


2017

Non-Equispaced Fast Fourier Transforms in Turbulence Simulation

Aditya M. Kulkarni

University of Massachusetts Amherst

Follow this and additional works at: https://scholarworks.umass.edu/masters_theses_2

 Part of the [Numerical Analysis and Computation Commons](#), and the [Other Mechanical Engineering Commons](#)

Recommended Citation

Kulkarni, Aditya M., "Non-Equispaced Fast Fourier Transforms in Turbulence Simulation" (2017). *Masters Theses*. 579.
https://scholarworks.umass.edu/masters_theses_2/579

This Open Access Thesis is brought to you for free and open access by the Dissertations and Theses at ScholarWorks@UMass Amherst. It has been accepted for inclusion in Masters Theses by an authorized administrator of ScholarWorks@UMass Amherst. For more information, please contact scholarworks@library.umass.edu.

NON-EQUISPACED FAST FOURIER TRANSFORMS IN TURBULENCE SIMULATION

A Thesis Presented
by
ADITYA MOHAN KULKARNI

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

September 2017

Mechanical and Industrial Engineering

©Copyright by Aditya M. Kulkarni 2017
All Rights Reserved

NON-EQUISPACED FAST FOURIER TRANSFORMS IN TURBULENCE SIMULATION

A Thesis Presented
By
ADITYA MOHAN KULKARNI

Approved as to style and content by:

Stephen de Bruyn Kops, Chair

Blair Perot, Member

Qian-Yong Chen, Member

Sundar Krishnamurty, Professor and Department Head
Mechanical and Industrial Engineering Department

DEDICATION

To my family.

ACKNOWLEDGMENTS

I would be ever grateful to my advisor, Prof. Stephen de Bruyn Kops, for his thoughtful and patient guidance. This work would not have been possible without his motivation and support.

Special thanks to Prof. Blair Perot and Prof. Qian-Yong Chen for kindly serving as my thesis committee and providing valuable suggestions.

I wish to thank my family without whom none of this would ever have been possible. I have had their support in everything I have done. Thank You.

Thanks to all my lab colleagues (Gavin, Felipe, Kedar and Abhishek) for helping me during the thesis. A special thanks to Gavin and Felipe for valuable guidance during the various stages of the work in aspects related to programming, mathematics and presentation. I would be thankful to all those whose support helped me to stay focused on completing the thesis and who have provided me with encouragement to continue when the going got tough.

ABSTRACT

NON-EQUISPACED FAST FOURIER TRANSFORMS IN TURBULENCE SIMULATION

SEPTEMBER 2017

ADITYA MOHAN KULKARNI, B. E., UNIVERSITY OF PUNE

M.S.M.E.

UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Prof. Stephen de Bruyn Kops

Fourier pseudo-spectral method is one of the approaches used to compute the derivative of a discrete data in Computational Fluid Dynamics. The Fourier transform of the data sampled on equispaced gridpoints is computed using Fast Fourier Transform (FFT), and the Fourier transform of derivative is obtained by multiplying each Fourier coefficient by its corresponding wavenumber and the imaginary number $i = \sqrt{-1}$. In a number of turbulent flows like wakes, jets etc., the dynamically important scales of motion are concentrated in some regions, which require a finer gridspacing and other regions may have a coarse grid. Use of non-equispaced grid can potentially lead to reduced memory usage without sacrificing accuracy, which is particularly important as memory throughput is a major limiting factor of the Direct Numerical Simulations (DNS) performed today. The aim of this thesis is to implement the non-equispaced grid in DNS, using the Non-Equispaced Fast Fourier Transform (NFFT)[1] algorithm.

In order to be able to achieve the similar accuracy using reduced number of gridpoints, the number of Fourier coefficients needs to be larger than that of the gridpoints. NFFT calculates the Fourier transform by solving a system of linear equations using a variant of conjugate gradient method, which in our case, becomes an under-determined system. A combination of NFFT and an iterative reconstruction algorithm, FOCUSS [2] is used to obtain the accurate Fourier transform of certain test functions, by solving an under-determined system of equations. The combination of NFFT and FOCUSS algorithm is also used to perform a small test case of Direct Numerical Simulation on a grid of 64^3 points, using Taylor Green initial conditions and the results are found to

be in agreement with the analytical solution. The speed of this simulation is slower than acceptable, which can be attributed to an increased condition number of the DFT matrix and various means that have the potential of increasing the computational performance are analyzed.

A test is also performed on slice of a 3 dimensional field of fluctuating density of a turbulent wake of high Reynold's number and larger size with 1024 gridpoints in the NFFT, which is similar to the ultimate expected application. The derivative computed for this slice using NFFT and FOCUSS is found to be inaccurate. The errors in the derivative can be attributed to the inaccurate computation of the Fourier transform using a non-equispaced grid in under-determined case. The FOCUSS algorithm is thus found incapable of computing the Fourier transform by solving an under-determined system of linear equations in cases like turbulence simulation where the fields have a wideband Fourier transform.

