An Order N log N Parallel Newton-Krylov Solver for Time Spectral Problems

Donya Ramezanian *
Behzad R. Ahrabi †
Dimitri Mavriplis ‡
Department of Mechanical Engineering, University of Wyoming, Laramie, WY 82071

The time-spectral method is a fast and efficient scheme for computing the solution to temporal periodic problems. Compared to traditional backward difference implicit time-stepping methods, time-spectral methods incur significant computational savings by using a temporal Fourier representation of the time discretization and solving the periodic problem directly. In the time-spectral discretization, all time instances are fully coupled to each other, resulting in a dense temporal discretization matrix, the evaluation of which scales as $O(N^2)$, where N denotes the number of time instances. However, by implementing the time-spectral method based on the fast Fourier transform (FFT) the computational cost can be reduced to $O(N \log_2 N)$ for even numbers of time instances and to $O(2N \log_3 N)$ for odd numbers of time instances. Furthermore, in parallel implementations, where each time instance is assigned to an individual processor, the wall-clock time necessary to converge time-spectral solutions is reduced to $O(\log_2 N)$ for even numbers of time instances and to $O(2 \log_3 N)$ for odd numbers of time instances using the FFT-based approach, as opposed to the $O(N)$ weak scaling incurred by previous discrete Fourier transform (DFT) parallel time-spectral solver implementations. However, as the number of time instances becomes larger or the period of the flow becomes shorter, the non-linear system associated with the time-spectral method becomes larger and stiffer to solve. Achieving superior efficiency with the time-spectral method based on the FFT requires a robust solver strategy that solves the large non-linear space-time system rapidly. In previous work, an FFT-based approximate factorization (AF) scheme was used to solve time-spectral problems with large numbers of time instances. In the current work, this solution stratey is reformulated as a preconditioner to be used in the context of a Newton-Krylov method applied directly to the complete non-linear space-time time-spectral residual. The use of the Generalized Minimal Residual Method (GMRES) Krylov method enables additional coupling between the various time instances running on different processors resulting in faster overall convergence. The new GMRES/AF scheme is shown to produce significantly faster convergence than the previous AF scheme used as a solver alone, and achieves order of magnitude gains in efficiency compared to previous DFT-based implementations of similar solution strategies for large numbers of time instances.

I. Introduction

Time periodic problems have a wide range of applications including analysis of turbomachinery flows, rotorcraft aerodynamics, and flow analysis around helicopter blades. To solve such time periodic problems, time-spectral methods offer an appropriate strategy that significantly reduces computational cost compared to traditional time implicit methods. In many cases, the time-spectral method, using small numbers of time instances per period, without the need to evolve through the transient part of the solution, shows the same or even better accuracy than traditional time stepping methods with a much higher number of time steps. To discretize the time domain, similar to the harmonic balance method,[1–3] a discrete Fourier analysis is used

*PhD Candidate, Member AIAA; email: dramezan@uwyo.edu
†AIAA Senior Member, Associate Research Scientist; email: brezaahr@uwyo.edu
‡Professor, AIAA Associate Fellow; email: mavriplis@uwyo.edu

in time-spectral methods where unsteady equations in the physical domain are first transferred to a set of steady equations in the frequency domain. The steady equations in the frequency domain are then transformed back to the physical domain by a time discretization operator which couples each time instance to all other time instances.[4] Higher-order accuracy and lower computational cost are the two main advantages of the time-spectral method compared to traditional time stepping methods. In the time-spectral method, spectral accuracy can be achieved as Fourier representations are used for time discretization.[5] In addition to having high accuracy, the time-spectral method has been shown to be computationally more efficient than dual-time stepping implicit methods (using backward difference in time) for various time periodic problems such as helicopter rotors,[6,7] oscillatory pitching airfoils,[8] turbomachinery flows,[3,9] and flapping wings[10] .

The time-spectral method based on the discrete Fourier transform has been implemented in the past in parallel by assigning each time instance to an individual processor.[4] In this work, simple implicit solvers such as block-Jacobi and Gauss-Seidel where initially employed but were found to produce slow convergence. Therefore, a Newton solution strategy was adopted, where a Generalized Minimal Residual method (GMRES) was used to solve the linear system at each Newton step, and the aforementioned linear iterative solvers were used as preconditioners for GMRES. Examining different preconditioning strategies, an approximate factorization (AF) scheme which solves the temporal and spatial components in two successive steps was found to be the most efficient approach overall, particularly for problems with large numbers of harmonics and/or high reduced frequency. However, the cost of all these methods scales as $O(N^2)$ where N denotes the number of time instances, due to the fully coupled nature of the time-spectral discretization. When implemented in parallel, using one time instance per processor, the wall-clock time of these methods scales as $O(N)$ due to the $O(N^2)$ communication.

Recently, the time-spectral method based on a fast Fourier transform has been implemented in parallel to reduce the cost of computations and wall-clock time from $O(N)$ for each processor to $O(\log_2 N)$.[11] The AF algorithm was used to separate the spatial and temporal parts of the discrete equations in order to transform the temporal part to the frequency domain using the FFT. This was done to take advantage of the fact that spectral matrices and their derivatives are strictly diagonal in the frequency domain, therefore the system of equations reduces to N decoupled equations.

In this paper, we seek further improvements in efficiency and scaling through the addition of GMRES. The FFT-based implementation of the approximate factorization scheme is reformulated as a preconditioner for GMRES, which is used in turn to solve the fully coupled space-time system at each non-linear step in the Newton method. The implementation is parallel in time, where each time instance is associated with a different processor or subset of processors.

In the following sections, first the governing equations and the base solver are presented, then the necessary parts of the time-spectral discretization based on the DFT and the FFT are shown including their parallel implementation. Subsequently, a brief explanation of two linear solvers, AF and GMRES is given. In this section, it is explained how the FFT-based AF is used as the preconditioner for GMRES. Then, the results for solving a pitching airfoil problem using the FFT-based time-spectral method for even and odd numbers of time instances are shown in comparison to the solution of the same problem using the DFT-based time-spectral method. Finally, the prospect for further improvements in the overall solver are also discussed.

II. Mathematical Formulation

A. Spatial Discretization

The integral form of the Euler equations for inviscid compressible flow over a moving control volume in two dimension can be written as:

$$\int_{\Omega(t)} \left(\frac{\partial U}{\partial t} \right) dV + \int_{\partial\Omega(t)} \left(F(U) \cdot \vec{n} \right) dS = 0 \tag{1}$$

in which $\Omega(t)$ is the control volume U is the vector of conserved quantities, $F(U)$ is the conserved flux, $\vec{n}$ is the normal vector of each edge and and S is the surface of each edge. The left hand side of equation (1) can be written as follows:

$$\frac{\partial}{\partial t}\int_{\Omega(t)} U dV = \int_{\Omega(t)} \frac{\partial U}{\partial t} dV + \int_{\partial\Omega(t)} U(\dot{x}.\vec{n}) \tag{2}$$

where $\dot{x}$ is the velocity of each edge which varies with time. Equation (1) then can be represented as:

$$\frac{\partial}{\partial t}\int_{\Omega(t)} U dV + \int_{\partial\Omega(t)} (F(U) - U\dot{x}).\vec{n} dS = 0 \tag{3}$$

The spatial part of the above equation can be illustrated as:

$$R(U, \dot{x}, \vec{n}) = \int_{\partial\Omega(t)} (F(U) - U\dot{x}) \tag{4}$$

where U is assumed to be a cell centered variable. Therefore, equation (3) is then given as follows:

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

In this equation, R contains the integrated convective fluxes in arbitrary Lagrangian-Eulerian(ALE) form and represents the spatial discretization, while V denotes the cell volume. In this work we employ a second-order accurate cell-centered discretization with added artificial dissipation on unstructured triangular meshes similar to that used in previous work.[4,12]

B. Time-Spectral Method

Using the time-spectral method for temporal differentiation achieves spectral accuracy in time. This differentiation can be implemented in two ways that will be explained briefly in the following sections.

