances and

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

for an odd number of time instances. Next, substitute equation (12) 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.

As was mentioned in the introduction, the time-spectral method has been extended to quasi-period problems in previous work;[10,11,12,13] however, since none of the test cases presented herein make use of the BDFTS method, a derivation of it has been omitted. This derivation is present in all of the works last referenced.

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 instance:

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

The impact of this restriction can be reduced by resorting to an implicit approach in pseudo-time. Such an approach has been derived in reference[16] 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:[16]

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

where a diagonal pseudo-time term can be included as shown for enhanced diagonal dominance of the Jacobian matrix. Equation (17) provides a suitable estimate for the local pseudo-time step required for diagonal dominance. Theoretically, use of this pseudo-time step should allow the same CFL number to be used regardless of the number of time instances. However, it has been found that a larger CFL number can be used when the number of time instances is small. As the number of time instances is increased, the CFL number asymptotically approaches a constant value such that it must no longer be decreased as more time instances are added.

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:

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

It should be readily apparent that the full Jacobian is of a size that could not be stored on modern computers for computational mesh sizes 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 (in 2D) $[4 \times 4 \times N_c]$ where N_c is the number of elements in the computational mesh, and an array of its off-diagonal blocks, which is 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 the exact Jacobian-vector products as required at each linear-solver iteration by storing three sets of diagonal and off-diagonal blocks, as shown in equation (22), which are then used to assemble Jacobian-vector products. 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:

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

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

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

where the block size corresponds to an entire 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 of 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:

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

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 all elements of each individual color sequentially, 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).

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 for any given color, some elements will have information from the previous iteration $\mathbf{x}_i$ while others will 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.

