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

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

The time-spectral (TS) method is a fast and efficient scheme for computing the solution to temporal periodic problems. Compared to traditional backward difference time-implicit methods, time-spectral methods incur significant computational savings by taking advantage of the temporal Fourier representation of the time discretization. However, the computational cost for current time-spectral solver implementations, which are based on the discrete Fourier transform (DFT), scales as $O(N^2)$ where N represents the number of time instances. For parallel implementations, where each time instance is assigned to an individual processor, the wall clock time necessary to converge time-spectral solutions is of order $O(N)$, whereas, the wall clock time for an optimal solver in this weak scaling approach is expected to be on the order of $O(1)$. The present paper is focused on making the current TS method more efficient by reducing the computational cost to $O(N \log_2 N)$ for even and $O(2N \log_3 N)$ for odd numbers of samples (where N is a power of 3) and hence for parallel implementations, achieving a wall clock time on the order of $O(\log_2 N)$ for even and $O(\log_3 N)$ for odd numbers of samples. This is achieved by developing a parallel fast Fourier transform (FFT) implementation instead of the current discrete Fourier transform approach for evaluating and solving the temporal derivative terms, which significantly decreases the computational cost and results in notably shorter wall clock time.

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 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 the 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.[4,11,12] In these previously implemented discretizations, in order to increase the efficiency and robustness of these solvers particularly for large number

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

of time instances, different pre-conditioners have been used to solve the implicit linear system over all coupled time instances using the Generalized Minimal Residual Method (GMRES).[4,12–16] However, due to the fact that the computational cost for methods based on the DFT is of the order of $O(N^2)$, where N represents the number of time instances within a period, all the previously developed methods scale as $O(N^2)$. This is not favourable since for cases with large numbers of time instances the computational cost becomes prohibitive even within a parallel computing framework.

In this paper, in order to achieve superior efficiency, the time-spectral method is implemented based on the fast Fourier transform that requires $O(N \log_2 N)$ number of operations for even and $O(2N \log_3 N)$ for odd numbers of samples (powers of 3). This approach offers a significant improvement in computational savings and can overcome the above mentioned limitations. The approximate factorization(AF) algorithm is 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.[17,18] This is 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. Furthermore, this idea has been extended to problems with temporal second-order derivative terms,[19] which are representative of structural dynamics problems.

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 for both even and odd numbers of samples are shown, including their parallel implementation. An approach for achieving additional efficiency is then presented, based on the fact that we are only concerned with the evaluation of real functional values within the complex FFT implementation. Subsequently, a brief explanation of the FFT-based approximate factorization solver is given. Finally, the implementation of the FFT approach for problems involving second-order temporal derivatives is described. Then, this new solver, based on the FFT with approximate factorization, is applied to a transonic pitching airfoil problem. The efficiency of the FFT-based solver is compared to that of the DFT based solver, for parallel implementations. Moreover, a second-order ordinary differential equation with a periodic forcing function is solved using the FFT approach as an example to demonstrate the applicability of this approach for periodic structural dynamics problems. Finally, prospects for further improvements in the overall solver are discussed.

II. Mathematical Formulation

II.A. Spatial Discretization

The Euler equations for inviscid compressible flow in integral form over a moving control volume $\Omega(t)$ can be written as follows:

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

where U is the vector of conserved quantities (mass, momentum, and energy), $F(U)$ is the conserved flux, $\vec{n}$ is the normal vector of each edge, $\Omega(t)$ is the volume of each control volume, and S is the surface of each edge, for two-dimensional discretizations. Rewriting the first term in the left hand side of equation (1) provides the following:

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

in which $\dot{x}$ is the velocity of each edge which varies with time; equation (1) thus becomes:

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

Assuming U as a cell centered variable, the spatial part of the above equation can be represented as:

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

Therefore, equation (3) can be written in the following form:

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

Here R represents the spatial discretization operator which corresponds to the integrated convective fluxes in arbitrary Lagrangian-Eulerian(ALE) form. 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.[4,11]

II.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.

II.B.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:[5,20,21]

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

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.

II.B.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.[22,23] 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.[23,24] 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.

