An Order NlogN Parallel Time-Spectral Solver for Quasi-Periodic Problems

Donya Ramezanian [*]
Dimitri Mavriplis [†]
Department of Mechanical Engineering, University of Wyoming, Laramie, WY 82071

The backward-difference time-spectral (BDFTS) method can be applied to problems with a slow transient in addition to a strong periodic behavior in time. This method is based on a collocation method that makes use of a combination of spectral and polynomial basis functions in time. This paper presents an extension of the recently developed parallel fast Fourier transform (FFT)-based time-spectral method for periodic problems to quasi-periodic problems. The FFT-based BDFTS scheme is implemented as a rank-1 modification of the FFT-based time-spectral scheme. Significant efficiency is gained in implementing the FFT-based BDFTS method compared to its original implementation which was based on the discrete Fourier transform (DFT) since the computational cost of implementing the spectral part of this method reduces from $O(N^2)$ to $O(NlogN)$. For problems with large number of time instances, the non-linear system becomes larger and/or stiffer to solve. A robust solver strategy is required that solves the large non-linear space-time system efficiently. An FFT-based approximate factorization (AF) scheme in space and time is used either directly as a solver or as a preconditioner for a Newton-Krylov method (GMRES) applied to the complete non-linear space-time residual. Results demonstrate the efficiency gained from the parallel FFT-based BDFTS scheme over the original implementation of this method. Furthermore, the performance of AF as the linear solver and as the preconditioner for GMRES is compared on a quasi-periodic problem. The accuracy and efficiency of the solution of the problem using linear and quadratic BDFTS formulations are also considered.

I. Introduction

Time periodic problems have a wide range of applications including analysis of turbo-machinery flows, rotorcraft aerodynamics, flow analysis of wind turbines and flapping wing unsteady aerodynamics.[1] 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,[2–4] a discrete Fourier expansion is used 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.[5] 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 temporal accuracy can be achieved as Fourier representations are used for the time discretization.[6] 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,[7,8] oscillatory pitching airfoils,[9] turbomachinery flows,[4,10] and flapping wings.[11]

[*]PhD Candidate, Member AIAA; email: dramezan@uwyo.edu
[†]Professor, AIAA Associate Fellow; email: mavriplis@uwyo.edu

The time-spectral method based on the discrete Fourier transform (DFT) has been implemented in the past in parallel by assigning each time instance to an individual processor.[5,12] The computational cost of implementing the parallel DFT-based time-spectral method scales as $O(N)$ in which N denotes the number of time instances. In our previous work the time-spectral method based on a fast Fourier transform (FFT) has been implemented in parallel to reduce the cost of computations and wall-clock time from $O(N)$ for each processor to $O(logN)$.[13]

A principal disadvantage of time-spectral methods is that they are only applicable to purely periodic problems. In previous work, a hybrid backward-difference time-spectral (BDFTS) approach has been proposed for solving problems with slow transients combined with a relatively fast periodic content in time.[12] The idea is to separate the periodic content from the quasi-periodic function and to obtain accurate resolution of the periodic component by making use of the properties of the spectral basis functions. Furthermore, the remaining transient portion of the quasi-periodic function is represented with polynomial basis functions.[12] The formulation is based on a collocation method which combines spectral and polynomial basis functions that are required to solve multiple successive periods to capture the transient behavior, while the fully coupled time instances within each individual period are solved simultaneously.

In this paper, the BDFTS formulation is shown to be equivalent to a rank-1 modification of the original periodic time-spectral matrix. Using the Sherman-Morrison formula, it is shown how the modified matrix can be inverted efficiently based on the parallel FFT-based approach for the temporal part of the AF algorithm. Using N time instances per period, the overall approach scales as $O(N\ logN)$ and requires $O(logN)$ wall clock time when implemented in parallel using one processor per time instance.

In the following sections, first the governing equations and the base solver are presented, then the necessary parts of the time-spectral and hybrid BDF/time-spectral discretizations based on the DFT and the FFT are shown including their parallel implementation. Subsequently, a brief explanation of the two linear solvers, the AF scheme used as a solver, and as a preconditioner for a Newton-Krylov method, is given. Next, the performance of the FFT-based AF solver is compared to the DFT-based implementation, and later the performance of the FFT-based GMRES/AF Newton-Krylov solver is compared to the performance of the FFT-based AF solver. Finally, the accuracy and efficiency of the solutions obtained by BDF1TS and BDF2TS implementations are studied.

II. Mathematical Formulation

A. Spatial Discretization

The integral form of the Euler equations for inviscid compressible flow over a moving control volume in two dimensions 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. Using the differential identity:

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

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, and V denotes the cell volume. In this work we employ a first-order accurate cell-centered discretization with added artificial dissipation on unstructured triangular meshes similar to that used in previous work.[13]

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:[2,5,6]

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

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:

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

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:[6,14,15]

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

in which

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

for an even number of time instances and

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

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, assuming no partitioning of the spatial mesh (although large problems will certainly involve both spatial and temporal parallel partitioning). 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 cases with power of 2 number of samples, the Fourier transform can be achieved with $O(N \log_2 N)$ number of operations, using the fast Fourier transform (FFT), as opposed to the discrete Fourier transform (DFT), or the straight forward evaluation of equation (12) which requires $O(N^2)$ number of operations, where N is the number of samples. The difference is significant especially for large numbers of samples.[16,17] The most commonly used FFT is the Cooley-Tukey algorithm. In this algorithm the DFT transform is decomposed into $\log_2 N$ levels with N computations on each level. In each level the samples are divided in two parts with size $N/2$ recursively, in which one part includes all the even numbered samples and the other one consists of the rest of the samples which are odd numbered. The total number of operations is thus $O(N \log_2 N)$. 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:

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

Figure 1 illustrates the pattern of recursive subdivision of $N = 8$ samples. Note that the outputs of the samples are not in the original order, due to the successive subdivision of the samples. In order to obtain the results in the original order, the Danielson-Lanczos lemma is used.[17,18] 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.