1. Discrete Fourier Transform Implementation

The discrete Fourier transform (DFT) of a temporally periodic variable, U, with the period T converts a sequence of samples in the time domain into the frequency domain and can be written as:[4]

$$\widehat{U}_k = \frac{1}{N}\sum_{n=0}^{N-1} U^n e^{-ikn\Delta t\frac{2\pi}{T}} \tag{6}$$

where N is the number of samples in the time domain, k is the frequency or wave number, and $\Delta t = T/N$. The original samples in the time domain can be recovered by the inverse discrete Fourier transform as:

$$U^n = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \widehat{U}_k e^{ikn\Delta t\frac{2\pi}{T}} \tag{7}$$

Taking the derivative of U^n with respect to t in equation (7), the derivative becomes:

$$\frac{\partial}{\partial t}U^n = \frac{2\pi}{T}\sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} ik\widehat{U}_k e^{ikn\Delta t\frac{s\pi}{T}} \tag{8}$$

The time-derivative formulation is most easily constructed when all the terms in the above equation are written in the time domain. Therefore, equation (6) is used to substitute $\widehat{U_k}$, in the above equation. Finally, the derivative of U with respect to time is obtained as:[5,13,14]

$$\frac{\partial U^n}{\partial t} = \sum_{j=0}^{N-1} d_n^j U^j \tag{9}$$

in which

$$\begin{cases} d_n^j = \frac{2\pi}{T}\frac{1}{2}(-1)^{n-j}cot(\frac{\pi(n-j)}{N}) & \text{if } n \neq j \\ 0 & \text{if } n = j \end{cases} \tag{10}$$

for an even number of time instances and

$$\begin{cases} d_n^j = \frac{2\pi}{T}\frac{1}{2}(-1)^{n-j}csc(\frac{\pi(n-j)}{N}) & \text{if } n \neq j \\ 0 & \text{if } n = j \end{cases} \tag{11}$$

for an odd number of time instances. By substituting this derivative into equation (5), the discretized governing equations reduce to a coupled system of N equations for N different time instances:

$$\sum_{j=0}^{N-1} d_n^j U^j V^j + R(U, \dot{x}, \vec{n}(t)) = 0 \quad n = 0, 1, 2, .., N-1 \tag{12}$$

The time-spectral method affects only the temporal part of these equations while the spatial discretization part remains unchanged. Additionally, in order to implement equation (12) in parallel, each time instance n is assigned to an individual processor. Therefore, N processors are needed to solve the entire time-spectral system. Since, each processor needs information from all other processors at each iteration (assuming no parallelism in the spatial domain for this argument), $O(N^2)$ communication takes place between the N processors.

2. Fast Fourier Transform Implementation for Even Numbers of Samples

For data sets with power of 2 numbers of samples, while the discrete Fourier transform of a variable with N samples requires $O(N^2)$ operations, the same result can be achieved with only $O(N \log_2 N)$ operations using the fast Fourier transform(FFT). The difference is significant especially for large numbers of time instances N.[15,16] The idea is that the discrete Fourier transform of length N can be written as the sum of two discrete Fourier transforms of length $N/2$, in which the first consists of all even numbered time instances, and the second comprises all odd numbered time instances. This splitting into odd and even groups is applied recursively until the length of the final subdivision is one. For N samples, $\log_2 N$ divisions are required and in each division N operations take place. Therefore, the total cost will be $O(N \log_2 N)$. In each level the Fourier transform is computed as:

$$\widehat{U}_k = \frac{1}{N}\sum_{n=0}^{\frac{N}{2}-1} U^{2n} e^{-ik2n\Delta t\frac{2\pi}{T}} + \frac{1}{N}\sum_{n=0}^{\frac{N}{2}-1} U^{2n+1} e^{-ik(2n+1)\Delta t\frac{2\pi}{T}} \tag{13}$$

In other words:

$$\widehat{U}_k = \text{even-indexed part} + W^k \text{odd-indexed part} \tag{14}$$

where:

$$W = e^{-i\frac{2\pi}{N}} \tag{15}$$

Successive subdivision of the samples into odd and even parts changes the order in which the samples must be considered. This is illustrated in Figure 1 for a recursive subdivision of $N = 8$ samples. The Danielson-Lanczos lemma provides a method to find the odd-even reordering pattern of each sample.[16,17] Letting the samples be denoted as U^n, the lemma shows that the new ordering is obtained by bit-reversal of the original sample index n as shown in Figure 1.

In order to take the derivative of U with respect to time we make direct use of equations (6) and (8). The forward FFT is applied to the variable U^n and the obtained $\widehat{U_k}$ is multiplied by its corresponding ik in which i is the imaginary unit and k is the corresponding frequency. Next, the inverse FFT (IFFT) is applied to the results of this multiplication. The result is the exact evaluation of equation (9) at a reduced cost afforded by the use of the FFT.

In order to implement the FFT in parallel, similar to the DFT parallel implementation, each time instance is assigned to an individual processor. The difference is that the FFT divides the calculations into $(\log_2 N)$ levels and in each level, each processor requires the information of just one other processor to calculate its own portion of the sequence. Therefore, $O(N)$ communication takes place at each level and hence the total amount of communications will be $O(N \log_2 N)$. In a traditional FFT implementation, the data is reordered according to the bit-reversal pattern either at the start or the end of the algorithm.[15,16]For a

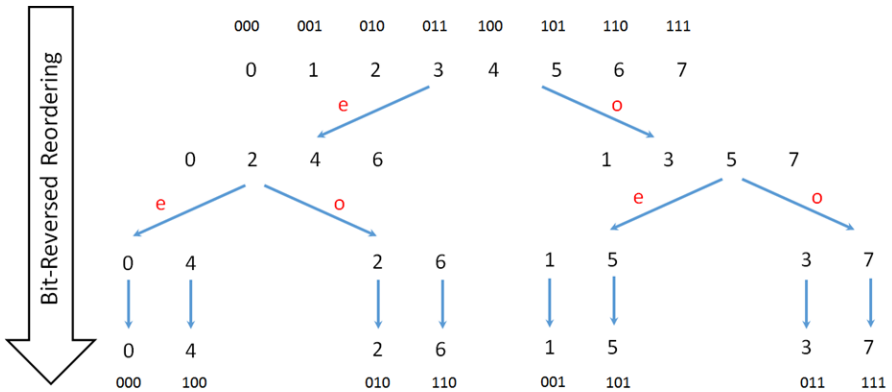


Figure 1. Recursive subdivision of $N = 8$ sample set and corresponding bit-reversal ordering

parallel FFT implementation of a time-spectral discretization, this implies significant communication, since the entire spatial grid data from each time instance on a given processor would need to be transferred to the corresponding bit-reversed processor location. However, since the time-spectral implementation always requires the application of a forward Fourier transform, followed by an inverse Fourier transform, as described in equations (6) and (8), the samples are brought back to the time domain afterwards via the inverse FFT and the reordering phase is not required. Rather, all that is required is the specification of the appropriate frequency k on each processor prior to the application of the IFFT, and the knowledge of the address of each processor to which communication must be done at each level in the FFT and IFFT process. These frequency values and processor addresses can easily be computed locally without the need for any additional communication. At each level of the FFT and IFFT, pairwise communication between processors occurs and the total volume of communication is the same for all levels. However, the pattern of communication varies for each level, as shown in Figure 2, for the case of the forward FFT with no data reordering. Application of the forward FFT corresponds to traversing the levels in Figure 1 from the bottom up. Thus in the first level, each processor must communicate with its neighbor in the bit-reversal ordering, which corresponds to a distant processor address in the original ordering. On the other hand, in the final level of the FFT, each processor communicates with its nearest neighbor in the original ordering. This widely varying communication pattern at each level can have significant effects on the achieved bandwidth for modern multi-core distributed memory computer architectures, as will be shown in the results section.

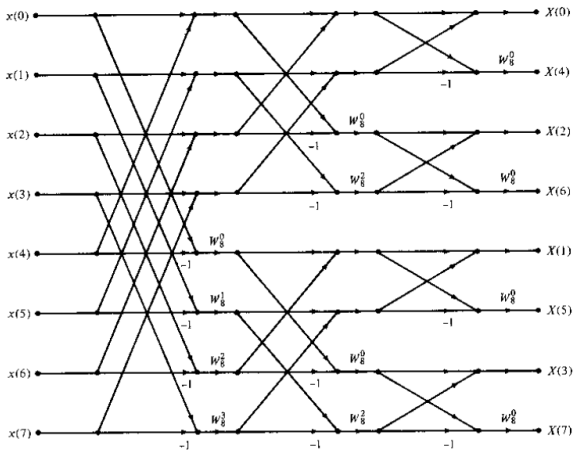


Figure 2. Pattern of communication for each level of the parallel FFT algorithm for sample size $N = 8$

3. Fast Fourier Transform Implementation for Odd Numbers of Samples

It is well known that time-spectral solutions using even numbers of samples perform suboptimally compared to time-spectral solutions using odd numbers of samples,[9] thus alternative FFT implementations that are not restricted to power 2 numbers of samples must be considered. For data sets with power of 3 numbers of samples, similar results can be achieved by using the FFT-base 3 instead of the DFT, in which the number of operations reduces from $O(N^2)$ to $O(2N \log_3 N)$. This implementation results in significant gains in computational efficiency particularly for large numbers of time instances N. The idea is that the discrete Fourier transform of length N can be written as the sum of three discrete Fourier transforms of length $N/3$. In these summations the first term consists of all samples with $3n$ indices, the second term consists of all samples with $3n+1$ indices, and the last term contains the rest of the samples with $3n+2$ indices. This subdivision is applied recursively until the length of the final individual sets is one. Similar to the FFT with even number of samples, the ordering of samples after recursive subdivision can be found by trit-reversing (in base 3) the index n of the samples.

