To the University of Wyoming:

The members of the Committee approve the dissertation of Donya Ramezanian presented on July 12, 2019.

Dr. Dimitri J. Mavriplis, Chairperson

Dr. Jayanarayanan Sitaraman, External Department Member

Dr. Jonathan W. Naughton

Dr. Michael Stoellinger

Dr. Joshua Leffell

Dr. Fred Furtado

APPROVED:

Dr. Carl P. Frick, Head, Department of Mechanical Engineering

Dr. Michael V. Pishko, Dean, College of Engineering and Applied Sciences

Ramezanian, Donya, An Order $NlogN$ Parallel Time Spectral Solver for Periodic and Quasi Periodic Problems, Ph.D., Department of Mechanical Engineering, July, 2019.

With the growing of computational resources within the last decades, the desire to develop scalable and faster algorithms has increased notably. One of the most computational costly category of Computational Fluid Dynamic (CFD) problems is unsteady flow problems, and periodic flow problems are one extensive category of unsteady problems. The time-spectral method is a fast and efficient scheme for computing this sub category of unsteady problems. Compared to traditional backward difference implicit time-stepping methods, time-spectral methods incur significant computational savings by using a temporal Fourier representation of the time discretization and solving the periodic problem directly. Exponential convergence trend is another advantage of time-spectral method over the time-accurate methods, which means that the rate of error decrease of this method accelerates as more time-instances are added. Therefore, convergence can become faster than any polynomial order. In the time-spectral discretization, all time instances are fully coupled to each other, resulting in a dense temporal discretization matrix, the evaluation of which scales as $O(N^2)$, where N denotes the number of time instances. This implementation of time-spectral method that is based on discrete Fourier transform (DFT) results in significant computational costs and wall-clock time especially in problems with large number of time instances. The goal of this dissertation is to provide a new parallel implementation of time-spectral methods that decreases computational expenses significantly. In this work the time-spectral method is implemented based on the fast Fourier transform (FFT) for power of two and three numbers of time-instances. The computational cost of this implementation can be reduced to $O(NlogN)$. Furthermore, in parallel implementations, where each time instance is assigned to an individual processor, the wall-clock time necessary to solve time-spectral problems is reduced to $O(logN)$ using the FFT-based approach, as opposed to the $O(N)$ weak scaling incurred by previous dense matrix or discrete Fourier transform (DFT) parallel time-spectral solver implementations.

Additionally, this work addresses an extension of the proposed parallel FFT-based time-spectral method to quasi-periodic problems. 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. 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 $O(NlogN)$.

Different parallel FFT-based solvers are developed and studied for periodic and quasi-periodic problems. An FFT-based approximate factorization (AF) scheme is used to solve time-spectral problems with large numbers of time instances. Subsequently, this solution strategy is reformulated as a preconditioner to be used in the context of a Newton-Krylov method applied directly to the complete non-linear space-time time-spectral residual. The use of the Generalized Minimal Residual Method (GMRES) Krylov method enables additional coupling between the various time instances running on different processors resulting in faster overall convergence. The GMRES/AF scheme is shown to produce significantly faster convergence than the AF scheme used as a solver alone, and achieves orders of magnitude gains in efficiency compared to previous DFT-based implementations of similar solution strategies for large numbers of time instances.

AN ORDER *NLOGN* PARALLEL TIME SPECTRAL SOLVER FOR PERIODIC AND QUASI PERIODIC PROBLEMS

by

Donya Ramezanian,
B.S. Mechanical Engineering, University of Tehran (2012)

A dissertation submitted to the
Department of Mechanical Engineering
and the
University of Wyoming
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY
in
MECHANICAL ENGINEERING

Laramie, Wyoming
July 2019

To Albert Camus

Who helped me find the invincible summer within me, in the midst of winter.

Contents

List of Figures

Acknowledgments

First, I am extremely grateful for my advisor, Professor Dimitri Mavriplis, for his consistent support and advice. It was a unique opportunity to learn CFD under his supervision. His guidance, patience and attitude made me passionate about considering academia as my future career goal.

Also, I am thankful to my family, my parents, Mahin and Abdolhossein and my sister Sara, for all their love, support and encouragement during these years. Without them, I would not be able to complete this work.

Further, I like to thank my committee members, Dr. Sitaraman, Dr. Naughton, Dr. Furtado, Dr. Leffell and Dr. Stoellinger for their reviews and comments, and my colleagues in the high altitude CFD lab, especially Soudeh Kamali. Without her this lab and Wyoming would not be as fun. I would like to extend my thanks to my friends all over the world, especially Reihaneh, Ghazal, Bahar, Soudabeh, Saeedeh, Ehsan and Amir for making my life full of hope and joy.

Finally I would like to acknowledge the institutions for the financial and computational resource support during the course of this research. I appreciate NASA NRA Grant NNX15AU23A by NASAs Transformational Tools and Technologies (TTT) Project of the Transformative Aeronautics Concepts Program. I am also thankful for ONR Grant N00014-14-1-0045 and the Alfred Gessow Rotorcraft Center of Excellence through a subcontract with the University of Maryland. Finally, I am grateful for computer time provided by the NCAR-Wyoming Supercomputer Center(NWSC) and by the University of Wyoming Advanced Research Computing Center(ARCC).

Donya Ramezanian

University of Wyoming

July 2019

Chapter 1

Introduction

Modern computing capabilities and numerical techniques have been refined over the years. With that, Computational Fluid Dynamics (CFD) plays a big role in giving outstanding insight to complex engineering problems, and yet many simulations exists that require more efficient CFD methods, in order to gain the desirable accuracy. The simulation of unsteady phenomena typically demands large computational investments to achieve suitable accuracy. Temporally periodic problems are one of the sub-categories of unsteady problems that have broad range of applications in industry, such as turbomachinery flows, rotorcraft problems, etc. Traditionally, time-marching methods were employed for unsteady flow problems including temporally periodic problems. This requires that the unsteady governing equations must be integrated forward in time for several periods until the initial transient part is resolved and a periodic steady state is obtained. In most realistic problems, solving the transient part is very time consuming, making time-marching methods inevitably expensive. Therefore, frequency-domain methods that directly solve for the periodic solution and avoid the initial transient parts are more favorable in these problems.

1.1 Previous Research

The earliest frequency-domain methods approached temporal periodic problems by splitting the flow variables into steady and unsteady parts. The steady set of flow equations are solved first and then used in the solution of the unsteady set of flow equations. The unsteady equations lead to a set of uncoupled equations for each mode that can be solved independently [1, 2]. Although this treatment has a very low computational cost and requires negligible storage, it does not predict accurate results for problems with significant nonlinear content. Another method was developed by Adamczyk that included some portion of the nonlinear effects [3]. In this method the flow variations were decomposed into a time-averaged part and an unsteady part. The time-average part creates deterministic stress terms, due to the non-linear nature of the Euler equations. The non-linear harmonic method was proposed by He [4] for aeroelastic applications [5,6] and blade-row interactions [7]. This method resolves all the non-linear effects. Similar to the method proposed by Adamczyk, this method decomposes the flow variables into a time-averaged part and an unsteady part. However, unlike Adamczyk's method, the deterministic stress terms were not modeled in this method. In this method the two inter-dependent equations were solved iteratively until the whole system is converged. Subsequently, the Harmonic Balance (HB) method has been developed by Hall et al.. This method linearizes the governing equations (potential, Euler or NavierStokes) about a nonlinear steady or mean operating condition. The result is a set of linearised equations in which time derivatives have been replaced by the operator $j\omega$. The resulting linearised equations can be discretised and solved very efficiently using conventional steady solution acceleration techniques. This method has evolved over the years. Initially this method was developed for the linearized unsteady Euler equations [8–10], while later development of this method could resolve the full nonlinear effects and time domain techniques [11, 12].

Thereafter, the Non-Linear Frequency Domain(NLFD) method was proposed by McMullen et al. [13,14], which solves the full nonlinear Euler/RANS equations in the frequency domain. Although this method accounts for all the non-linear effects of the flow, it is computationally expensive as two Fourier transforms are required in each iteration and it requires large storage since the solution at all time levels must be stored. The Time-Spectral method is an extension of the NLFD method, with the difference that it solves the governing equations in the time domain, and recasts the frequency domain equations back to the time domain. The time-spectral derivative term can be added to any spatial solver, which enables the use of any state-of-the-art solver for the spatial part. In many cases, the time-spectral method, using small number 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 [13–15], a discrete Fourier analysis is used in time-spectral methods where unsteady equations in the time 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 [16]. Higher order accuracy and lower computational cost are two main advantages of the time-spectral method compared to traditional time-stepping. In the time-spectral method spectral accuracy can be achieved since Fourier representations are used for the time discretization [16]. 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 [17, 18], oscillatory pitching airfoils [19], turbomachinery flows [20, 21], and flapping wings [22]. 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. Simple implicit solvers such as block-Jacobi and Gauss-Seidel were initially employed but found to produce slow convergence. Therefore, a Newton solution strategy was adopted, where a Generalized Minimal Residual method (GMRES) was used to solve the linear system at each Newton step, and the aforementioned linear iterative solvers were used as preconditioners for GMRES. Examining different preconditioning strategies, an approximate factorization (AF) scheme solves the temporal and spatial components in two successive steps was found to be the most efficient approach overall, particularly for problems with large numbers of harmonics and/or high reduced frequency [16, 23–25].

However, the cost of all these methods scales as $O(N^2)$ where N denotes the number of time-instances, due to the fully coupled nature of the time-spectral discretization. When implemented in parallel, using one time-instance per processor, the wall-clock time of these methods scales as $O(N)$ due to the $O(N^2)$ communication.

A principal disadvantage of time-spectral methods is that they are only applicable to purely periodic problems. 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 [24]. 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 [24]. 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. The aeroelastic flutter problem is an example of a coupled fluid/structure problem with strong periodic content and a slow transient motion.

1.2 Dissertation Overview

The objective of this thesis is to develop the time-spectral method based on the parallel fast Fourier transform (FFT) for solving periodic and quasi-periodic problems. This approach offers a significant improvement in computational savings, reducing the cost of computations and wall-clock time from $O(N)$ for each processor to $O(logN)$ [26]. Through the course of this work, different TS solvers that are suggested by previous researches are developed based on the parallel FFT. The AF scheme is implemented based on the parallel FFT and its performance is studied in solving different periodic and quasi-periodic problems. In many realistic problems, large numbers of time-instances are needed to resolve both low and high frequency periodic content. As the number of time-instances becomes larger or the period of the flow becomes shorter, the non-linear system associated with the time- spectral method becomes larger and stiffer to solve. Achieving superior efficiency with the time-spectral method requires a robust solver strategy that solves the large non-linear space-time system rapidly. The AF scheme is suggested to be used as a preconditioner for GMRES in previous studies [23]. In the present work the FFT-based AF scheme is reformulated as a preconditioner for GMRES, which is used in turn to solve the fully coupled space-time system at each non-linear step in the Newton Method. The performance of the parallel FFT-based GMRES/AF solver is investigated in periodic and quasi-periodic problems. In what follows the primary contributions of the research are highlighted:

1.2.1 An Order of $NlogN$ Parallel Time Spectral Method for Periodic Problems

In this work, the time-spectral method is derived using a parallel base-2 and -3 fast Fourier transform (FFT). In this new implementation the wall-clock time necessary to converge time-spectral solutions is reduced to $O(logN)$, for N number of time-instances as opposed to $O(N)$ weak scaling incurred by previous DFT parallel time-spectral solver implementations. The parallel FFT implementation devised in this work allows for the efficient computation of time-spectral problems with large numbers of time instance for increased temporal accuracy. Although in many cases time-spectral methods are used for problems where only a small number of harmonics are of interest, there exist many practical applications for which larger numbers of time instances are required in order to deliver the temporal accuracy that is competitive with traditional implicit time-stepping methods. These most often involve periodic problems with a rapid localized change in time, for which spectral basis functions are less efficient. Practical examples include dynamic stall on the retreating blade of a helicopter rotor [27], or periodic interactions of a rotor as it passes close to a stationary object such as a wind turbine tower, or a rotor-stator interaction in turbomachinery. The new implementation of TS method accomplished in this work, made this method competitive with implicit time-stepping methods in this categories of problems.

1.2.2 An Order of $NlogN$ Parallel BDF/Time-Spectral method for Quasi-Periodic Problems

As discussed previously, the hybrid BDF/time-spectral (BDFTS) approach can be used to simulate flows with a strong periodic content in addition to a slow mean transients, such as fixed-wing flutter problems. This work attempts to develop a FFT-based algorithm for quasi-periodic problems. 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 $O(NlogN)$ and requires $O(logN)$ wall-clock time when implemented in parallel using one processor per time instance.

1.2.3 Dissertation Outline

$\boxed{\text{Chapter 2}}$ Provides the mathematical formulations involved in spatial and temporal discretization of the Euler equations. The backward-difference formulas are provided as well as TS and BDFTS methods and the parallel implementation of FFT-based TS and BDFTS methods are presented.

$\boxed{\text{Chapter 3}}$ Covers all the solution methods that are used in this thesis, including implementing FFT-based AF algorithm in periodic and quasi-periodic problems and the GMRES method.

$\boxed{\text{Chapter 4}}$ Presents the results of solving periodic pitching airfoil problems using FFT-based TS solvers. First the performance of TS solvers are examined in a problem with a single frequency prescribed motion and then in a problem with Gaussian bump prescribed motion. At the end of this chapter the performance of FFT-based TS method is studied in a simple second-order derivative problem.

$\boxed{\text{Chapter 5}}$ Examines the results of solving a quasi-periodic pitching airfoil problems using FFT-based quasi-periodic TS solvers.

$\boxed{\text{Chapter 6}}$ Summarizes the entire thesis, draws conclusions and suggests future directions of this work.

Chapter 2

Mathematical Formulations

2.1 Governing Equation

Since this work studies inviscid compressible flows, the Euler equations are the governing equations. The two dimensional conservative form of the Euler equations can be written as:

$$\frac{\partial U}{\partial t} + \boldsymbol{\nabla} \cdot F(U) = 0 \tag{2.1}$$

in which U is the vector of conserved variables (mass, momentum and energy) and $F(U)$ represents the conservative fluxes. Equation (2.1) can be written as:

$$\begin{pmatrix} \rho \\ \rho u \\ \rho v \\ E \end{pmatrix}_t + \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (E+p)u \end{pmatrix}_x + \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ (E+p)v \end{pmatrix}_y = 0 \tag{2.2}$$

where ρ, p, and E denote the density, pressure and total energy and u, and v indicate the Cartesian velocity components of the flow. Since many computational fluid dynamics problems are discretized over unstructured meshes, finite volume method which is a conservative method is easier to be formulated to allow for unstructured meshes. Finite volume refers to the small volume surrounding each node point on a mesh. In the finite volume method, volume integrals in a partial differential equation that contain a divergence term are converted to surface integrals, using the divergence theorem. These terms are then evaluated as fluxes at the surfaces of each finite volume. The integral form of the Euler equations over a moving control volume in two dimensions can be written as:

$$\int_{\Omega(t)} \left(\frac{\partial U}{\partial t}\right) dV + \int_{\partial\Omega(t)} (F(U) \cdot \vec{n}))\, dS = 0 \tag{2.3}$$

in which $\Omega(t)$ is the control volume, $\vec{n}$ is the normal vector of each surface and S is the surface of each control volume. The first term in the left hand side of equation (2.3) can be rewritten as:

$$\int_{\Omega(t)} \frac{\partial U}{\partial t} dV = \frac{\partial}{\partial t} \int_{\Omega(t)} U dV - \int_{\partial\Omega(t)} U(\dot{x}.\vec{n}) dS \tag{2.4}$$

where $\dot{x}$ is the velocity of each surface of each control volume cell which varies with time. Using equation (2.4) the Euler equations can be written in a form that the temporal derivation is taken outside the integral. then Equation (2.3) can be represented as:

$$\frac{\partial}{\partial t} \int_{\Omega(t)} U dV + \int_{\partial\Omega(t)} (F(U) - U\dot{x}).\vec{n} dS = 0 \tag{2.5}$$

The second term of the above equation can be written as:

$$R(U, \dot{x}, \vec{n}) = \int_{\partial\Omega(t)} (F(U) - U\dot{x}).\vec{n} dS \tag{2.6}$$

where U is assumed to be a cell-centered variable. Therefore, equation (2.5) is then given as follows:

$$\frac{\partial(UV)}{\partial t} + R(U, \dot{x}, \vec{n}(t)) = 0 \tag{2.7}$$

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 cell-centered discretization with added artificial dissipation on unstructured triangular meshes similar to that used in previous work [16, 28]. In the cell-centered

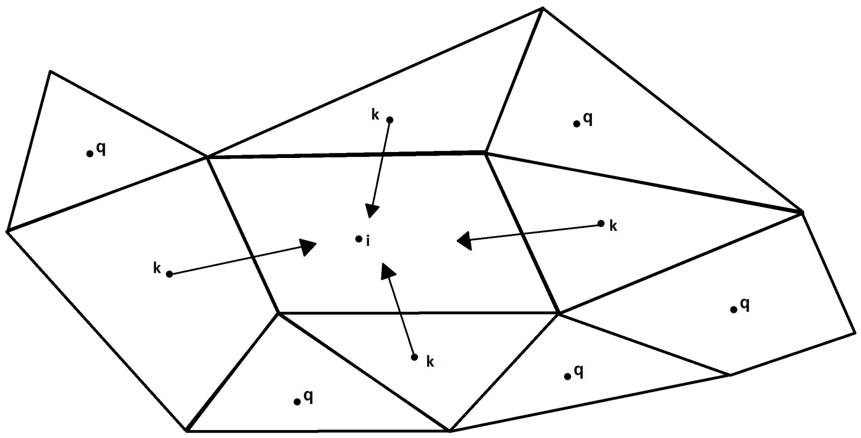


Figure 2.1: Analysis of fluxes through the surfaces of a cell-centered control volume

finite volume method the cells serve directly as control volumes containing the degrees of freedom stored on a per-cell basis as shown in Figure 2.1.

2.2 Spatial Discretization

The spatial part of the Euler equations are discretized by a cell-centered central-difference finite-volume scheme. All the problems in this work are two dimensional and are solved on unstructured grids. To evaluate fluxes at the cell faces, computing the convective and diffusive fluxes is required. For control volume ”i”, with its closest neighbors ”k” and the neighbors of neighbors ”q”, as shown in Figure 2.1, the total flux is calculated from:

$$F_{ik} = F_{conv,ik} + F_{Diff,ik} \tag{2.8}$$

In this work, the convective term at the cell face are computed using central approximation.

$$F_{conv,ik} = \frac{1}{2}[F_i(U_i) + F_k(U_k)] \tag{2.9}$$

and the diffusive flux can be obtained from the matrix dissipation model of:

$$F_{Diff,ik} = \kappa \underline{T}\,|\underline{\Lambda}|\,\underline{T}^{-1}(U_i - U_k) \tag{2.10}$$

In this evaluation, the dissipative terms of each discrete equation are scaled by the appropriate eigenvalues of the flux Jacobian matrix. Since the Euler equations are a strongly hyperbolic system, the Jacobian matrix can be diagonalized. $\underline{T}$ is a matrix that contains the eigenvectors of the Jacobian matrix normal to the control-volume face between i and k, and matrix $|\underline{\Lambda}|$ is the diagonal matrix containing the absolute values of the eigenvalues associated with the Euler equations, which are the forward acoustic, backward acoustic, and convective eigenvalues. Since one eigenvalue is repeated, there are three distinct eigenvalues: $u_n - \dot{x_n}$, $u_n - \dot{x_n} + c$, $u_n - \dot{x_n} - c$ where u_n is the fluid velocity normal to the control volume face, $\dot{x_n}$ is the grid velocity normal to the control volume face and c is the speed of sound at the face. One of the eigenvalues approaches zero, near stagnation points or near sonic lines. A zero artificial viscosity would create instabilities. To solve this problem, the eigenvalues are modified as:

$$\begin{aligned} |u_n - \dot{x_n}| &= max[|u_n - \dot{x_n}|\,, \delta(|u_n - \dot{x_n}| + c)] \\ |u_n - \dot{x_n} + c| &= max[|u_n - \dot{x_n} + c|\,, \delta(|u_n - \dot{x_n}| + c)] \\ |u_n - \dot{x_n} - c| &= max[|u_n - \dot{x_n} - c|\,, \delta(|u_n - \dot{x_n}| + c)] \end{aligned} \tag{2.11}$$

where $|u_n - \dot{x_n}| + c$ is the maximum eigenvalue and δ is an empirical factor with a value between zero and one [23].

2.3 Temporal Discretization

There are many ways to discretize the temporal derivative term in equation (2.7). In explicit methods, the size of the time steps is limited by the stability criteria, which depends on the size of the computational cell and the speed of wave. On the other hand, implicit methods are unconditionally stable and are not limited by the stability criteria. However, implicit methods require information that has not been computed yet. Although the focus of this thesis is to look at the efficiency and accuracy of spectral discretization of the time-derivative term of equation (2.7), the implicit time discretization is also described here for the comparison purposes. In this section, first the most popular time marching methods are described, later DFT and FFT discretizations of the time-spectral method are presented.

Finally, the implementations of linear and quadratic BDFTS for quasi-periodic problems are shown.