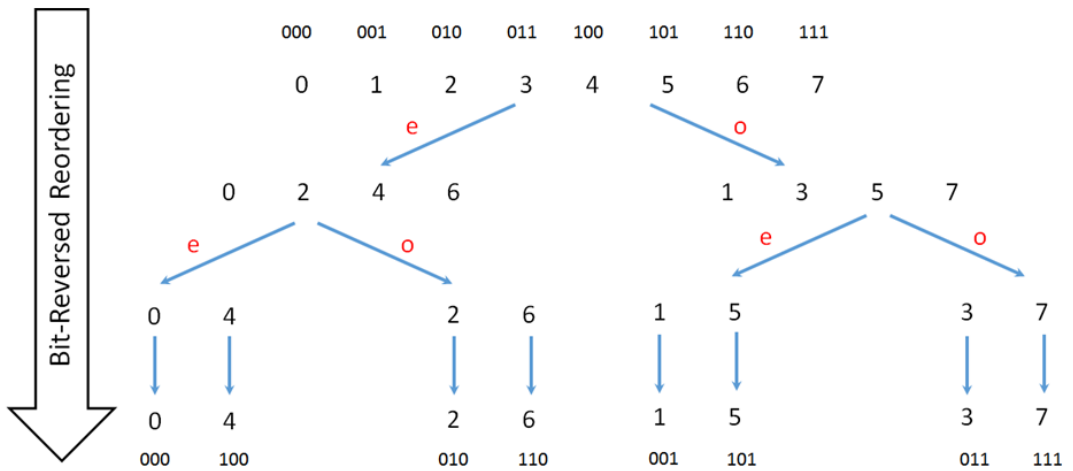

Figure 1: Recursive subdivision of $N = 8$ sample set and corresponding bit-reversal ordering

Taking the derivative of U with respect to time occurs in three steps. First, the forward FFT is applied to the variable U^n. Next, 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.

The idea of parallel implementation of the time-spectral method is the same in both DFT and FFT based implementations. Each time instance is assigned to an individual processor. However, the amount of communication is different. Since the FFT decomposes the calculations into ($\log_2 N$) levels and $O(N)$ communication takes place in each level, the total amount of communication will be $O(N \log_2 N)$, which results in large savings in the amount of communication compared to the DFT parallel implementation which requires $O(N^2)$ communication.

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.[16, 17] For a 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.

From Figure 2, it can be seen that the amount of communication is the same in different levels of the FFT algorithm. However, the pattern of communication varies for each level. 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 shown in previous work.[13]

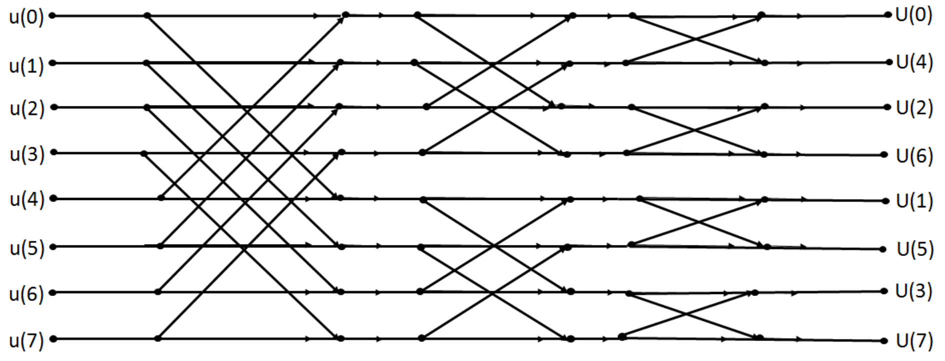

Figure 2: Pattern of communication for each level of the parallel FFT algorithm for sample size $N = 8$

C. Hybrid BDF/Time-Spectral Method

The time-spectral method may be extended to problems that involve the combination of periodic motion with a slower transient. The idea of the quasi-periodic time-spectral formulation is to subtract out the non-periodic transient part, which can be modeled using a polynomial basis set, and approximate the remaining purely periodic component with a spectral basis.[12] The formulation is based on a collocation method that makes use of a combination of spectral and polynomial basis functions.

We separate the periodic part, and the slowly varying mean flow part of a temporal quasi-periodic problem as:

$$U(t) = \sum_{k=\frac{-N}{2}}^{k=\frac{N}{2}} \widehat{U}_k e^{ikn\Delta t \frac{2\pi}{T}} + \bar{U}(t) \tag{16}$$

in which the slowly varying mean flow for a linear variation is approximated by a collocation method using a polynomial basis set as:

$$\bar{U}(t) = \Phi_{12}(t)U^{m+1} + \Phi_{11}(t)U^m \tag{17}$$

and for a quadratic variation in time, it is approximated as:

$$\bar{U}(t) = \Phi_{23}(t)U^{m+1} + \Phi_{22}(t)U^{m} + \Phi_{21}(t)U^{m-1} \tag{18}$$

in which U^m and U^{m+1} represent discrete solution instances in time usually taken as the beginning and ending points of the considered period in the quasi-periodic motion (and U^{m-1} corresponds to the beginning point of the previous period). In the first case, $\Phi_{12}(t)$ and $\Phi_{11}(t)$ correspond to the linear interpolation function given as:

$$\Phi_{11}(t) = \frac{t^{m+1} - t}{T} \tag{19}$$

$$\Phi_{12}(t) = \frac{t - t^{m}}{T} \tag{20}$$

with the period given as $T = t^{m+1} - t^m$. Similarly for quadratic interpolation, the $\Phi_{23}(t)$, $\Phi_{22}(t)$, $\Phi_{21}(t)$ are given as:

$$\phi_{21}(t) = 0.5[-\frac{(t - t_m)}{T} + \frac{(t - t_m)^2}{T^2}] \tag{21}$$

$$\Phi_{22}(t) = 1 - \frac{(t - t_m)^2}{T^2} \tag{22}$$

$$\phi_{23}(t) = 0.5[\frac{(t - t_m)}{T} + \frac{(t - t_m)^2}{T^2}] \tag{23}$$

Note that in this case, the collocation approximation leads to the determination of the Fourier coefficients as:

$$\widehat{U}_k = \frac{1}{N}\sum_{n=0}^{N-1} \tilde{U}^n e^{-ikn\Delta t \frac{2\pi}{T}} \tag{24}$$

with $\tilde{U^n} = U^n - \bar{U}^n$ defined as the remaining periodic component of the function after polynomial subtraction. Differentiating equation (16) and making use of equation (9) and (24) we obtain the following expression for the time derivative:

$$\frac{\partial U^n}{\partial t} = \sum_{j=0}^{N-1} d_n^j \tilde{U}^j + \Phi'_{12}(t_n)U^{m+1} + \Phi'_{11}(t_n)U^m \tag{25}$$

for the case of a linear polynomial functions in time. The $\Phi'_{12}(t_n)$ and $\Phi'_{12}(t_n)$ represent the time derivatives of the polynomial basis functions (resulting in the constant values $\frac{-1}{T}$ and $\frac{1}{T}$ in this case), and the various time instances are given by:

$$t_j = t_m + \frac{j}{N}(t_{m+1} - t_m) \tag{26}$$

We also note that $\bar{U}(t_m) = U^m = U(t_m)$ and thus we have $\tilde{U^0} = 0$. In other words, the constant mode in the spectral representation must be taken as zero, since it is contained in the polynomial component of the function representation. Therefore, the $j = 0$ component in the summation can be dropped, and rewriting the equation (25) in terms of the original time instances U^n we obtain:

$$\frac{\partial U^n}{\partial t} = \sum_{j=1}^{N-1} d_n^j U^j - (\sum_{j=1}^{N-1} d_n^j \Phi_{12}(t_j) - \Phi'_{12}(t_n))U^{m+1} - (\sum_{j=1}^{N-1} d_n^j \Phi_{11}(t_j) - \Phi'_{11}(t_n))U^m \tag{27}$$

Finally, the above expression for the time derivative is substituted into equation (5) which is then required to hold exactly at time instances $j = 1, 2, ..., N-1$ and $j = N$ (which corresponds to the $m+1$ time instances):

$$\begin{aligned} &\sum_{j=1}^{N-1} d_n^j V^j U^j - \\ &(\sum_{j=1}^{N-1} d_n^j \Phi_{12}(t_j) - \Phi'_{12}(t_n))V^{m+1}U^{m+1} - \\ &(\sum_{j=1}^{N-1} d_n^j \Phi_{11}(t_j) - \Phi'_{11}(t_n))V^m U^m + \\ &R(U^n, \bar{x}^n, \vec{n}^n) = 0 \end{aligned} \tag{28}$$

with $n = 1, 2, .., N$. As previously, we have N coupled equations for the N unknown time instances, although in this case the $j = 0$ time instance which corresponds to U^m values are known from the solution of the previous period, while the $j = N$ or U^{m+1} values are unknown, since these are not equal to the $j = 0$ values as they would be in a purely periodic flow. In the case of vanishing periodic content, summation involving the d_n^j coefficients vanish by virtue of equation (25) with $\tilde{U}^j = 0$ and it is easily verified that the above formulation reduces to a first-order backward difference scheme with a time step equal to T. On the other hand, for purely periodic motion, we have $U^{m+1} = U^m$ which results in cancellation of the polynomial derivative term $\Phi'_{12}(t_n)$ and $\Phi'_{11}(t_n)$. Furthermore, using the identities $\Phi_{21}(t_j) + \Phi_{11}(t_j) = 1$, and $\sum_{j=0}^{N-1} d_n^j = 0$, it can be seen that the remaining polynomial terms reduce to the missing $j = 0$ equation and the time-spectral method given by equation (5) is recovered.

In this description, the accuracy of the implementation based on linear polynomials corresponds to a BDF1 time-stepping scheme. For accuracy purposes, the transient part of the derivative can be implemented based on a quadratic polynomial which gives the accuracy of a BDF2 time-stepping approach in the absence of the periodic component. In this case, equation (28) changes to:

$$\begin{aligned} &\sum_{j=1}^{N-1} d_n^j V^j U^j - \\ &(\sum_{j=1}^{N-1} d_n^j \Phi_{23}(t_j) - \Phi'_{23}(t_n))V^{m+1}U^{m+1} - \\ &(\sum_{j=1}^{N-1} d_n^j \Phi_{22}(t_j) - \Phi'_{22}(t_n))V^m U^m - \\ &(\sum_{j=1}^{N-1} d_n^j \Phi_{21}(t_j) - \Phi'_{21}(t_n))V^{m-1}U^{m-1} + \\ &R(U^n, \bar{x}^n, \vec{n}^n) = 0 \end{aligned} \tag{29}$$

The quasi-periodic derivatives described in equations (28) and (29) can be rewritten in the following form:

$$[D_{qp}]\vec{U} = [D_{pp}]\vec{U} + [qp]\vec{U} + \vec{const}. \tag{30}$$

where $[D_{qp}]\vec{U}$ is the quasi-periodic derivative, and $[D_{pp}]\vec{U}$ is the periodic derivative of U which is obtained from:

$$[D_{pp}]\vec{U} = \begin{bmatrix} d_0^0 & d_0^1 & . & . & . & d_0^{N-1} \\ d_1^0 & d_1^1 & . & . & . & d_1^{N-1} \\ . & & & & & \\ . & & & & & \\ . & & & & & \\ d_{N-1}^0 & d_{N-1}^1 & . & . & . & d_{N-1}^{N-1} \end{bmatrix}_{N\times N} \begin{Bmatrix} U^N \\ U^1 \\ . \\ . \\ . \\ U^{N-1} \end{Bmatrix} \tag{31}$$