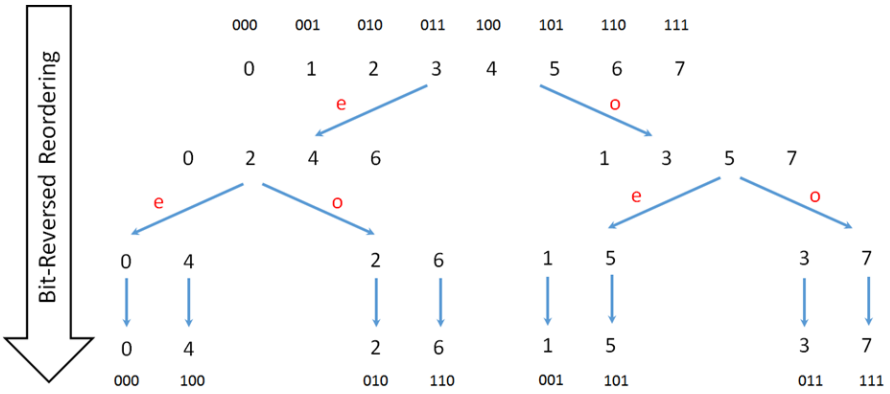


Figure 1. Recursive subdivision of $N = 8$ sample set and corresponding bit-reversal ordering

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.[22,23] 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.

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.

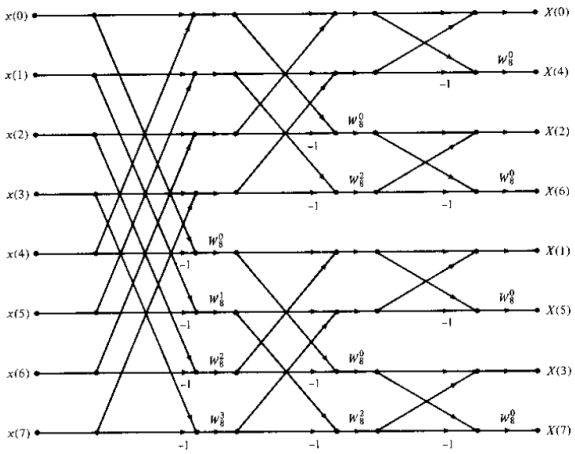


Figure 2. Pattern of communication for each level of the parallel FFT algorithm for sample size $N = 8$

II.B.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 a new rank to each processor, the extra cost of communication due to exchanging data for reordering can be avoided. The new rank in this case can be obtained by trit reversing the rank of the processor.

II.C. Optimization for real valued samples

In order to further increase the speed of the code we can take advantage of the fact that both inputs and outputs of the spectral derivative are real valued. Hence it is possible to treat N real data like $N/2$ 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. After the application of the forward and inverse FFT, the outputs from the derivative routine are rearranged to match the original set of input data. Theoretically, this splitting trick can yield a factor of 2 decrease in communication and computational cost.

II.D. 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, 11, 13, 18]

Adding a pseudo-time term to the left hand side of equation (12) and linearizing by 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 system and

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

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, 12, 25, 26]

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

which separates the contribution of the spatial and temporal parts in the calculation of ΔU. However, this separation 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. 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 colored block Gauss-Seidel 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 the use of inverse fast Fourier transfer (IFFT) to obtain ΔU.

II.E. Second-Order Derivative

The steps required to compute the second-order derivative of U with respect to time are similar to those used in the first-order derivative formulation. The only difference is that $\widehat{U_k}$ is multiplied by the term $-(\omega k)^2$, in which k is the corresponding frequency and ω is equal to $2\pi/T$. The rest of the steps are exactly the same as what has been described for the first-order derivative formulation.

Parallel implementation in second-order derivative problems is the same as for first-order derivative problems. Each time instance is assigned to an individual processor. Therefore, $O(N \log_2 N)$ communication takes place, which is a significant improvement compared to the DFT based second-order derivative algorithm which requires $O(N^2)$ communication. This implementation is required for structural dynamics equations that contain temporal second-order derivative terms.