For N samples, $\log_3 N$ divisions are required and in each division $2N$ operations take place. Therefore, the total cost will be $O(2N \log_3 N)$. In each level the Fourier transform is computed as:

$$\widehat{U}_k = \frac{1}{N}\sum_{n=0}^{\frac{N}{3}-1} U^{3n} e^{-ik3n\Delta t\frac{2\pi}{T}} + \frac{1}{N}\sum_{n=0}^{\frac{N}{3}-1} U^{3n+1} e^{-ik(3n+1)\Delta t\frac{2\pi}{T}} + \frac{1}{N}\sum_{n=0}^{\frac{N}{3}-1} U^{3n+2} e^{-ik(3n+2)\Delta t\frac{2\pi}{T}} \tag{16}$$

Similar to even numbers of samples, the same steps are done for taking the derivative of U with respect to time. The parallel implementation is done with the difference that FFT-base 3 implementation divides the calculations into $(\log_3 N)$ levels and in each level, each processor requires the information of two other processors to calculate its own portion of the sequence. Therefore, $O(2N)$ communication takes place at each level and hence the total amount of communications will be $O(2N \log_3 N)$. Moreover, for the same reason that has been explained in the previous section, the samples do not require reordering prior to or after application of the FFT. Thus by tagging an appropriate rank to each processor, the extra cost of communication due to exchanging data for reordering can be avoided. The processor rank in this case can be obtained by trit reversing the original rank of the processor.

For both even and odd number of time instances, since both inputs and outputs are real, in order to further increase the speed of the code we can use optimization for real valued samples. As has been used in our previous work,[11] it is possible to treat N real data as $N/2$ numbers of complex data. By splitting the N real input data and putting the first half of the data set into the real locations of the FFT and the second half into the imaginary locations of a $N/2$ set of complex numbers, the size of the input as well as the cost of the FFT derivative subroutine can be reduced substantially. Theoretically, this splitting trick can yield a factor of 2 decrease in communication and computational cost.

C. Approximate Factorization Algorithm

The above sections focus on the computation of the time-spectral derivative, which corresponds to the evaluation of the temporal part of the residual operator. While this is sufficient for explicit in time TS solvers, a more effective solution technique is required for problems with large numbers of time instances and/or high reduced frequencies.[4, 12, 18, 19]

Adding a pseudo-time term to the left hand side of equation (12) and linearizing by the Newton-Raphson approach, equation(12) yields:

$$[A]\Delta U = -Res(U, \dot{x}, \vec{n}(t)) = -\sum_{j=0}^{N-1} d_n^j U^j V^j - R(U, \dot{x}, \vec{n}(t)) \tag{17}$$

where Res is the total residual of the time-spectral space-time system and

$$[A] = \left[\frac{V}{\Delta\tau} + J + V\left[D_{TS}\right]\right] \tag{18}$$

is the complete time-spectral Jacobian matrix. Here $\Delta\tau$ denotes the pseudo-time step, J refers to the Jacobian of the spatial discretization, and $[D_{TS}]$ corresponds to the matrix of the time-spectral coefficients

d_n^j as defined in equation (12). Approximate factorization is an efficient algorithm that factors the Jacobian into the following form:[4, 20–22]

$$[A] \approx [I + \Delta\tau D_{TS}][\frac{V}{\Delta\tau} I + J] \tag{19}$$

which separates the contribution of the spatial and temporal parts in the Jacobian. This algorithm is implemented in two steps. In the first step, the spatial matrix is solved to find an intermediate value $\Delta\Delta U$:

$$\Delta\Delta U = [\frac{V}{\Delta\tau} I + J]^{-1}(-Res(U, \dot{x}, \vec{n}(t))) \tag{20}$$

This can be achieved using any existing direct or iterative solver previously implemented for steady-state problems. In this work we employ a simple block Jacobi iterative solver. In the second step, using the previously computed intermediate value, the temporal matrix is inverted to find ΔU:

$$\Delta U = [I + \Delta\tau D_{TS}]^{-1} \Delta\Delta U \tag{21}$$

When solving this part of the equation using the DFT approach, the spectral matrix is usually inverted or factorized directly. However, for the FFT implementation, by transferring the equation to the frequency domain, the spectral matrix becomes diagonal and subsequently the system of coupled equations changes to N decoupled equations. Therefore, the $\widehat{\Delta U_k}$ is calculated as:

$$\widehat{\Delta U_k} = \frac{1}{1 + ik\omega\Delta\tau}(\Delta\Delta\widehat{U}_k) \tag{22}$$

where i is the imaginary unit and ω is the angular frequency ($\omega = 2\pi/T$). Next $\widehat{\Delta U_k}$ is transferred back to the time domain by use of the inverse fast Fourier transfer (IFFT) to obtain ΔU.

If the inversion of both spatial and temporal Jacobian matrices in the AF scheme is exact, the order of solving these two parts does not effect the result. In this paper, first the spatial part is solved using block-Jacobi to find the intermediate value. Subsequently, the temporal component is solved based on the previously obtained intermediate value. This algorithm is not exact and includes an error which is given as $\Delta\tau J D_{TS}$. By choosing a small $\Delta\tau$ the error can be reduced although the system must be solved iteratively.

D. Generalized Minimal Residual Method

Although the approximate factorization scheme is relatively effective and can be implemented efficiently using an FFT approach as described above, this scheme still suffers from the requirement of using a small pseudo-time step or CFL number due to the limitations of the factorization error incurred by the scheme. Following previous work, one way to overcome these limitations is to use the approximate factorization scheme as a preconditioner for a GMRES solver within the context of a Newton-Krylov method. In this approach, the entire non-linear space-time system of equations resulting from the time-spectral method is linearized and solved using this Newton method. The linear system arising at each step of the Newton solution procedure is solved using the GMRES approach with the approximate factorization-FFT solver used as a preconditioner. The flexible GMRES algorithm that allows an iterative method as a preconditioner has been described by Saad and shown here in Algorithm 1.[4, 23]

Algorithm 1: FGMRES

1 Given $Ax = b$
2 Compute $r_0 = b - Ax_0$, $\beta = \|r_0\|$ and $v_1 = \frac{r_0}{\beta}$
3 for $j = 1, ..., , n$ do
4 compute $z_j = P^{-1}v_j$
5 compute $w = Az_j$
6 for $i = 1, ..., j$ do
7 $h_{i,j} = (w, v_i)$
8 $w = w - h_{i,j}v_j$
9 end for
10 Compute $h_{j+1,j} = \|w\|_2$ and $v^{j+1} = \frac{w}{h_{j+1,i}}$
11 Define $Z_m = [z_1, ...z_m]$, $H_m = h_{i,j\,1\leq i\leq j+1;1\leq j\leq m}$
12 end for
13 Compute $y_m = argmin_y \|\beta e_1 - H_m y\|_2 =$ and $x_m = x_0 + Z_m y_m$
14 if satisfied Stop, else set $x_0 = xm$

In the given algorithm, A is the time-spectral Jacobian matrix that includes both the temporal and spatial parts, as defined in equation (18). The pseudo-time step term may or may not be included in A. b corresponds to the negative sign of the non-linear time-spectral residual and x is the non-linear update of ΔU. The FFT-AF scheme is applied in line 4 of the algorithm as the preconditioner. In this case a pseudo-time step must be applied to guarantee the diagonal dominance of the system and to limit the AF factorization error. Also, Given's rotation is used in order to solve the minimization problem in line 13 of the algorithm.[23]

Aside from the pseudo-time term in the preconditioner, another pseudo-time step is used in the GMRES algorithm. Unlike the constant pseudo-time step inside the approximate factorization, the GMRES pseudo-time step is allowed to grow as the non-linear residual decreases. Therefore, two CFL values are involved in this algorithm. The first is used in the calculation of pseudo-time step of A, which is used in the FGMRES algorithm and is increased to a very large value so that quadratic convergence of the non-linear problem can be achieved. The second, which is constant and always smaller than or equal to the first one is used to manage the factorization error and guarantee diagonal dominance in the preconditioner(FFT-AF). The pseudo-time step in the FGMRES algorithm grows rapidly so that an exact Newton method can be recovered after several orders of magnitude decrease in the non-linear residual, provided the linear system is solved exactly. However, for efficiency reasons, we generally employ an inexact Newton approach where the linear system is only solved approximately, as discussed in the next section.