2.3.1 Backward Difference Formula

The first-order backward difference formula (BDF1) requires the values of the conserved vectors in each control volume at the current time step and the previous time step. This method treats the time derivative term in the following way:

$$\frac{\partial(VU)}{\partial t} = \frac{(VU)^n - (VU)^{n-1}}{\Delta t} \tag{2.12}$$

where n denotes the values of the conserved vectors in the current time step, and $n-1$ denotes the values of the same vectors from the previous time steps. The accuracy of the temporal error obtained by this method is first-order, $O(\Delta t)$. For accuracy purposes second-order backward difference (BDF2) formula is more favorable in most of the studies. This formula gives the temporal error accuracy of $O(\Delta t^2)$. The BDF2 formula is given as follows:

$$\frac{\partial(VU)}{\partial t} = \frac{3(VU)^n - 4(VU)^{n-1} + (VU)^{n-2}}{2\Delta t} \tag{2.13}$$

Observe that in this formula, the time derivative at time step n depends on the solution at the previous two time steps ($n-1$, $n-2$) as well as the solution at the current time step (n). Using BDF2 as the time scheme for descritizing the Euler equations changes the equation (2.7) to:

$$\frac{3(VU)^n - 4(VU)^{n-1} + (VU)^{n-2}}{2\Delta t} + R(U, \dot{x}, \vec{n}(t))^n = 0 \tag{2.14}$$

Higher-order backward difference formulas require variables from further previous time steps. Although these methods give higher accuracies, they are not stable [29]. This instability is proved by the theorem known as the Second Dahlquist barrier. This theorem says that the highest order of an A-stable multi-step method is 2. Therefore, in this study the BDF2 time-stepping scheme is used where comparisons are needed.

2.3.2 Time Spectral Method

For problems with temporal periodic content, the time-spectral method is a strong alternative for calculating the temporal derivative term. Some properties of the time-spectral method make this method favorable to study. This method achieves spectral accuracy in time and solves for all the time-instances simultaneously. Also, an important property of this method is that it can be achieved in parallel both in space and time. The time-spectral derivation can be implemented in two ways that will be explained in the following sections. But before that it is useful to briefly review a few mathematical ideas.

Mathematical Preliminaries: Periodicity

In general, the function U(t) is said to be periodic with period T if:

$$U(t+T) = U(t) \qquad \forall t \tag{2.15}$$

where T refers to the smallest possible value that satisfies equation (2.15).

Discrete Fourier Transform Implementation

Taking the Fourier transform of a signal $U(t)$ returns information about its spectrum. If $U(t)$ happens to be a discrete signal of length N:

$$U = [U^0, U^1, ..., U^{N-1}] \tag{2.16}$$

and if T is the period of U, the k^{th} component of its discrete Fourier transform (DFT) is given by: [16, 30]

$$\widehat{U}_k = \frac{1}{N} \sum_{n=0}^{N-1} U^n e^{-ikn\Delta t \frac{2\pi}{T}} \tag{2.17}$$

where k is the frequency or wave number, i is the imaginary unit, and $\Delta t = T/N$. The definition of the DFT is slightly different in different sources. Some authors define the DFT including the factor of $\frac{1}{N}$, others include this factor in the definition of inverse of the DFT. It is important to mention that this computation can be conceptualized as a matrix-vector multiplication. Considering U as a vector of size N, the DFT of vector U which is the result of the operations in equation (2.17) also can be found by multiplying it with an N by N matrix whose elements are given by:

$$DFT_{Matrix}(n,k) = \frac{1}{N} e^{\frac{-2\pi ink}{N}} \tag{2.18}$$

and the complete Fourier transform of vector U, which has been represented by $\hat{U}_k$ earlier is:

$$\hat{U} = [\hat{U}_0, \hat{U}_1, \hat{U}_2, ..., \hat{U}_K, \hat{U}_{-K}, ..., \hat{U}_{-1}]^T \tag{2.19}$$

K is the highest wave number that N number of time instances can accommodate, which is called the Nyquist frequency and when N is odd, it is obtained from:

$$K_{Nyquist} = \frac{N-1}{2} \tag{2.20}$$

for odd values of N and

$$K_{Nyquist} = \frac{N}{2} \tag{2.21}$$

for even values of N. The frequency corresponding to each wavenumber is given by:

$$f_k = k\frac{2\pi}{T} \tag{2.22}$$

Therefore, there are only K frequencies, represented by $N = 2K + 1$ time-instances. The original samples in the time domain can be recovered by the inverse discrete Fourier transform (IDFT). For even number of samples the IDFT becomes:

$$U^n = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \widehat{U}_k e^{ikn\Delta t \frac{2\pi}{T}} \tag{2.23}$$

Taking the derivative of U^n with respect to t in equation (2.23), the derivative becomes:

$$\frac{\partial}{\partial t} U^n = \frac{2\pi}{T} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} ik\widehat{U}_k e^{ikn\Delta t \frac{2\pi}{T}} \tag{2.24}$$

The time-derivative formulation is most easily constructed when all the terms in the above equation are written in the time domain. Therefore, equation (2.17) is used to substitute $\widehat{U}_k$, in the above equation. The derivation of time-spectral formulations for odd and even number of samples are slightly different, in this work the steps of the derivation of time-spectral formulation for odd number of time-instances is shown.

Rewriting equation (2.24) for odd number of time-instances and by applying the chain rule, the following is obtained [31]:

$$\frac{\partial}{\partial t}U^n = \frac{2\pi}{T}\sum_{k=-\frac{N-1}{2}}^{\frac{N-1}{2}} ik\widehat{U}_k e^{ikn\Delta t\frac{2\pi}{T}} \tag{2.25}$$

Using the definition of the DFT for $\widehat{U}_k$, in the above equation:

$$\frac{\partial}{\partial t}U^n = \frac{2\pi}{T}\sum_{k=-\frac{N-1}{2}}^{\frac{N-1}{2}} ik(\frac{1}{N}\sum_{j=0}^{N-1}U^j e^{\frac{-2\pi ikj\Delta t}{T}})e^{\frac{2\pi ikn\Delta t}{T}} \tag{2.26}$$

By rearranging the terms, equation (2.26) becomes:

$$\frac{\partial}{\partial t}U^n = \frac{2\pi}{T}\frac{1}{N}\sum_{k=-\frac{N-1}{2}}^{\frac{N-1}{2}}\sum_{j=0}^{N-1} ikU^j e^{\frac{-2\pi ikj\Delta t}{T}}e^{\frac{2\pi ikn\Delta t}{T}} \tag{2.27}$$

Considering:

$$\frac{\Delta tj}{T} = \frac{j}{N} \tag{2.28}$$

and by properties of exponentiation, equation (2.27) becomes:

$$\frac{\partial}{\partial t}U^n = \frac{2\pi}{T}\frac{1}{N}\sum_{k=-\frac{N-1}{2}}^{\frac{N-1}{2}}\sum_{j=0}^{N-1} ikU^j e^{\frac{2\pi ik(n-j)}{N}} \tag{2.29}$$

and finally by interchanging the order of summation:

$$\frac{\partial}{\partial t}U^n = \sum_{j=0}^{N-1}(\frac{2\pi}{T}\frac{1}{N}\sum_{k=-\frac{N-1}{2}}^{\frac{N-1}{2}} ike^{\frac{2\pi ik(n-j)}{N}})U^j \tag{2.30}$$

In other words:

$$\frac{\partial U^n}{\partial t} = \sum_{j=0}^{N-1} d_n^j U^j \tag{2.31}$$

where

$$d_n^j = \frac{2\pi}{T}\frac{1}{N}\sum_{k=-\frac{N-1}{2}}^{\frac{N-1}{2}} ike^{2\pi ik\frac{(n-j)}{N}} \tag{2.32}$$

The coefficients, d_n^j for odd number of time-instances can be represented as:

$$d_n^j = \begin{cases} \frac{2\pi}{T}\frac{1}{2}(-1)^{n-j}csc(\frac{\pi(n-j)}{N}) & \text{if } n \neq j \\ 0 & \text{if } n = j \end{cases} \tag{2.33}$$

Following the same steps for even number of time-instances, the coefficients of time-spectral derivative matrix, for even number of time-instances are obtained as:

$$d_n^j = \begin{cases} \frac{2\pi}{T}\frac{1}{2}(-1)^{n-j}cot(\frac{\pi(n-j)}{N}), & \text{if } n \neq j \\ 0 & \text{if } n = j \end{cases} \tag{2.34}$$

The summation in equation (2.31) describes a matrix-vector multiplication. By substituting this derivative into equation (2.7), the discretized governing equations reduce to a coupled system of N non-linear equations for N different time-instances:

$$\sum_{j=0}^{N-1} d_n^j U^j V^j + R_n(U, \dot{x}, \vec{n}(t)) = 0 \quad n = 0, 1, 2, .., N-1 \tag{2.35}$$

where R_n denotes the spatial residual for the n^{th} time instance. 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 (2.35) in parallel, each time-instance, n, is assigned to an individual processor. Therefore, N processors are needed to solve the entire time-spectral system. It is assumed that there is no parallelism in the spatial domain for this argument, since each processor needs information from all other processors at each non-linear iteration, $O(N^2)$ communication takes place between the N processors.

There are three different ways to implement the spectral matrix vector multiplication to obtain the temporal derivative term. In the first approach, while the coefficients of the spectral matrix is stored in all the processors, the data of each processor is broadcast to all other processors, and then the data is multiplied by its corresponding spectral coefficient, and summed together. The following algorithm describes the implementation of the first approach:

Algorithm 2-1: Broadcast implementation of time-spectral derivative

```
1 Given $U^n$ for $0 \leq n < N$
2 Set $dU^n = 0$
3 Store spectral matrix coefficients, ($D(0:N-1, 0:N-1)$ from equation (2.34) and (2.33))
4 for $n = 0, ..., N-1$ do, (where $N$ is the number of time instances or processors)
5       Broadcast $U^n$ from processor n to all other processors
6       $dU^n = dU^n + D(my-rank, n) \times U^n$
7 end for
```

In the second approach, similar to the first approach, while the coefficients of the spectral matrix is stored in all the processors, first, data in each processor is multiplied to its corresponding spectral coefficient, and then the results are Reduced and summed in all the processors. The computational cost of this approach is the same as the first approach. The implementation of this approach is shown in the following algorithm:

Algorithm 2-2: Reduce implementation of time-spectral derivative

```
1 Given $U^n$ for $0 \leq n < N$
2 Set $dU^n = 0$
3 Store spectral matrix coefficients, ($D(0:N-1, 0:N-1)$ from equation (2.34) and (2.33))
4 for $n = 0, ..., N-1$ do, (where $N$ is the number of time instances or processors)
5       $dU^n = D(n, my-rank) \times U^n$
6       Reduce and sum $dU^n$ from all the processors in the processor n
7 end for
```

In the third implementation, instead of calculating the spectral matrix-vector multiplication, the derivative is calculated by taking the steps that are explained in equations (2.23) and (2.24). In this approach first the direct application of the DFT is used, then the Fourier transform of the data is multiplied by their corresponding coefficients that are introduced in equation (2.24). Later the results are brought back to the time domain using the inverse of the DFT. The cost of this approach is about twice that of the last two approaches. This is due to the fact that while the cost of broadcast and reduce functions are the same, in the third approach the broadcast function is used twice. It is used once in applying the DFT routine and next in applying inverse of the DFT routine. In the next section, it is shown that the steps taken to calculate the FFT-based time-spectral derivative are similar to the third implementation of the spectral matrix-vector multiplication. Thus, in order to be consistent in comparison, the third approach is used in solving the problem using the DFT based time-spectral solver. The algorithm of the third approach is shown here:

```
Algorithm 2-3: DFT-IDFT implementation of time-spectral derivative
1 Given $U^n$ for $0 \leq n < N$
2 Calculate $\widehat{U}_k$ using DFT in which $0 \leq k < N$
3 Calculate $d\widehat{U}_k$ by multiplying $\widehat{U}_k$ by $i\omega k$ in which $\omega = 2\pi/T$
4 Obtain $dU^n$ by transferring $d\widehat{U}_k$ to the time domain using the inverse of DFT
```

Fast Fourier Transform Implementation for Even Number 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 [32, 33]. 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,

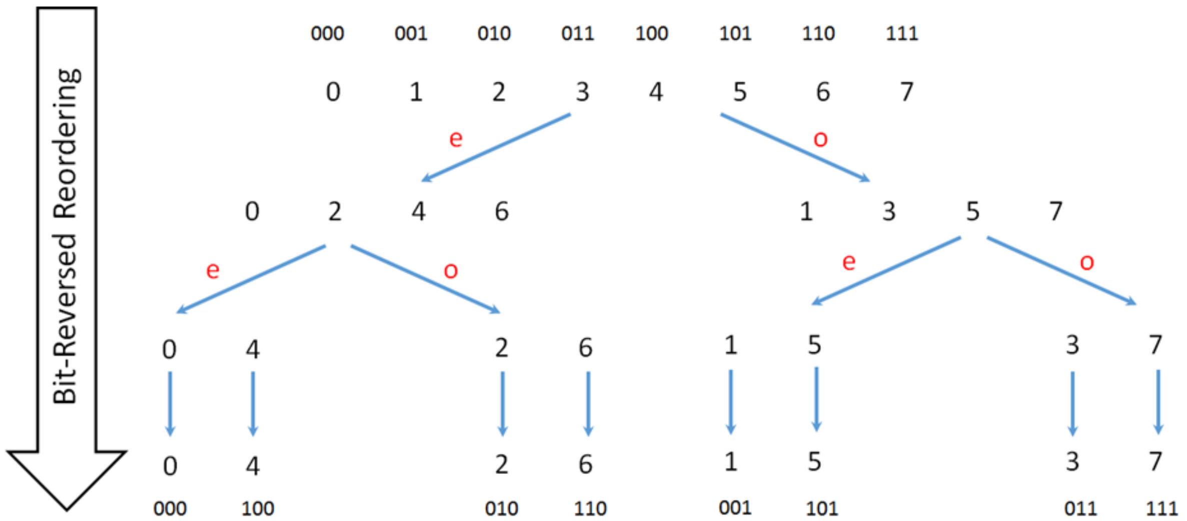


Figure 2.2: Recursive subdivision of $N = 8$ sample set and corresponding bit-reversal ordering

$\log_2 N$ 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:

$$\widehat{U}_k = \frac{1}{N}\sum_{n=0}^{\frac{N}{2}-1} U^{2n} e^{-ik2n\Delta t\frac{2\pi}{T}} + \frac{1}{N}\sum_{n=0}^{\frac{N}{2}-1} U^{2n+1} e^{-ik(2n+1)\Delta t\frac{2\pi}{T}} \tag{2.36}$$

In other words:

$$\widehat{U}_k = \text{even-indexed part} + W^k \text{odd-indexed part} \tag{2.37}$$

where:

$$W = e^{-i\frac{2\pi}{N}} \tag{2.38}$$

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 2.2 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 [33,34]. 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 2.2.

In order to take the derivative of U with respect to time, we make direct use of equations (2.17) and (2.24). The forward FFT is applied to the variable U^n and the obtained $\widehat{U}_k$

are multiplied by their 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 (2.31) 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 $O(N \log_2 N)$. The Cooley-Tukey algorithm is used for the parallel implementation of the FFT [32]. The following algorithm shows this implementation:

Algorithm 2-4: Parallel Fast Fourier transform for even number of samples

```
1 Given $U^n$ for $0 \leq n < N$
2 for $level = 1, ..., NumLevel$ do
3 each processor finds its partner in the current level
4 Send and receive data to and from partner
5 $\widehat{U}_k = U_n + e^{-i\frac{2\pi k}{N}} U_{partner}$
6 $\widehat{U}_{k+\frac{N}{2}} = U_{partner} + e^{-i\frac{2\pi k}{N}} U_n$
7 end for
```

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 [32, 33]. 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 (2.17) and (2.24), 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,

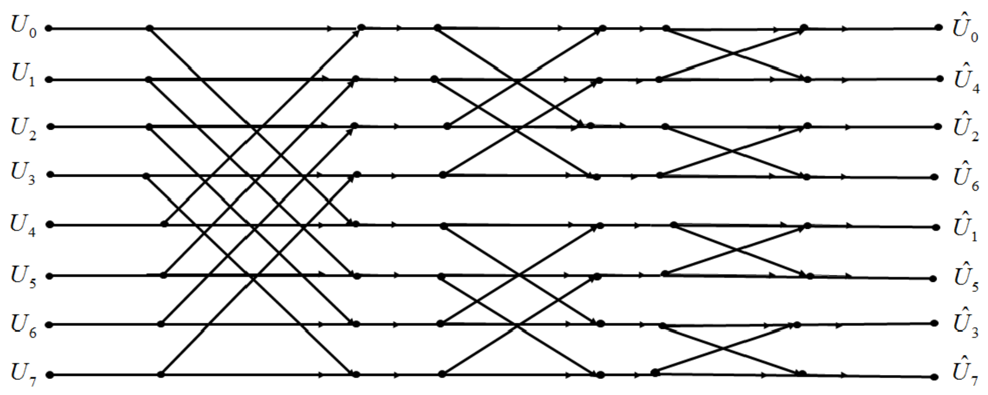


Figure 2.3: Pattern of communication for each level of the parallel FFT algorithm for sample size $N = 8$

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 2.3, for the case of the forward FFT with no data reordering. Application of the forward FFT corresponds to traversing the levels in Figure 2.2 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 Chapter 4.

Odd-Even Decoupling

Considering the matrix form of equations (2.33) and (2.34):

$$D^{even} = \begin{bmatrix} 0 & d_1^{even} & ... & d_{\frac{N-1}{2}}^{even} & 0 & -d_{\frac{N-1}{2}}^{even} & ... & -d_1^{even} \\ -d_1^{even} & 0 & d_1^{even} & ... & d_{\frac{N-1}{2}}^{even} & 0 & ... & -d_2^{even} \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ d_1^{even} & d_2^{even} & ... & 0 & ... & -d_2^{even} & -d_1^{even} & 0 \end{bmatrix} \tag{2.39}$$

$$D^{odd} = \begin{bmatrix} 0 & d_1^{odd} & ... & d_{\frac{N}{2}-1}^{odd} & -d_{\frac{N}{2}-1}^{odd} & ... & -d_1^{odd} \\ -d_1^{odd} & 0 & -d_1^{odd} & d_2^{odd} & ... & ... & -d_2^{odd} \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ d_1^{odd} & d_2^{odd} & ... & ... & -d_2^{odd} & -d_1^{odd} & 0 \end{bmatrix} \tag{2.40}$$

By comparing the even and odd matrices shown in (2.39) and (2.40), it can be seen that matrix D^{even} contains two zeros in each row, while each row of D^{odd} contains only one zero. Matrix D^{even} has two zero eigenvalues with eigenvectors $e_1 = (1, 1, ..., 1)^T$ and $e_2 = (1, 0, 1, 0, ..., 0)^T$. While eigenvector e_1 corresponds to a zero time derivative for a constant solution, eigenvector e_2 results in a discrete zero time-derivative, which results in an odd-even decoupled solution which is not desirable. On the other hand, matrix D^{odd} has only one zero eigenvalue with corresponding eigenvector $e_1 = (1, 1, ..., 1)^T$, meaning that for D^{odd}, a discrete zero time-derivative does not occur as the result of odd-even decoupling.
In problems such as pitching airfoil and wings [30], where the temporal derivative is relatively small, both D^{even} and D^{odd} are stable. However in turbumachinery problems with high RPM, the odd-even decoupling creates instabilities. For these categories of problems only an odd number of time instances can be used [21].

Fast Fourier Transform Implementation for Odd Number of Samples

For the reason explained in section 2.3.2, in some problems time-spectral solutions using even numbers of samples perform suboptimally compared to time-spectral solutions using odd numbers of samples [21,35], 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 base-3 FFT instead of the DFT, in which the number of operations reduces from $O(N^2)$ to $O(2N \log_3 N)$. This implementation also 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 remaining 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:

$$\widehat{U}_k = \frac{1}{N}\sum_{n=0}^{\frac{N}{3}-1} U^{3n} e^{-ik3n\Delta t\frac{2\pi}{T}} + \frac{1}{N}\sum_{n=0}^{\frac{N}{3}-1} U^{3n+1} e^{-ik(3n+1)\Delta t\frac{2\pi}{T}} + \frac{1}{N}\sum_{n=0}^{\frac{N}{3}-1} U^{3n+2} e^{-ik(3n+2)\Delta t\frac{2\pi}{T}} \quad (2.41)$$

The algorithm of parallel fast Fourier transform for numbers of samples that are a power of 3 is given as follows:

Algorithm 2-5: Parallel Fast Fourier transform for odd number of samples

```
1 Given $U^n$ for $0 \leq n < N$
2 for $level = 1, ..., NumLevel$ do
3 each processor finds its partner#1 and partner#2 in the current level
4 Send and receive data to and from partner#1 and partner#2
5 $\widehat{U}_k = U_n + e^{-i\frac{2\pi k}{N}} U_{partner\#1} + e^{-i\frac{4\pi k}{N}} U_{partner\#2}$
6 $\widehat{U}_{k+\frac{N}{3}} = U_{partner\#2} + e^{-i\frac{2\pi k}{N}} U_n + e^{-i\frac{4\pi k}{N}} U_{partner\#1}$
7 $\widehat{U}_{k-\frac{N}{3}} = U_{partner\#1} + e^{-i\frac{2\pi k}{N}} U_{partner\#2} + e^{-i\frac{4\pi k}{N}} U_n$
8 end for
```

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 the 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 section 2.3.2, 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.