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
```

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 fully-implicit BCGS affords over methods that are only spatially implicit, the time-spectral Euler equations still become difficult to solve as the number of time instances or the reduced frequency of the problem increases. With an increasing number of time instances and/or higher reduced frequencies, 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 Minimal 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[17] 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, **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 three 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 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. 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, we in fact 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 used by the FGMRES algorithm itself and another in the first-order Jacobian 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 (if at all) than the FGMRES pseudo-time step. Conversely, the pseudo-time step used in the FGMRES algorithm grows rapidly such that the diagonal pseudo-time 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 $[\mathbf{A}]$ matrix, given by equation (20), uses the first-order spatial Jacobian with a pseudo-time step that is different, and smaller, than the pseudo-time step used in FGMRES. Additionally, $[\mathbf{A}]$ 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 $\Delta\mathbf{U}_i^{l+1}$, 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. 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 only comes through a source term, 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.

One of the drawbacks of the TS/BDFTS 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 solution vector to a single neighboring processor. The received time-instance solution vector 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 1×10^{-11}. 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 that same 1×10^{-11} level of precision. While this convergence tolerance would be considered to be overly tight by most industry standards, we feel it is good practice for algorithm development and scientific inquiry.

III. Results

A. Test Case 1: 1st and 7th Harmonic Pitching Motion

A two-dimensional inviscid flow test case is constructed with the forced, pitching oscillation of the NACA-0012 airfoil at two reduced frequencies $k_{c_1}=0.02$ and $k_{c_2}=0.14$. It should be noted that the second frequency is $7\times$ the first; in other words, motion occurs in the first and seventh harmonics. This case is run at a freestream Mach number $M_\infty=0.50$ and pitches about the airfoil quarter chord. The amplitude of pitching in the first harmonic is $\alpha_1=2.5^o$ and $\alpha_7=1.0^o$ in the seventh harmonic; additionally, $\alpha_0=0.0^o$. This motion is given by the following equation:

$$\alpha(t)=\alpha_0+\alpha_1\sin(\omega t)+\alpha_7\sin(7\omega t). \tag{28}$$

The angular frequency is related to the reduced frequency as follows:

$$\omega=\frac{2U_\infty k_c}{c} \tag{29}$$