As mentioned, while U^0 in each period is known and obtained from the previous period, U^N which is equal to U^{m+1} is unknown. Since the coefficients of the time-spectral derivative matrix are the same for U^0

and U^N, the U^0 term is replaced by U^N in equation (31) in order to avoid changing the structure of the derivative routine. Furthermore, the matrix $[qp]$, is a rank-1 matrix and represents the contribution of U^{m+1} in equations (28) and (29). This matrix-vector product is given as:

$$[qp]\vec{U} = \begin{bmatrix} \alpha_N - d_0^0 & 0 & . & . & . & 0 \\ \alpha_1 - d_1^0 & 0 & . & . & . & 0 \\ . & & & & & \\ . & & & & & \\ . & & & & & \\ \alpha_{N-1} - d_{N-1}^0 & 0 & . & . & . & 0 \end{bmatrix}_{N \times N} \begin{Bmatrix} U^N \\ U^1 \\ . \\ . \\ . \\ U^{N-1} \end{Bmatrix} \tag{32}$$

in which the α_n are obtained for (28) as:

$$\alpha_n = \sum_{j=1}^{N-1} d_n^j \Phi_{12}(t_j) - \Phi'_{12}(t_n) \tag{33}$$

and for equation (29) as:

$$\alpha_n = \sum_{j=1}^{N-1} d_n^j \Phi_{23}(t_j) - \Phi'_{23}(t_n) \tag{34}$$

Moreover, the constant vector in equation (30) is given as:

$$\vec{const.} = (\sum_{j=1}^{N-1} d_n^j \Phi_{11}(t_j) - \Phi'_{11}(t_n))U^m \tag{35}$$

and in equation (29) can be calculated as follows:

$$\vec{const.} = (\sum_{j=1}^{N-1} d_n^j \Phi_{22}(t_j) - \Phi'_{22}(t_n))U^m - (\sum_{j=1}^{N-1} d_n^j \Phi_{21}(t_j) - \Phi'_{21}(t_n))U^{m-1} \tag{36}$$

Noting that the volume terms V^m have been removed for simplicity. Therefore, the quasi-periodic time-spectral derivative corresponds to a rank-1 modification of the original periodic time-spectral derivative matrix, and can be evaluated in parallel using the FFT-based approach which incurs $O(NlogN)$ communications, followed by a rank-1 update which itself requires one additional broadcast operation.

D. Approximate Factorization Algorithm

1. AF Algorithm in Purely Periodic Problems

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 time-spectral solvers, a more effective solution technique is required for problems with large numbers of time instances and/or high reduced frequencies.[5, 12, 19, 20]

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

$$[A]\Delta U = -Res(U, \dot{x}, \vec{n}(t)) = -\sum_{j=0}^{N-1} d_n^j U^j V^j - R(U, \dot{x}, \vec{n}(t)) \tag{37}$$

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

$$[A] = \left[\frac{V}{\Delta\tau} + J + V\left[D_{pp}\right]\right] \tag{38}$$

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 $[D_{pp}]$ corresponds to the time-spectral matrix. Approximate factorization is an efficient algorithm that factors the Jacobian into the following form:[5, 21–23]

$$[A] \approx [\frac{V}{\Delta\tau}I + J][I + \Delta\tau D_{pp}] \tag{39}$$

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 $\Delta\Delta U$:

$$\Delta\Delta U = [\frac{V}{\Delta\tau}I + J]^{-1}(-Res(U, \dot{x}, \vec{n}(t))) \tag{40}$$

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

$$\Delta U = [I + \Delta\tau D_{pp}]^{-1}\Delta\Delta U \tag{41}$$

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 $\Delta\widehat{U}_k$ is calculated as:

$$\Delta\widehat{U}_k = \frac{1}{1 + ik\omega\Delta\tau}(\Delta\Delta\widehat{U}_k) \tag{42}$$

where i is the imaginary unit and ω is the angular frequency ($\omega = 2\pi/T$). Next $\Delta\widehat{U}_k$ 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 affect the result. However, in general the spatial component is solved approximately using an iterative method. In our implementation, the spatial component is solved first (approximately) following the direct solution of the temporal component. The AF inversion is not exact and includes an error which is given as $\Delta\tau J D_{pp}$. By choosing a small $\Delta\tau$ the error can be reduced although the approximately factorized system must be solved iteratively.

2. AF Algorithm in Quasi-Periodic Problems

Proceeding along the same steps as in the purely periodic problems, equation (37) is recovered with some differences:

$$[A]\Delta U = -Res(U, \dot{x}, \vec{n}(t)) = -[D_{qp}]U^jV^j - R(U, \dot{x}, \vec{n}(t)) \tag{43}$$

in which D_{qp} is the quasi-periodic derivative term that is obtained from equation (27) or (28). Matrix $[A]$ in equation (43) is given as:

$$[A] = \left[\frac{V}{\Delta\tau} + J + V\left[D^*_{qp}\right]\right] \tag{44}$$

where

$$[D^*_{qp}] = [D_{pp}] + [qp] \tag{45}$$

and the matrix $[qp]$ comes from equation (30), noting that the constant vector in this equation drops out during the process of Newton-Raphson linearization. The factorization process in this case is the same as in the previous section which leads to the following approximation:

$$[A] \approx [\frac{V}{\Delta\tau}I + J][I + \Delta\tau D^*_{qp}] \tag{46}$$

Solving the spatial matrix in order to find the intermediate solution proceeds exactly as in equation (40), except that the residual is obtained from the right hand side of the equation (43). In this work we employ a simple block-Jacobi iterative solver, as the focus of the current work is on the efficient inversion of the temporal operator. By following the same steps as in purely periodic problems for solving the temporal component of the AF scheme, equation (41) for a quasi-periodic problem changes to:

$$[I + \Delta\tau D^*_{qp}]\Delta U = \Delta\Delta U \tag{47}$$

or

$$[I + \Delta\tau([D_{pp}] + [qp])]\Delta U = \Delta\Delta U \tag{48}$$

Equation (48) is transferred to the frequency domain in order to benefit from this fact that the spectral matrix is diagonal in the frequency domain:

$$[FT]([D^*_{pp}] + \Delta\tau[qp])\Delta U = [FT]\Delta\Delta U \tag{49}$$

where $[FT]$ shows the Fourier transform matrix and $[D^*_{PP}]$ is:

$$[D^*_{pp}] = (I + \Delta\tau[D_{pp}]) \tag{50}$$

which is the temporal component of the AF scheme in purely periodic problems. Matrices $[D^*_{PP}]$ and $[qp]$ can be written as:

$$[D^*_{pp}] = [IFT][\tilde{D^*_{pp}}][FT] \tag{51}$$

and

$$[qp] = [IFT][\tilde{qp}][FT] \tag{52}$$

where $[IFT]$ represents the inverse of the Fourier transform matrix. Therefore, equation (49) is simplified to:

$$[\tilde{D^*_{qp}}](\Delta\widehat{U}_k) = [[\tilde{D^*_{PP}}] + \Delta\tau[\tilde{qp}]](\Delta\widehat{U}_k) = (\Delta\Delta\widehat{U}_k) \tag{53}$$

Next step is taking the inversion of matrix $[\tilde{D^*_{qp}}]$. In this case, since matrix $[\tilde{qp}]$ can be written as:

$$\Delta\tau[\tilde{qp}] = \vec{\widehat{u}_k}\vec{\widehat{v}_k^T} \tag{54}$$

matrix $[\tilde{D^*_{qp}}]$ corresponds to a rank-1 modification of the matrix $[\tilde{D^*_{pp}}]$, and it is no longer diagonal in the frequency domain. Therefore, it is not possible to calculate the inverse of the temporal matrix using equation (42). However, since the inverse of the matrix, $[\tilde{D^*_{pp}}]$, is easy to obtain in the frequency domain, following the equation (42) and matrix $[\tilde{qp}]$ is a rank-1 matrix, it is possible to calculate the inverse of the temporal matrix efficiently, using the Sherman-Morrison formula.[24] The Sherman-Morrison formula gives the inverse of the matrix $[\tilde{D^*_{qp}}]$ as follows:

$$[\tilde{D^*_{qp}}]^{-1} = ([\tilde{D^*_{pp}}] + \vec{\widehat{u}_k}\vec{\widehat{v}_k^T})^{-1} = [\tilde{D^*_{pp}}]^{-1} - \frac{[\tilde{D^*_{pp}}]^{-1}\vec{\widehat{u}_k}\vec{\widehat{v}_k^T}[\tilde{D^*_{pp}}]^{-1}}{1 + \vec{\widehat{v}_k^T}[\tilde{D^*_{pp}}]^{-1}\vec{\widehat{u}_k}} \tag{55}$$

We note that the term on the denominator is a constant scalar that can be precomputed, and one instance of the $[\tilde{D^*_{pp}}]^{-1}$ matrix can be factored out in the overall expression. Therefore this formula requires two matrix-vector products of $[\tilde{D^*_{pp}}]$, which can be obtained as previously, using the parallel FFT approach. Finally, $\Delta\widehat{U}_k$ is transferred back to the time domain by use of the inverse fast Fourier transform (IFFT) to obtain ΔU. The AF inversion error in this case depends on $\Delta\tau J D^*_{qp}$ which can be reduced by choosing a smaller $\Delta\tau$.