Fast Fourier Transform Implementation for Values That are Not a Power of 2 or 3

The parallel FFT can be implemented for N values that are not a power of 2 or 3. The implementations are similar to the Danielson-Lanczos lemma that recursively divides N to smaller problems. In this case, N is divided into smaller groups based on the smallest prime number factors of N. The larger the largest prime factor of N is, the slower the resulting FFT. The worst case is when N is a large prime number, which means no subdivision can happen, and $O(N^2)$ operations are required to calculate the Fourier transform, which is equivalent to DFT implementation [33].

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 using $N/2$ 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 set of $N/2$ complex numbers, as shown in Figure 2.4. This way 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

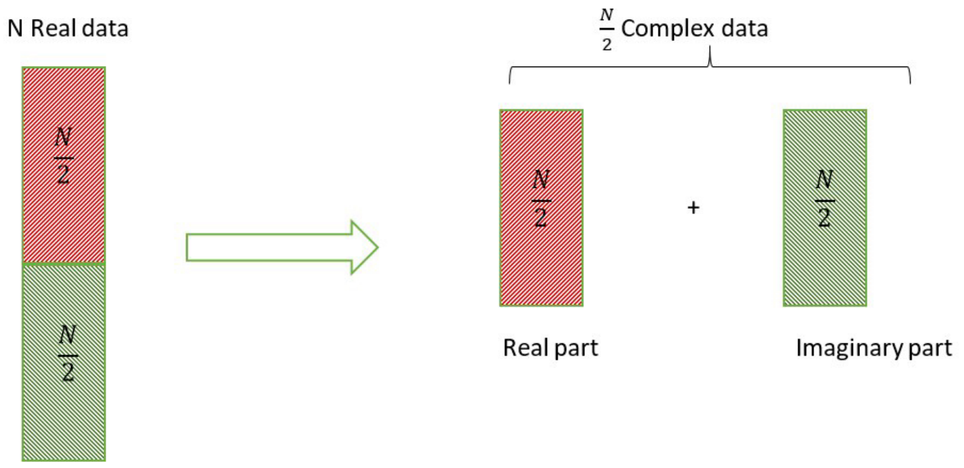


Figure 2.4: Splitting the N real valued samples and creating a $N/2$ complex data

derivative routine are rearranged to match the original set of input data. Theoretically, this splitting can yield a factor of 2 decrease in communication and computational cost.

2.3.3 Time-Spectral 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. Equation (2.42) gives the second-order differentiation of U in time, using the Fourier transform:

$$\frac{\partial^2}{\partial t^2}U^n = -(\frac{2\pi}{T})^2 \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} k^2 \widehat{U}_k e^{ikn\Delta t \frac{2\pi}{T}} \tag{2.42}$$

Algorithm 2 depicts the steps in order to calculate this formulation using the FFT [36]:

```
Algorithm 2-6: Second-Order Derivative
1 Given $U^n$ for $0 \leq n < N$
2 Calculate $\widehat{U}_k$ using FFT in which $0 \leq k < N$
3 Calculate $\widehat{U}''_k$ by multiplying $\widehat{U}_k$ by $-(\omega k)^2$ in which $\omega = 2\pi/T$
4 Obtain $U''^n$ by transferring $\widehat{U}''_k$ to the time domain using the inverse of FFT
```

Parallel implementation in second-order derivative problems is thus analogous to first-order derivative problems with each time-instance assigned to an individual processor. Therefore, the method scales $O(\log_2 N)$, which is a significant improvement compared to the DFT based second-order derivative algorithm which requires $O(N)$ communication. This implementation is required for structural dynamics equations that contain temporal second-order derivative terms.

2.3.4 Hybrid BDF/Time-Spectral Method

The time-spectral method can 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 [28]. 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:

$$U^n(t) = \sum_{k=\frac{-N}{2}}^{k=\frac{N}{2}} \widehat{U}_k e^{ikn\Delta t\frac{2\pi}{T}} + \bar{U}(t) \tag{2.43}$$

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

$$\bar{U}(t) = \Phi_{12}(t)U^{m+1} + \Phi_{11}(t)U^m \tag{2.44}$$

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

$$\bar{U}(t) = \Phi_{23}(t)U^{m+1} + \Phi_{22}(t)U^m + \Phi_{21}(t)U^{m-1} \tag{2.45}$$

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:

$$\Phi_{11}(t) = \frac{t^{m+1} - t}{T} \tag{2.46}$$

$$\Phi_{12}(t) = \frac{t - t^m}{T} \tag{2.47}$$

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:

$$\phi_{21}(t) = 0.5[-\frac{(t - t_m)}{T} + \frac{(t - t_m)^2}{T^2}] \tag{2.48}$$

$$\Phi_{22}(t) = 1 - \frac{(t - t_m)^2}{T^2} \tag{2.49}$$

$$\phi_{23}(t) = 0.5[\frac{(t - t_m)}{T} + \frac{(t - t_m)^2}{T^2}] \tag{2.50}$$

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

$$\widehat{U}_k = \frac{1}{N} \sum_{n=0}^{N-1} \tilde{U}^n e^{-ikn\Delta t \frac{2\pi}{T}} \tag{2.51}$$

with $\tilde{U}^n = U^n - \bar{U}^n$ defined as the remaining periodic component of the function after polynomial subtraction. Differentiating equation (2.43) and making use of equation (2.31) and (2.51) we obtain the following expression for the time derivative:

$$\frac{\partial U^n}{\partial t} = \sum_{j=0}^{N-1} d_n^j \tilde{U}^j + \Phi'_{12}(t_n) U^{m+1} + \Phi'_{11}(t_n) U^m \tag{2.52}$$

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:

$$t_j = t_m + \frac{j}{N}(t_{m+1} - t_m) \tag{2.53}$$

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 equation (2.52) in terms of the original time instances U^n we obtain:

$$\frac{\partial U^n}{\partial t} = \sum_{j=1}^{N-1} d_n^j U^j - (\sum_{j=1}^{N-1} d_n^j \Phi_{12}(t_j) - \Phi_{12}'(t_n))U^{m+1} - (\sum_{j=1}^{N-1} d_n^j \Phi_{11}(t_j) - \Phi_{11}'(t_n))U^m \tag{2.54}$$

Finally, the above expression for the time derivative is substituted into equation (2.35) 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):

$$\begin{aligned} \sum_{j=1}^{N-1} d_n^j V^j U^j - \\ (\sum_{j=1}^{N-1} d_n^j \Phi_{12}(t_j) - \Phi_{12}'(t_n))V^{m+1}U^{m+1} - \\ (\sum_{j=1}^{N-1} d_n^j \Phi_{11}(t_j) - \Phi_{11}'(t_n))V^m U^m + \\ R(U^n, \bar{x}^n, \vec{n}^n) = 0 \end{aligned} \tag{2.55}$$

with $n = 1, 2, .., N$. As previously, N coupled equations are used 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, whereas the $j = N$ or U^{m+1} values are unknown. The reason is that the values at $j = N$ 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 (2.52) 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 (2.35) 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 (2.55) becomes [23]:

$$\begin{aligned} \sum_{j=1}^{N-1} d_n^j V^j U^j - \\ (\sum_{j=1}^{N-1} d_n^j \Phi_{23}(t_j) - \Phi'_{23}(t_n))V^{m+1}U^{m+1} - \\ (\sum_{j=1}^{N-1} d_n^j \Phi_{22}(t_j) - \Phi'_{22}(t_n))V^{m}U^{m} - \\ (\sum_{j=1}^{N-1} d_n^j \Phi_{21}(t_j) - \Phi'_{21}(t_n))V^{m-1}U^{m-1} + \\ R(U^n, \bar{x}^n, \vec{n}^n) = 0 \end{aligned} \tag{2.56}$$

with $n = 1, 2, .., N$. The quasi-periodic derivatives described in equations (2.55) and (2.56) can be rewritten in the following form:

$$[D_{qp}]\vec{U} = [D_{pp}]\vec{U} + [qp]\vec{U} + co\vec{n}st. \tag{2.57}$$

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:

$$[D_{pp}]\vec{U} = \begin{bmatrix} d_0^0 & d_0^1 & . & . & . & d_0^{N-1} \\ d_1^0 & d_1^1 & . & . & . & d_1^{N-1} \\ . & & & & & \\ . & & & & & \\ . & & & & & \\ d_{N-1}^0 & d_{N-1}^1 & . & . & . & d_{N-1}^{N-1} \end{bmatrix}_{N\times N} \begin{Bmatrix} U^N \\ U^1 \\ . \\ . \\ . \\ U^{N-1} \end{Bmatrix} \tag{2.58}$$

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 (2.58) 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 (2.55) and (2.56). This matrix-vector product is given as:

$$[qp]\vec{U} = \begin{bmatrix} \alpha_N - d_0^0 & 0 & . & . & . & 0 \\ \alpha_1 - d_1^0 & 0 & . & . & . & 0 \\ . & & & & & \\ . & & & & & \\ . & & & & & \\ \alpha_{N-1} - d_{N-1}^0 & 0 & . & . & . & 0 \end{bmatrix}_{N\times N} \begin{Bmatrix} U^N \\ U^1 \\ . \\ . \\ . \\ U^{N-1} \end{Bmatrix} \tag{2.59}$$

in which the α_n are obtained for equation (2.55) as:

$$\alpha_n = \sum_{j=1}^{N-1} d_n^j \Phi_{12}(t_j) - \Phi'_{12}(t_n) \tag{2.60}$$

and for equation (2.56) as:

$$\alpha_n = \sum_{j=1}^{N-1} d_n^j \Phi_{23}(t_j) - \Phi'_{23}(t_n) \tag{2.61}$$

Moreover, the constant vector in equation (2.57) is given as:

$$\vec{const.} = (\sum_{j=1}^{N-1} d_n^j \Phi_{11}(t_j) - \Phi'_{11}(t_n))U^m \tag{2.62}$$

for equation (2.55) and in equation (2.56) can be calculated as follows:

$$\vec{const.} = (\sum_{j=1}^{N-1} d_n^j \Phi_{22}(t_j) - \Phi'_{22}(t_n))U^m - (\sum_{j=1}^{N-1} d_n^j \Phi_{21}(t_j) - \Phi'_{21}(t_n))U^{m-1} \tag{2.63}$$

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.

Chapter 3

Solution Methods

This research attempts to seek the best solution strategy in solving periodic and quasi-periodic problems. The solution strategies are studied and compared based on the efficiency and accuracy of the overall solver. For this purpose, this section describes the development of different solvers.

3.1 Explicit and Implicit Schemes

In compressible Euler calculations, the most popular method for solving unsteady problems is adding a pseudo-time stepping scheme to sufficiently reduce the unsteady residual at each physical time step. The pseudo-time stepping is approximated by adding the first-order backward difference formula to the existing Euler equations as:

$$\frac{(VU)^{s+1} - (VU)^{s}}{\Delta\tau} + \frac{\partial(VU)}{\partial t} + R(U(t), x\dot{t}, \tilde{n}(t)) = 0 \tag{3.1}$$

where the first term in the left hand side of equation (3.1) is the pseudo-time derivative that is discretized using BDF1 and $\frac{\partial(VU)}{\partial t}$ is the actual time derivative term that can be obtained using any time derivative scheme based on the nature of the problem. If the Euler equations' spatial residual terms in equation (3.1) are evaluated from the iteration $s+1$, then the scheme is implicit. While the implicit dual-time stepping approach is stable using any size of pseudo-time step, the explicit schemes have limitations in choosing the pseudo-time step, for stability reasons. These limitations results in convergence degradation as the number of time instances or time steps increase or the length of the period of the problem decreases. It has been shown that if BDF2 is used for calculation of the real time derivative term, then the maximum size of the pseudo-time term is restricted by [23, 37, 38]:

$$\Delta\tau_n = CFL \frac{V^n}{\parallel \lambda \parallel + \frac{V^n}{\Delta t}} \tag{3.2}$$

in which index n represents the index of each cell, V is the volume of the cell, $\parallel \lambda \parallel$ is the spectral radius of the largest absolute eigenvalue of spatial discretization matrix, and Δt is the real time-stepping size. If the real time-stepping derivative term is evaluated with either time-spectral or BDFTS discretization, then the size of pseudo-time step is limited by the following:

$$\Delta\tau_n = CFL \frac{V^n}{\parallel \lambda \parallel + V^n k'} \tag{3.3}$$

As equation (3.3) shows, the restriction on the size of the pseudo-time step is determined by the largest wave number k', that is defined as:

$$k' = \begin{cases} \frac{\pi N}{T} & \text{for even N} \\ \frac{\pi(N-1)}{T} & \text{for odd N} \end{cases} \tag{3.4}$$

where T is the length of the period of the problem [21,23]. In order to avoid these restrictions and consequently convergence degradation, the implicit approach can be used as implemented in reference [39] and discussed in the next section.

3.2 Newton-Raphson Method

The non-linear unsteady flow equations are linearized using Newton-Raphson method where the linear system is solved approximately at each Newton step using linear solvers that will be discussed later in this chapter. The set of fully coupled non-linear equations are linearized using the Newton-Raphson method as:

$$[\frac{\partial R_T}{\partial U}]^n \Delta U^{n+1} = -R_T(U(t), x\dot{t}, \tilde{n}(t)) \tag{3.5}$$

In the above equation R_T is the total residual including spatial and temporal residuals, which is a vector with the size of spatial grid for each n and the matrix $[\frac{\partial R_T}{\partial U}]$ is the Jacobian of the entire system, including the Jacobian of the spatial and temporal parts. If the BDF2 scheme is used for discretization of the temporal derivative term, then the complete Jacobian matrix over all the time and space is given as:

$$[\frac{\partial R_T}{\partial U}] = \begin{bmatrix} \frac{V^0}{\Delta\tau^0}I + \frac{3V^0}{2\Delta t}I + J_0 & 0 & ... & 0 \\ 0 & \frac{V^1}{\Delta\tau^1}I + \frac{3V^1}{2\Delta t}I + J_1 & ... & 0 \\ . & & & \\ . & & & \\ . & & & \\ 0 & 0 & ... & \frac{V^{N-1}}{\Delta\tau^{N-1}}I + \frac{3V^{N-1}}{2\Delta t}I + J_{N-1} \end{bmatrix} \tag{3.6}$$

Matrix (3.6) is a diagonal block-matrix, in which each block element of the diagonal corresponds to one time step over all the grids. The $\frac{V^j}{\Delta\tau^j}$ are pseudo-time terms that are discretized using BDF1 scheme, and are added to enhance the diagonal dominance of the Jacobian matrix. Furthermore, matrices J_j correspond to the Jacobian of the spatial part at time step j, and I is the identity matrix with the same size as spatial Jacobian matrix.

In cases that the temporal derivative term is obtained from the time-spectral formulations, the complete Jacobian matrix is not diagonal anymore. In these cases the Jacobian matrix is given as follows:

$$[\frac{\partial R_T}{\partial U}] = \begin{bmatrix} \frac{V^0}{\Delta\tau^0}I + J_0 & V^1 d_0^1 I & ... & V^{N-1} d_0^{N-1} I \\ V^0 d_1^0 I & \frac{V^1}{\Delta\tau^1}I + J_1 & ... & V^{N-1} d_1^{N-1} I \\ . & & & \\ . & & & \\ . & & & \\ V^0 d_{N-1}^0 I & V^1 d_{N-1}^1 I & ... & \frac{V^{N-1}}{\Delta\tau^{N-1}}I + J_{N-1} \end{bmatrix} \tag{3.7}$$

In matrix (3.7), matrices J and I and the term $\frac{V^j}{\Delta\tau^j}$ are the matrix of Jacobian of the spatial part, identity matrix and pseudo-time term. The terms d_i^j are the coefficient of the spectral matrix that have been derived in (2.33) and (2.34). The full Jacobian matrix of the linear BDFTS (BDF1TS) discretization is similar to matrix (3.7). The only difference is in the first column:

$$[\frac{\partial R_T}{\partial U}] = \begin{bmatrix} \frac{V^0}{\Delta\tau^0}I + J_0 + \alpha_N & V^1 d_0^1 I & ... & V^{N-1}d_0^{N-1}I \\ V^0 d_1^0 I + \alpha_1 & \frac{V^1}{\Delta\tau^1}I + J_1 & ... & V^{N-1}d_1^{N-1}I \\ . & & & \\ . & & & \\ . & & & \\ V^0 d_{N-1}^0 I + \alpha_{N-1} & V^1 d_{N-1}^1 I & ... & \frac{V^{N-1}}{\Delta\tau^{N-1}}I + J_{N-1} \end{bmatrix} \tag{3.8}$$

where α_j are obtained from equation (2.60). In all the cases studied in this work and Reference [23], the Jacobian matrices are complete, meaning that the temporal term is included in the structure of the Jacobian as well as the pseudo-time and spatial discretization terms. In all the above full Jacobian matrices, matrix J which is the Jacobian of the spatial discretization, can be calculated with first-order or higher order accuracies. In the first-order accurate implementation of the Jacobian, the value of each cell is influenced by the values of its closest neighbor, as in Figure 3.1, the value of cell "i" is influenced by cells labeled "k". In second-order accurate spatial discretization schemes, the value of each cell is influenced by both the closest neighbors and the neighbors of its neighbors. Using this scheme, the value of the same cell in Figure 3.1 depends on the values of cells named "k" and "q". In this sense, equation (3.5) can be obtained when the right hand side is evaluated using a second-order spatially accurate scheme and the Jacobian is chosen to correspond to a first-order spatially accurate scheme. The right-hand-side can also be written as a combination of a first-order scheme and second-order correction terms [40]. If the full Jacobian is evaluated with second-order accuracy, it may not be practical to store it explicitly. In this case, the calculation of the exact Jacobian-vector product is possible in each linear-solver iteration by storing three sets of diagonal and off-diagonal blocks, as shown in reference [23]. In all the cases studied in this work, both the left and right hand side of the equation (3.5) are

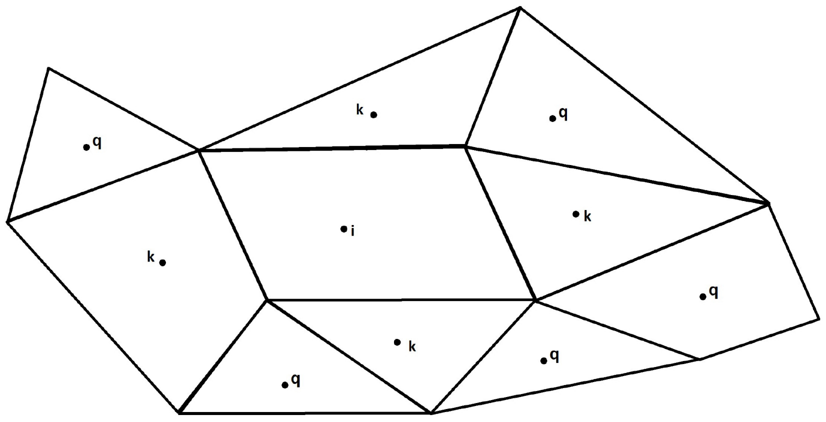


Figure 3.1: Cell "i", its neighbors and its neighbors of neighbors

obtained using first-order spatially accurate schemes.

3.3 Approximate Factorization Scheme

One of the most effective solution technique for solving time-spectral problems with large numbers of time-instances and/or high reduced frequencies is approximate factorization. In this section the implementations of this solver in purely periodic and quasi-periodic problems are demonstrated [16, 25, 28, 39].

3.3.1 AF Algorithm in Purely Periodic Problems

Considering the linearized Euler equations given in equation (3.5) and assuming that the temporal derivative term is calculated using time-spectral method, equation (3.5) yields:

$$[A]\Delta U = -R_T(U, \dot{x}, \vec{n}(t)) = -\sum_{j=0}^{N-1} d_n^j U^j V^j - R(U, \dot{x}, \vec{n}(t)) \qquad (3.9)$$

where R_T is the total residual of the time-spectral space-time system, R is the residual of the spatial part and matrix [A] which is obtained from:

$$[A] = \left[\frac{V}{\Delta\tau_{AF}} + J + V\,[D_{pp}]\right] \tag{3.10}$$

is the complete time-spectral Jacobian matrix. In this equation, $\Delta\tau_{AF}$ 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 [16, 41–43]:

$$[A] \approx [\frac{V}{\Delta\tau_{AF}}I + J][I + \Delta\tau_{AF}D_{pp}] \tag{3.11}$$

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 $\Delta\Delta U$:

$$\Delta\Delta U = [\frac{V}{\Delta\tau_{AF}}I + J]^{-1}(-R(U, \dot{x}, \vec{n}(t))) \tag{3.12}$$

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

$$\Delta U = [I + \Delta\tau_{AF}D_{pp}]^{-1}\Delta\Delta U \tag{3.13}$$

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 transforming the equation to the frequency domain, the spectral matrix becomes diagonal and subsequently the system of coupled equations changes to N decoupled equations. Equation(3.13) is transferred to the frequency domain using the FFT:

$$[FT](I + \Delta\tau_{AF}[D_{pp}])\Delta U = [FT]\Delta\Delta U \tag{3.14}$$

where $[FT]$ denotes the Fourier transform operation. Matrices I and D_{pp} can be written as:

$$[D_{pp}] = [IFT][\tilde{D_{pp}}][FT] \tag{3.15}$$

$$[I] = [IFT][\tilde{I}][FT] \tag{3.16}$$

where $[IFT]$ is the inverse of Fourier transform operation, and both $[\tilde{D_{pp}}]$ and $[\tilde{I}]$ are diagonal matrices. By substituting equation (3.15) and (3.16) into equation (3.14), the following is obtained:

$$([\tilde{I}] + \Delta\tau_{AF}[\tilde{D_{pp}}])[FT]\Delta U = [FT]\Delta\Delta U \tag{3.17}$$

In other words:

$$([\tilde{I}] + \Delta\tau_{AF}[\tilde{D_{pp}}])\Delta\widehat{U}_k = \Delta\Delta\widehat{U}_k \tag{3.18}$$

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

$$\Delta\widehat{U}_k = \frac{1}{1 + ik\omega\Delta\tau_{AF}}(\Delta\Delta\widehat{U}_k) \tag{3.19}$$

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.