where U_∞ is the freestream velocity and c is the airfoil chord length. The unstructured, computational mesh consists of 8747 triangles. Figure (1) shows this near field mesh. Figure (2) plots the lift and moment coefficients as functions of non-dimensional time for a reference, time-implicit solution which uses 512 time steps per period and time-spectral solutions using $N = 3, 11, 13$, and 15 time instances. As can be seen, $N = 15$ time instances matches the reference time-implicit solution quite well for both lift and moment coefficients. This is the expected result as the number of time instances required is $N = H \times 2 + 1$ where H is the highest harmonic of content that can be resolved by these N time instances. Since lift is primarily a function of angle-of-attack, and the 7th harmonic is the highest harmonic of pitching motion, 15 time instances can accurately resolve the lift. In this case, the moment coefficient is primarily a function of lift; thus 15 time instances can also resolve the moment coefficient.

Figure (3) plots the convergence history of the non-linear residual versus both the non-linear iteration count and wall-clock time for $N = 3, 7, 11$, and 15 time instances obtained using the GMRES-IM solver, which was the best performing solver for most numbers of time instances. As can be seen, all cases converge very smoothly. Additionally, this case converges along very similar non-linear iteration paths for all four time instance numbers shown. Wall-clock time convergence scales along expected lines since all cases were run on the same hardware. Finally, Figure (4) plots the wall-clock time needed to converge this test case for $32, 64, 128, 256$, and 512 time steps per period (top axis) using the time-implicit method and for $3, 5, 7, 9, 11, 13$, and 15 time instances (bottom axis) using the time-spectral method. As can be seen, the time-spectral method, even using the maximum of $N = 15$ time instances, converges more quickly than the time-implicit method in all cases.

1. *Performance comparison of solver variants*

Figure (5) plots the convergence of the time-spectral method for the this test case using $N = 3$ time instances for the five different solver configurations summarized in Table (1). In order to resolve any details for 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. However, it can readily be seen how much more efficient the GMRES based solvers are. Also note that all solvers were run on identical hardware. Table (2) summarizes the convergence of the various solvers that are shown in Figure (5), including at what point both BCGS-EX and BCGS-IM eventually reach convergence.

Table 2. Convergence of the First Test Case with $N = 3$ Time Instances

Solver	Non-linear Iter.	BCGS Iter.	Krylov Vectors	Wall-clock Time (s)
BCGS-EX	6,699	21,442	N/A	161
BCGS-IM	2,777	16,870	N/A	98
GMRES-EX	12	2,907	651	12
GMRES-IM	12	2,986	501	13
GMRES-DC	11	2,683	447	13

It can be seen that the GMRES-based solvers are by far the fastest in every measure. In comparison to the BCGS-EX solver, the BCGS-IM solver reduces the number of non-linear iterations by a factor of 2.4, the number of BCGS iterations by a factor of 1.3, and the wall-clock time by a factor of 1.6. These gains are accomplished almost exclusively by the increased CFL number allowed by BCGS-IM. A CFL number of 7.2 can be used for BCGS-EX, while a CFL number $4.17\times$ higher, i.e. 30.0, can be used for the BCGS-IM solver.

The GMRES-EX method gains over the BCGS-IM by using 17% the number of BCGS iterations, while decreasing the number of non-linear iterations by a factor of $231\times$ and the wall-clock time by a factor of about $8.2\times$. The gains of GMRES-EX first occur by allowing each BCGS iteration to use a higher CFL number, i.e. 20.0, than can be used by BCGS-EX as the linear solver, itself. This solver also allows each non-linear iteration to make use of more BCGS iterations. This aspect decreases the overhead cost of each BCGS iteration, which decreases the total wall-clock time even further. Because this test case using three time instances is well-conditioned for time-spectral solution, the more complicated GMRES-IM and GMRES-DC solvers exhibit few efficiency gains. Most notably, the number of Krylov vectors used is reduced, but the increased overhead costs (because of increased MPI communications) of these two methods more than account for this computational savings when it comes to wall-clock time.