III. Computational Results

The problem being studied consists of the inviscid flow over a pitching NACA0012 airfoil at a Mach number of 0.755 and a mean incidence angel of $\alpha_0 = 0.016$. The pitching motion is prescribed about the quarter chord of the airfoil with the following formula:

$$\alpha(t) = \alpha_0 + \alpha_A sin(\omega t) \tag{23}$$

in which the reduced frequency is 0.1628 and the pitching amplitude α_A is equal to 2.51. This test case corresponds to the AGARD test case No.5.[24] The periodic solution is obtained with various numbers of time instances on an unstructured spatial mesh of 15573 triangles, as shown in Figure 3(a). The airfoil pitching motion is prescribed as a solid body rotation applied to the entire mesh. The contours of Mach number computed using the time-spectral solution with 64 time instances at a given location in time are shown in Figure 3(b).

In a first set of runs, the performance of the approximate-factorization (AF) scheme used directly as an iterative solver for the time-spectral problem is examined. In this and all subsequent cases, an iteration of the AF scheme includes 20 block Jacobi sweeps to approximately invert the spatial factor, followed by the direct inversion of the time-spectral factor. Figures 4 and Figures 5 depict the convergence rates of the DFT

and FFT-based AF implementations for even and odd numbers of time instances. As expected, the DFT and the FFT implementations result in identical convergence rates as measured by the computed residuals at each iteration.[11] However, when measured in terms of wall-clock time, the FFT solver is significantly more efficient than the DFT based solver. This is illustrated in Figures 6 and 7, where the wall-clock time to achieve a fully converged solution is plotted versus the number of time instances used in the time-spectral discretization for even and odd numbers of time instances, respectively. For even numbers of time instances the code was run for N equal to different powers of 2 up to 2048, and for odd numbers of time instances the code was run for N equal to different powers of 3 up to 2187. These figures demonstrate the significant efficiency gains of the FFT based method over the DFT based method, noting that the FFT approach is more efficient even for low values of N, as seen in Figures 6(b) and 7(b).[11]

The performance of the GMRES/AF scheme is examined next. In this case, the full space-time non-linear system is solved using a Newton-Krylov approach, using the previous FFT-based AF scheme as a preconditioner for GMRES. Figure 8 compares the non-linear convergence obtained by the GMRES/AF scheme versus that produced by the AF scheme used directly as a solver, for a case using 8 time instances, and for a case using 1024 time instances. In both cases, convergence is plotted as the L2 norm of the space-time residual versus the number of iterations for the AF solver, and versus the number of Krylov vectors for the GMRES/AF scheme, since in the latter case the cost of a Krylov vector is roughly equivalent to an iteration of the AF scheme. As seen from these results, using the AF scheme as a preconditioner for GMRES results in significantly faster convergence than using the AF scheme directly as a solver, requiring 4 to 5 times fewer iterations or Krylov vectors to reach the final convergence tolerance.

In these results, the same CFL value is used for the AF scheme when used either as a solver or as a preconditioner for GMRES. On the other hand, for the GMRES/AF scheme, the CFL in the Newton linearization used for GMRES is increased at each nonlinear iteration using a simple geometric progression, and reaches a maximum of 10^{15} after roughly 10 non-linear iterations. At each non-linear iteration, multiple Krylov vectors are used in the GMRES algorithm in order to solve the resulting linear system. An important consideration for the overall efficiency of the solver is the determination of the precision to which the linear system is solved at each non-linear step in the GMRES algorithm. In the previous results, the linear system solution tolerance was set to 0.1, based on the experience and recommendations from previous work.[4]

This effect of the linear system solution tolerance is studied in more detail for this case in Figure 9, where the same problem using 256 time instances has been solved using tolerances of 0.5, 0.1 and 0.01. In this figure, the L2 norm of the space-time residual is plotted versus the number of Krylov iterations for all three cases. Figures 10, 11 and 12 provide more detail for each case by showing the residual convergence in terms of non-linear updates, the number of Krylov vectors at each non-linear update, and the CFL evolution as a function of non-linear updates. For this case, the linear tolerance of 0.5 is the most efficient, achieving full convergence in approximately 2000 Krylov vectors compared to over 2800 Krylov vectors for the case using the lowest linear system tolerance of 0.01, and with the tolerance case of 0.1 falling in between these two cases. As seen from Figures 10, 11 and 12, the tighter linear tolerance cases result in fewer non-linear iterations overall, but this must be balanced by an increase in the number of Krylov vectors required at each non-linear step in order to satisfy the prescribed linear system solution tolerance. Figure 13 compares the required wall-clock time to solve the same problem using different numbers of time instances for the three values of the linear system tolerance. This plot shows that the wall-clock time for linear tolerance settings of 0.5 and 0.1 are similar for small numbers of time instances, where they are both lower than the time required by the smallest tolerance case of 0.01. However, for larger numbers of time instances, the wall-clock time required for the linear tolerance case of 0.1 is the lowest of all three cases. Therefore, the linear system tolerance is set to 0.1 for the remainder of the cases presented in this paper, since this represents the best compromise between consistent convergence behavior and efficiency over a wide range of test cases.

Figure 14 compares the wall-clock time required to converge the pitching airfoil problem for various numbers of time instances using the GMRES/AF scheme and the AF scheme alone. The wall-clock time is plotted versus the log of the number of time instances, and for both schemes, the wall-clock time varies roughly linearly with the log of the number of time instances, at least up to $N = 1024$. This is the expected behavior, since the FFT scheme, which dominates the cost of either solver scales as O(NlogN). However, the GMRES/AF scheme is seen to be consistently 4 to 5 times more efficient than the AF scheme alone over the entire range of the number of time instances up to $N = 2048$.

To further illustrate the efficiency of the current approach, Figure15 plots the required wall-clock time for the solution of the same problem versus the number of time instances for the GMRES/AF solver implemented

using the FFT approach and the DFT approach. This latter case corresponds closely to the solution scheme reported in references,[4,22] and the current approach shows significant improvement over the DFT approach, particularly for large numbers of time instances where up to two orders of magnitude improvement in the wall-clock time can be obtained.

Figure 16 compares the wall-clock time versus the number of time instances for different reduced frequencies using either the FFT-based GMRES/AF solver or the FFT-based AF solver alone. This plot indicates that the performance of both solvers is relatively insensitive to the reduced frequency of the problem while the GMRES/AF solver retains a factor of 4 or more speed-up compared to the AF solver.

To further study the behavior of the GMRES/AF solver, the portion of wall-clock time due to the computation and communications phases of the solver are examined. In this case, there is no spatial partitioning, and each time instance runs on a single individual processor or core. Thus the majority of the communication occurs in the FFT routines. These are invoked in the computation of the time-spectral residual evaluation, and also in the time-spectral portion of the Jacobian-vector product that occurs in the GMRES routine. Additionally, solution of the approximate factorization problem in the preconditioner involves the use of the FFT routines. Finally, additional communication is required in the GMRES routine for forming inner products over Krylov vectors which span all time instances. Figure 17 depicts the total wall-clock time, as well as the time due to communication and computation, versus the logarithm of the number of time instances. As the costs are dominated by the FFT algorithm, both communication and computation time are expected to scale as O(log N). This trend is observed approximately, although the costs increase more rapidly for larger numbers of time instances. This is attributed to a slight growth in the number of Krylov vectors required for the solution of cases above $N = 512$.