III. Description of the Problem and Computational Results

III.A. First-Order Derivative Problem Results

The problem being studied is 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. 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 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 all cases, identical solutions are obtained with the DFT based time-spectral method and with the FFT based time-spectral method. This is seen in Figures 4 and Figures 5, where the residual convergence is identical for both DFT and FFT based implementations for odd and even numbers of samples. Figure 6 depicts the wall clock time versus the number of time instances for the FFT-based approach as originally coded, and for the version optimized for real valued data, by splitting the data into two groups which are used as real and imaginary inputs to the FFT routine, as described previously. The figure illustrates the O(log N) complexity of the algorithm and indicates that the real data splitting optimization provides a speedup which asymptotes to just under a factor of 2 at high processor counts.

Figure 7 and Figure 8 compare the wall clock time versus number of time instances for the DFT-based time-spectral solver versus the FFT-based time-spectral solver 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 7(b) and 8(b). The figures also indicate that the wall clock time grows as $O(N)$ for each processor in the DFT based method while it grows as order $O(\log N)$ in the FFT based method as the number of time instances increases. This is further demonstrated in Figure 9 which shows an approximate linear trend of wall clock time when plotted versus $log(N)$. However, as seen from the figure, the slope of the curve increases substantially for values of N greater than 1024. To further examine this behavior, the portion of wall clock time due to the various phases of the parallel FFT routine are plotted in Figure 10 versus $log(N)$. The 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 11 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.[27] 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.

Finally, Figure 12 compares the wall clock time of the FFT and DFT-based time-spectral solvers versus the number of time instances for different reduced frequencies, indicating that the performance of the approximate factorization solver is relatively insensitive to the reduced frequency of the problem.

III.B. Second-Order Derivative Problem Results

The following second-order ODE has been studied as an example of second-order problems to investigate the efficiency of the FFT-based time-spectral method for computational structural dynamics problems.

$$y'' - y' - y = sin(\omega t) \tag{24}$$

The forcing term is periodic with a period of $T = 2\pi/\omega$, which allows the use of the time-spectral method. In order to solve the above equation we define the residual Res as:

$$y'' - y' - y - sin(\omega t) = Res \tag{25}$$

The derivatives are discretized using the time-spectral approach and the resulting Newton scheme is written as:

$$[DD_{TS} - D_{TS} - I]\,\Delta y = -Res \tag{26}$$

with

$$y = y_{initial} + \Delta y \tag{27}$$

Here $[DD_{TS}]$ and $[D_{TS}]$ are the matrices containing the time-spectral coefficients for the second and first-order derivative terms of y, respectively, and $y_{initial}$ is the initial value for y. Applying the same idea used previously in the approximate factorization algorithm, Δy can be found by taking equation(26) to the frequency domain. Therefore the system of coupled equations changes to N decoupled equations since the spectral matrices are strictly diagonal in the frequency domain. Denoting $\widehat{\Delta y_k}$ and $\widehat{Res_k}$ as the Fourier transforms of Δy and Res, respectively, the solution update in frequency space is computed as:

$$\widehat{\Delta y_k} = \frac{\widehat{Res_k}}{-1 - (\omega k)^2 - i\omega k} \tag{28}$$

where i is the imaginary unit. Next $\widehat{\Delta y_k}$ is transferred back to the time domain using the inverse Fourier transform to obtain Δy.

In order to simulate the solution of a partial differential equation with a corresponding spatial discretization, a total of 20,000 ODEs are solved simultaneously for each time instance and processor. Figure 13 shows the wall-clock time required for the solution of these equations when the temporal derivative terms are implemented based on the FFT and the DFT approaches. The solutions of mid y (unknown number 10000) obtained from these two implementations are shown in Figure 14. These figures illustrate that identical solutions are obtained in both cases, while the FFT-based solver is significantly more efficient that the DFT-based solver, particularly for large values of N, just as in the previous first-order computational fluids dynamics (CFD) results.

IV. Conclusion and Future Work

In the present work a new approach for parallel solution of the time-spectral discretization has been developed that drastically decreases the amount of computation and communication required for the parallel solution of time-periodic problems. The approach is based on the fast Fourier transform and scales as $O(N \log_2 N)$ for even and $O(2N \log_3 N)$ for odd (powers of 3) numbers of time instances. This approach results in significant savings compared to previous implementations that are based on the discrete Fourier transform requiring $O(N^2)$ operations and communication. An approximate factorization algorithm has been used to separate the spatial and temporal parts of the solution, providing an effective technique for problems with large numbers of time instances and/or high reduced frequencies. Further gains in wall-clock time appear by taking advantage of the fact that all inputs and outputs of the FFT routines are real-valued.