Results for the various solvers when $N = 15$ are presented in Table (3) except BCGS-EX is omitted since it is so clearly inefficient.

Table 3. Convergence of the First Test Case with $N = 15$ Time Instances

Solver	Non-linear Iter.	BCGS Iter.	Krylov Vectors	Wall-clock Time (s)
BCGS-IM	16,730	38,566	N/A	3,550
GMRES-EX	12	8,687	2,502	316
GMRES-IM	13	5,469	1,832	204
GMRES-DC	13	5,859	1,111	196

The convergence of this test case, run with $N = 15$ time instances, follows similar trends to $N = 3$. Because of the increased stiffness of the problem introduced by the additional time instances, the GMRES-IM and GMRES-DC solvers both show efficiency gains. This benefit occurs in reduced numbers of BCGS iterations and Krylov vectors primarily as a result of the implicit coupling among the time-instances present in these solvers. Overall, it can be seen that the addition of GMRES as the linear solver greatly increases the efficiency of time-spectral methods.

B. Test Case 2: AGARD test case No. 5

The second test case uses the same computational mesh as the first. A forced, pitching oscillation of the NACA-0012 airfoil at a Mach number of 0.755 and a mean incidence α_0 of 0.016 degrees is prescribed at the quarter chord of the airfoil as follows:

$$\alpha(t) = \alpha_0 + \alpha_A \sin(\omega t). \tag{30}$$

The reduced frequency k_c is equal to 0.0814 and the amplitude α_A is equal to 2.51 degrees. This test case corresponds to the AGARD test case No. 5. Figure (6a) 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 generated by the time-spectral method with even 1 harmonic or 3 time instances shows reasonable agreement with the reference solution. Figure (6b) 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 (6c) 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 converges to the reference solution as the number of time instances increases.

Figure (7) compares the wall-clock time required to complete a converged solution for different numbers of time instances for the time-spectral method using the GMRES-DC solver and for different numbers of time steps per period for the time-implicit method. 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. 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 in fact 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 complicated-flow test cases like the AGARD 5 test case.

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.

If the convergence of the first and second test cases are compared, it is obvious that the first test case converges much more quickly than the second case when using the time-spectral method. Two factors are the main causes of the increased convergence time of the second test case: first, the transonic nature of the freestream Mach number of the second case, and second, the higher reduced frequency of the first harmonic of its airfoil motion.