E. 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. 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 the use of an iterative method as a preconditioner has been described by Saad and shown here in Algorithm 1.[5, 25]

```
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 (38). The pseudo-time step term may or may not be included in A. b corresponds to the negative non-linear time-spectral residual and x is the non-linear update 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.[25]

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. Numerical Results

The two dimensional inviscid flow over a pitching NACA0012 airfoil test case is studied in this section. The solution is obtained with various numbers of time instances on an unstructured spatial mesh of 15573 triangles, as shown in Figure 3. A periodic pitching motion is prescribed at the quarter chord point of the airfoil, and the mean angle of attack varies over a sequence of five periods. The prescribed angle of attack is as follows:

$$\alpha(t) = \alpha_0 + \bar{\alpha}(t) + \alpha_1 sin(\omega_1 t) \tag{56}$$

in which the mean angle of attack is:

$$\bar{\alpha}(t) = \begin{cases} 0 & \text{if } t < t_1 \\ \alpha_m \frac{1}{2}(1 - cos(\omega_m(t - t_1))) & \text{if } t \geq t_1 \end{cases} \tag{57}$$

where t_1 is the time when the transient motion begins. Here, we assume it to be one period. This condition is set in a way that the problem is purely periodic in the first period, and the quasi periodic solution starts from the second period. The constants are assumed as:

$$\alpha_0 = 0.016^\circ, \quad \alpha_m = 2^\circ, \quad \alpha_1 = 2.51^\circ, \quad \omega_1 = 0.1628, \quad \omega_m = 0.1\omega_1 \tag{58}$$

Figure 4 shows the comparison of the computed lift coefficient in the first five periods using different number of time instances. From the figure, it can bee seen that the BDF1TS scheme has poor accuracy using only 8 number of time instances. However, the accuracy improves for greater number of time instances. In the first set of runs, the AF scheme is used directly as an iterative solver for the solution of the BDF1TS equations for this problem. For all of the runs performed for this test case, the AF scheme employed 50 Jacobi iterations for inverting the spatial matrix followed by the direct inversion of the temporal matrix. Figure 5 shows the convergence history of the DFT and FFT-based AF scheme using 16 time instances, for the first five periods of this quasi-periodic problem. Both of the implementations show identical convergence rates as expected. The first period is the solution of the corresponding purely periodic problem ($\bar{\alpha}(t) = 0$). The solution of the purely periodic problem is required for initializing the solution of the quasi-periodic motion. While both implementations (DFT and FFT) produce exactly the same convergence histories, the convergence wall-clock time is significantly improved for the FFT-based implementation as shown in Figure 6. This plot shows the total wall clock time of the solution of 5 periods using even numbers of time instances up to 512 for both the DFT and FFT-based solvers. Since the same spatial solver is used in both cases, the difference in the wall-clock time is entirely due to the solution of the temporal part. Table 1 compares the convergence rates of the quasi-periodic AF scheme for solving the problem to the residual level of 10^{-8} with different numbers of time instances. From this table it can be seen that the convergence rate provided by this scheme is relatively insensitive to the number of time instances, as the number of iterations required for convergence changes only slightly with increasing number of time instances.

Number of Time Instances	Number of Iterations
8	80791
16	82868
32	81256
64	80012
128	81998
256	87322
512	92164

Table 1: Comparison of convergence rate of the quasi-periodic AF scheme over first five periods for different number of time instances per period for the BDF1TS scheme