To examine the performance of the solver in more detail, the parallel FFT routine is isolated and the wall-clock time for communication and computation parts are monitored. Figure 18 shows the same plot for the parallel FFT routine alone. This figure shows that the computation component of the FFT scales optimally as $log(N)$, whereas the communication component grows faster than $log(N)$, particularly for higher processor counts. Referring back to Figure 2, the communication time required for the first and last level in the parallel FFT routine are compared in Figure 19 as a function of the number of time instances or processors N. Keeping in mind that both levels involve the same volume of communication data but across significantly different communication schedules, the increase in time for the first level can be attributed to the non-local nature of the communication, as illustrated in Figure 2. These results were run on the NCAR-Wyoming Yellowstone supercomputer, for which the intra-node communication bandwidth has been measured to be 60GBps, while the peak one-way network speed (inter-node) is about 6GBps.[25] Note that the communication time for both levels is nearly identical for $N = 16$, when all MPI ranks are within a single shared memory node. Thus, the additional communication time of the first level of the FFT routine for cases with more than 16 time instances can be attributed to the non-local nature of this schedule, which results in increased inter-node versus intra-node communication as N increases.[11]

IV. Conclusion and Future Work

In this work, we have developed a parallel FFT-based approximate factorization scheme for time-spectral problems that scales as $O(NlogN)$, where N denotes the number of time instances. This FFT-based AF scheme can be used either directly as a solver, or as a preconditioner for a GMRES linear solver, within the context of a Newton-Krylov approach applied to the complete space-time time-spectral non-linear problem. When used as a preconditioner for the Newton-Krylov approach, the GMRES/AF scheme is consistently and significantly more efficient than the AF scheme alone. Furthermore, the GMRES/AF scheme delivers convergence rates that are insensitive to the number of time instances and to the reduced frequency of the problem. The overall solver performance can be more than an order of magnitude more efficient than previous DFT-based implementations which scale as $O(N^2)$,[4,22,26] allowing for the effective solution of time-spectral problems using large numbers of time instances. Future work will investigate the effect of mesh resolution on the performance of the GMRES/AF solver as well as the extension of the solver to three-dimensional problems with different spatial discretizations.

V. Acknowledgements

This research was sponsored under NASA NRA Grant NNX15AU23A by NASA's Transformational Tools and Technologies (TTT) Project of the Transformative Aeronautics Concepts Program under the Aeronautics Research Mission Directorate.

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]Jameson, A., Alonso, J., and McMullen, M., "Application of a non-linear frequency domain solver to the Euler and Navier–Stokes equations," 2002, AIAA paper 2002-0120, 40th AIAA Aerospace Science Meeting and exhibit, Reno,NV, January 2002.

[3]Alonso, J. J., Mcmullen, M., and Jameson, A., "Acceleration of convergence to a periodic steady state in turbomachinery flows," 2001, AIAA paper 2001-0152, 39th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January 2001.

[4]Mundis, N. L. and Mavriplis, D. J., "Toward an Optimal Solver for Time-spectral Solutions on Unstructured Meshes," 2016, AIAA Paper 2016-0069, 54th AIAA Aerospace Sciences Meeting, San Diego, CA, January 2016.

[5]Gopinath, A. and Jameson, A., "Time spectral method for periodic unsteady computations over two-and three-dimensional bodies," 2005, AIAA paper 2005-1220, 43th AIAA Aerospace Science Meeting and Exhibit, Reno,NV, January 2005.

[6]Choi, S. and Datta, A., "CFD prediction of rotor loads using time-spectral method and exact fluid-structure interface," 2008, AIAA paper 2008-7325, 26th AIAA Applied Aerodynamics Conference, Honolulu, Hawaii, August 2008.

[7]Choi, S., Lee, K., Potsdam, M. M., and Alonso, J. J., "Helicopter rotor design using a time-spectral and adjoint-based method," *Journal of Aircraft*, Vol. 51, No. 2, 2014, pp. 412–423.

[8]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," 2006, AIAA paper 2006-692, 44th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January 2006.

[9]Van Der Weide, E., Gopinath, A., and Jameson, A., "Turbomachinery applications with the time spectral method," 2005, AIAA paper 2005-4905, 35th AIAA Fluid Dynamics Conference and Exhibit, Toronto, Ontorio, June 2005.

[10]Tomlin, C. and Jameson, A., "Aerodynamics and Flight Control of Flapping Wing Flight Vehicles: A Preliminary Computational Study," 2005, AIAA paper 2005-0841, 43th AIAA Aerospace Sciences Meeting, REno, NV, January 2005.

[11]Ramezanian, D. and Mavriplis, D. J., "An Order N log (N) Parallel Solver for Time Spectral Problems," *55th AIAA Aerospace Sciences Meeting*, 2017, p. 1219.

[12]Mavriplis, D. J. and Yang, Z., "Time Spectral Method for Periodic and Quasi-Periodic Unsteady Computations on Unstructured Meshes," *Mathematical Modeling of Natural Phenomena*, Vol. 6, No. 3, May 2011, pp. 213–236, DOI: http://dx.doi.org/10.1051/mmnp/20116309.

[13]Canuto, C., Hussaini, M. Y., Quarteroni, A. M., Thomas Jr, A., et al., *Spectral methods in fluid dynamics*, Springer Science & Business Media, 2012.

[14]Hesthaven, J. S., Gottlieb, S., and Gottlieb, D., *Spectral methods for time-dependent problems*, Vol. 21, Cambridge University Press, 2007.

[15]Cooley, J. W. and Tukey, J. W., "An algorithm for the machine calculation of complex Fourier series," *Mathematics of computation*, Vol. 19, No. 90, 1965, pp. 297–301.

[16]Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., "Numerical Recipes," 1992.

[17]Danielson, G. C. and Lanczos, C., "Some improvements in practical Fourier analysis and their application to X-ray scattering from liquids," *Journal of the Franklin Institute*, Vol. 233, No. 5, 1942, pp. 435–452.

[18]Sicot, F., Puigt, G., and Montagnac, M., "Block-Jacobi implicit algorithms for the time spectral method," *AIAA journal*, Vol. 46, No. 12, 2008, pp. 3080–3089.

[19]Leffell, J., *An Overset Time-Spectral Method for Relative Motion*, Ph.D. thesis, Stanford University, 2014.

[20]Thomas, J. P., Custer, C. H., Dowell, E. H., Hall, K. C., and Corre, C., "Compact implementation strategy for a harmonic balance method within implicit flow solvers," *AIAA journal*, Vol. 51, No. 6, 2013, pp. 1374–1381.

[21]Leffell, J. I., Murman, S. M., and Pulliam, T. H., "Time-Spectral Rotorcraft Simulations on Overset Grids," *32nd AIAA Applied Aerodynamics Conference*, 2014, p. 3258.

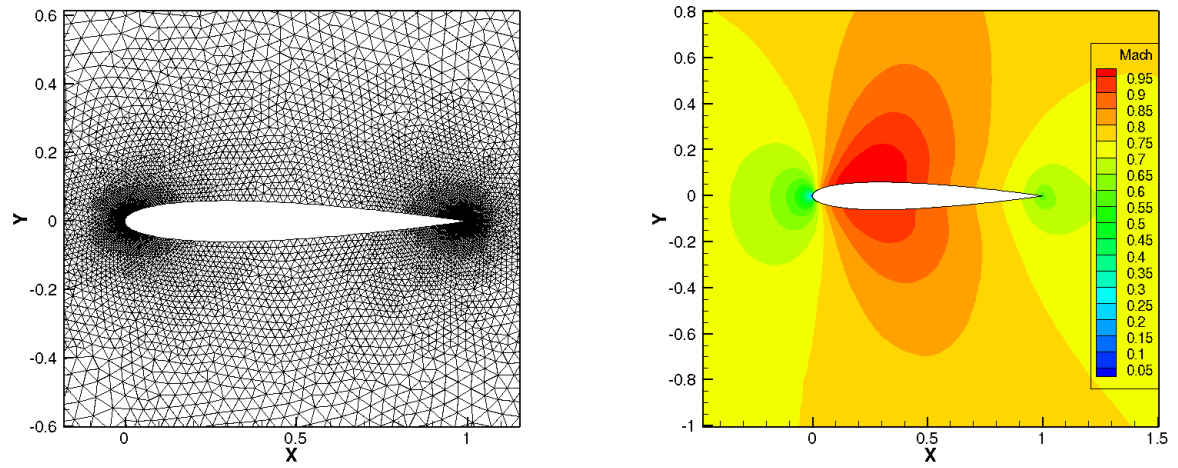
[22]Leffell, J., Sitaraman, J., Lakshminarayan, V., and Wissink, A., "Towards Efficient Parallel-in-Time Simulation of Periodic Flows," 2016, AIAA paper 2016-0066, 54th AIAA Aerospace Science Meeting, San Diego, California, January 2016.

[23]Saad, Y., *Iterative methods for sparse linear systems*, SIAM, 2003.

[24]Landon, R., "Compendium of unsteady aerodynamic measurements," *AGARD Report*, Vol. 702, 1982.