The AF inversion is not exact and includes an error which depends on the size of $\Delta\tau_{AF}$. By choosing a small $\Delta\tau_{AF}$, the error can be reduced although the approximately factorized system must be solved iteratively. Also, the sequence of the factorization determines the magnitude of error. Depending on whether the spatial part or the temporal part is solved first, the factorization error is either $\Delta\tau_{AF}[J][D_{pp}]$ or $\Delta\tau_{AF}[D_{pp}][J]$. The magnitude of the error resulting from these two implementations depends on the problem. In this work, by examining both implementations, we realized that it is more efficient if the spatial part is solved first. Therefore, in our implementation, the spatial component is solved first (approximately) followed by the direct solution of the temporal component as discussed above.

3.3.2 AF Algorithm for Quasi-Periodic Problems

Proceeding along the same steps as in purely periodic problems, for quasi-periodic problems equation (3.5) is recovered with some differences as:

$$[A]\Delta U = -Res(U, \dot{x}, \vec{n}(t)) = -[D_{qp}]U^j V^j - R(U, \dot{x}, \vec{n}(t)) \tag{3.20}$$

in which D_{qp} is the quasi-periodic derivative term that is obtained from the temporal terms of equations (2.55) or (2.56). Matrix $[A]$ in equation (3.20) is now given as:

$$[A] = \left[\frac{V}{\Delta\tau_{AF}} + J + V\left[D^*_{qp}\right]\right] \tag{3.21}$$

where

$$[D^*_{qp}] = [D_{pp}] + [qp] \tag{3.22}$$

and the matrix $[qp]$ is given by equations (2.57) and (2.59), noting that the constant vector in the equation (2.57) 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:

$$[A] \approx [\frac{V}{\Delta\tau_{AF}}I + J][I + \Delta\tau_{AF}D^*_{qp}] \tag{3.23}$$

Solving the spatial matrix in order to find the intermediate solution proceeds exactly as in equation (3.12), except that the residual is obtained from the right hand side of equation (3.20). In this work we employ a simple block-Jacobi iterative spatial 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 (3.13) for a quasi-periodic problem becomes:

$$[I + \Delta\tau_{AF}D^*_{qp}]\Delta U = \Delta\Delta U \tag{3.24}$$

or

$$[I + \Delta\tau_{AF}([D_{pp}] + [qp])]\Delta U = \Delta\Delta U \tag{3.25}$$

Equation (3.25) is transferred to the frequency domain in order to benefit from the fact that the spectral matrix is diagonal in the frequency domain:

$$[FT]([D^*_{pp}] + \Delta\tau_{AF}[qp])\Delta U = [FT]\Delta\Delta U \tag{3.26}$$

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

$$[D^*_{pp}] = (I + \Delta\tau_{AF}[D_{pp}]) \tag{3.27}$$

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

$$[D^*_{pp}] = [IFT][\tilde{D^*_{pp}}][FT] \tag{3.28}$$

and

$$[qp] = [IFT][\tilde{qp}][FT] \tag{3.29}$$

where $[IFT]$ represents the inverse Fourier transform matrix, $[\tilde{D^*_{pp}}]$ and $[\tilde{qp}]$ are diagonal matrices corresponding to $[D^*_{pp}]$ and $[qp]$. Therefore, equation (3.26) simplifies:

$$[\tilde{D^*_{qp}}](\Delta\widehat{U}_k) = [[\tilde{D^*_{PP}}] + \Delta\tau_{AF}[\tilde{qp}]](\Delta\widehat{U}_k) = (\Delta\Delta\widehat{U}_k) \tag{3.30}$$

Here we have named the summation of $[\tilde{D^*_{pp}}]$ and $[\tilde{qp}]$, $[\tilde{D^*_{qp}}]$. The next step is taking the inverse of matrix $[\tilde{D^*_{qp}}]$. In this case, since matrix $[\tilde{qp}]$ is a rank-1 matrix, it can be written as:

$$\Delta\tau_{AF}[\tilde{qp}] = \vec{\widehat{u}}_k \vec{\widehat{v}}^T_k \tag{3.31}$$

Here, 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 (3.19). However, since the inverse of the matrix, $[\tilde{D^*_{pp}}]$, is easy to obtain in the frequency domain, following equation (3.19) 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 [44]. The Sherman-Morrison formula gives the inverse of the matrix $[\tilde{D^*_{qp}}]$ as follows:

$$[\tilde{D^*_{qp}}]^{-1} = ([\tilde{D^*_{pp}}] + \vec{\widehat{u}}_k \vec{\widehat{v}}_k^T)^{-1} = [\tilde{D^*_{pp}}]^{-1} - \frac{[\tilde{D^*_{pp}}]^{-1}\vec{\widehat{u}}_k \vec{\widehat{v}}_k^T [\tilde{D^*_{pp}}]^{-1}}{1 + \vec{\widehat{v}}_k^T [\tilde{D^*_{pp}}]^{-1}\vec{\widehat{u}}_k} \tag{3.32}$$

We note that the term in the denominator is a constant scalar that can be precomputed and we name it SC, and one instance of the $[\tilde{D^*_{pp}}]^{-1}$ matrix can be factored out in the overall expression as:

$$[\tilde{D^*_{qp}}]^{-1} = [\tilde{D^*_{pp}}]^{-1}([I] - \frac{\vec{\widehat{u}}_k \vec{\widehat{v}}_k^T [\tilde{D^*_{pp}}]^{-1}}{SC}) \tag{3.33}$$

Thus, $\Delta\widehat{U}_k$ is obtained from:

$$\Delta\widehat{U}_k = [\tilde{D^*_{pp}}]^{-1}([I] - \frac{\vec{\widehat{u}}_k \vec{\widehat{v}}_k^T [\tilde{D^*_{pp}}]^{-1}}{SC})(\Delta\Delta\widehat{U}_k) \tag{3.34}$$

As can be noted, this formula requires two matrix-vector products of $[\tilde{D^*_{pp}}]^{-1}$, 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. In this case, similarly to the AF implementation for purely-periodic problems, the AF inversion error depends on the size of $\Delta\tau_{AF} J D^*_{qp}$, and the sequence of solving the temporal and spatial part.

3.4 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 FFT-based approximate factorization solver used as a preconditioner. The linear system of equations obtained in equation (3.5) is written as:

$$[A]x = b \tag{3.35}$$

where A is the Jacobian obtained by linearizing the full space-time residual, x is the Newton update and b is the negative of the space-time residual. GMRES seeks to minimize the least squares norm of the residual of the linear system that is obtained from:

$$r = [A]x - b \tag{3.36}$$

over the space span $\Big\{r_0, [A]r_0, [A]^2 r_0, .., A^{k-1} r_0\Big\}$. For a right preconditioning system the system is solved as:

$$[A][P]^{-1}y = b \tag{3.37}$$

where

$$x = [P]^{-1}y \tag{3.38}$$

P is the preconditioner in the above equation. The Krylov subspace in this case is the span $\Big\{r_0, [A][P]^{-1}r_0, ([A][P]^{-1})^2 r_0, .., (A[P]^{-1})^{k-1} r_0\Big\}$. Preconditioning consists of solving the following linear system in order to compute the Krylov subspace:

$$[P]z = q \tag{3.39}$$

in which q is a Krylov vector and z is the preconditioned Krylov vector. The flexible GMRES algorithm that allows the use of an iterative method as a preconditioner is used here, as described by Saad and shown here in Algorithm 1 [16, 45].

Algorithm 1: Flexible GMRES (FGMRES)

```
1 Given $Ax = b$
2 Compute $r_0 = b - Ax_0$ , $\beta = \|r_0\|$ and $v_1 = \frac{r_0}{\beta}$
3 for $j = 1, ..., m$ 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        Compute $h_{j+1,j} = \|w\|_2$ and $v^{j+1} = \frac{w}{h_{j+1,j}}$
10   end for
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 $x_0 = x_m$
```

In the given algorithm, A is the time-spectral Jacobian matrix that includes both the temporal and spatial parts, as defined in the following:

$$[A] = \left[\frac{V}{\Delta\tau_{Newton}} + J + V\left[D_{TS}\right]\right] \tag{3.40}$$

A well-known characteristic of Krylov methods is that only Jacobian-vector products are required. In this case, exact Jacobian-vector products are performed for the above space-time Jacobian by first computing the spatial Jacobian-vector product, followed by the temporal Jacobian-vector product, and then adding together these two resulting vectors. For the spatial product, the exact Jacobian-vector product is obtained through exact differentiation of the spatial discretization. A pseudo-time step term, $\Delta\tau_{Newton}$, is included in A. The right hand side of the equation in the first line of the algorithm, b, corresponds to the negative of the residual and x is the non-linear update of ΔU. The FFT-AF scheme is applied in line 4 of the algorithm as the preconditioner. In this case a pseudo-time step must be applied to guarantee the diagonal dominance of the system and to limit the AF factorization error. Also, Given's rotation is used in order to solve the minimization problem in line 13 of the algorithm [45].

Aside from the pseudo-time term in the preconditioner, $\Delta\tau_{AF}$, another pseudo-time step is used in the GMRES algorithm, which is denoted as $\Delta\tau_{Newton}$. Unlike $\Delta\tau_{AF}$, which is the constant pseudo-time step inside the approximate factorization, the FGMRES pseudo-time step, $\Delta\tau_{Newton}$, 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 $\Delta\tau_{Newton}$. A geometric progression is used to control the growth of this CFL value. When the non-linear residual decreases, the CFL is increased using a constant factor of 1.5, otherwise it is held constant. The resulting CFL in the FGMRES algorithm is increased to very large values $O(10^{15})$ so that quadratic convergence of the non-linear problem can be achieved. The second CFL, which is used in the calculation of $\Delta\tau_{AF}$ is constant and always smaller than or equal to the first one. This CFL is used to manage the factorization error and guarantee diagonal dominance in the preconditioner (FFT-AF). The pseudo-time step in the FGMRES algorithm, $\Delta\tau_{Newton}$, 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 Results section [23].

Chapter 4

Purely Periodic Flow Results

4.1 First-Order Derivative Problem Results

4.1.1 Single Frequency Prescribed Motion

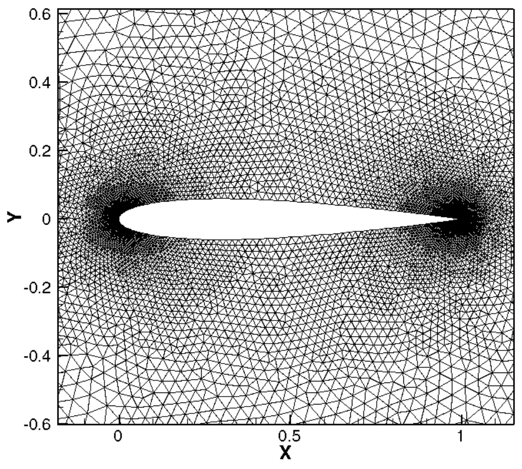
A two-dimensional inviscid pitching airfoil case is studied using NACA0012 airfoil at a Mach number of 0.755 and a mean incidence angle of $\alpha_0 = 0.016°$. The pitching motion is prescribed about the quarter chord of the airfoil with the following formula:

$$\alpha(t) = \alpha_0 + \alpha_A sin(\omega t) \tag{4.1}$$

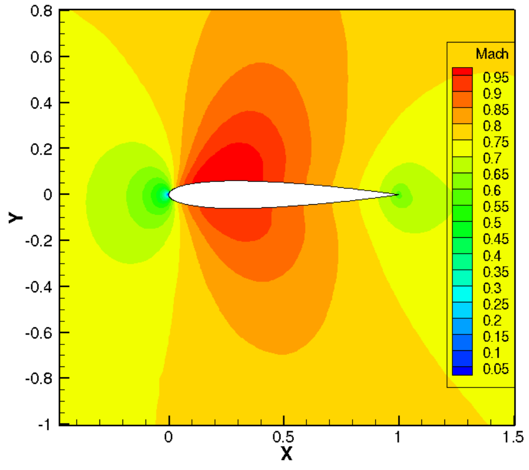
in which ω is the frequency of the pitch motion and is obtained from the reduced frequency, k_c as:

$$\omega = \frac{2U_\infty k_c}{c} \tag{4.2}$$

The reduced frequency for this problem is 0.1628 and the pitching amplitude α_A is equal to 2.51°. This test case corresponds to the AGARD (Advisory Group for Aerospace Research and Development) test case No.5 [46]. The periodic solution is obtained with various numbers of time instances on an unstructured spatial mesh of 15573 triangles, as shown in Figure 4.1(a). The airfoil pitching motion is prescribed as a solid body rotation applied to the entire mesh. The Mach number, computed using the time-spectral solution with 64



(a) Unstructured mesh

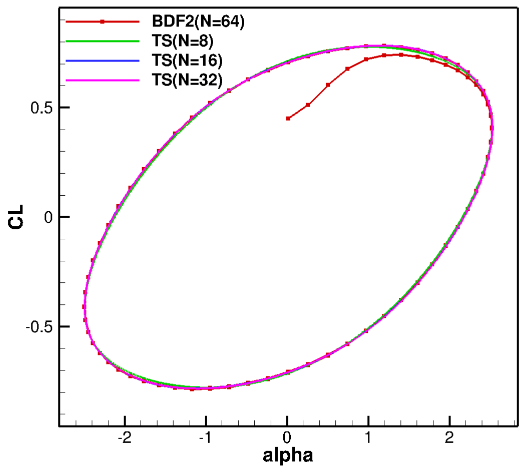


(b) Computed Mach contours

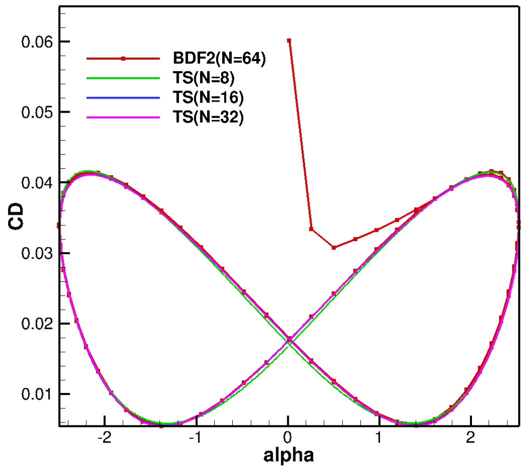
Figure 4.1: Computational mesh and computed Mach contours at $t = 25.7s$, for time-spectral solution of pitching airfoil problem using 64 time instances

time instances at a given location, in time shown in Figure 4.1(b). Figures 4.2 show the comparison of the lift and drag coefficients versus angle of attack between the second-order backward difference time-accurate method using 64 time steps per period, and the time-spectral method with 8, 16, and 32 number of time instances. For this case, the results from the time-spectral method using 16 time instances shows equivalent accuracy, compared to the results obtained from the time accurate method.

In a first set of runs, the performance of the approximate-factorization (AF) scheme used directly as an iterative solver for the time-spectral problem is examined. In this and all subsequent cases (except the mesh study test cases), an iteration of the AF scheme includes 20 block Jacobi sweeps to approximately invert the spatial factor, followed by the direct inversion of the time-spectral factor. Figure 4.3 depicts the convergence rates of the DFT and FFT-based AF implementations for even and odd numbers of time instances. As expected, the DFT and the FFT implementations result in identical convergence rates as measured by the computed residuals at each iteration [26, 47]. However, when measured in terms of wall-clock time, the FFT solver is significantly more efficient than the DFT based solver.

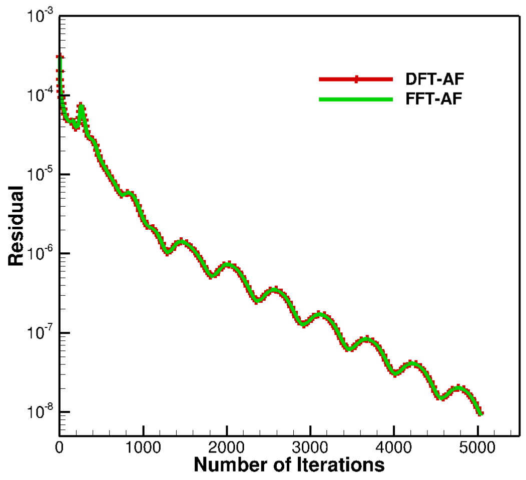


(a) Lift coefficient versus alpha

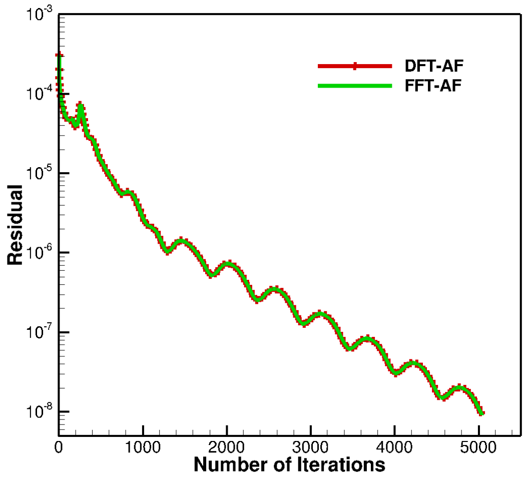


(b) Drag coefficient versus alpha

Figure 4.2: Comparison of time histories of the flow coefficients using the time-spectral method and BDF2

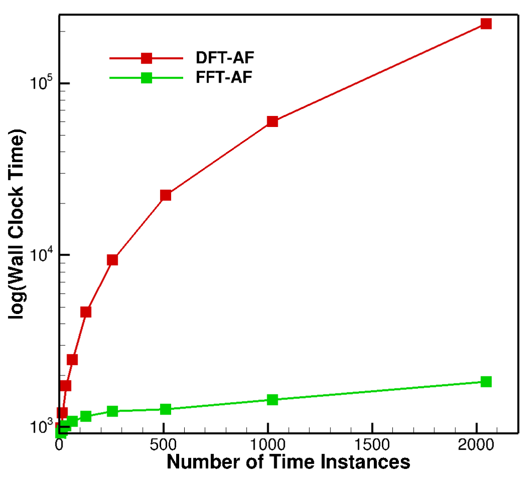


(a) 256 time instances

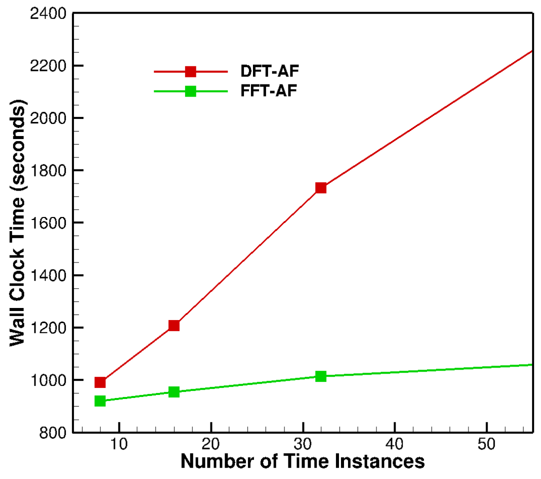


(b) 243 time instances

Figure 4.3: Comparison of residual history versus number of iterations for different numbers of time instances, using the FFT- and DFT-based AF solvers



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

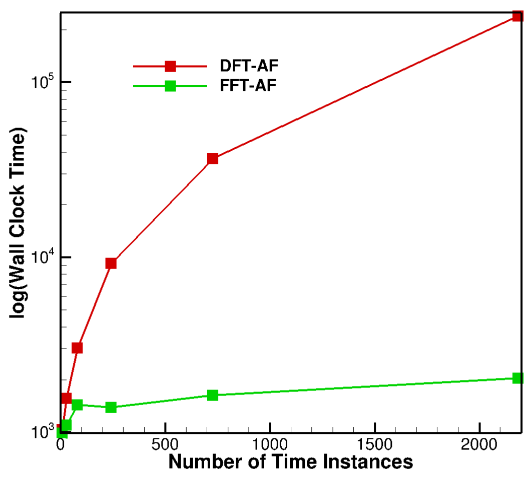


(b) Wall-clock time for small number of samples

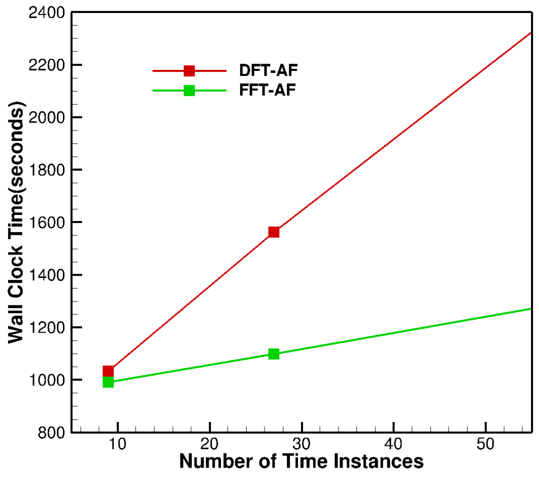
Figure 4.4: Wall-clock time versus number of time instances for even numbers of samples using FFT- and DFT-based AF solvers

This is illustrated in Figures 4.4 and 4.5, where the wall-clock time to achieve a fully converged solution is plotted versus the number of time instances used in the time-spectral discretization for even and odd numbers of time instances, respectively. For even numbers of time instances the code was run for N equal to different powers of 2 up to 2048, and for odd numbers of time instances the code was run for N equal to different powers of 3 up to 2187. These figures demonstrate the significant efficiency gains of the FFT-based method over the DFT-based method, noting that the FFT approach is more efficient even for low values of N, as seen in Figures 4.4(b) and 4.5(b) [26, 47]. Furthermore, as mentioned in the previous chapter, the TS method based on parallel base-2 FFT implementation scales as $O(logN_2)$, while it scales as $O(2logN_3)$ in parallel base-3 implementation of the FFT. Therefore, solving the problem using even number of samples shows slightly more efficiency in terms of running wall-clock time compared to solution of the problem using odd number of samples.