The transonic Mach number naturally means that shock waves will be present in the flow. For this particular test case, the shock waves develop on the the upper and lower surfaces at different moments in time, periodically depending on the angle-of attack of the airfoil at each moment. These moving, disappearing and reappearing shock waves complicate the flow solution and introduce higher frequency (possibly non-smooth) content into the time dependent solution. The increased reduced frequency of the second test case also leads to convergence degradation since the coupling between the time-spectral instances is increased, as determined by the coefficients in equations (13) and (14), which are directly proportional to the reduced frequency. It can also be seen from equations (17) and (18) that an increase in the reduced frequency results in a smaller pseudo-time step size for a given CFL number. Despite these phenomena that cause the second test case to be more difficult to converge time-spectrally, time-spectral methods still prove competitive when compared to time-implicit methods.

1. Convergence Study

Figure (8) 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), just as was done for the first test case. 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 (4) summarizes the convergence of the various solvers that were shown in Figure (8). For the second test case, it can be seen that

Table 4. 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

the GMRES-DC solver is the most computationally efficient in three of the four measures. 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 time-implicit BCGS. 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 were already outlined for the first test case but are henceforth 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, again, because of the implicit coupling among time instances.

The gains of adding a defect-correction step to the GMRES preconditioner, which were not significant for the first test case, 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 compare 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 (5) except BCGS-EX is again omitted. As can be seen once again, the same trends hold for the $N = 15$ solutions as were discussed for the $N = 3$ solutions. 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.

Table 5. 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

2. *An Investigation of the Effects of Additional Preconditioning*

To establish a better idea of how the time-spectral solver might be further improved, particularly for stiffer cases such as the second test case, the convergence effects of the amount of preconditioning performed on each Krylov vector is examined. It is expected that increasing the number of defect-correction steps will reduce the number of Krylov vectors needed to converge a solution. Figure (9) plots the non-linear iteration and Krylov vector convergence for $N = 3$ time instances using $2, 4, 6$, and 8 defect-correction steps per Krylov vector. This data is then summarized with additional data points in Table (6).

Table 6. Comparison of the Nature of Convergence for Test Case 2 with 3 Time Instances

Defect Correction Steps	Non-linear Iter.	BCGS Iter.	Krylov Vectors	Wall-clock Time (s)
2	27	7,529	1,385	35.9
3	24	7,044	994	32.7
4	24	6,923	767	31.8
5	22	6,937	659	30.5
6	21	6,800	540	29.7
7	21	7,529	509	32.1
8	21	7,902	476	34.7
9	21	7,994	431	35.6
10	21	8,376	405	36.5

As can be seen from this figure and table, the number of cumulative Krylov vectors needed to produce a converged solution can be reduced substantially when more preconditioning iterations are used. Additionally, added preconditioning also moderately reduces the number of non-linear iterations needed. However, both the cumulative number of BCGS iterations used and the wall-clock time are little affected. These results indicate that one way to make the solver more efficient is to use a better stationary iterative method than BCGS, such that each stationary iterative solve accomplishes a greater residual reduction, i.e. is more computationally efficient.