In previous work it has been shown that for purely-periodic problems, using AF as a preconditioner for GMRES, within the context of a Newton-Krylov method results in a more effective solver compared to using AF directly.[26]Therefore, the performance of the FFT-based GMRES/AF scheme in solving the quasi-periodic problem 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. 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 about 20 non-linear iterations. Figure 7 shows the convergence rate of the quasi-periodic problem obtained using the DFT-based GMRES/AF and the FFT-based GMRES/AF solvers, with the exact same settings in both cases. The plot shows that the convergence history is identical in both cases. 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 previous work for solving the purely-periodic problem,a linear tolerance of 0.1 was found to be the best in terms of number of iterations and wall clock time.[26] The effect of the linear system solution tolerance is studied in more detail for the quasi-periodic problem. Figure 8 shows the convergence history for solving the quasi-periodic problem using 16 time instances per period with different linear tolerances of 0.1, 0.01 and 0.001. As seen from this figure, the tighter linear tolerance cases result in fewer non-linear iterations overall and also the convergence history becomes more monotone. Figure 9 compares the required wall clock time to solve the same problem using different numbers of time instances for the three values of linear tolerance of 0.05, 0.1 and 0.5. This plot shows that the wall clock time for the linear tolerance of 0.1 is lower than the wall clock time for the other two linear tolerances for cases with greater number of time instances. Therefore, the linear tolerance is set to 0.1 for the remainder of the cases presented in this paper. Figure 10 plots the required wall-clock time for the solution of the problem using the FFT- and DFT-based implementation of GMRES/AF. The FFT-based approach shows significant savings in wall-clock time over the DFT-based approach. In cases with larger number of time instances the efficiency gained by the FFT-based approach becomes more remarkable. Figure 11 compares the wall clock time versus number of time instances using the FFT-based AF solver used directly as the non-linear solver, and used as a preconditioner for GMRES. This plot indicates that the GMRES/AF solver provides a factor of 2 to 3 speed-up compared to the AF solver.

Table 2 compares the convergence rates of the quasi-periodic GMRES/AF scheme for solving the problem to the residual level of 10^{-8} with different numbers of time instances. From this table it can be seen that the convergence rate changes slightly with the number of time instances.

Number of Time Instances	Number of Non-Linear Iterations
8	1278
16	304
32	333
64	371
128	387
256	415
512	439

Table 2: Comparison of convergence rate of the quasi-periodic GMRES/AF scheme over the first five periods for different number of time instances per period

For all the cases studied so far, the linear BDF1TS formula has been used for calculating the temporal derivative term, which corresponds to the accuracy of BDF1 time-stepping for the non-periodic content of the problem. For accuracy purposes the problem has also been solved using the BDF2TS formulation. Figure 12 plots the running wall-clock time of the solution of the same problem using BDF1TS and BDF2TS, for different numbers of time instances. The FFT-based GMRES/AF is used as the linear solver for the Newton-Raphson method in both cases. Since the preconditioner CFL in the BDF2TS formulation can only be about half of the size in the BDF1TS formulation, the running wall clock time for cases based on BDF2TS formulation is longer than cases which are based on BDF1TS. Figure 13 compares the error of the computed lift coefficient using linear and quadratic BDFTS solvers for different number of time instances per period. This error was computed as the difference between the current solution and a reference solution obtained using 1024 number of time instances. For smaller number of time instances, the error obtained in both cases are almost identical, which means the solution is dominated by the spectral content of the solver. However, for greater values of time instances, the BDF2TS solver has a steeper slope than the BDF1TS which shows that the solution is influenced more by the polynomial basis functions. Given the slower convergence of the BDF2TS acheme, a precise accuracy study of both schemes is required to determine the overall most efficient approach for a given level of accuracy.

IV. Conclusions

In this work, we have developed a parallel FFT-based approximate factorization scheme for quasi-periodic problems. The BDFTS equations correspond to a rank-1 update of the fully-periodic time-spectral equations, and can be solved effectively by leveraging the FFT-based periodic AF solver using the Sherman-Morrison formulation.[24] The FFT-based AF scheme can be used either directly as a solver, or as a preconditioner for the GMRES linear solver within a space-time Newton-Krylov scheme. The parallel FFT formulation enables large savings in wall clock time compared to the traditional DFT or time-spectral matrix formulation. Additionally, it has been shown that using the FFT-based AF scheme as a preconditioner for a Newton-Krylov approach is consistently and significantly more efficient than the AF scheme alone. Furthermore, the