Figure 4.6 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 $O(logN)$ complexity of the



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



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

Figure 4.5: Comparison of wall-clock time versus number of time instances for odd numbers of samples, using the FFT- and DFT-based AF solvers

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.

The performance of the GMRES/AF scheme is examined next. In this case, the full space-time non-linear system is solved using a Newton-Krylov approach, using the previous FFT-based AF scheme as a preconditioner for GMRES. Figure 4.7 compares the non-linear convergence obtained by the GMRES/AF scheme versus that produced by the AF scheme used directly as a solver, for a case using 8 time instances, and for a case using 1024 time instances. In both cases, convergence is plotted as the L2 norm of the space-time residual versus the number of iterations for the AF solver, and versus the cumulative number of Krylov vectors for the GMRES/AF scheme, since in the latter case the cost of a Krylov vector is roughly equivalent to an iteration of the AF scheme. As seen from these results, using the AF scheme as a preconditioner for GMRES results in significantly faster convergence than using the AF scheme directly as a solver, requiring 3 to 4 times fewer iterations or Krylov vectors to reach the final convergence tolerance. In these results, the same CFL value is used for the AF scheme when used either as a solver or as a preconditioner for GMRES. On the other hand, for the GMRES/AF scheme, the CFL in the Newton linearization used

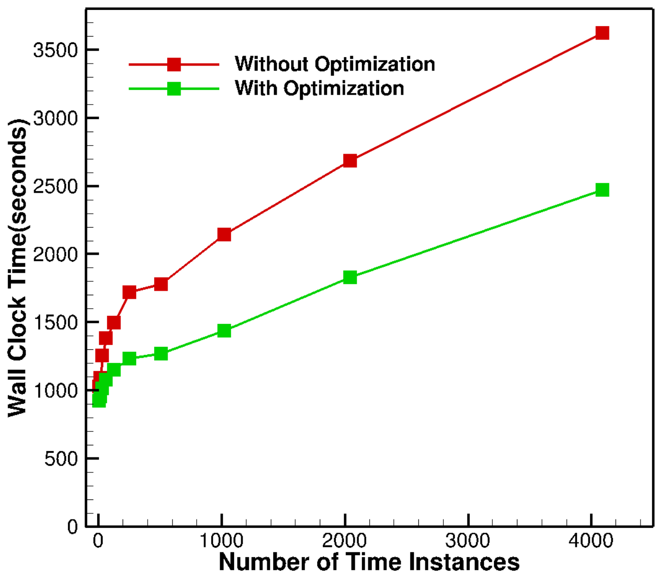
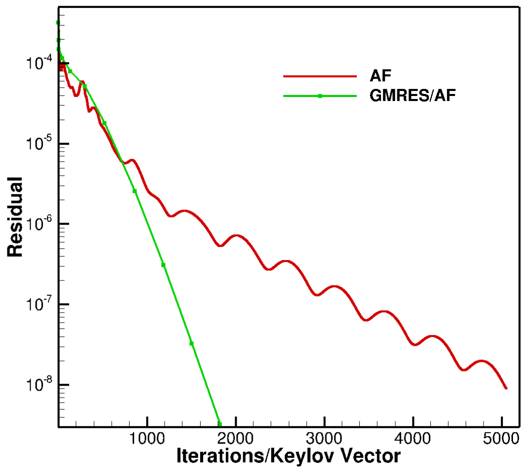


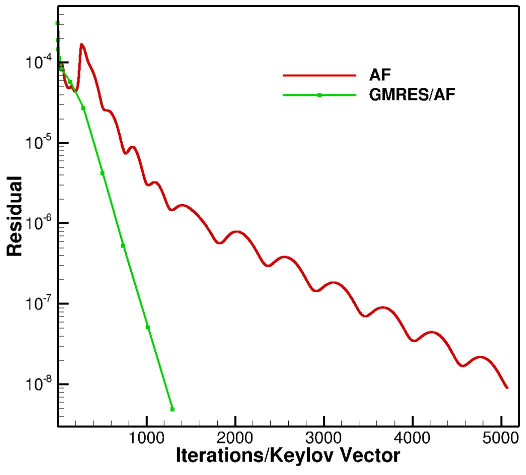
Figure 4.6: Wall-clock time versus number of time instances for original complex FFT and real-data split FFT implementations.

for GMRES is increased at each nonlinear iteration using a simple geometric progression, with the growth factor of 0.01 and reaches a maximum of 10^{15} after roughly 10 non-linear iterations. At each non-linear iteration, multiple Krylov vectors are used in the GMRES algorithm in order to solve the resulting linear system.

An important consideration for the overall efficiency of the solver is the determination of the precision to which the linear system is solved at each non-linear step in the GMRES algorithm. In the previous results, the linear system solution tolerance was set to 0.1, based on the experience and recommendations from previous work [35]. This effect of the linear system solution tolerance is studied in more detail for this case in Figure 4.8, where the same problem using 256 time instances has been solved using tolerances of 0.5, 0.1 and 0.01. In this figure, the L2 norm of the space-time residual is plotted versus the cumulative number of Krylov iterations for all three cases. Figure 4.9, provide more detail for each case by showing the residual convergence in terms of non-linear updates, the number of Krylov vectors at each non-linear update, and the CFL evolution as a function of non-linear updates. For this case,



(a)



(b)

Figure 4.7: Comparison of the non-linear residual versus iterations for the AF solver and versus Krylov vectors for the GMRES/AF solver with 8 number of time instances (a), and 1024 number of time instances (b).

the linear tolerance of 0.5 is the most efficient, achieving full convergence in approximately 1300 Krylov vectors compared to over 2000 Krylov vectors for the case using the lowest linear system tolerance of 0.01, and with the tolerance case of 0.1 falling in between these two cases. As seen from Figure 4.9, the tighter linear tolerance cases result in fewer non-linear iterations overall, but this must be balanced by an increase in the number of Krylov vectors required at each non-linear step in order to satisfy the prescribed linear system solution tolerance. Figure 4.10 compares the required wall-clock time to solve the same problem using different numbers of time instances for the three values of the linear system tolerance. This plot shows that the wall-clock time for linear tolerance settings of 0.5 and 0.01 are similar, where they are both higher than the time required by the tolerance case of 0.1. This admits the above statement that linear tolerance is balanced by number of Krylov vectors. Therefore, the linear system tolerance is set to 0.1 for the remainder of the cases presented in this work, since this represents the best compromise between consistent convergence behavior and efficiency over a wide range of test cases. Figure 4.11 compares the wall-clock time required to converge the pitching airfoil problem for various numbers of time instances using the GMRES/AF scheme and the AF scheme alone. The wall-clock time is plotted versus the log of the number of time instances, and for both schemes, the wall-clock time varies roughly linearly with the log of the number of time instances, at least up to $N = 1024$. This is the expected behavior, since the FFT scheme, which dominates the cost of either solver scales as O(NlogN). A very slight decrease appears in the trend of the wall clock time of GMRES/AF scheme, by increasing the number of time-instances for N equals to 8, 16 and 32. This is due to the fact that the solver requires moderately more number of Krylov-vectors to solve the problem for 8 and 16 number of time-instances compared to $N = 32$ to reach the required accuracy, and consequently results in slightly longer running time. However, the GMRES/AF scheme is seen to be consistently over 2 times more efficient than the AF scheme alone over the entire range of the number of time instances up to $N = 4096$. To further illustrate the efficiency of the current approach, Figure 4.12 plots the required wall-clock time for the solution of the same problem versus the number of time instances for the GMRES/AF solver implemented using the FFT approach and the DFT approach. This latter case corresponds closely to the solution scheme reported in references [16,43], and the current approach shows significant improvement over the DFT approach, particularly for large numbers of time instances where up to two orders of magnitude improvement in the wall-clock time can be obtained.

Figure 4.13 compares the wall-clock time versus the number of time instances for different reduced frequencies using either the FFT-based GMRES/AF solver or the FFT-based AF solver alone. This plot indicates that the performance of both solvers is relatively insensitive to the reduced frequency of the problem while the GMRES/AF solver retains a factor of 2 to 3 speed-up compared to the AF solver.

To further study the behavior of the GMRES/AF solver, the portion of wall-clock time due to the computation and communication phases of the solver are examined. In this case, there is no spatial partitioning, and each time instance runs on a single individual processor or core. Thus the majority of the communication occurs in the FFT routines. These are invoked in the computation of the time-spectral residual evaluation, and also in the time-spectral portion of the Jacobian-vector product that occurs in the GMRES routine. Additionally,

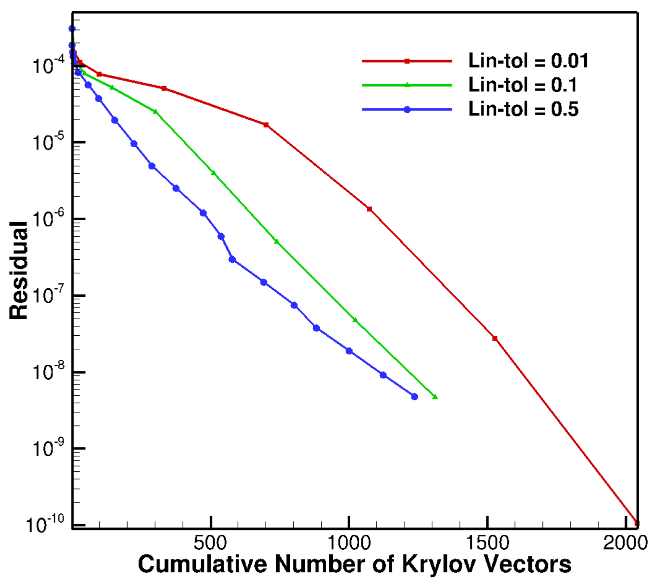


Figure 4.8: Residual versus cumulative Krylov vectors for different linear tolerances for $N = 256$

solution of the approximate factorization problem in the preconditioner involves the use of the FFT routines. Finally, additional communication is required in the GMRES routine for forming inner products over Krylov vectors which span all time instances. Figure 4.14 depicts the total wall-clock time, as well as the time due to communication and computation, versus the logarithm of the number of time instances for the same test case. As the costs are dominated by the FFT algorithm, both communication and computation time are expected to scale as $O(logN)$. This trend is observed approximately, although the costs increase more rapidly for larger numbers of time instances. This is attributed in past to a slight growth in the number of Krylov vectors required for the solution of cases above $N = 512$.

To examine the performance of the solver in more detail, the parallel FFT routine is isolated and the wall-clock time for the communication and computation parts are monitored. Figure 4.15 shows the same plot for the parallel FFT routine alone. This figure shows that the computation component of the FFT scales optimally as $log(N)$, whereas the communication component grows faster than $log(N)$, particularly for higher processor counts. Referring back to Figure 2.3, the communication time required for the first and last level in the parallel FFT routine are compared in Figure 4.16 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 2.3. 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 [48]. 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.

4.1.2 Mesh Resolution Study

While the previous results have been obtained on a mesh of relatively coarse spatial resolution (15,573 cells), in order to examine the effect of grid size on the convergence behavior of the FFT-based GMRES/AF time-spectral solver, an additional finer mesh with over four times the number of cells (85,974 cells) is introduced. Figure 4.17(a) provides a comparison of the convergence histories obtained by the GMRES/AF time-spectral solver on the coarse and fine meshes using 64 time instances in both cases. As can be seen, the finer mesh produces somewhat slower overall convergence. However, in both cases a total of 20 Jacobi iterations were used in the AF preconditioner and it is well known that simple smoothers such as block Jacobi do not scale optimally with grid size. As a numerical experiment, the same problem is recomputed on both meshes using a large number of Jacobi iterations in the AF preconditioner such that the spatial problem within the AF scheme is converged to machine zero for each Krylov vector. These results are depicted in Figure 4.17(b) and show that similar asymptotic rates of convergence are achieved on both the fine grid and coarse grid in this case. These results provide evidence that the spatial scaling of the solver is governed by the spatial solver component of the AF scheme, and that a GMRES/AF time-spectral solver that is both optimal in space and time can be constructed provided an optimal spatial solver (i.e. multigrid) is employed in the AF preconditioner.

4.1.3 Gaussian Bump Test Case

The previous test cases were based on a relatively simple prescribed pitching motion. In this section we examine the performance of the TS solver for a periodic Gaussian bump pitching motion. In this case the prescribed pitching motion as a function of time is given as:

$$\alpha(t) = \frac{1}{\sqrt{20\pi}} e^{-\frac{(t-10)^2}{2}} \tag{4.3}$$

where $\alpha(t)$ represents the angle of attack of the airfoil, and the period is set to $T = 20$, corresponding to a reduced frequency of 0.208. Here the maximum pitch angle occurs at the middle of the period at $t = 10$, and the width of the Gaussian is determined by the denominator in the exponential function. The signal of the pitching motion is shown in the time domain and in the frequency domain in Figure 4.18. This example provides a more stringent test case for the time-spectral solver, since the prescribed motion contains significant spectral content over a range of frequencies and the rapid variation near the middle of the period mimics more realistic important temporal events such as dynamic stall or gust response events, which in turn can be expected to require a larger number of time-instances for achieving good temporal accuracy. Therefore, this test case demonstrates the importance of using larger numbers of time-instances, while provides a more realistic test bed for comparing the efficiency of time-spectral methods versus implicit time-domain methods such as BDF2.

Figure 4.19 depicts the time history of the airfoil lift coefficient obtained using the time-spectral solver for different numbers of time instances. As can be seen from the figure, using $N = 8$ time instances for this case results in significant under-resolution with associated poor accuracy. At $N = 16$, although the peak C_L values are better approximated, lower frequency oscillations are visible in the tails of the time history. However, using $N = 32$ or more time instances results in good accuracy over the entire period for this case and Figure 4.19(b) shows very small differences between the solutions obtained using $N = 64$ and $N = 256$. The parallel FFT-based AF/GMRES solver is employed in all cases, and the convergence as a function of cumulative Krylov vectors is shown in Figure 4.20. In this case, convergence is seen to degrade slightly with larger N values, using the same solver settings as described in the previous section.

In order to compare the performance of time-spectral and implicit time domain (i.e. BDF2) solvers, time-to-solution for both approaches for equivalent accuracy levels is needed to be compared. For this purpose, we use the TS solution with $N = 256$ time-instances as the reference solution, and calculate the error as the RMS difference between the computed TS or BDF2 solutions and this reference solution. For the BDF2 solutions, the integration must be carried out through an appropriate number of periods in order to reduce the effect of the non-periodic start-up transients. Figure 4.21 depicts the calculated error for the BDF2 solutions over the first and fifth periods, using different number of time-steps per period.

As the figure shows, the BDF2 solver requires more than one period to accurately solve the periodic problem as the error over the first period stalls out with increasing numbers of time-steps. However, the solution error obtained solving the problem for 5 periods drops consistently with a slope approximately equal to 2 with increasing numbers of time-steps. Figure 4.22 shows a comparison between the solutions obtained from solving the problem using BDF2 for 5 periods, using different number of time-steps per period with the reference TS $N = 256$ solution.

Figure 4.23 compares the accuracy of the solutions obtained using BDF2 for 5 periods with the solutions obtained by the TS method, using different numbers of time-instances N. As expected, the error convergence of the TS method is much faster than the BDF2 scheme, which is second-order accurate in time, resulting in about 5 orders of magnitude lower error in the solution of the TS method at $N = 128$ compared to the BDF2 method using 128 time steps per period. We note that the TS method does not achieve spectral accuracy since the Gaussian profile is only C_0 continuous at the start and end of each period.

Table 4.1 depicts the wall-clock run time for solving this problem using BDF2 with different numbers of time steps per period for 5 periods and using the TS solver with different numbers of time-instances. The most obvious characteristic of these timing results is the fact that the wall-clock time for the TS method is much lower than that for the BDF2 method in all cases. The wall-clock time for the TS method grows slightly faster than logarithmically with N, while the wall-clock time for the BDF2 method grows slightly slower than linearly with the number of time steps (due to the faster convergence per time-step obtained with smaller time steps). Thus, in terms of wall-clock time, the TS method is clearly superior to the BDF2 method. However, the TS method makes use of larger numbers of parallel hardware cores at higher N values. In terms of total computational resources required, (i.e. core-hours), the TS solver is seen to require more resources when compared with the BDF2 method using the same number of time steps per period as TS time-instances. (This cam be seen by multiplying the TS timings by the value of N for each case). However, the TS method is significantly more accurate when compared on this basis. For example, from Figure 4.23 the TS solution at $N = 32$ is seen to be more accurate than the BDF2 solution using 256 time steps but only requires approximately half the total number of core hours of this BDF2 solution, demonstrating superior overall efficiency for a given accuracy level.

N	wall-clock time of BDF2 for 5 periods	wall-clock time of TS	Core hours for TS
8	2155.04	275.84	2206.72
16	3985.31	533.49	8535.84
32	7866.85	757.30	24233.6
64	14932.49	906.58	58021.1
128	28164.49	978.86	125286.4
256	52678.29	1129.60	289177.6

Table 4.1: Run time for solving the Gaussian bump problem using BDF2 solver for 5 periods, and TS solver for 8 to 256 time-steps per period or time-instances

Finally, the efficiency and accuracy comparisons of TS versus BDF2 schemes must be qualified. In this example, the implicit problems at each time step for the BDF2 scheme were converged to the same level (10 orders of magnitude reduction) as the TS space-time residuals. In general lower convergence tolerances may be used, although a proportional decrease in run time for both schemes can be expected and the overall conclusions should hold. On the other hand, the relative width of the Gaussian signal (as determined by the denominator in the exponent and the overall length of the period) will significantly affect this efficiency comparison. For example, a faster (more narrow) Gaussian within a longer period will result in the need for more time-instances N in the TS method for equivalent accuracy, while this will reduce the number of periods required by the BDF2 scheme to achieve a fully periodic solution. Therefore, a more narrow Gaussian pulse favors the BDF2 scheme. Overall, the ability of the proposed TS solver to handle large numbers of time-instances represents a significant step forward in the competitiveness of time-spectral methods for problems with sharp transients.

4.2 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.

$$y'' - y' - y = sin(\omega t) \tag{4.4}$$

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

$$y'' - y' - y - sin(\omega t) = Res \tag{4.5}$$

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

$$\left[DD_{TS} - D_{TS} - I\right] \Delta y = -Res \tag{4.6}$$

with

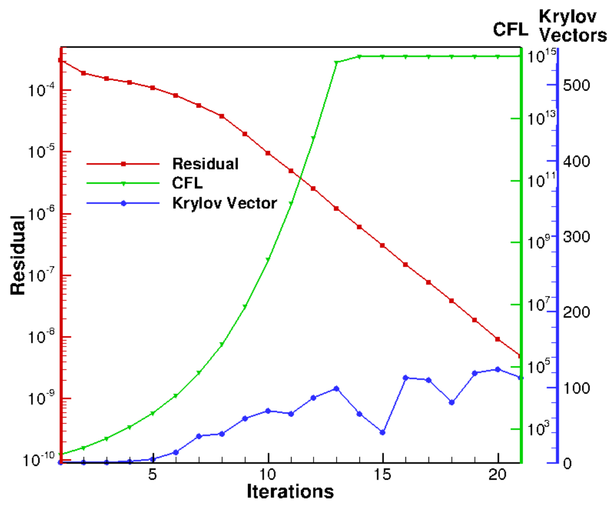
$$y = y_{initial} + \Delta y \tag{4.7}$$

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 (4.6) 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 $\Delta \widehat{y}_k$ and $\widehat{Res}_k$ as the Fourier transforms of Δy and Res, respectively, the solution update in frequency space is computed as:

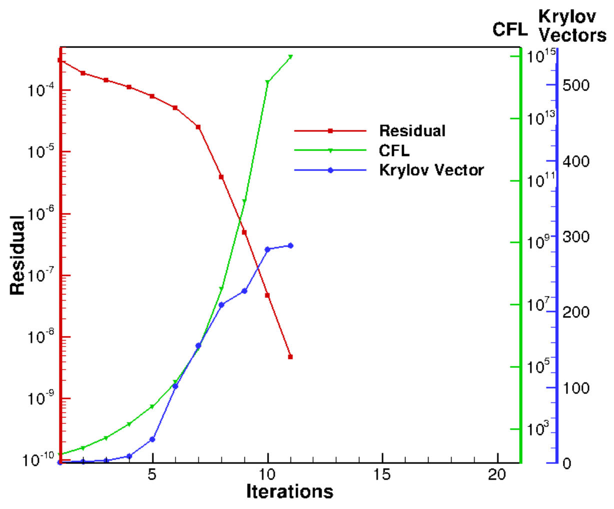
$$\Delta \widehat{y}_k = \frac{\widehat{Res}_k}{-1 - (\omega k)^2 - i\omega k} \tag{4.8}$$

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

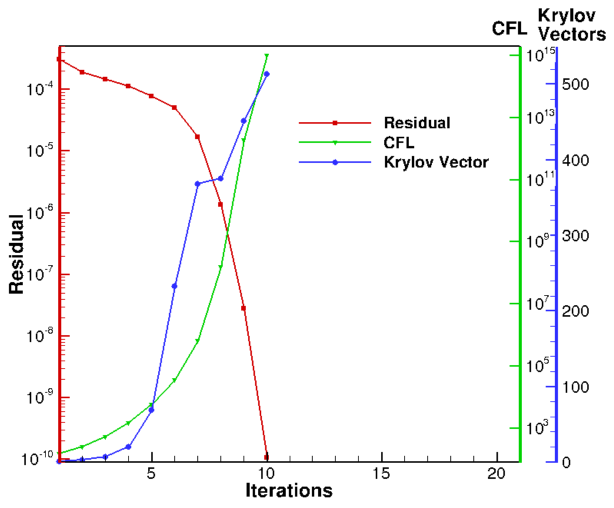
In order to simulate the computation and communication pattern for the solution of a partial differential equation with a corresponding spatial discretization, a total of 20,000 ODEs are solved simultaneously for each time-instance and processor. Figure 4.24 shows the wall-clock time required for the solution of these equations when the temporal derivative terms are implemented based on the FFT and the DFT approaches. The solutions of $y(t)$ obtained from these two implementations are shown in Figure 4.25. These figures illustrate that identical solutions are obtained in both cases, while the FFT-based solver is significantly more efficient than the DFT-based solver, particularly for large values of N, just as in the previous computational fluids dynamics (CFD) results based on a first-order temporal derivative. The temporal second-order derivative term appears in all the aero-structural problems where inertia or mass effects are not negligible. In problems with periodic excitation where the modes of acceleration are the same as displacement modes, the second-order temporal derivative terms can be calculated using the time-spectral implementation. Examples include modal analysis of vibration of buildings in the occurrence of earthquakes, aeroelastic flutter problems, analysis of flow around oscillating blades, etc.