Figure (10) plots the same information as the last figure except for $N = 15$ time instances using $6, 12, 18$, and 24 defect-correction iterations per Krylov vector, and Table (7) summarizes these results with more data points. Similarly, Figure (11) is a comparison of the convergence for $N = 31$ time instances using $5, 15, 25$, and 35 defect-correction preconditioning iterations per Krylov vector. Again, this data is summarized in Table (8), with additional data points.

The same Krylov vector trend as was seen with the $N = 3$ case holds for the $N = 15$ and $N = 31$ cases. It can also be seen that the number of BCGS iterations increases as the number of time instances increases, while the number of non-linear iterations needed remains roughly the same (for the 3 and 15 time instance cases) when enough preconditioning iterations are used, and the number of cumulative Krylov vectors needed does not necessarily grow as more time instances are utilized. When examining the data for the $N = 31$ case, it is seen that the number of non-linear iterations decreases monotonically at first, then spikes at 30 defect-correction steps per Krylov vector, after which it again decreases monotonically. For some as yet unknown reason, when 30 or more defect-correction steps are used, the GMRES CFL number growth rate must be reduced for the solution to be convergent. Why this reduction in CFL number growth occurs and how non-linear convergence can be restored warrants future investigation.

Table 7. Comparison of the Nature of Convergence for Test Case 2 with 15 Time Instances

Defect Correction Steps	Non-linear Iter.	BCGS Iter.	Krylov Vectors	Wall-clock Time (s)
6	37	13,351	1,055	538
9	30	13,592	713	524
12	29	14,311	565	461
15	25	14,668	463	478
18	22	14,525	383	423
21	22	14,963	338	479
24	21	14,596	289	453
27	22	15,940	282	468

Table 8. Comparison of the Nature of Convergence for Test Case 2 with 31 Time Instances

Defect Correction Steps	Non-linear Iter.	BCGS Iter.	Krylov Vectors	Wall-clock Time (s)
5	106	15,650	1,490	888
10	58	16,270	794	1,033
15	51	16,342	530	865
20	49	17,259	423	935
25	40	18,342	361	974
30	79	19,432	350	1,068
35	66	20,320	308	1,103
40	41	19,899	250	1,232

It can also be observed that the number of cumulative BCGS iterations needed for solution convergence is inversely proportional to BCGS CFL number. Table (9) summarizes this proportionality in its fourth column which contains the product of the CFL number and the cumulative BCGS iteration count. The inverse proportionality is shown by the numbers in this column being roughly equal. This result indicates that another avenue to obtain a better preconditioner would eliminate this CFL number constraint altogether through the use of a more global preconditioner, such as ILU.

Table 9. Proportionality of BCGS iteration count to BCGS CFL number

Time Instances	BCGS CFL	Fewest BCGS Iter. (NBCGS)	BCGS CFL $\times$ NBCGS
3	2.6	6,923	18,000
15	1.4	13,351	18,691
31	1.2	15,650	18,780

IV. Conclusions and Future Work