TABLE OF CONTENTS

	Page
ACKNOWLEDGEMENTS	v
ABSTRACT	vi
LIST OF TABLES	x
LIST OF FIGURES	xi
CHAPTER	
1 MOTIVATION	1
1.1 Introduction	1
1.2 Discretization Methods	2
1.2.1 Finite Volume Method	2
1.2.2 Finite Difference Method	2
1.2.3 Finite Element Method	3
1.2.4 Spectral Method	5
1.3 Fourier Series	5
1.4 Fast Fourier Transform	7
1.5 Other Approaches	8
1.6 Non-Equispaced Grid	9
1.7 Non-Equispaced FFT	10
1.8 NFFT in Turbulence	11
2 NFFT AS LINEAR SYSTEM OF EQUATIONS	13
2.1 Moore-Penrose Pseudoinverse	13
2.1.1 Linear Systems of Equations	14
2.1.2 Solution of Linear Systems	15

2.1.3	Fourier Transform as Linear System	16
2.2	Iterative Method for Linear Systems	17
2.2.1	Conjugate Gradient	18
2.2.2	Normal Equations	18
2.2.3	Conjugate Gradient on Normal Equations	19
2.3	Minimum Norm Solution	20
3	FOCUSS ALGORITHM AND TESTING	23
3.1	NFFT with Minimum Norm Solution	23
3.2	FOCUSS Algorithm	24
3.3	Introduction to Turbulence and its Numerical Simulation	29
3.4	Implementation on a test case	30
4	CONDITIONING OF NFFT	34
4.1	Condition Number	34
4.2	Condition Number in NFFT	35
4.3	Analysis	37
4.4	Methods for Improving Speed	40
4.4.1	Preconditioning	41
4.4.2	Selection of Number of Fourier Coefficients	43
4.4.3	Splitting the Problem	44
5	TESTS ON TURBULENT WAKE	47
5.1	Slicing the Wake Field	47
5.2	Accuracy	48
5.3	Analysis	50
6	CONCLUDING REMARKS	54
	APPENDIX: 'C' CODE FOR NFFT	57
	BIBLIOGRAPHY	59

LIST OF TABLES

Table	Page
3.1 CPU time requirement for Taylor Green vortex simulation for 32^3 grid-points in real space, $32^2 \times 40$ Fourier coefficients, carried out for 50 timesteps.	32

LIST OF FIGURES

Figure	Page
1.1 Comparison of Error in Spectral Derivative (taken using Fourier series) and Finite Difference Derivative. A demonstration of h -type and p -type convergence.	5
2.1 Graphical illustration of minimum-norm solution	22
3.1 (a,c): Absolute values of the Fourier coefficients against the wavenumbers for function $\sin(2\pi x)$ and $x \in [-0.5, 0.5]$; (b,d): Spectral derivative of the function with respect to $2\pi x$	25
3.2 (a,c): Absolute values of the Fourier coefficients against the wavenumbers for the Gaussian $\exp(-50x^2)$ and $x \in [-0.5, 0.5]$; (b,d): Spectral derivative of the function with respect to $2\pi x$	28
3.3 The grid and a typical function observed in turbulence	29
3.4 The Solution of 3D Navier Stokes equation with Taylor Green initial conditions. The solid lines give the results of DNS using NFFT. Stars (*) denote the solution obtained independently using an equispaced grid. Dashed lines give the analytical solution. W' and T represent the dimensionless forms of dissipation rate and time, respectively. .	33
4.1 (a,b): Condition number of K ; (c,d): Number of conjugate gradient iterations required for convergence for $n/m = 1.75$	36
4.2 (a,b): Condition number of K ; (c,d): Number of conjugate gradient iterations required for convergence for $n/m = 1.5$	37
4.3 Left: Real part, Right: Imaginary part of the modified coefficient matrix K before application of weights. This matrix is independent of the function but depends only on the grid.	38