(a)



(b)



(c)

Figure 4.9: Non-linear convergence, CFL history, and number of Krylov vectors in each iteration for linear tolerance of 0.5 (a), 0.1 (b) and 0.01 (c) for 256 number of time instances.

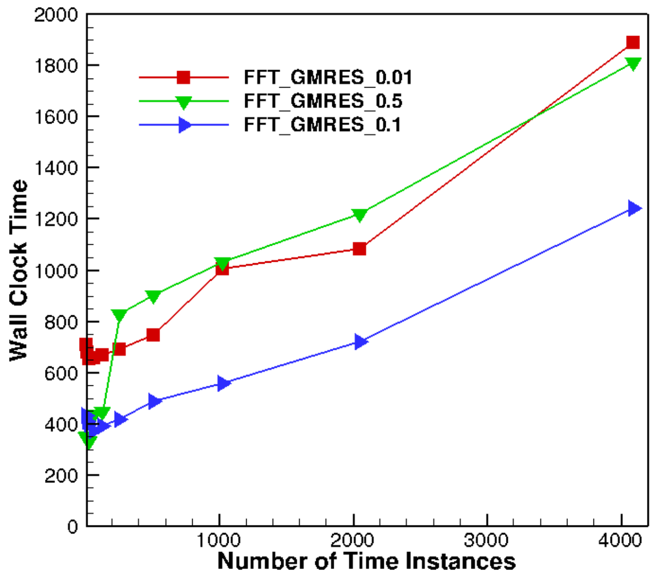


Figure 4.10: Wall-clock time versus number of time instances for different linear tolerances

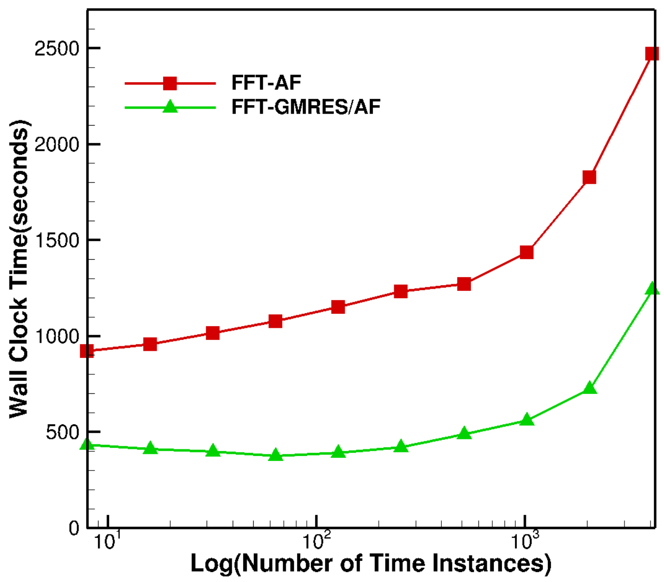


Figure 4.11: Wall-clock time versus number of time instances for FFT-based GMRES/AF and FFT-based AF solver

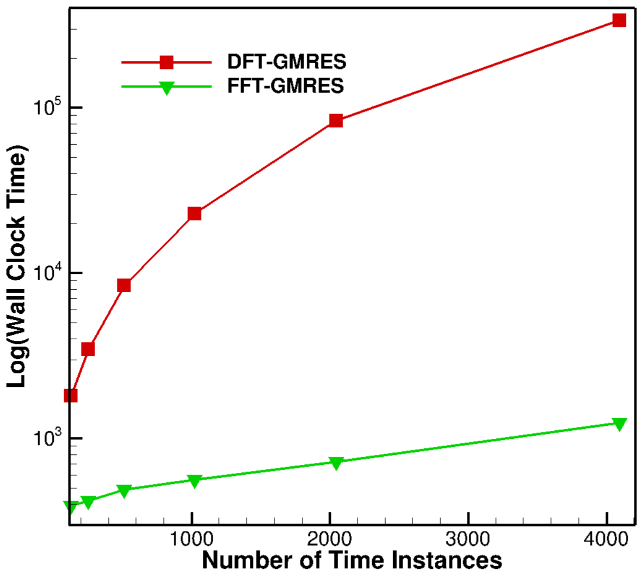


Figure 4.12: Wall-clock time for DFT- and FFT-based GMRES/AF solvers

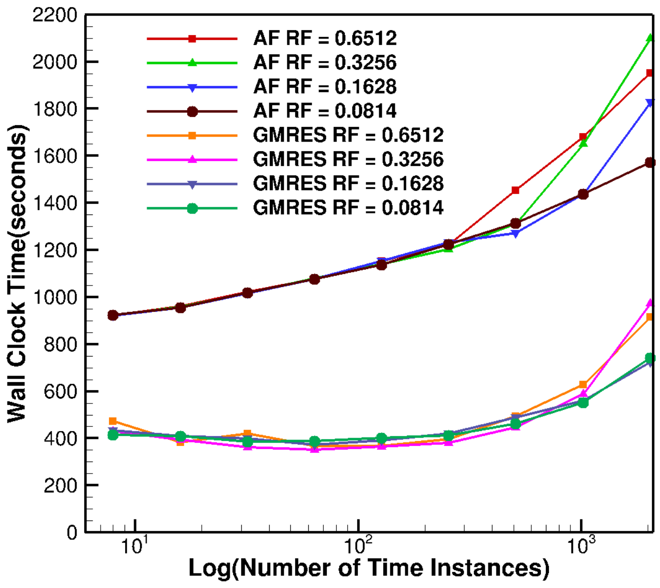


Figure 4.13: Wall-clock time versus log of number of time instances for FFT-based GMRES/AF and FFT-based AF solver for different reduced frequencies

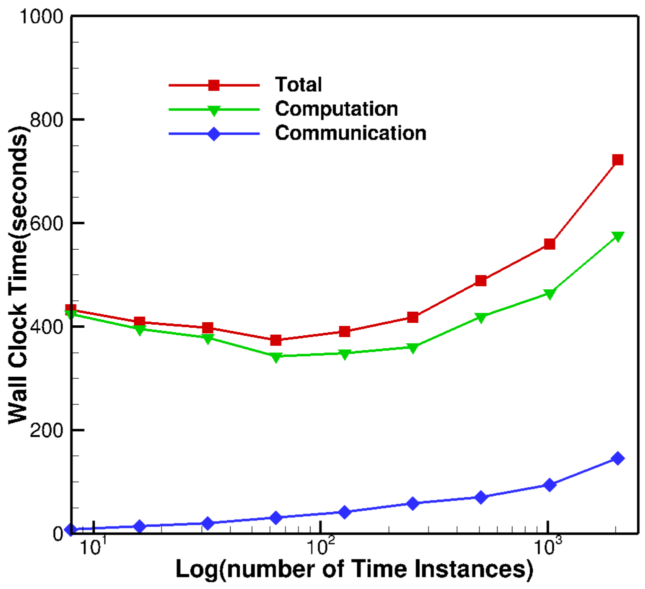


Figure 4.14: Breakdown of wall-clock time for computation and communication of the solver running on NCAR- Wyoming Yellowstone supercomputer using up to 2048 processors

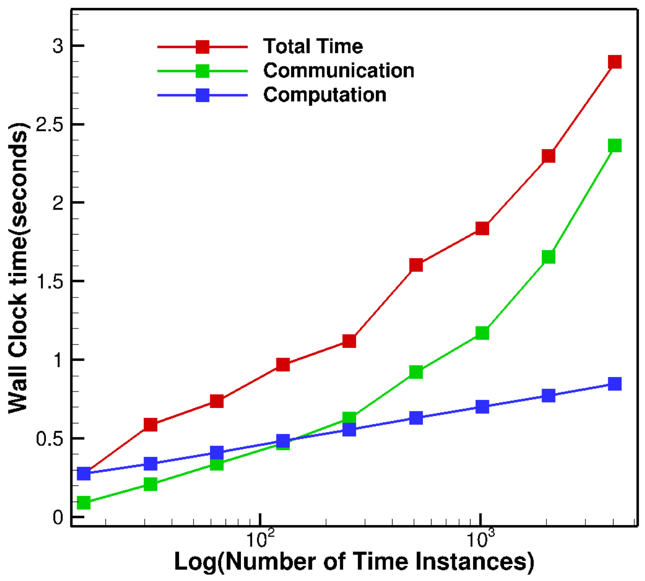


Figure 4.15: Breakdown of wall-clock time for computation and communication of parallel FFT routine running on NCAR- Wyoming Yellowstone supercomputer using up to 4096 processors

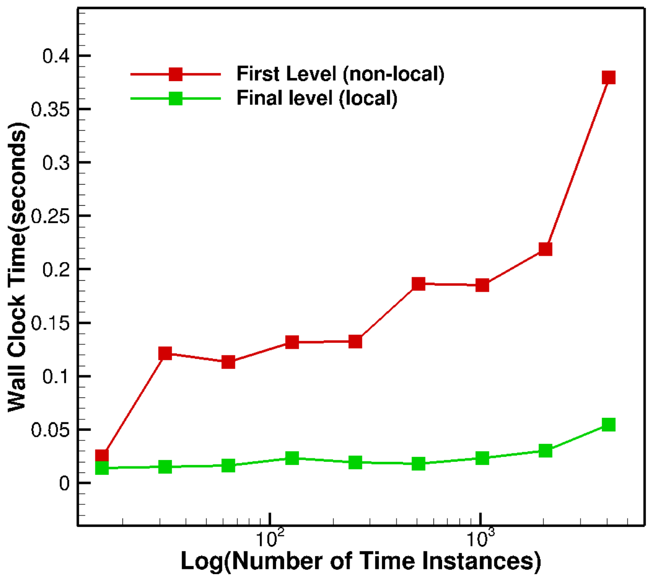
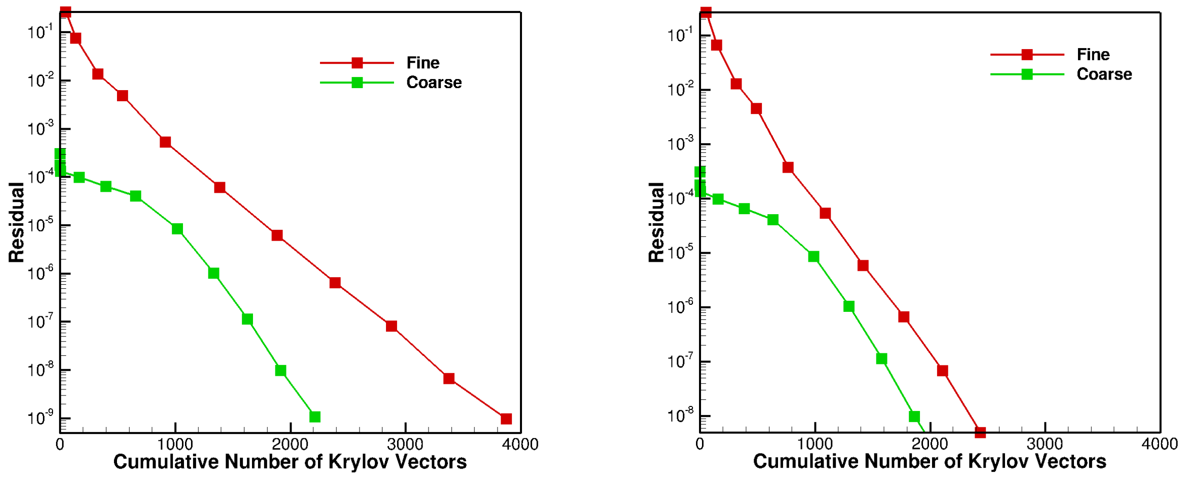


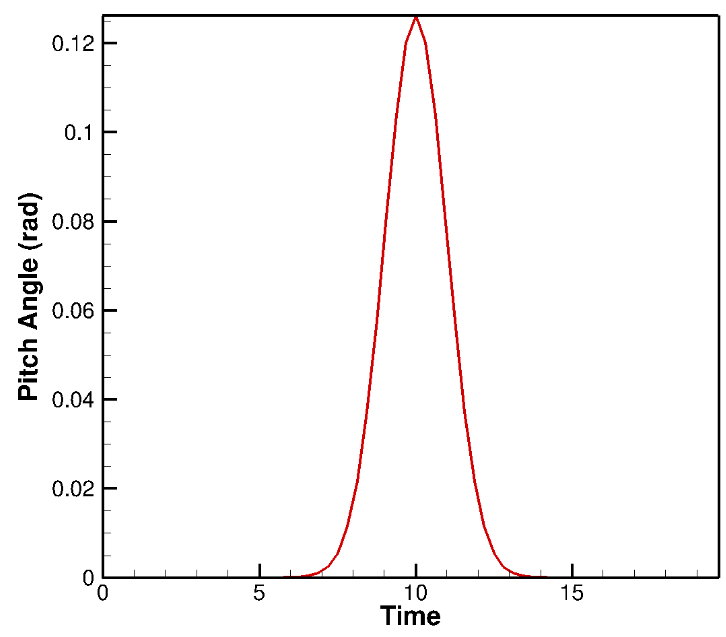
Figure 4.16: Comparison of communication time for first and last level of parallel FFT routine using up to 4096 processors



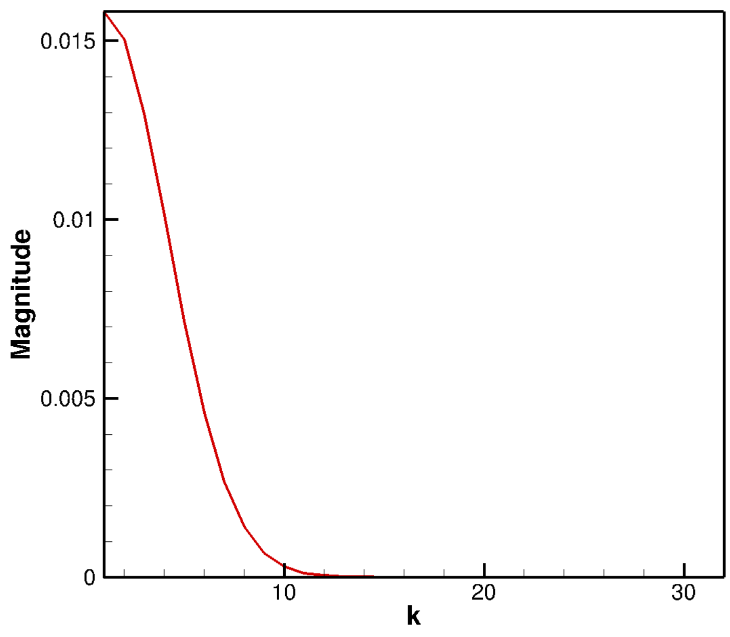
(a)

(b)

Figure 4.17: Non-linear residual versus cumulative Krylov vectors of the fine and coarse grids using 64 number of time instances and linear tolerance of 0.1, with: 20 block-Jacobi sweeps in the preconditioner(a) solving Jacobi to machine zero in the preconditioner(b)

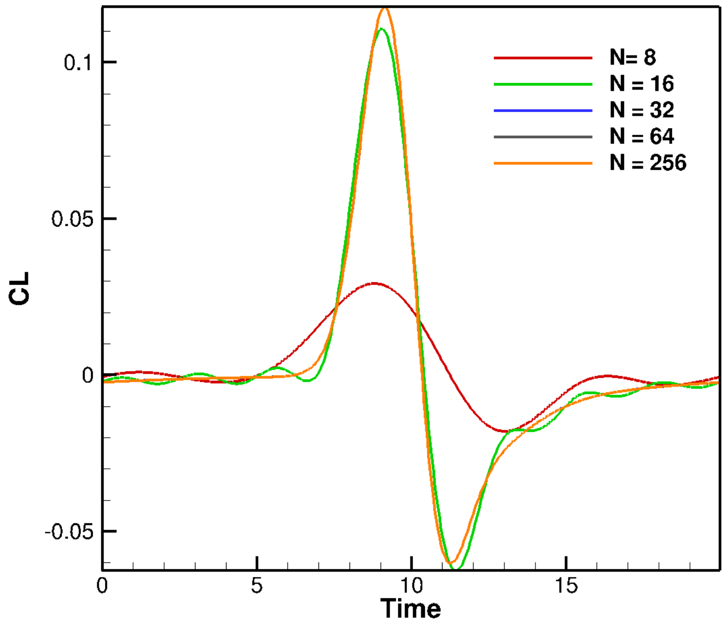


(a)

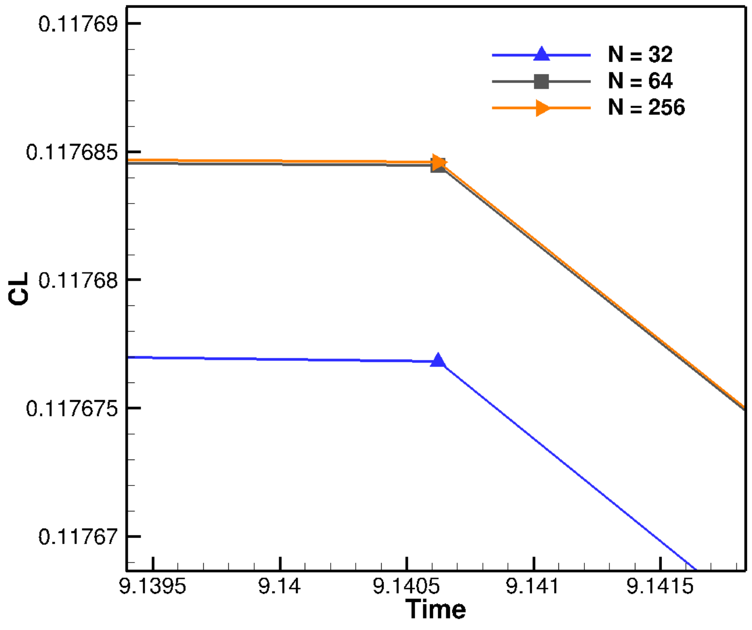


(b)

Figure 4.18: (a) Time history of Gaussian bump prescribed pitching motion and (b) frequency content of prescribed motion signal



(a)



(b)

Figure 4.19: (a) Computed lift coefficient history using TS solver with different number of time instances and (b) details of differences between TS solutions for N = 32, 64 and 256