The GMRES-DC solver described in the present work has been successfully applied to the time-spectral Euler equations. It has been demonstrated in this work that the current solver can converge time-spectral test problems using as many as $N = 47$ time instances. In fact, in work that was very recently presented,[18] solution of $N = 79$ time-instances was demonstrated on a fully-coupled time-spectral, aeroelastic problem.[18] In addition, it has been shown that the GMRES-DC solver is much more efficient than the time-spectral solvers presented in our own previous work.[10,11,12,13] For the first-test case, a time-spectral solution was obtained much more quickly than a time-implicit solution of similar accuracy. However, for the second test case, the time-spectral solver is not more efficient than the time-implicit solver when a similar level of accuracy is sought. The decreased efficiency of the time-spectral solver when applied to this test case results from its transonic nature as well the higher reduced frequency of its first harmonic; more study is needed to quantify the effects of Mach number and reduced frequency on the cost of equivalent accuracy of time-spectral methods when compared to time-implicit methods.

To further improve the efficiency of the GMRES solver, more efficient preconditioners than simple BCGS should be considered. Since it has been demonstrated that adding defect-correction steps mainly improves efficiency by reducing the number of Krylov vectors while leaving the total number of BCGS iterations needed for convergence roughly unchanged, an obvious improvement could be obtained by replacing BCGS with a more efficient stationary iterative method, such as linear multigrid, or resorting to global preconditioners such as ILU. Another possibility that may increase preconditioning effectiveness would be to use the second-order spatial Jacobian in the defect-correction step instead of the first-order Jacobian approximation currently used. Of course, there are innumerable combinations of all the aforementioned preconditioners that could be combined to produce even further increases in efficiency.

Even if it proves impossible to make the time-spectral method more efficient than the time-implicit method for all practical time-accurate problems when run on the same number of computational cores, the time-spectral method will in all likelihood be able to converge these cases in less wall-clock time by taking advantage of the additional parallelism in time that the time-spectral method affords. This is especially likely since all the current computing trends seem to indicate that, while processor clock speed is not going to increase substantially, processor core count will, and with it, opportunities for increased parallelism. For a time-implicit solver, the advantages of additional computational cores will eventually deteriorate as parallelism in the spatial dimension is exhausted.

Overall, the use of GMRES as the linear solver for the implicit time-spectral method has addressed the main limitations of the method heretofore: the ability to handle large numbers of time instances and robustness when applied to stiff problems. With the practical limitations addressed, future work will be focused on efficiency increases through better preconditioner selection.

V. Acknowledgments

This work was partly funded by the Alfred Gessow Rotorcraft Center through a partnership between the University of Maryland and the University of Wyoming. Partial support was also provided under AFOSR STTR contract FA9550-13-C-0015.

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]Yang, Z. and Mavriplis, D. J., "Time spectral method for quasi-periodic unsteady computation on unstructured meshes." AIAA Paper 2010-5034, June 2010.

[11]Yang, Z., Mavriplis, D. J., and Sitaraman, J., "Prediction of Helicopter Maneuver Loads Using BDF/Time Spectral Method on Unstructured Meshes," AIAA Paper 2011-1122, June 2011.