[25]Loft, R., Andersen, A., Bryan, F., Dennis, J. M., Engel, T., Gillman, P., Hart, D., Elahi, I., Ghosh, S., Kelly, R., et al., "Yellowstone: A dedicated resource for earth system science," *Contemporary High Performance Computing: From Petascale Toward Exascale, Volume Two*, Vol. 2, 2015, pp. 262.

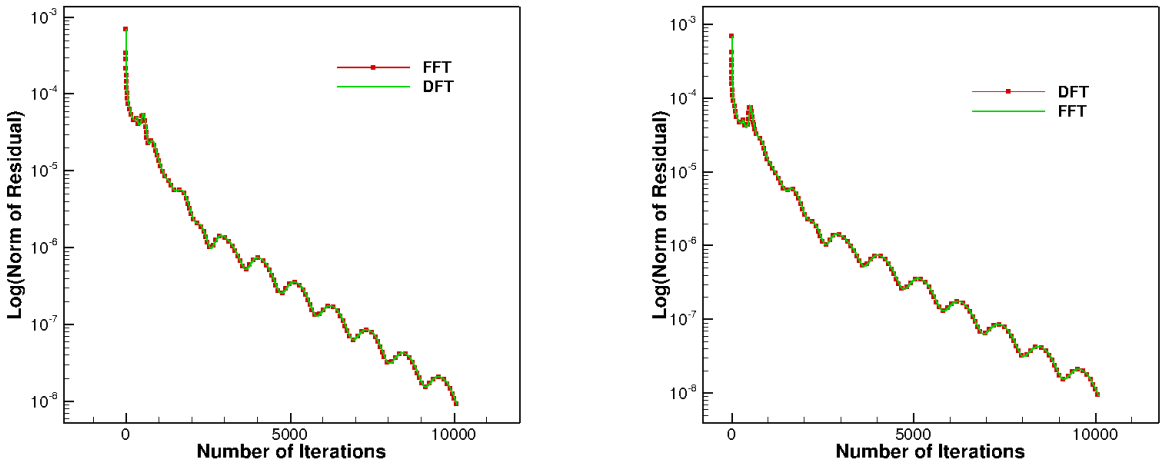
[26]Mundis, N. L. and Mavriplis, D. J., "GMRES applied to the Time-spectral and Quasi-periodic Time-spectral Methods," 2013, AIAA paper 2013-0638.



(a) Unstructured mesh

(b) Computed Mach contours

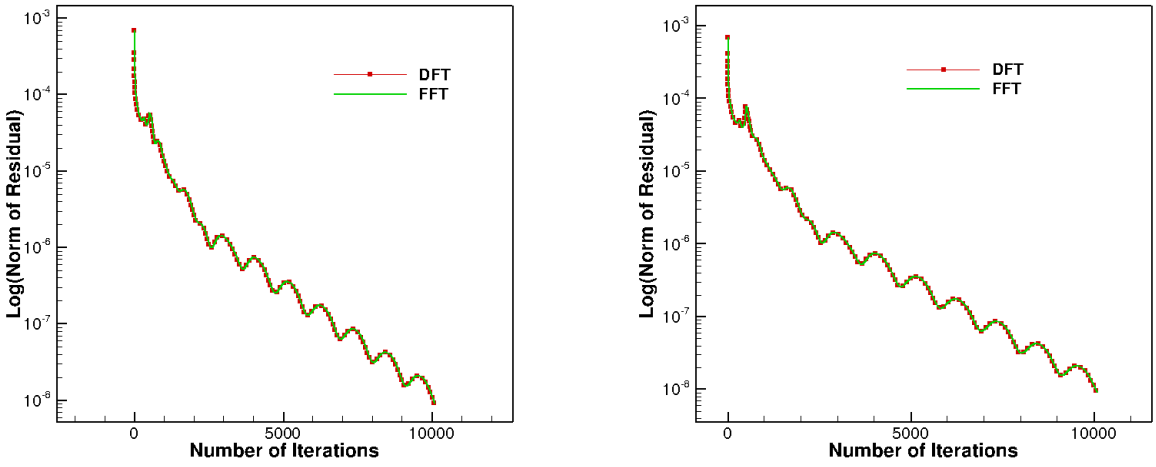
Figure 3. Computational mesh and computed Mach contours for time-spectral solution of pitching airfoil problem using 64 time instances



(a) 64 time instances

(b) 256 time instances

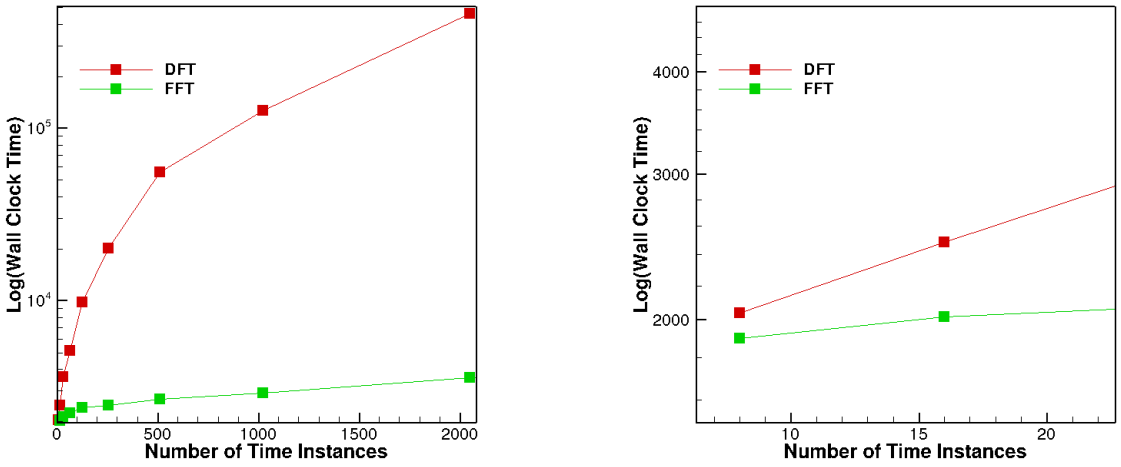
Figure 4. Residual versus number of iterations using the AF scheme as a solver for different even numbers of time instances



(a) 81 time instances

(b) 243 time instances

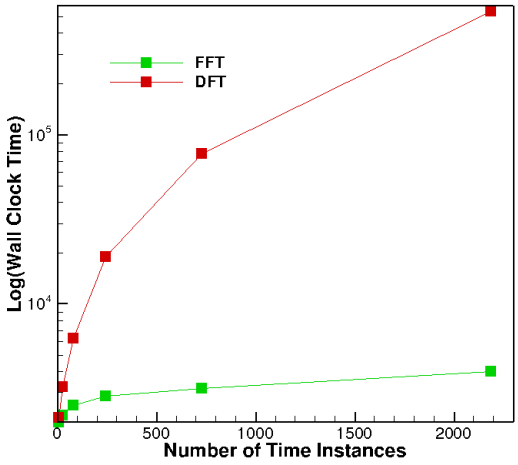
Figure 5. Residual versus number of iterations using the AF scheme as a solver for different odd numbers of time instances



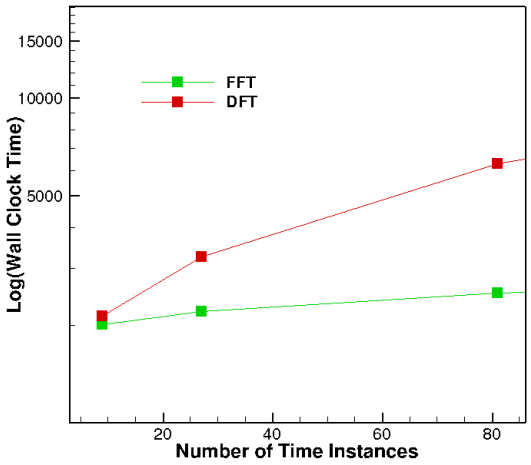
(a) Wall-clock time for up to 2048 number of samples

(b) Wall-clock time for small number of samples

Figure 6. Wall-clock time versus number of time instances using the AF scheme as a solver for even numbers of samples