Solutions for a pitching airfoil problem discretized with the Euler equations on an unstructured mesh using the FFT-based and DFT-based time-spectral approaches have been performed, using large even and odd numbers of time instances (up to N=4096). The convergence history of solutions for both methods illustrates that both implementations give identical convergence histories and solutions. However, the wall-clock time required for the FFT based time-spectral method is significantly lower than that required for the DFT-based time-spectral method in all cases. The FFT-based time-spectral solver has also been extended to second-order ODE problems, demonstrating its applicability to structural dynamics problems.

Further increases in overall solution efficiency can be obtained by formulating the FFT-based approximate factorization time-spectral solver as a preconditioner for a Newton-Krylov method applied to the complete time-space discretization, as demonstrated in previous work using the DFT approach.[4,12,15] Finally, by combining the temporal parallelism afforded by this approach with spatial parallelism, it is anticipated that the solution of periodic and quasi-periodic problems of moderate spatial size can be effectively scaled to hundreds of thousands of cores.

V. Acknowledgements

This work was partly funded by ONR Grant N00014-14-1-0045 and by the Alfred Gessow Rotorcraft Center of Excellence through a subcontract with the University of Maryland. Computer resources were provided by the University of Wyoming Advanced Research Computing Center and by the NCAR-Wyoming Supercomputer Alliance.

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]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.

[12]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.

[13]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.

[14]Mundis, N. L. and Mavriplis, D. J., "An Efficient Flexible GMRES Solver for the Fully-coupled Time-spectral Aeroelastic System," 2014, AIAA Scitech Forum 2014-1427, 52nd Aerospace Sciences Meeting, National Harbor, Maryland, 2014.

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

[16]Mundis, N. L. and Mavriplis, D. J., "Wave-number Independent Preconditioning for GMRES Time-spectral Solvers," 2015, AIAA SciTech Forum 2015-0568, 53th AIAA Aerospace Sciences Meeting, Kissimmee, Florida, January 2015.

[17]Thomas, J. P., Custer, C. H., Dowell, E. H., and Hall, K. C., "Unsteady flow computation using a harmonic balance approach implemented about the OVERFLOW 2 flow solver," 2009, AIAA paper 2009-4970, 19th AIAA Computational Fluid Dynamics, San Antonio, Texas, June 2009.

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

[19]Johnson, S. G., "Notes on fft-based differentiation," 2011, MIT Applied Mathematics.

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

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

[22]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.

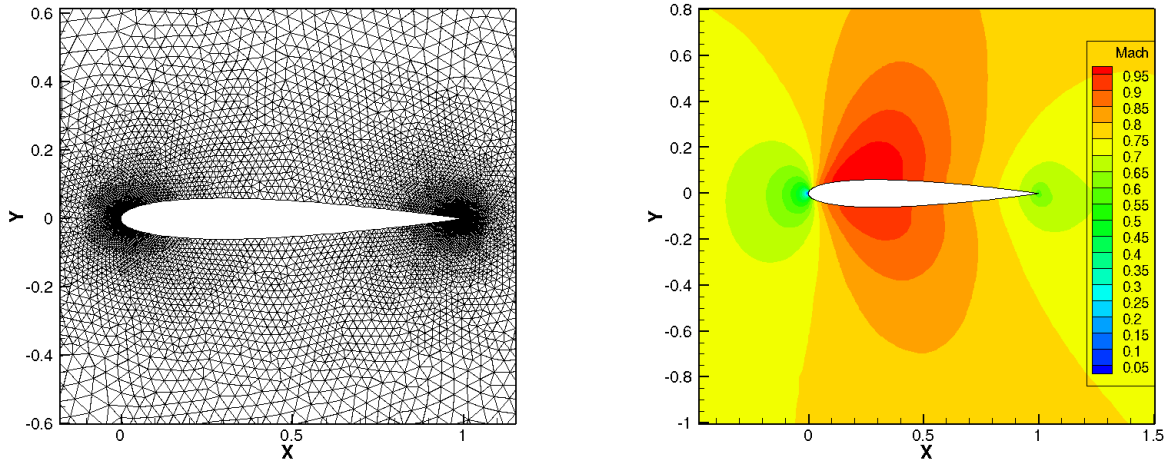
[23]Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., "Numerical Recipes," 1992.