[12]Mavriplis, D. J., Yang, Z., and Mundis, N. L., "Extensions of Time Spectral Methods for Practical Rotorcraft Problems," AIAA paper 2012-0423, Jan. 2012.

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

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

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

[16]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.

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

[18]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.

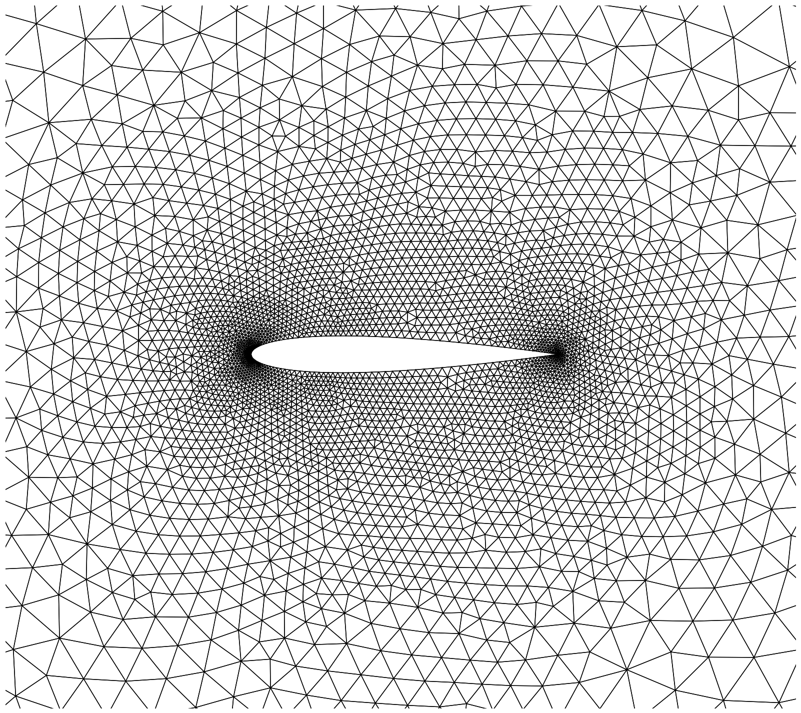
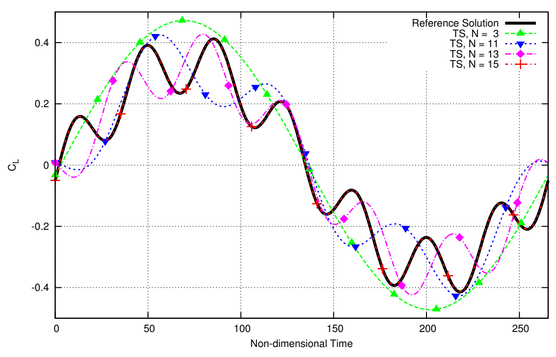
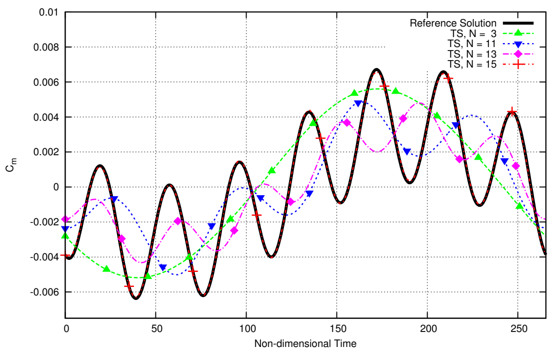


Figure 1. Near field mesh for the NACA-0012 airfoil

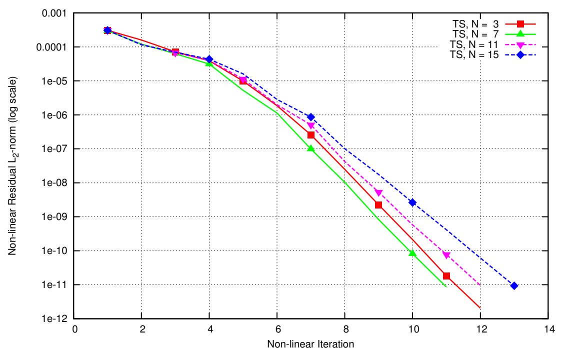


(a)

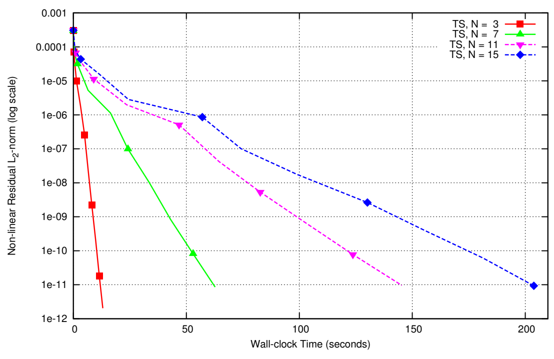


(b)

Figure 2. Comparison of computed lift coefficient (a) and moment coefficient (b) using the time-spectral method to a reference, time-implicit solution for Test Case 1



(a)



(b)

Figure 3. Comparison of convergence of Test Case 1 using different number of time-spectral time instances with Non-linear Iterations (a) and Wall-clock time (b) on the x-axis

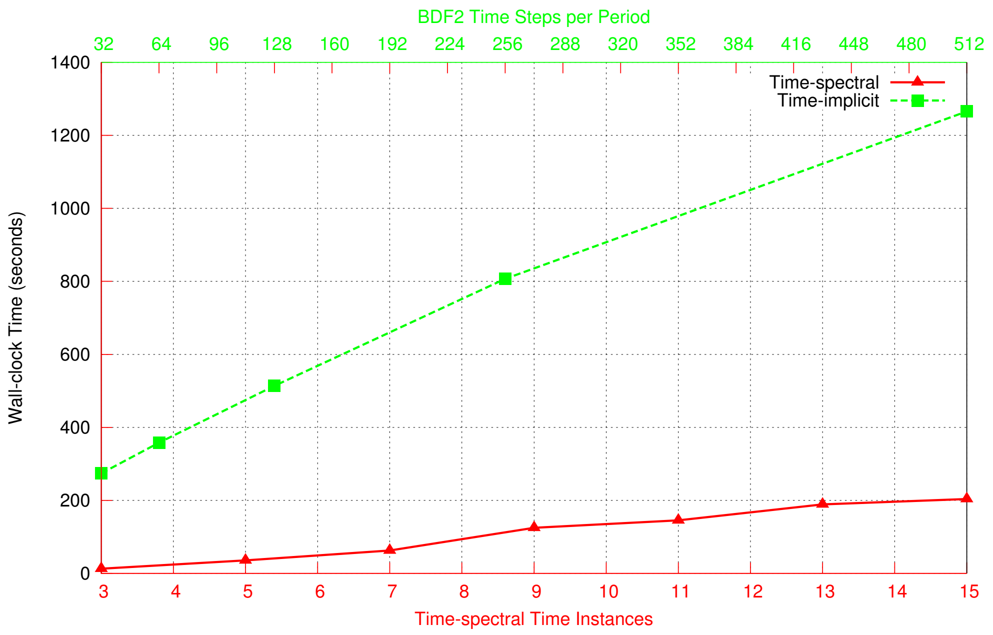
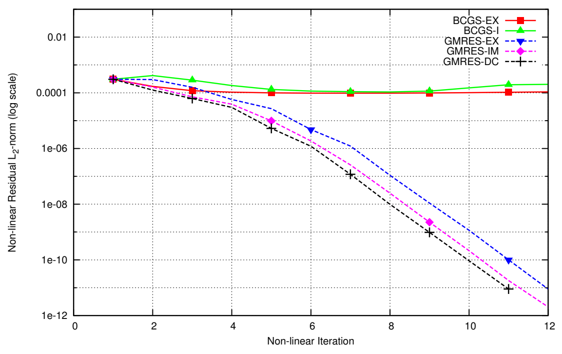
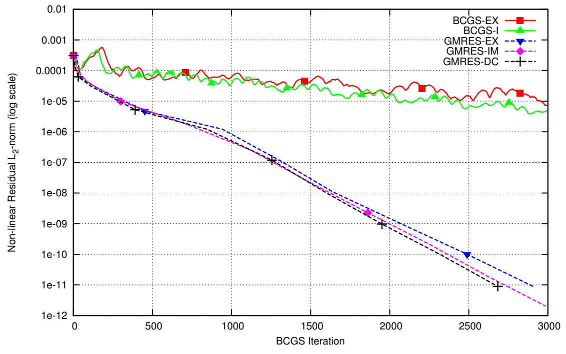


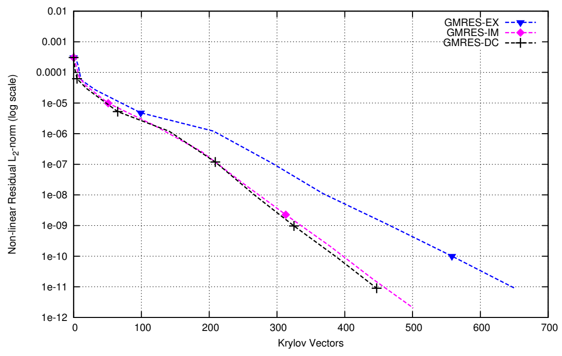
Figure 4. Comparison of Wall-clock time to convergence for time-implicit and time-spectral with different numbers of time steps per period (top axis) and time instances (bottom axis) for Test Case 1



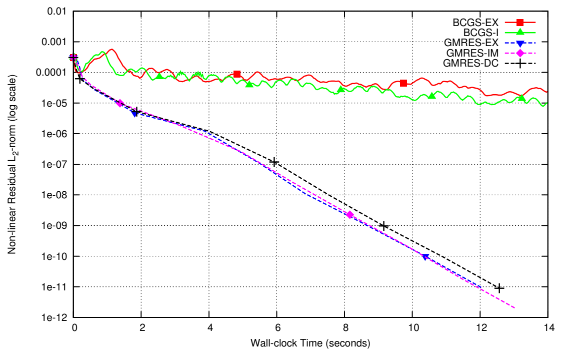
(a)



(b)

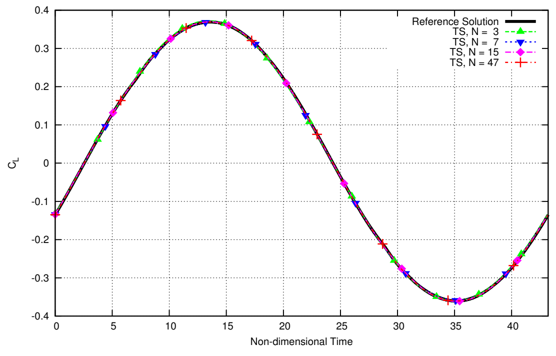


(c)

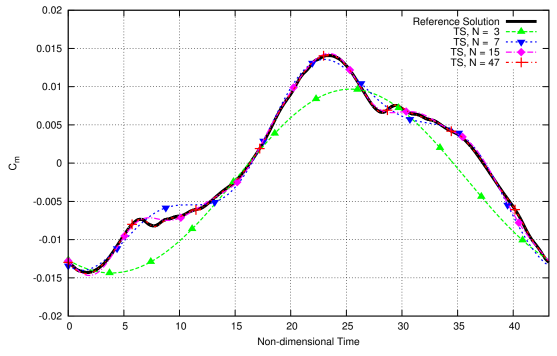


(d)

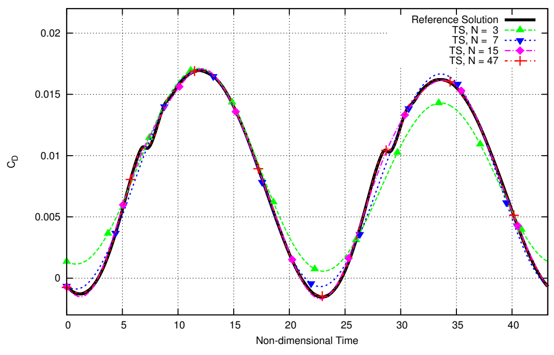
Figure 5. Comparison of convergence for Test Case 1 with 3 time-spectral time instances using the 5 different solvers discussed with Non-linear Iterations (a), BCGS Iterations (b), Krylov Vectors (c), and Wall-clock time (d) on the x-axis



(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 Test Case 2

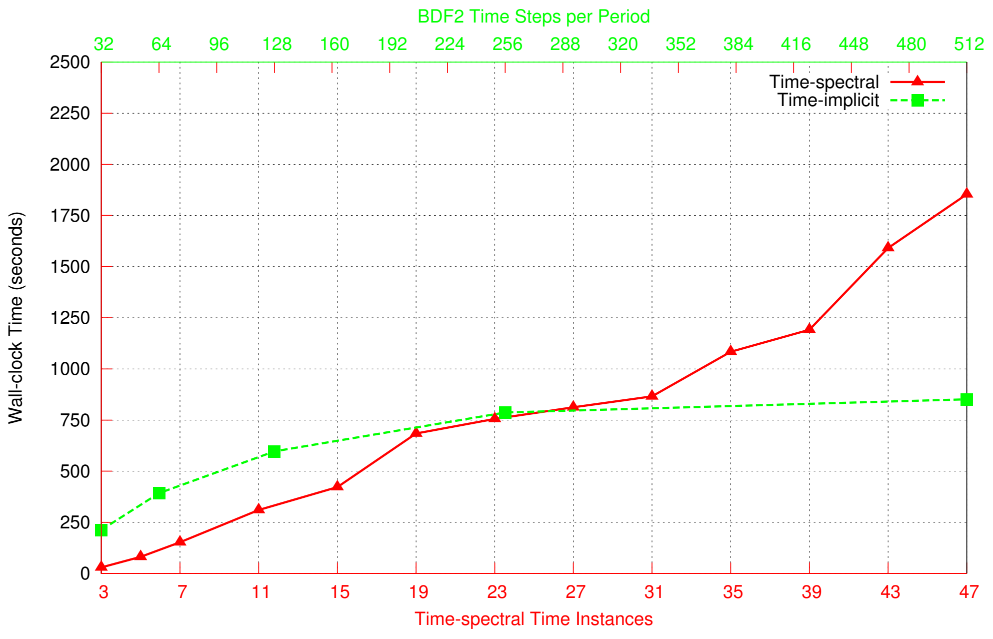
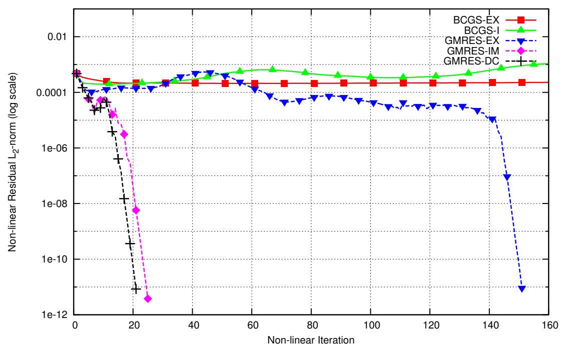
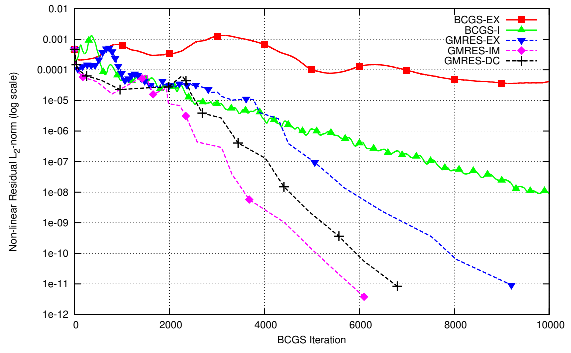


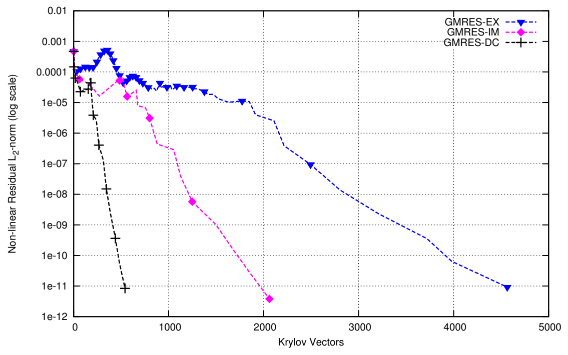
Figure 7. Comparison of Wall-clock time to convergence for time-implicit and time-spectral with different numbers of time steps per period (top axis) and time instances (bottom axis) for Test Case 2



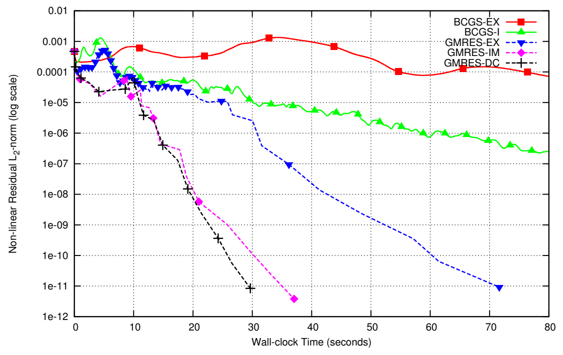
(a)



(b)

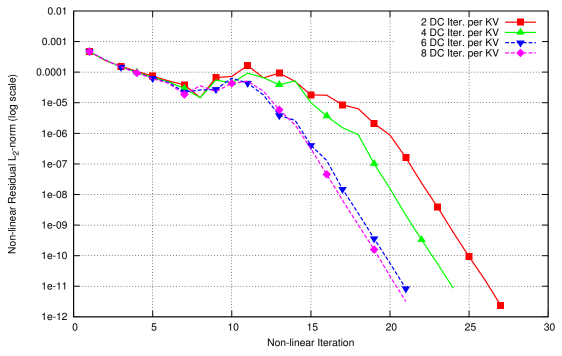


(c)

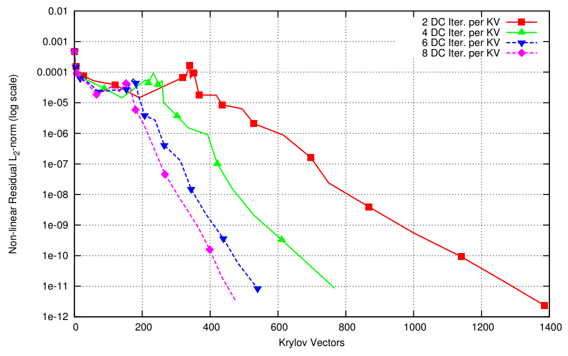


(d)

Figure 8. 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

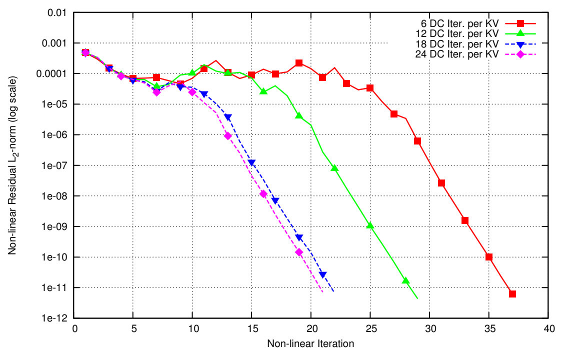


(a)



(b)

Figure 9. Comparison of convergence of Test Case 2 using 3 time-spectral time instances and the GMRES-DC solver using different numbers of defect-correction iterations per Krylov vector with Non-linear Iterations (a) and Krylov Vectors (b) on the x-axis


(a)

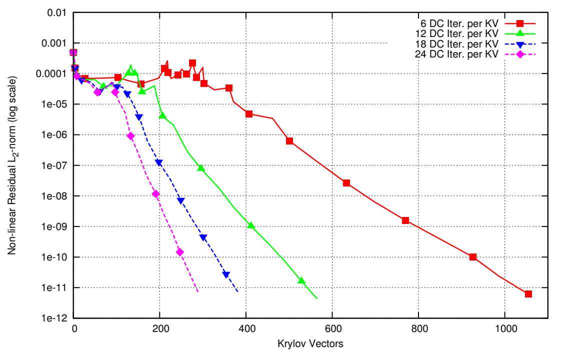
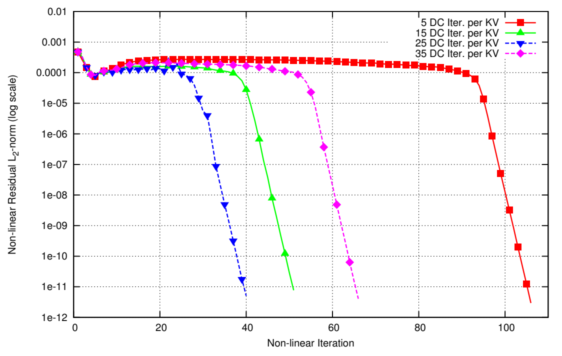

(b)

Figure 10. Comparison of convergence of Test Case 2 using 15 time-spectral time instances and the GMRES-DC solver using different numbers of defect-correction iterations per Krylov vector with Non-linear Iterations (a) and Krylov Vectors (b) on the x-axis


(a)

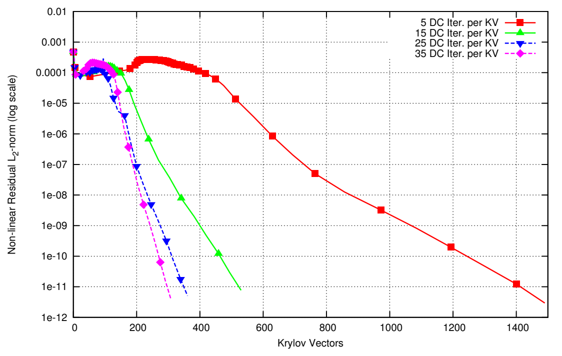

(b)

Figure 11. Comparison of convergence of Test Case 2 using 31 time-spectral time instances and the GMRES-DC solver using different numbers of defect-correction iterations per Krylov vector with Non-linear Iterations (a) and Krylov Vectors (b) on the x-axis