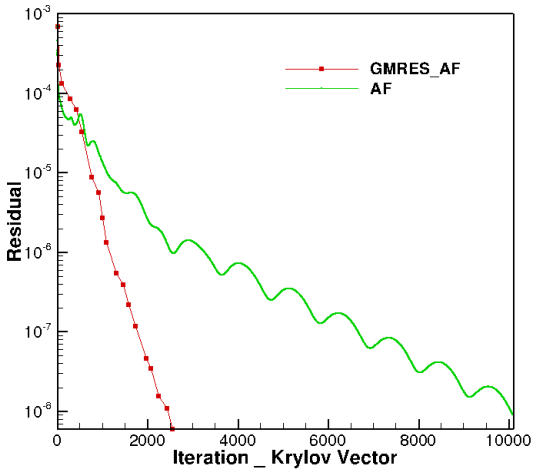


(a) Wall-clock time for up to 2187 number of samples

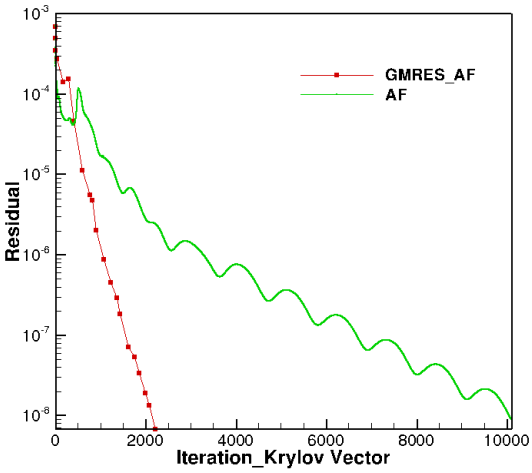


(b) Wall-clock time for small number of samples

Figure 7. Wall-clock time versus number of time instances using the AF scheme as a solver for odd numbers of samples



(a)



(b)

Figure 8. Comparison of the non-linear residual versus iterations for the AF solver and versus Krylov vectors for the GMRES/AF solver with 8 number of time instances (a), and 1024 number of time instances (b).

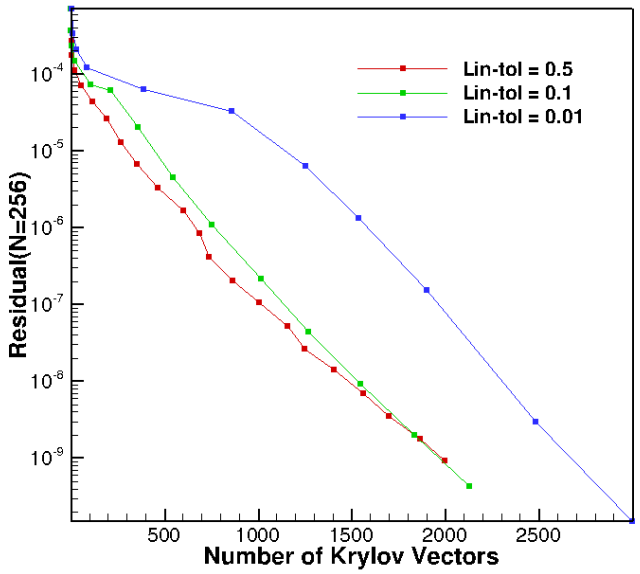


Figure 9. Residual versus Krylov vectors for different linear tolerances for $N = 256$

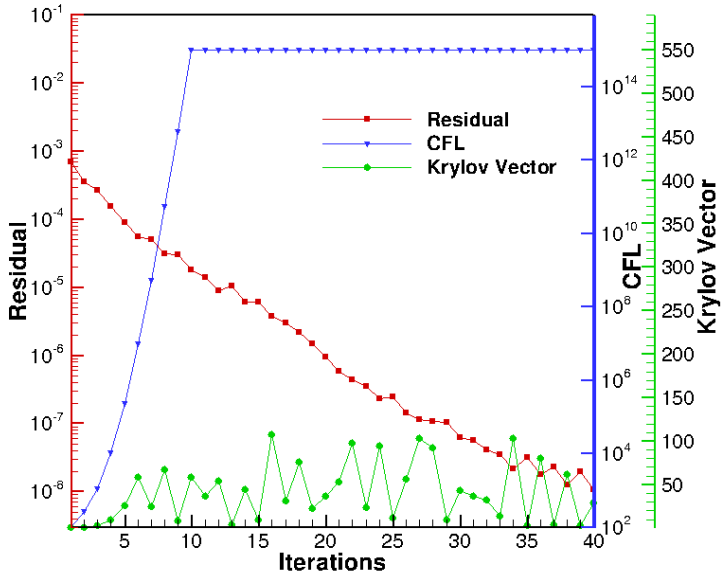


Figure 10. Non-linear convergence, CFL history, and number of Krylov vectors in each iteration for linear tolerance of 0.5

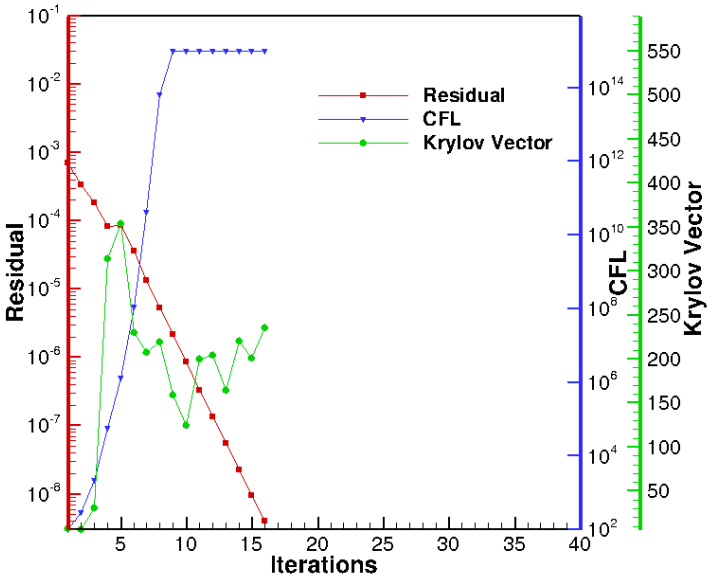


Figure 11. Non-linear convergence, CFL history, and number of Krylov vectors in each iteration for linear tolerance of 0.1

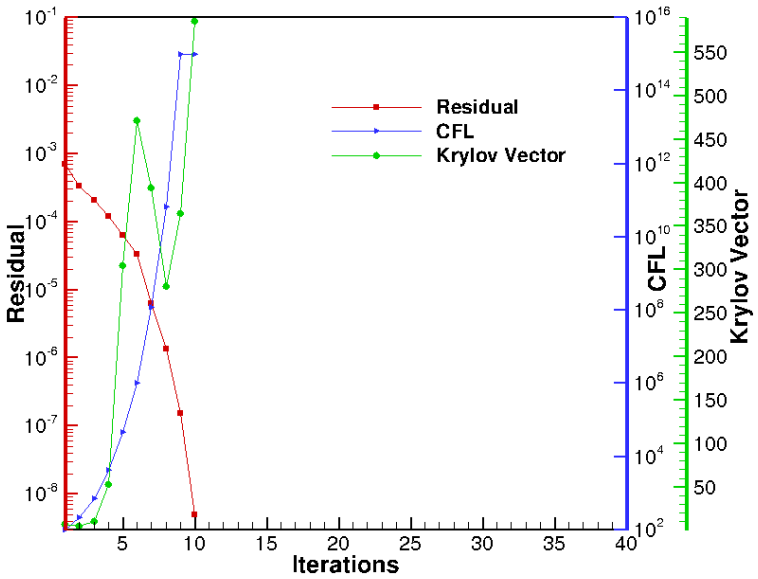


Figure 12. Non-linear convergence, CFL history, and number of Krylov vectors in each iteration for linear tolerance of 0.01

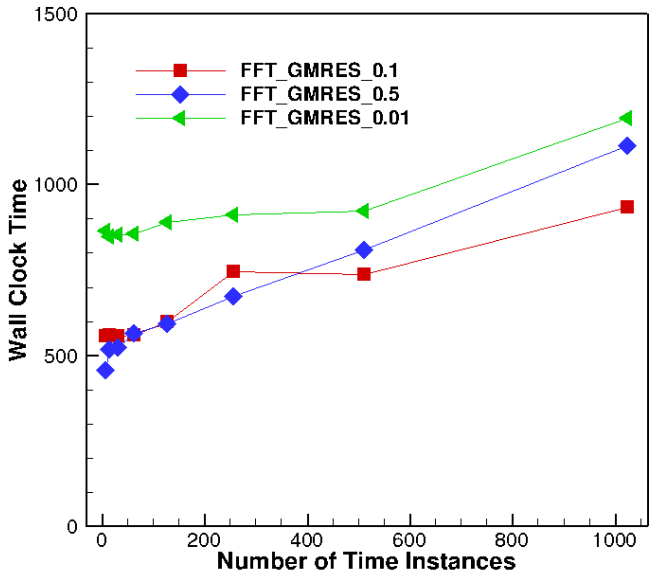


Figure 13. Wall-clock time versus number of time instances for different linear tolerances for $N = 256$