[24]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.

[25]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.

[26]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.

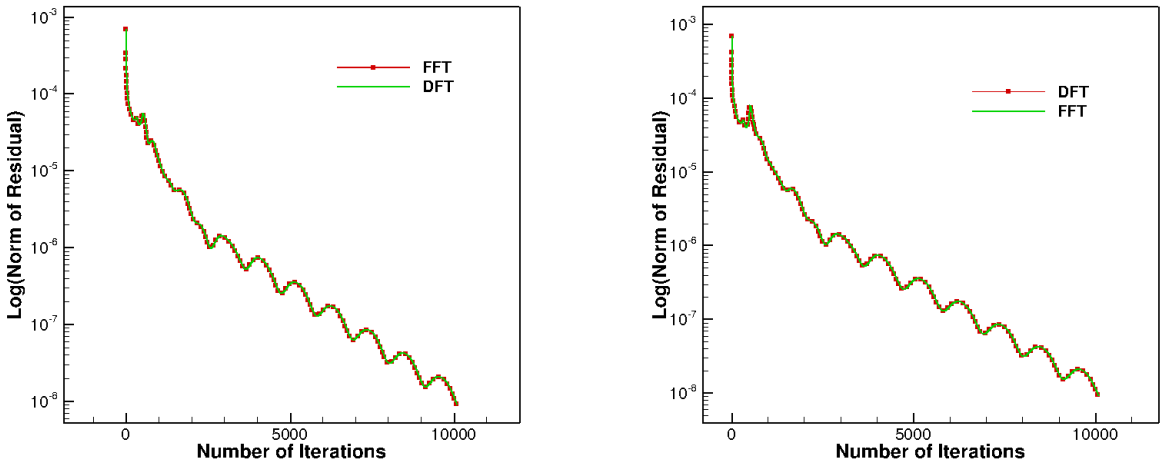
[27]Loft, R., Andersen, A., Bryan, F., Dennis, J. M., Engel, T., Gillman, P., Hart, D., Elahi, I., Ghosh, S., Kelly, R., et al., "Yellowstone: A dedicated resource for earth system science," *Contemporary High Performance Computing: From Petascale Toward Exascale, Volume Two*, Vol. 2, 2015, pp. 262.



(a) Unstructured mesh

(b) Computed Mach contours

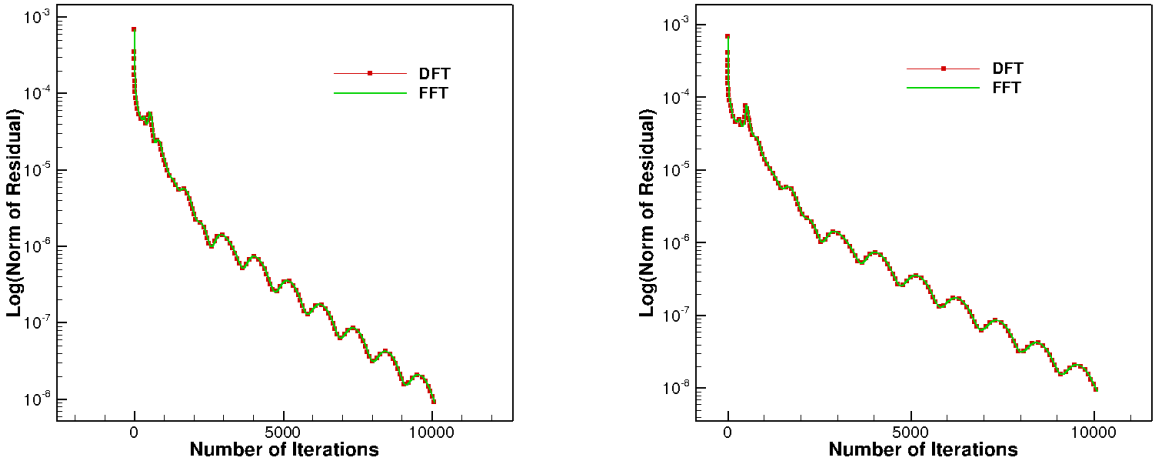
Figure 3. Computational mesh and computed Mach contours for time-spectral solution of pitching airfoil problem using 64 time instances