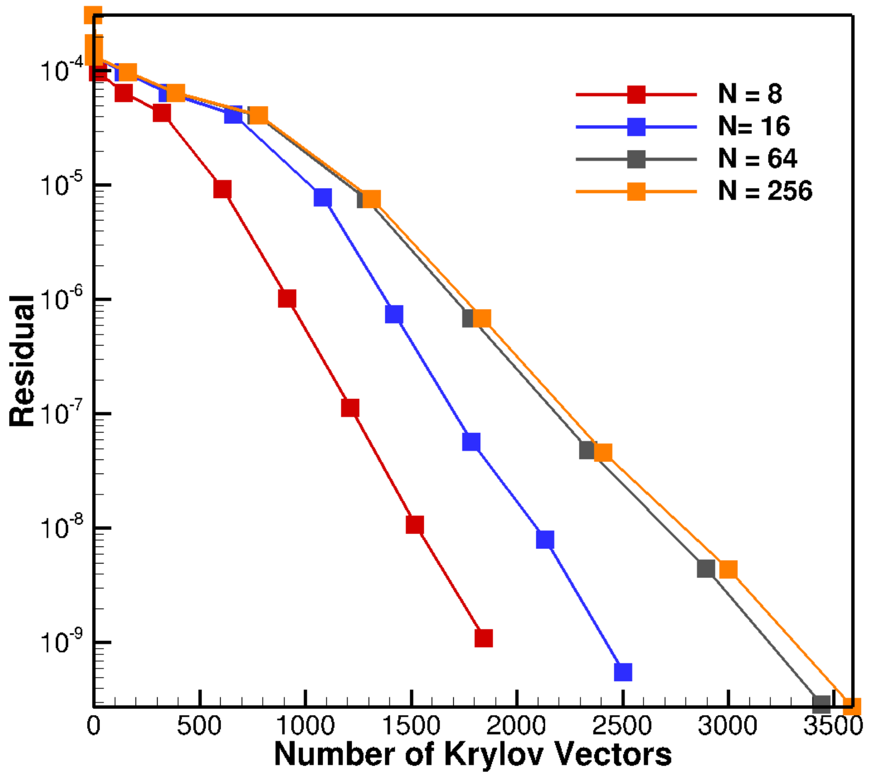


Figure 4.20: Convergence histories for TS solver as measured by residual versus cumulative number of Krylov vectors, using different number of time-instances

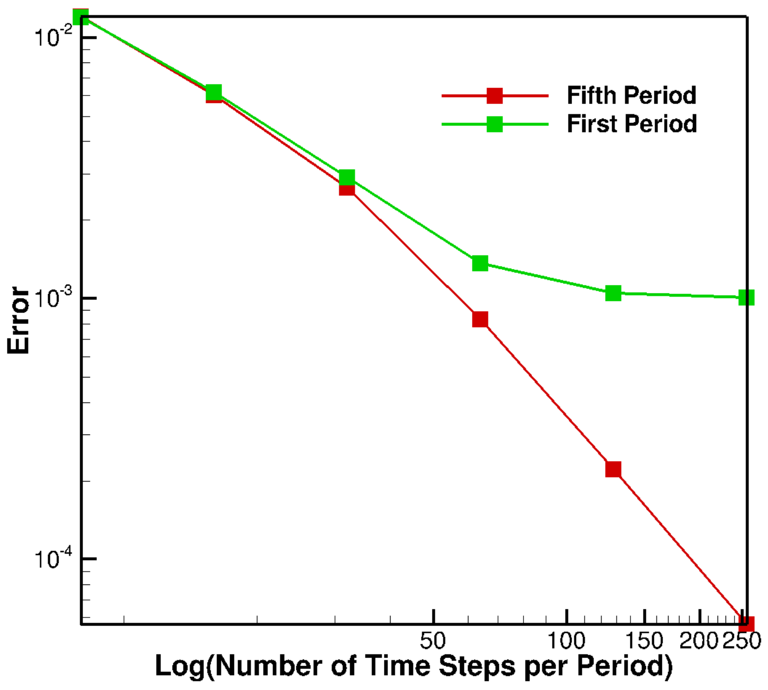
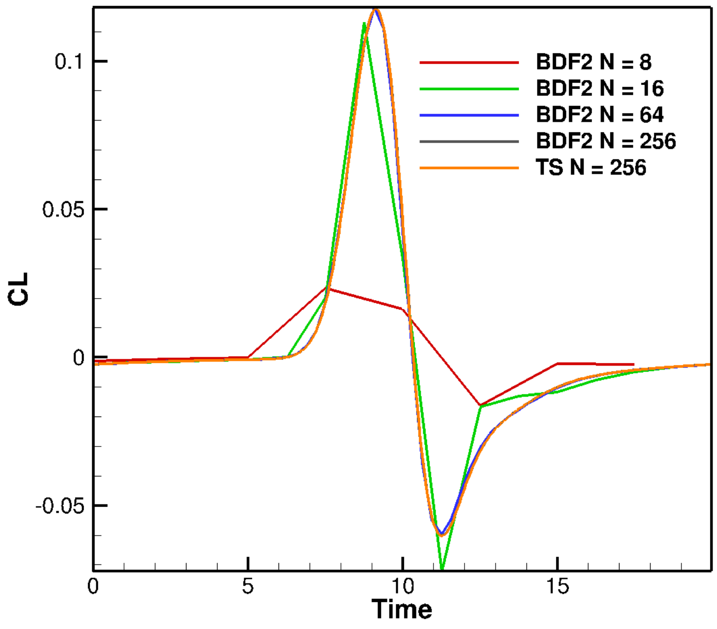
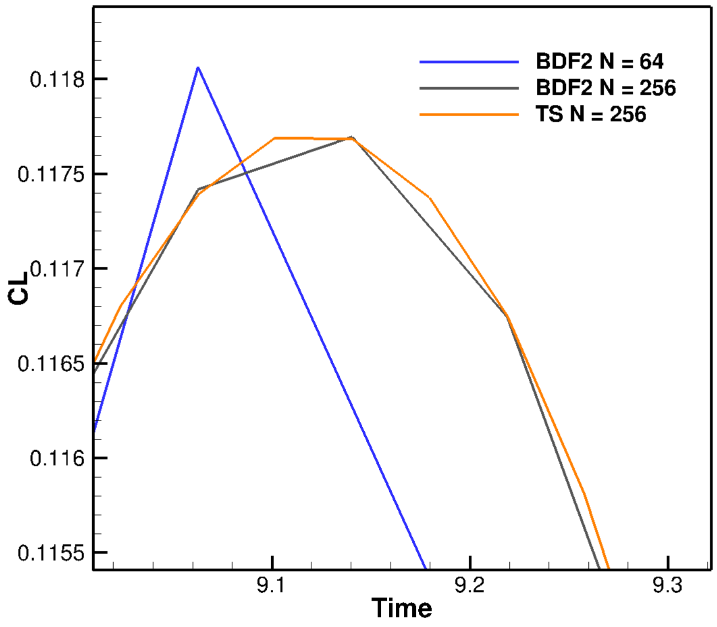


Figure 4.21: Temporal error of BDF2 solution for the first and fifth periods using different number of time-steps



(a)



(b)

Figure 4.22: (a) Computed lift coefficient time histories using the BDF2 scheme over last of 5 periods for different numbers of time steps and (b) detail of time histories near peak CL value.

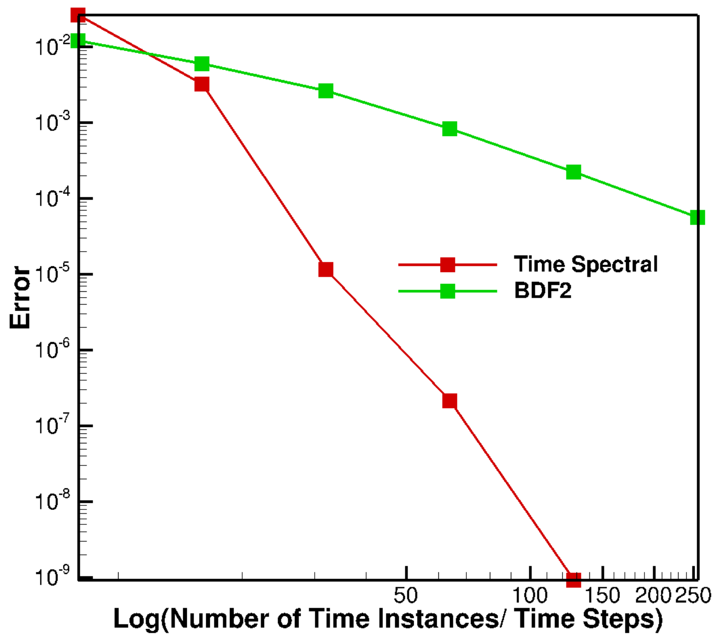
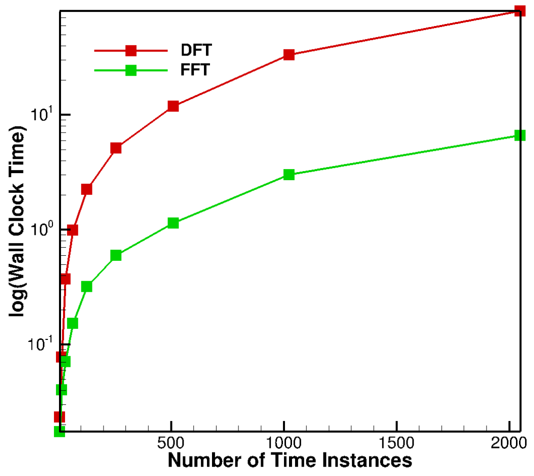
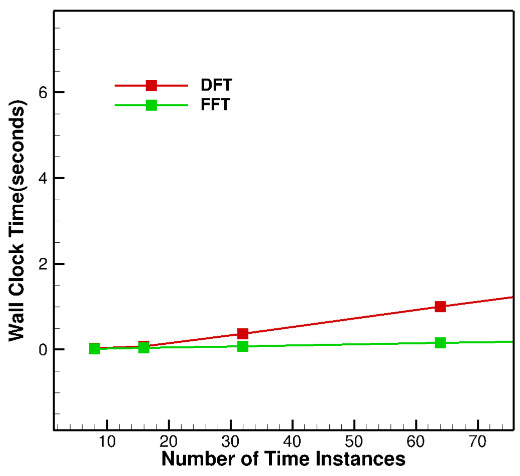


Figure 4.23: Temporal error of TS and BDF2 solutions as a function of the number of time-instances or time steps

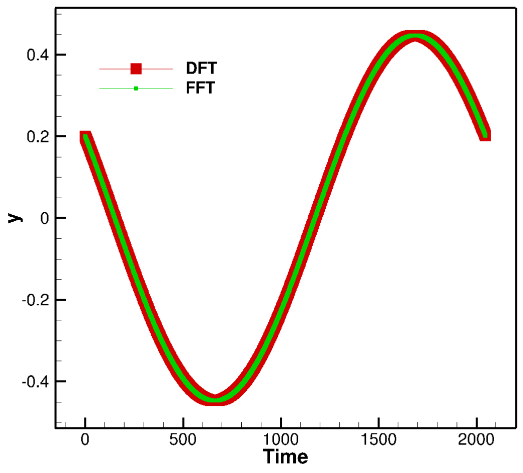


(a) Wall-clock time for up to 2048 number of samples

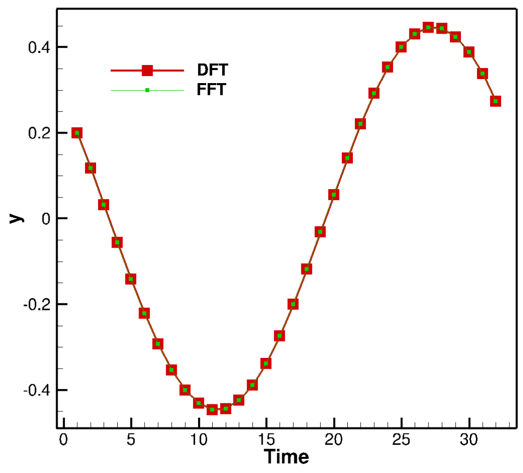


(b) Wall-clock time for small number of samples

Figure 4.24: Comparison of wall-clock time versus number of time-instances using the FFT- and DFT- based solvers for the solution of second-order problem



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



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

Figure 4.25: Comparison of solutions of y versus time for different numbers of time-instances/ processors

Chapter 5

Quasi-Periodic Flow Results

The quasi-periodic 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 4.1(a). The quasi-periodic motion is constructed as a periodic pitching motion 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 given as:

$$\alpha(t) = \alpha_0 + \bar{\alpha}(t) + \alpha_1 sin(\omega_1 t) \tag{5.1}$$

in which the mean angle of attack is:

$$\bar{\alpha}(t) = \begin{cases} 0 & \text{if } t < t_1 \\ \alpha_m \frac{1}{2}(1 - cos(\omega_m(t - t_1))) & \text{if } t \geq t_1 \end{cases} \tag{5.2}$$

where t_1 is the time when the transient motion begins. Here, we assume $t_1 = T$. This condition is set in a way that the quasi-periodic motion begins at the end of the first period. The constants are assumed as:

$$\alpha_0 = 0.016^\circ, \quad \alpha_m = 2^\circ, \quad \alpha_1 = 2.51^\circ, \quad \omega_1 = 0.1628, \quad \omega_m = 0.1\omega_1 \tag{5.3}$$

Figure 5.1 compares the computed lift coefficients of BDFTS solution in the first five periods using different number of time-instances per period. From the figure, it can be seen that the BDF1TS scheme has poor accuracy using only 8 time-instances. However, the accuracy improves for greater number of time instances.

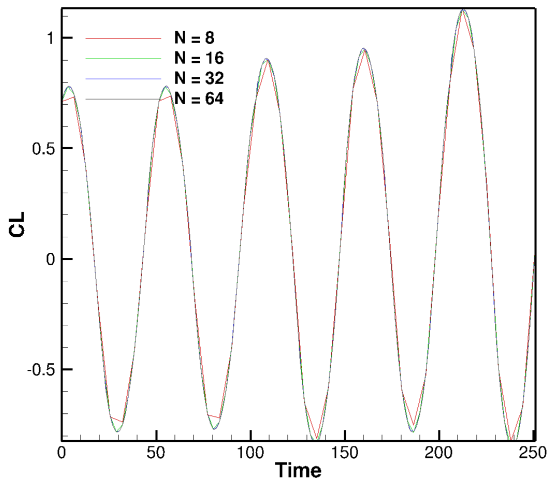


Figure 5.1: Lift coefficient versus time for the first 5 periods using different numbers of time-instance per period, using BDFTS solver

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 5.2 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 represents 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 5.3. 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

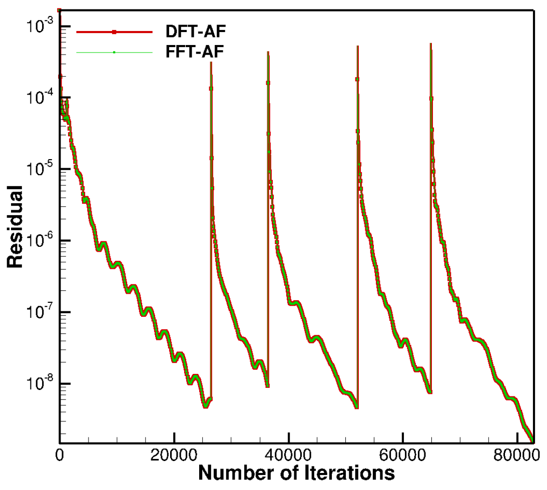


Figure 5.2: Residual versus number of iterations for DFT and FFT-based AF solver using 16 time instances per period for 5 periods of a quasi-periodic airfoil problem

time is entirely due to the solution of the temporal part.

Table 5.1 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 5.1: 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

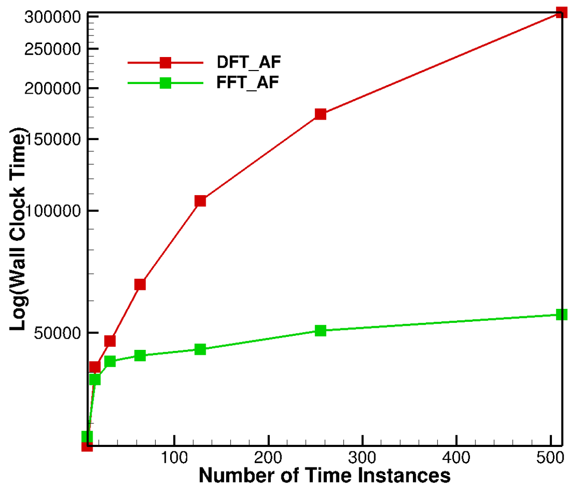


Figure 5.3: Log of total wall-clock time versus number of time instances for FFT and DFT-based AF solution for quasi-periodic airfoil problem

In the previous chapter 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 [47]. 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 for each period, 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 5.4 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.

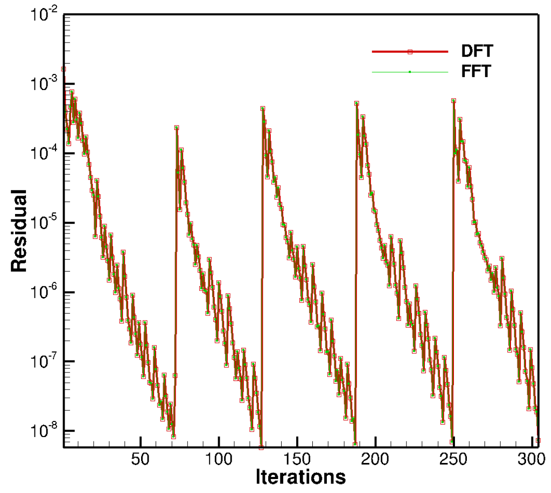


Figure 5.4: Residual versus iterations for DFT and FFT based GMRES/AF solvers using 16 time instances per period

An important consideration for the overall efficiency of the solver is the determination of the precision to which the linear system is solved at each non-linear step in the GMRES algorithm. In the previous chapter 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 [47]. The effect of the linear system solution tolerance is studied in more detail for the quasi-periodic problem. Figure 5.5 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.

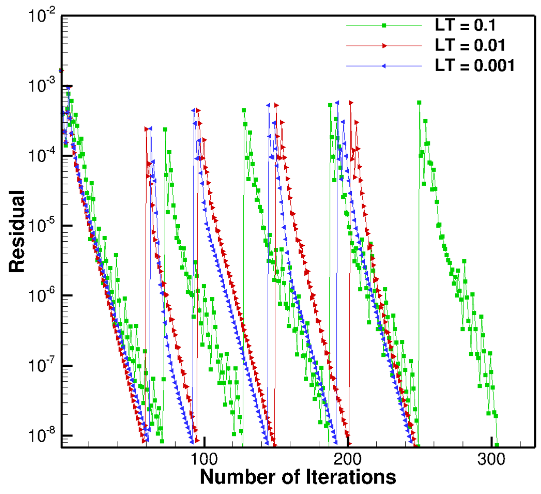


Figure 5.5: Non-linear residual versus number of iterations for different linear tolerance of 0.1, 0.01 and 0.001, using N=16

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 5.6 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 5.7 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 a larger number of time instances, the efficiency gained by the FFT-based approach becomes more remarkable. Figure 5.8 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 comparison indicates that the GMRES/AF solver provides a factor of 2 to 3 speed-up compared to the AF solver.

Table 5.2 compares the convergence rates of the quasi-periodic GMRES/AF scheme for

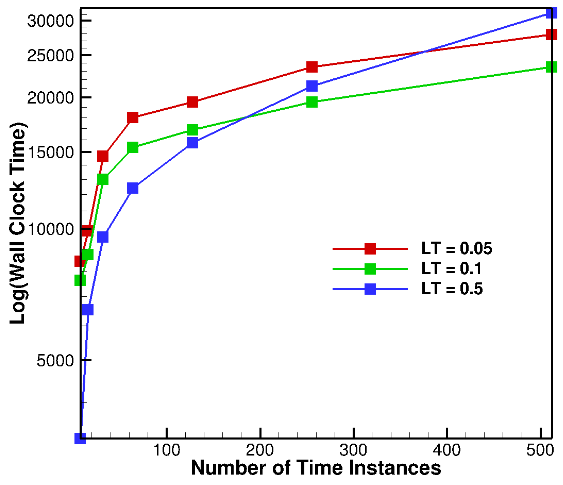


Figure 5.6: Log of wall-clock time versus number of time instances for different linear tolerances of 0.5, 0.1 and 0.05

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 grows 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 5.2: 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

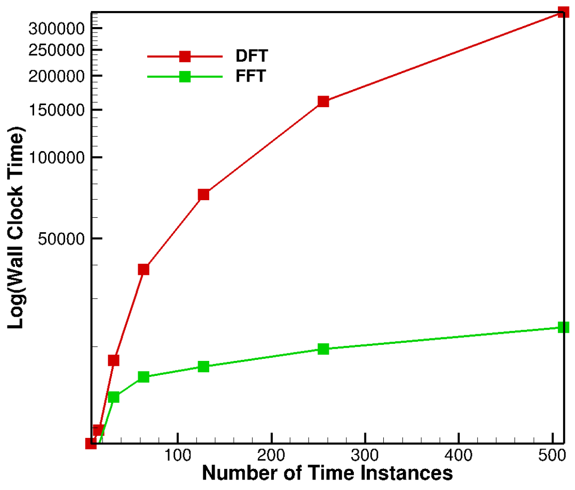


Figure 5.7: Log of wall-clock time versus number of time instances for DFT and FFT based GMRES/AF solvers using linear tolerance of 0.1

the non-periodic content of the problem. For accuracy purposes the problem has also been solved using the BDF2TS formulation. Figure 5.9 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 could only be set to half the size of that used in the BDF1TS runs, the running wall-clock time for cases based on the BDF2TS formulation is longer than cases which are based on BDF1TS.