BDFTS scheme based on quadratic polynomials is developed and compared to the BDFTS based on linear polynomials in terms of accuracy and performance. It has been shown that although BDF2TS requires longer wall-clock time for convergence, it provides better accuracy for cases with larger number of time instances, comparing to BDF1TS scheme.

References

[1]Rezaei, A. S. and Taha, H. E., "Computational Study of Lift Frequency Responses of Pitching Airfoils at Low Reynolds Numbers," 2017, 55th AIAA Aerospace Sciences Meeting.

[2]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.

[3]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.

[4]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.

[5]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.

[6]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.

[7]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.

[8]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.

[9]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.

[10]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.

[11]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.

[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]Ramezanian, D. and Mavriplis, D. J., "An Order NlogN Parallel Solver for Time Spectral Problems," 2017, AIAA 2017-1219, 55th AIAA Aerospace Sciences Meeting, January.

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

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

[16]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.

[17]Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., *Numerical Recipes*, Vol. 2, Cambridge Univ. Press, 1992.

[18]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.

[19]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.

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

[21]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.

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

[23]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.

[24]Maponi, P., "The solution of linear systems by using the Sherman–Morrison formula," *Linear algebra and its applications*, Vol. 420, No. 2-3, 2007, pp. 276–294.

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

[26]Ramezanian, D. and Mavriplis, D. J., "An Order NlogN Parallel Newton Krylov Solver for Time Spectral Problems," 2017, AIAA 2017-4509, 23rd AIAA Computational Fluid Dynamics Conference, June.

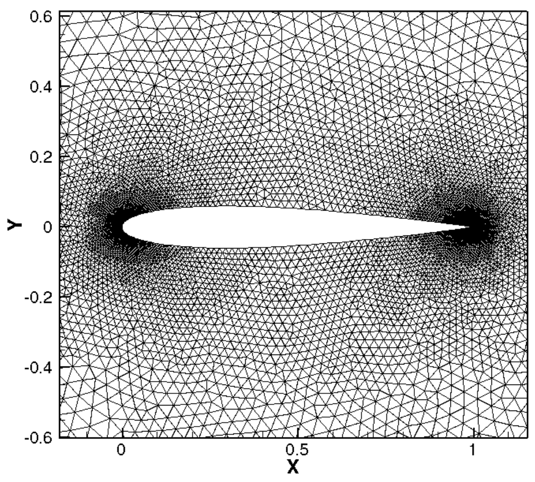

Figure 3: Computational mesh

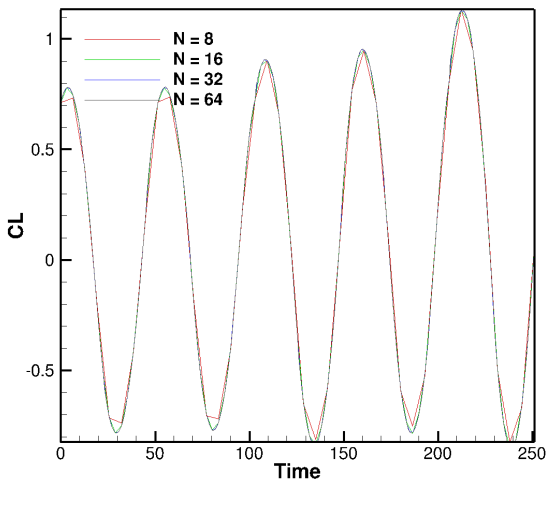

Figure 4: Lift coefficient versus time for the first 5 periods using different numbers of time instance per period

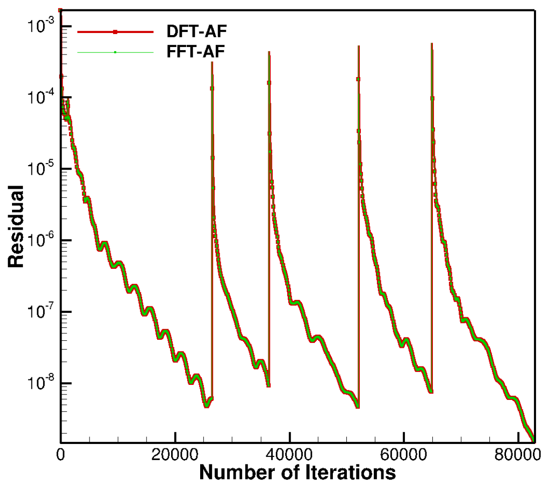

Figure 5: Residual versus number of iterations for DFT and FFT-based AF solver using 16 time instances per period for 5 periods of a quasi-periodic airfoil problem

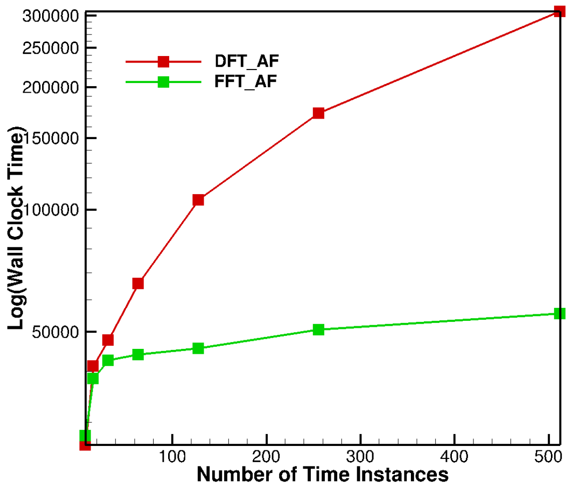

Figure 6: Log of total wall-clock time versus number of time instances for FFT and DFT- based AF solution for quasi-periodic airfoil problem

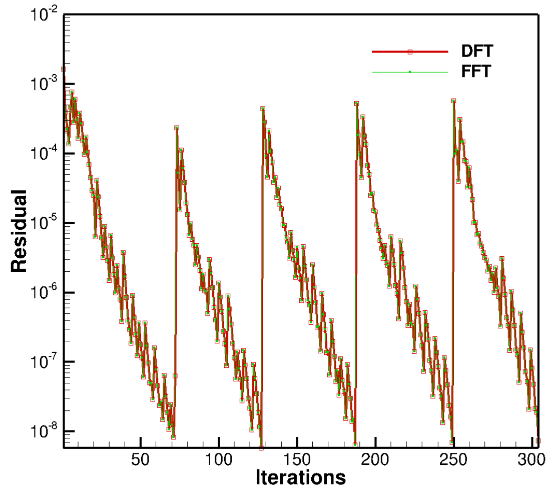

Figure 7: Residual versus iterations for DFT and FFT based GMRES/AF solvers using 16 time instances per period

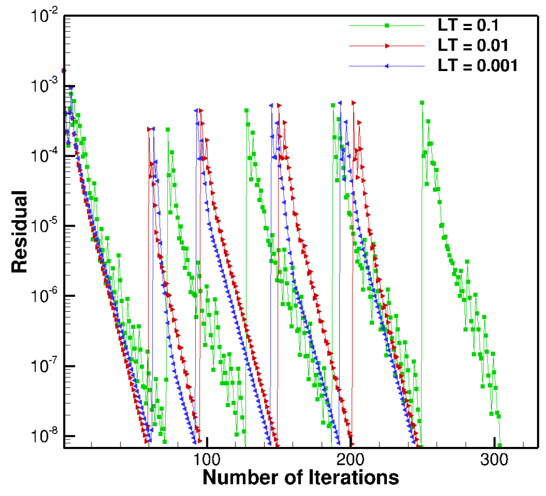

Figure 8: Non-linear residual versus number of iterations for different linear tolerance of 0.1, 0.01 and 0.001, using N=16

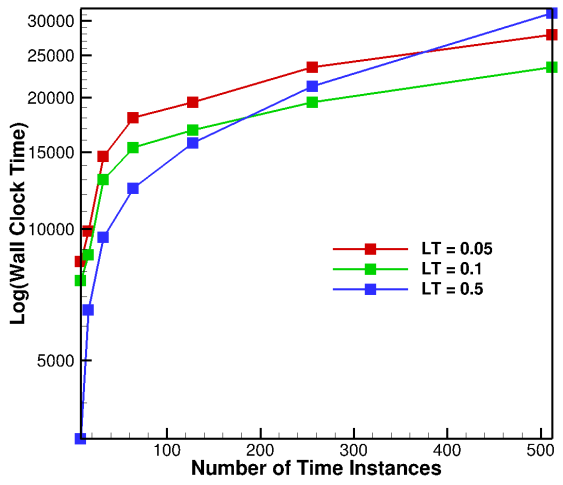

Figure 9: Log of wall clock time versus number of time instances for different linear tolerances of 0.5, 0.1 and 0.05

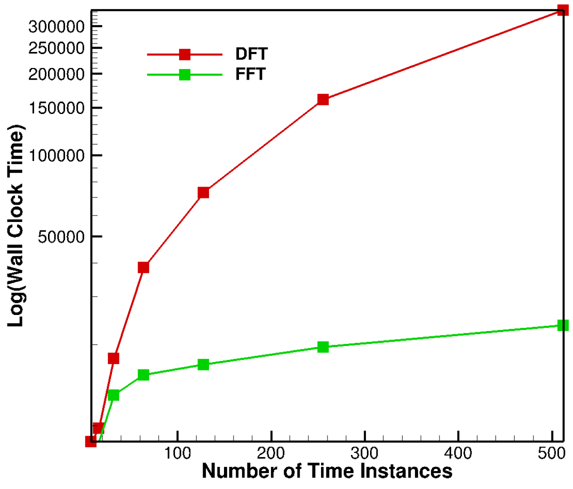

Figure 10: Log of wall clock time versus number of time instances for DFT and FFT based GMRES/AF solvers using linear tolerance of 0.1

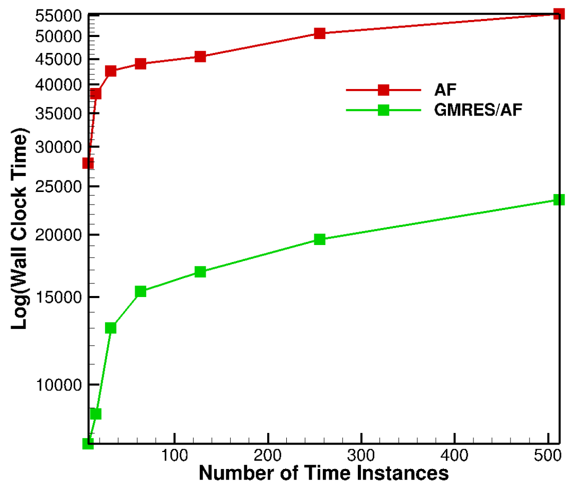

Figure 11: Log of total wall-clock time versus number of time instances for FFT-based AF and GMRES/AF solution for quasi-periodic airfoil problem

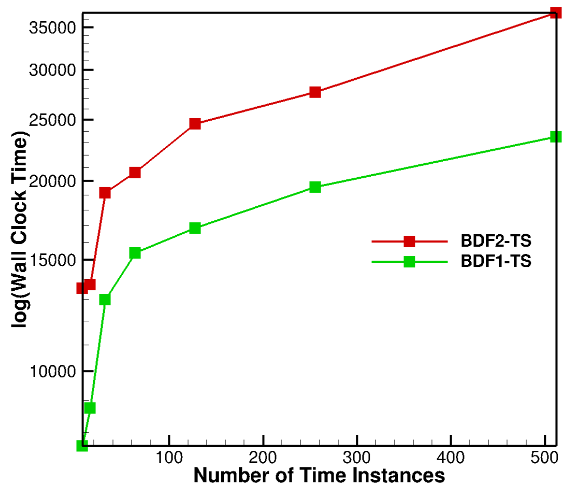

Figure 12: Log of wall clock time versus number of time instances using BDF1TS and BDF2TS

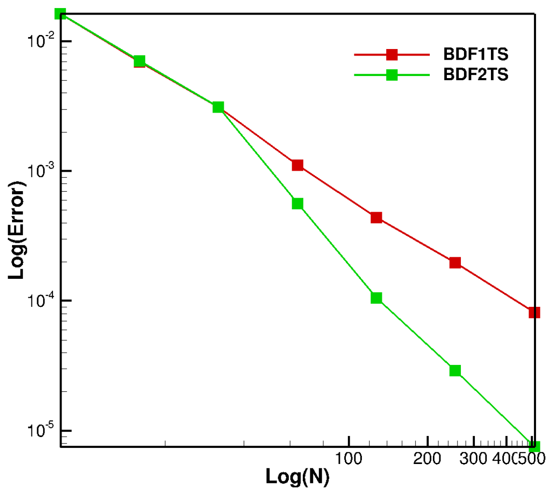

Figure 13: Lift coefficient error versus log of number of time instances using BDF1TS and BDF2TS solvers