(a) 64 time instances

(b) 256 time instances

Figure 4. Residual versus number of iterations for different even numbers of time instances



(a) 81 time instances

(b) 243 time instances

Figure 5. Residual versus number of iterations for different odd numbers of time instances

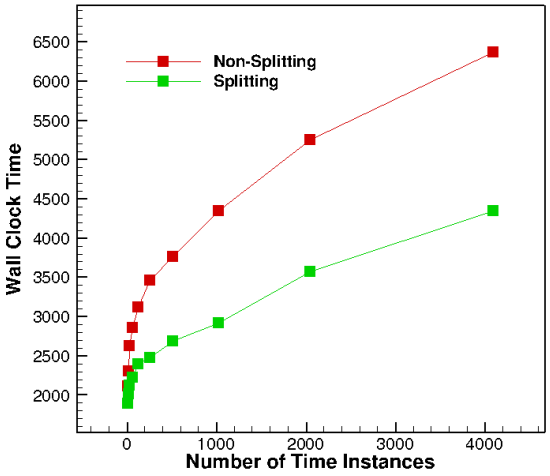
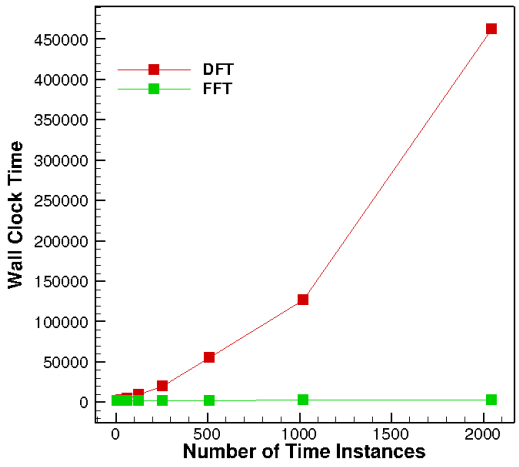
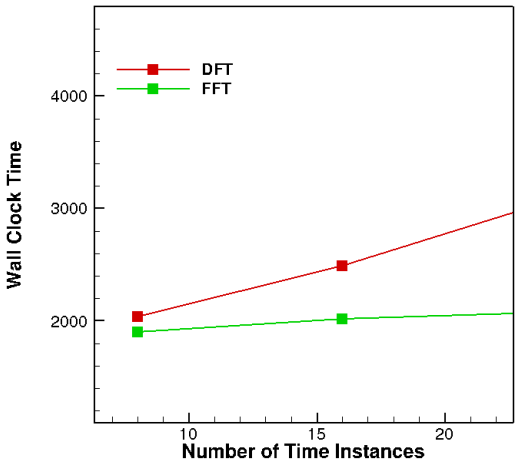


Figure 6. Wall clock time versus number of time instances for original complex FFT and real-data split FFT implementations.