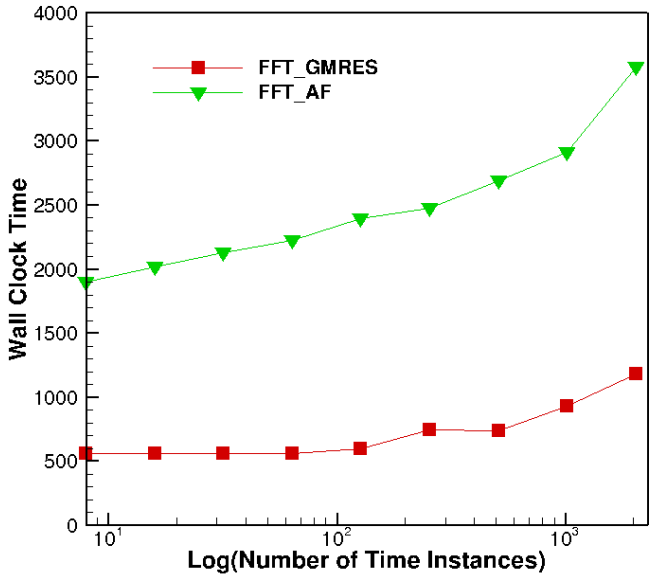


Figure 14. Wall-clock time versus number of time instances for FFT based GMRES/AF and FFT based AF solver

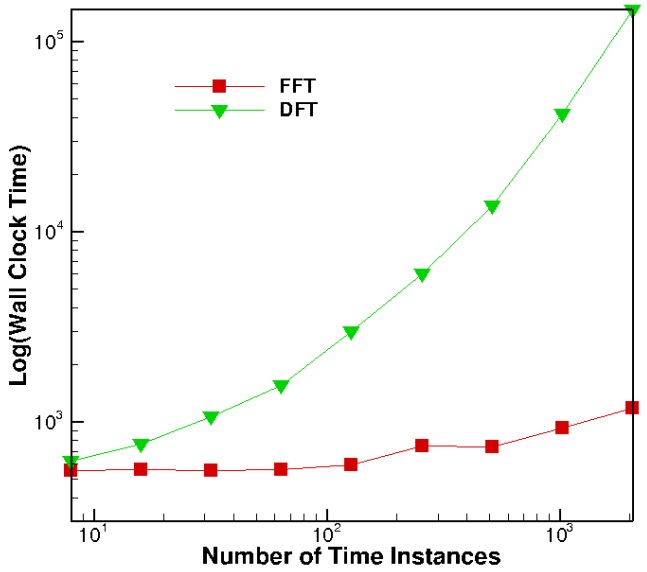


Figure 15. Wall-clock time for DFT and FFT based GMRES/AF solvers

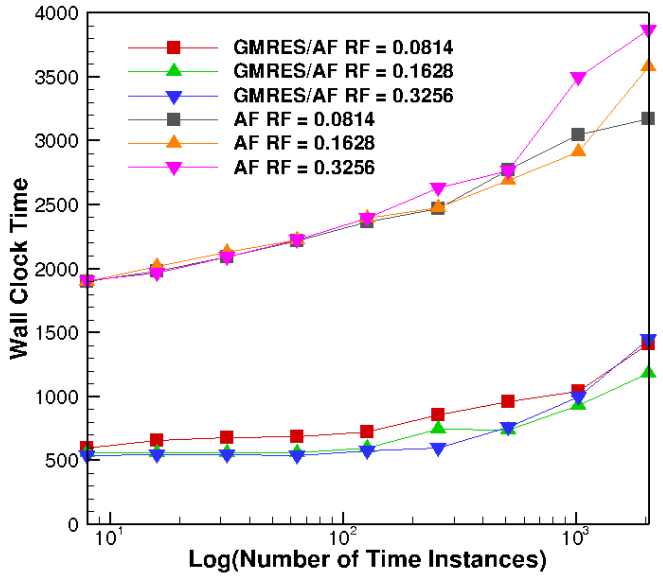


Figure 16. Wall-clock time versus number of time instances for FFT based GMRES/AF and FFT based AF solver using different reduced frequencies

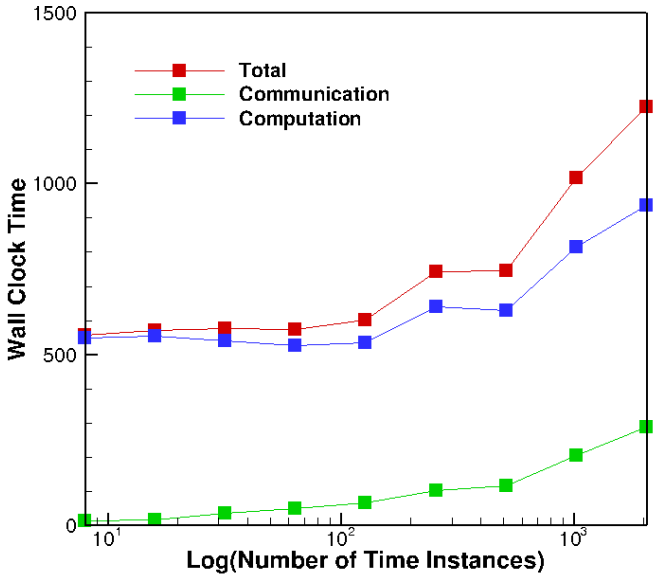


Figure 17. Breakdown of wall-clock time for computation and communication of the solver running on NCAR-Wyoming Yellowstone supercomputer using up to 2048 processors

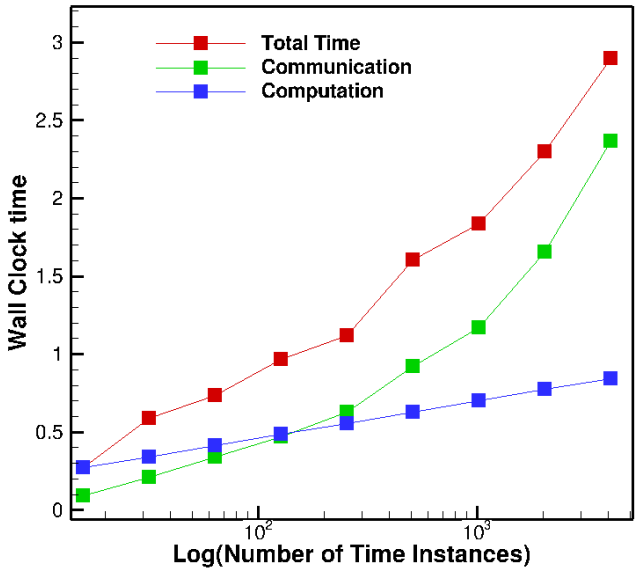


Figure 18. Breakdown of wall-clock time for computation and communication of parallel FFT routine running on NCAR- Wyoming Yellowstone supercomputer using up to 4096 processors

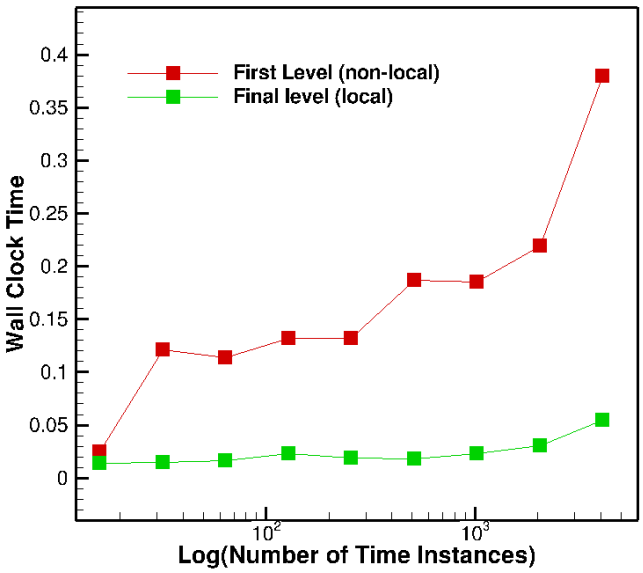


Figure 19. Comparison of communication time for first and last level of parallel FFT routine using up to 4096 processors