Figure 5.10 compares the temporal 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 suggests that the solution is dominated by the spectral content of the solver. However, for greater numbers of time instances, the BDF2TS solver has a steeper slope than the BDF1TS, which shows that the accuracy is influenced by the polynomial basis functions. Given the slower residual convergence of the BDF2TS scheme, a precise accuracy study of both schemes over a range of practical problems would

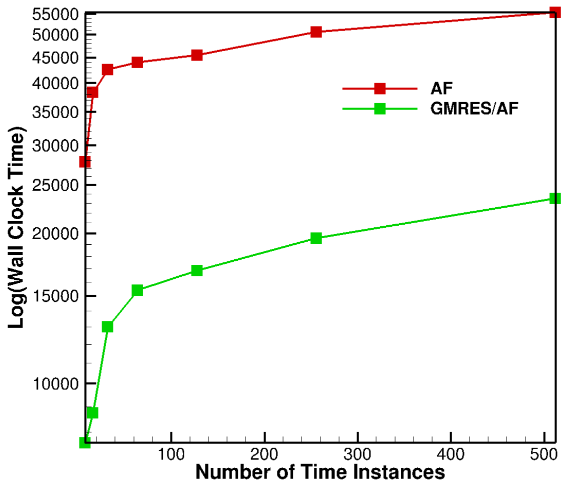


Figure 5.8: Log of total wall-clock time versus number of time instances for FFT-based AF and GMRES/AF solution for quasi-periodic airfoil problem

be required to determine the overall most efficient approach for a given level of accuracy. In this test case, if 5×10^{-5} (half count) is considered as the target error level, the desired accuracy can be obtained using the BDF2TS scheme with $N = 256$ or the BDF1TS scheme using 512 (or more) number of time instances. By comparing the run times of these two methods, we can see that the BDF2TS scheme is the more efficient choice.

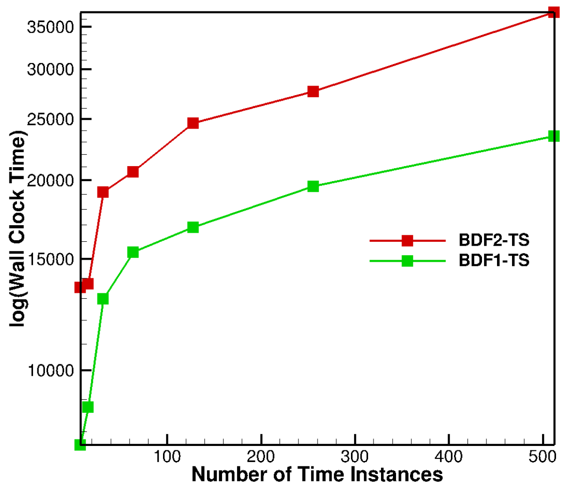


Figure 5.9: Log of wall-clock time versus number of time instances using BDF1TS and BDF2TS

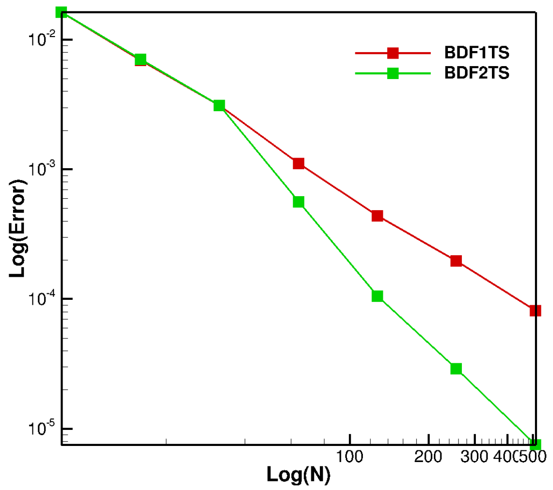


Figure 5.10: Comparison of temporal error in computed lift versus log of number of time instances using BDF1TS and BDF2TS solvers

Chapter 6

Conclusions

6.1 Summary

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 and -quasi-periodic problems using large number of time-instances. Different solution methods have been developed based on this approach and studied for solving the Euler equations in 2D for pitching airfoil problems.

The new parallel time-spectral algorithm that is developed in this work is based on the fast Fourier transform and scales as $O(NlogN)$ for N number of time-instances as opposed to the traditional implementation of this method which was based on the discrete Fourier transform and scales as $O(N^2)$. The new approach results in significant savings compared to previous implementations in terms of wall-clock time. Furthermore, in order to avoid instabilities caused by odd-even decoupling in problems with high reduced frequencies, a base-3 implementation of this approach is developed in addition to the base-2 implementations. The proposed approach is a parallel-in-time scheme in which each time-instance is assigned to one processor, and enables the use of both space and time parallelism.

Following previous work that studied the performance of different solvers in time-spectral problems [23], in the present work, first an approximate factorization algorithm is developed based on the fast Fourier transform. AF separates the spatial and temporal parts of the solution procedure, providing an effective technique for problems with large number of time-instances and/or high reduced frequency. This solver is used to solve the Euler equations for a pitching airfoil problem with single harmonic excitation, which is equivalent to the AGARD No. 5 test case. It is shown that while the FFT-based AF solver has identical convergence rate to the previously implemented AF solver, it is significantly more efficient than the DFT-AF solver in terms of wall-clock time.

In the AF scheme, factorization error is controlled by the size of the pseudo-time step, and therefore large values cannot be used to accelerate the convergence rate. Furthermore, this scheme can not sufficiently overcome the stiffness of the non-linear system in problems with large number of time-instances or short periods. For these reasons, the entire non-linear space-time system of equations resulting from the time-spectral method is linearized and solved incorporating GMRES as the linear solver and the FFT-based AF scheme reformulated as a preconditioner for GMRES. The GMRES/AF scheme is shown to be consistently and significantly more efficient than the AF scheme alone. Additionally it is shown that scaling of the overall solution with spatial resolution is mostly dependent on the spatial component of the AF solver/preconditioner, and optimal scaling should be possible given an optimal spatial solver. The overall solver performance can be more than an order of magnitude more efficient than previous DFT-based implementations which scale as $O(N^2)$, allowing for the effective solution of time-spectral problems using large numbers of time-instances.

To investigate the performance of the proposed solvers in problems with prescribed motion including a wide range of frequency spectrum, the Euler equations are solved for a 2D pitching airfoil test case based on a Gaussian bump pitching function. The performance of the parallel FFT-based GMRES/AF scheme is compared with the performance of a time-accurate scheme in this test case. Previous work has shown that time-spectral methods can solve periodic problems more efficiently than time-implicit method, especially for problems with few harmonics content [16]. By comparing the performance of the FFT-based time-spectral solvers to the BDF2 scheme in solving the aforementioned problem, it can be seen that by improvements made in time-spectral solvers done in this work, these solvers can outperform the time-accurate solvers in problems with high frequency content as well, which is one of the challenging areas for time-spectral method. However, the viability of time-spectral methods over time-implicit methods depends on the frequency content of the problem and remains to be studied in more details.

The new implementation of time-spectral methods for purely periodic problems was extended to quasi-periodic problems as well. A parallel FFT-based approximate factorization scheme was developed for this category of problems. The BDFTS equations correspond to a rank-1 modification of the fully-periodic time-spectral equations, and can be solved effectively by leveraging the FFT-based periodic AF solver using the Sherman-Morrison formula. Similar to purely-periodic test cases, the performance of the parallel FFT-based AF scheme was studied when used directly as a solver and as a preconditioner for a GMRES solver within the context of a Newton-Krylov method for a quasi-periodic problem. The results obtained from this test case are compatible with the results from the purely-periodic test case. Additionally, the BDFTS scheme based on quadratic polynomials is developed and compared to a BDFTS based on linear polynomials in terms of accuracy and performance. It has been shown that although BDF2TS requires somewhat longer wall-clock times for convergence, it provides better accuracy for cases with larger number of time-instances, compared to the BDF1TS scheme for the simple test case used in this work.

6.2 Contributions

• Implementation of time-spectral method based on parallel FFT

The previously developed parallel time-spectral method was based on the DFT and scales as $O(N)$ per processor for N number of time-instances [23]. Time-spectral method has also been implemented based on the FFT in serial in past [25]. This thesis has presented a parallel base-2 and -3 FFT-based implementation of the time-spectral method. This new parallel implementation scales as $O(logN)$ per processor which results in significant efficiency in the running wall-clock time compared to the parallel DFT based TS method. The outstanding improvement in wall-clock time obtained by the new implementation makes time-spectral method a competitive method in problems with high frequency content as well as few harmonic contents. This is a remarkable accomplishment of this work.

- **Implementation of BDFTS method based on parallel FFT**

The BDFTS method for quasi-periodic problems was previously implemented based on DFT. Another novelty of this work is presenting a parallel FFT-based implementation of the BDFTS method. This parallel implementation scales as $O(logN)$ per processor, as opposed to the former DFT-based implementation of this method that scales as $O(N)$ per processor. Furthermore in this research, the BDFTS formulation is shown to be equivalent to a rank-1 modification of the original periodic time-spectral matrix. Using Sherman-Morison formula, the modified matrix is inverted efficiently based on the parallel FFT-based approach for the temporal part of the AF algorithm.

6.3 Future Work

- **Three dimensional parallel in space and time problems**

The present work introduced a new parallel solver for time-spectral problems. The performance of the new approach was tested for 2D problems, and, since the goal was to study the temporal efficiency of the solvers, in all the test cases the spatial component was solved in serial. In addition, the 2D test cases studied in the current work with solution of the spatial part on one core are representative of the size of a spatial portion in a parallel 3D run. Since the 2D test cases were small problems, it was possible to solve the spatial part in serial. However, in real 3D problems, the spatial component should be implemented in parallel as well. 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.

- **Extension to other flow regimes**

Since the goal of current work is to introduce a new algorithm for implementing time-spectral solvers, the solution of the Euler equations are presented in all the test cases. Future work should attempt to use the proposed solvers in other flow regimes. For turbulent flow problems, the spatial part becomes harder to solve, and requires more sophisticated spatial solvers. In all the test cases studied in this work, a block-Jacobi scheme is used to solve the spatial part of AF, both when it is used as the preconditioner in the Krylov context, and when it is directly used as the solver. However, this is not an efficient spatial solver for problems with high complexity. Other elaborate spatial solvers such as multigrid, or other solvers that do not need diagonal dominance can make the AF scheme a much stronger candidate as a preconditioner for GMRES in solving the problems with more complex spatial part, such as turbulence problems. In many realistic problems, most of the solution run time is due to solving the spatial part. For these categories of problems using even large number of time-instances can not show the viability of the proposed implementations since the solution of the spatial part dominates most of the expenses. Furthermore, in many applications it is enough to obtain the harmonics of interest from a time-accurate solution, and therefore large number of time-instances are not needed. For these categories of problems the capabilities of the proposed solvers can not be conveyed. Therefore, applying the proposed solvers is reasonable in cases with expensive temporal solution and/or when large number of time-instances are needed to achieve high accuracy.

- **Studying the viability of BDFTS**

As mentioned, one advantage of the time-spectral method is that it skips the transient part of the solution whereas in time-accurate methods usually the majority of the CPU resources could be spent resolving the transient part of the solution. This advantage does not exist for the BDFTS method, since in most cases the problem needs the same number of periods as required in time-accurate methods to resolve the slow transient content. Therefore, in the BDFTS method, each period must be solved faster than time-accurate methods, in order to outperform them. Therefore, the viability of BDFTS remains to be studied in more details for different applications. In the current work, linear and quadratic BDFTS are implemented and their performance regarding run time and accuracy are compared for solving the Euler equations in a 2D problem. The comparison remains to be done for 3D problems and for other flow regimes as well.

• **Implementing time derivative based on Wavelet Transform**

One disadvantage of time-spectral method is that all the time-instances need to be the same size, which means all the time-instances are solved with the same resolution. In problems with high frequency content in part of the period, it is not efficient to use small time-instance size for all the period. Time-spectral method is based on the Fourier transform, which represents functions by superposing sines and cosines. The Fourier transform of a function tells us precisely the size of the component of frequency, but loses all control on the time domain. In other words, the Fourier transform output is localized in frequency, but has no time localization. The Wavelet transform that have been used in CFD particularly in solving turbulent flow in the recent years can overcome the deficiencies of Fourier transform [49]. The basis functions of wavelet transforms are not limited to sines and cosines, they can be more complicated functions. In wavelet analysis, the scale that we use to look at the data plays a special role. Wavelet algorithms process data at different scales or resolutions. These characteristics enable analysis of signals both with very short basis functions and some very large basis functions at the same time to isolate signal discontinuities and capture detailed frequency analysis [50]. Another big advantage of wavelet transforms over Fourier transforms is that wavelet functions are localized both in the time and in the frequency domain, which means we know about the behavior of a signal locally in time as well as frequency domain. Furthermore, the fast wavelet transform is an optimal transform which requires only $O(N)$ operations to transform an N sample vector, as opposed to the fast Fourier transform that requires $O(NlogN)$ operations. Wavelet transform for discrete-time periodic signals have been derived in past research [51].

Overall, despite it's complexity, it looks like wavelet transforms can deal with some specific problems more efficiently compared to Fourier transforms. However, since the idea of implementing time derivatives based on wavelet transforms is branching out from the basic idea of time-spectral methods, it should be studied in more detail and can be considered as a long term future plan.

References

[1] J. M. Verdon and J. R. Caspar, “A linearized unsteady aerodynamic analysis for transonic cascades,” *Journal of Fluid Mechanics*, vol. 149, pp. 403–429, 1984.

[2] K. C. Hall, W. S. Clark, and C. B. Lorence, “A linearized euler analysis of unsteady transonic flows in turbomachinery,” *ASME Journal of Turbomachinery*, vol. 116, no. 3, pp. 477–488, 1994.

[3] J. J. Adamczyk, “Model equation for simulating flows in multistage turbomachinery,” *ASME Paper*, 1988.

[4] L. He, “I. modelling issues for computation of unsteady turbomachinery flows ii. time-marching calculations of unsteady flows blade row interaction and flutter iii. flow-structure coupled approach. rotating stall/stall flutter calculation,” *Lecture series-van Kareman Institute for fluid dynamics*, vol. 5, pp. C1–C27, 1996.

[5] W. Ning and L. He, “Computation of unsteady flows around oscillating blades using linear and nonlinear harmonic euler methods,” pp. V004T14A039–V004T14A039, 1997.

[6] L. He and W. Ning, “Efficient approach for analysis of unsteady viscous flows in turbomachines,” *AIAA journal*, vol. 36, no. 11, pp. 2005–2012, 1998.

[7] L. He, “Three-dimensional unsteady navierstokes analysis of statorrotor interaction in axial-flow turbines,” *Proceedings of the Institution of mechanical Engineers, Part A: Journal of Power and Energy*, vol. 214, no. 1, pp. 13–22, 2000.

[8] K. C. Hall and E. F. Crawley, “Calculation of unsteady flows in turbomachinery using the linearizedeuler equations,” *AIAA journal*, vol. 27, no. 6, pp. 777–787, 1989.

[9] K. Hall, C. Lorence, and W. Clark, “Nonreflecting boundary conditions for linearized unsteady aerodynamic calculations,” *31st Aerospace Sciences Meeting*, 1993.

[10] K. C. Hall and P. D. Silkowski, “The influence of neighboring blade rows on the unsteady aerodynamic response of cascades,” *Journal of Turbomachinery*, vol. 119, no. 1, pp. 85–93, 1997.

[11] P. D. Silkowski and K. C. Hall, “A coupled mode analysis of unsteady multistage flows in turbomachinery,” *Journal of Turbomachinery*, vol. 120, no. 3, pp. 410–421, 1998.

[12] K. C. Hall and K. Ekici, “Multistage coupling for unsteady flows in turbomachinery,” *AIAA Journal*, vol. 43, no. 3, pp. 624–632, 2005.

[13] M. M. Alonso, J. J. and A. Jameson, “Acceleration of convergence to a periodic steady state in turbomachinery flows,” *AIAA Paper 2001-0152, 39th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January 2001.*

[14] A. J. Jameson, A. and M. McMullen, “Application of a non-linear frequency domain solver to the euler and navierstokes equations,” *AIAA Paper 2002-0120, 40th AIAA Aerospace Science Meeting and exhibit, Reno, NV, January 2002.*

[15] T. J. P. Hall, K. C. and W. S. Clark, “Computation of unsteady nonlinear flows in cascades using a harmonic balance technique,” *AIAA Journal*, vol. 40, no. 5, pp. 879–886, 2002.

[16] N. L. Mundis and D. J. Mavriplis, “Toward an optimal solver for time-spectral solutions on unstructured meshes,” *AIAA Paper 2016-0069 54th AIAA Aerospace Sciences Meeting, San Diego, CA, January 2016.*

[17] S. Choi and A. Datta, “Cfd prediction of rotor loads using time-spectral method and exact fluid-structure interface,” *AIAA paper 2008-7325, 26th AIAA Applied Aerodynamics Conference, Honolulu, Hi, August 2008.*

[18] S. Choi, M. Potsdam, K. Lee, G. Iaccarino, and J. Alonso, “Computation of unsteady nonlinear flows in cascades using a harmonic balance technique,” *Journal of Aircraft*, vol. 51, no. 2, pp. 412–423, 2014.

[19] K.-H. Lee, J. J. Alonso, and E. van der Weide, “Mesh adaptation criteria for unsteady periodic flows using a discrete adjoint time-spectral formulation,” *AIAA Paper 2006-692, 44th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January (2006)*, 2006.

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

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

[22] C. Tomlin and A. Jameson, “Aerodynamics and flight control of flapping wing flight vehicles: A preliminary computational study,” *AIAA paper 2005-0841, 43th AIAA Aerospace Sciences Meeting, Reno, NV, January 2005.*

[23] N. L. Mundis, “The development of a robust, efficient solver for spectral and spectral-element time discretizations,” Ph.D. dissertation, University of Wyoming, 2014.

[24] D. J. Mavriplis and Z. Yang, "Time spectral method for periodic and quasi-periodic unsteady computations on unstructured meshes," *Mathematical Modelling of Natural Phenomena*, vol. 6, no. 3, pp. 213–236, 2011.

[25] J. Leffell, "An overset time-spectral method for relative motion," Ph.D. dissertation, Stanford University, 2014.

[26] D. Ramezanian and D. J. Mavriplis, "An order n log 2 (n) parallel solver for time spectral problems," *AIAA paper 2005-0841, 55th AIAA Aerospace Sciences Meeting, Grapevine, TX, January 2017.*

[27] Z. Yang, D. Mavriplis, and J. Sitaraman, "Prediction of helicopter maneuver loads using BDF-time spectral method on unstructured meshes," *AIAA Paper 2011-1122; 49th AIAA Aerospace Sciences Meeting and Exhibit, Orlando FL, January*, 2011.

[28] D. J. Mavriplis and Z. Yang, "Time spectral method for periodic and quasi-periodic unsteady computations on unstructured meshes," *Mathematical Modeling of Natural Phenomena*, vol. 6, no. 3, pp. 213–236, 2011.

[29] J. C. Butcher, *Numerical methods for ordinary differential equations.* John Wiley & Sons, 2016.

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

[31] K. Naik, "The time-spectral method: A primer," 2011.

[32] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965.

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

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

[35] N. L. Mundis and D. J. Mavriplis, "Toward an optimal solver for time-spectral fluid-dynamic and aeroelastic solutions on unstructured meshes," *Journal of Computational Physics*, vol. 345, pp. 132–161, 2017.

[36] S. G. Johnson, "Notes on FFT-based differentiation," *MIT Applied Mathematics, (2011)*, 2011.

[37] B. Helenbrook, "Artificial compressibility preconditioning for incompressible flows under all conditions," *AIAA Paper 2006-0689, 44th AIAA Aerospace Science Meeting and Exhibit, Reno, NV, January 2006.*

[38] B. T. Helenbrook and G. W. Cowles, "Preconditioning for dual-time-stepping simulations of the shallow water equations including coriolis and bed friction effects," *Journal of Computational Physics*, vol. 227, no. 9, pp. 4425–4440, 2008.

[39] F. Sicot, G. Puigt, and M. Montagnac, "Block-jacobi implicit algorithms for the time spectral method," *AIAA journal*, vol. 46, no. 12, pp. 3080–3089, 2008.

[40] S. Chakravarthy and S. Osher, "High resolution applications of the osher upwind scheme for the euler equations."

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

[42] J. I. Leffell, S. M. Murman, and T. H. Pulliam, "Time-spectral rotorcraft simulations on overset grids," *32nd AIAA Applied Aerodynamics Conference Atlanta, GA, June*, p. 3258, 2014.

[43] J. Leffell, J. Sitaraman, V. Lakshminarayan, and A. Wissink, "Towards efficient parallel-in-time simulation of periodic flows," *AIAA Paper 2016-0066, 54th AIAA Aerospace Science Meeting, San Diego, California, January (2016)*, 2016.

[44] P. Maponi, "The solution of linear systems by using the sherman–morrison formula," *Linear algebra and its applications*, vol. 420, no. 2-3, pp. 276–294, 2007.

[45] Y. Saad, *Iterative Methods for Sparse Linear Systems.* SIAM, 2003.

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

[47] D. Ramezanian, B. Reza Ahrabi, and D. Mavriplis, "An order nlogn parallel newton_krylov solver for time spectral problems," *AIAA paper, 23rd AIAA Computational Fluid Dynamics Conference, Denver, CO, June 2017.*

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

[49] K. Schneider and O. V. Vasilyev, "Wavelet methods in computational fluid dynamics," *Annual review of fluid mechanics*, vol. 42, pp. 473–503, 2010.

[50] A. Graps, "An introduction to wavelets," *IEEE computational science and engineering*, vol. 2, no. 2, pp. 50–61, 1995.

[51] J. A. Gubner and W.-B. Chang, "Wavelet transforms for discrete-time periodic signals," *Signal Processing*, vol. 42, no. 2, pp. 167–180, 1995.