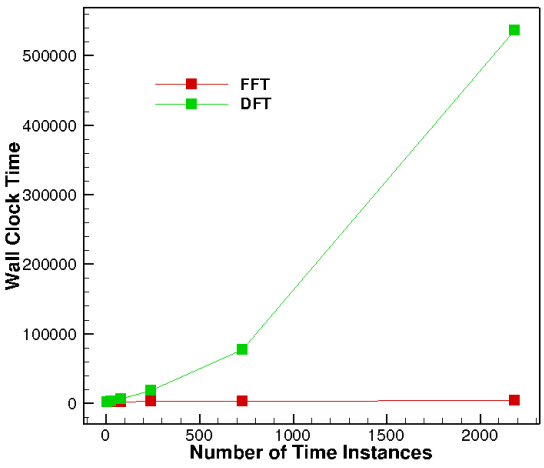


(a) Wall clock time for up to 2048 number of samples

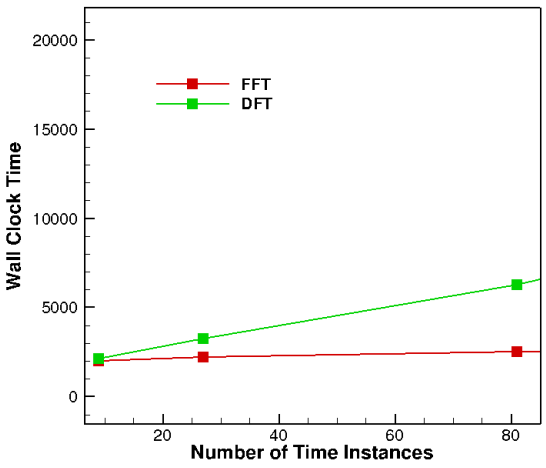


(b) Wall clock time for small number of samples

Figure 7. Wall clock time versus number of time instances 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 8. Wall clock time versus number of time instances for odd numbers of samples

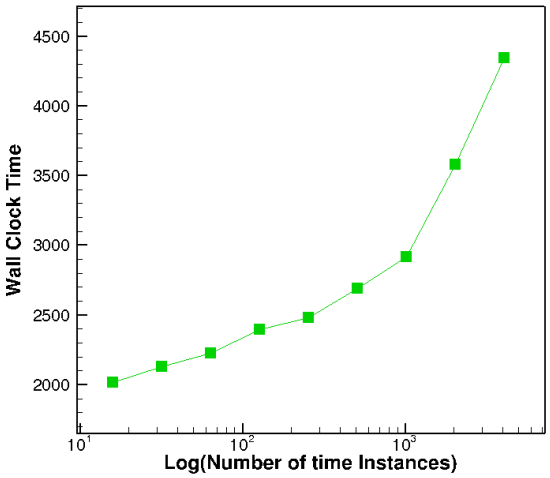


Figure 9. Wall clock time versus log of number of time instances for even number of time instances up to N=4096 on NCAR-Wyoming Yellowstone supercomputer

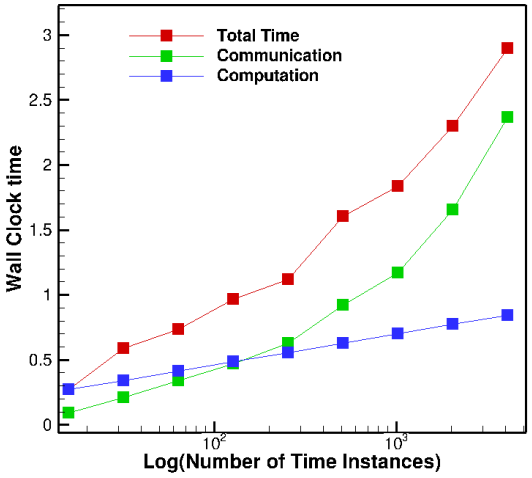


Figure 10. Breakdown of wall-clock time for computation and communication within parallel FFT routine running on NCAR- Wyoming Yellowstone supercomputer using up to 4096 processors

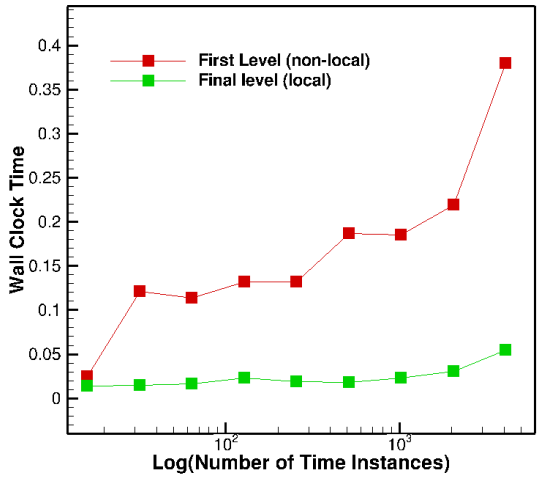


Figure 11. Comparison of communication time for first and last level of parallel FFT routine using up to 4096 processors

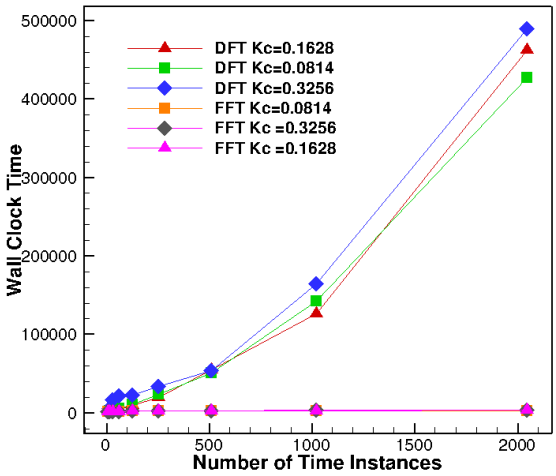
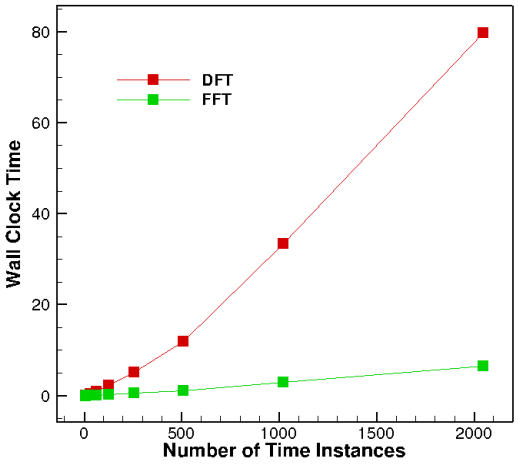
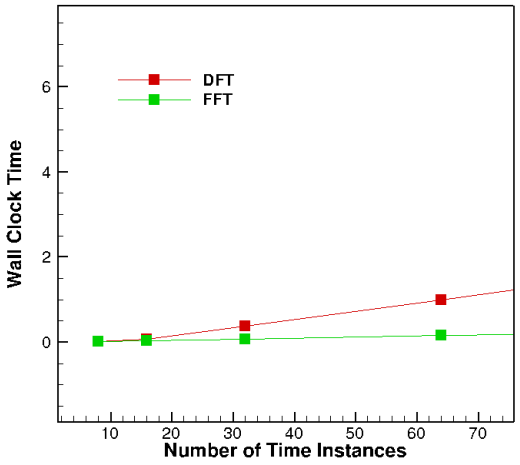


Figure 12. Wall clock time versus number of time instances for different reduced frequencies

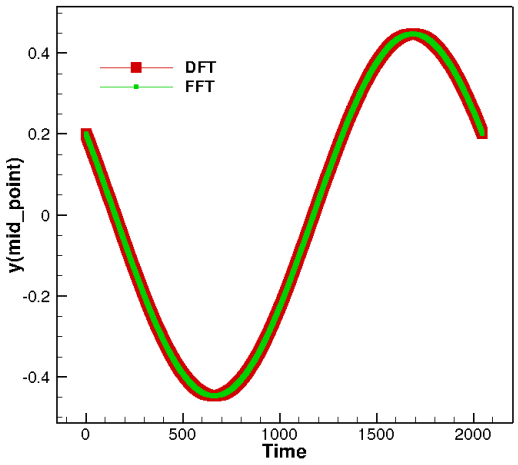


(a) Wall clock time for up to 2048 number of samples

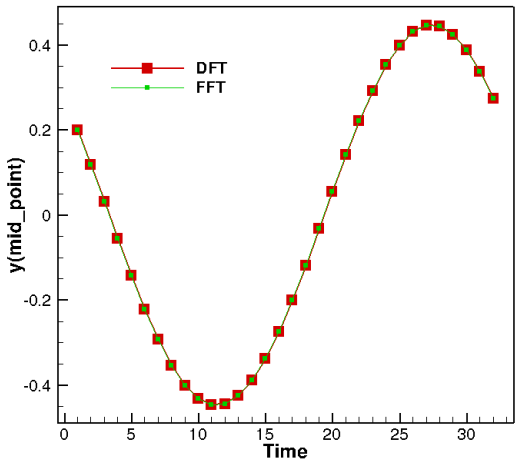


(b) Wall clock time for small number of samples

Figure 13. Wall clock time versus number of time instances for the solution of second order problem



(a) solution of mid y for 2048 number of time instances



(b) solution of mid y for 32 number of time instances

Figure 14. Solutions of mid y versus time for different numbers of time instances/ processors