4.4	Left: Real part, Right: Imaginary part of the modified coefficient matrix K after application of weights in computation of Fourier transform of a Gaussian	39
4.5	Eigenvalue distribution of matrix K ; Left: before application of weights (independent of the function), right: After application of weights for computation of Fourier transform of a Gaussian	39
4.6	Left: Real part, Right: Imaginary part of the modified coefficient matrix for $m = 1024$ non-equispaced gridpoints, $n = 2048$ Fourier coefficients before application of weights	40
4.7	Left: Real part, Right: Imaginary part of the modified coefficient matrix for $m = 1024$, $n = 2048$ after application of weights for NFFT of a slice of turbulent wake	40
4.8	Eigenvalue distribution of modified coefficient matrix K for NFFT of slice of turbulent wake before application of weights (Left) and after application of weights for 4 FOCUSS iterations(Right)	41
4.9	The reduction in conjugate gradient iterations required with preconditioner applied. Numbers in legend indicate the number of FOCUSS loops for which preconditioner was calculated, for the test case of a Gaussian.	42
4.10	The reduction in condition number of modified coefficient matrix K with application of preconditioner. Numbers in legend indicate the number of FOCUSS loops for which preconditioner was calculated, for the test case of a Gaussian.	43
4.11	The variation of $\kappa(K)$ with varying n . The grid has $m = 128$ non-equispaced points and $1/\Delta x_{\min} = 192$. The condition number depends only on the grid before application of weights.	44
4.12	Fourier coefficients for higher wavenumbers computed by splitting the matrix, compared with the FFT on equispaced grid. Left: k from -127 to -64, Right: k from 65 to 128	46
5.1	The line on a slice of the fluctuating density field in a turbulent wake. Y-axis shows the density variations.	48
5.2	(a, c): Blue lines indicate a line on a slice of turbulent wake mapped onto non-equispaced grid, red dots indicate the original line on equispaced grid. (b, d): Blue lines indicate the derivative of lines (a) and (c) respectively sampled on non-equispaced grid, red lines indicate the derivative of original slice sampled on equispaced grid. . .	49
5.3	Spectrum of a line on a slice of turbulent wake, $m = 1024$, $n = 2048$	50
5.4	The spectrum of a line on a slice of turbulent wake, for wavenumbers from 900 to 1024.	51

5.5	The spectrum of derivative of a line on a slice of turbulent wake, for wavenumbers from 900 to 1024	52
5.6	Fourier spectrum of a sinewave, sampled on a non-equispaced grid, for first three FOCUSS iterations. Legend indicates the number of FOCUSS iterations.	52
5.7	Effect of FOCUSS algorithm on the Fourier transform of a line on a slice of turbulent wakes for wavenumbers ranging from 900 to 1024.	53
5.8	Left: The derivative of a line on a slice of turbulent wake, Right: Relative error in the derivative when it is computed using one FOCUSS loop and NFFT	53

CHAPTER 1

MOTIVATION

1.1 Introduction

Computational fluid dynamics is a widely used tool that solves the governing equations of fluid flow numerically, as they are too complicated to solve analytically when applied to general problems. With the increase in processing power of computers, we are able to solve more and more complicated fluid dynamics problems without sacrificing accuracy. The basic approach to CFD can be explained by following four steps:

1. Define geometry of the problem
2. Discretize the geometry, that is, divide the geometry into finitely small discrete elements.
3. Identify the governing equations
4. Solve the equations using appropriate numerical algorithms

The basic underlying idea here is to discretize the domain (volume, area etc.) of our focus into discrete small elements, use mathematical methods to convert partial and/or ordinary differential equations into algebraic equations and solve them numerically. The governing equations are equations of conservation of some physical property like mass, momentum, energy etc. There are various methods of discretisation, the most commonly used ones being Finite Volume Method, Finite Element Method, Finite Difference Method etc.

From the study of numerical analysis[3] we know that the accuracy of the solution depends upon the size of the discrete elements. Generally, smaller the elements, better is the accuracy and vice versa. In turbulence simulation, finer discretization allows us to resolve the smaller length scales. In many turbulence problems including but not limited to jets, wake, plumes etc., the smaller and dynamically important scales are concentrated in some regions than others. In other words, there are some regions with smaller lengthscales and some regions without them, for which a coarser discretization is sufficient. The finer discretization can only be used in only the regions containing

the small lengthscales to be able to accurately resolve them. A coarser discretization can be used in other regions, as fine discretization increases the memory and computational requirements.

If the domain is divided using a finite number of points called gridpoints, the above approach will make the grid non-equispaced. The purpose of this thesis is to assess the feasibility of using Fourier spectral method on a non-equispaced grid in turbulence simulation.

1.2 Discretization Methods

There are various discretization methods with their own advantages and disadvantages. It is the job of fluid dynamicist to select the best method for his/her purpose.

1.2.1 Finite Volume Method

The Finite Volume Method (FVM) is the most widely used method for fluid problems. In this method, the volume occupied by the fluid is divided into finite discrete non overlapping volumes called as elements or cells. Partial differential equations are converted into algebraic equations enforcing the requirement that fluxes between adjacent cells satisfy simple algebraic relationships for most quantities of interest in fluid dynamics[4]. FVM solves the integral conservation law equation numerically.

$$\frac{\partial}{\partial t} \iiint_V \mathbf{Q} dV + \iint_A \mathbf{F} d\mathbf{A} = \iiint_V \mathbf{S} dV$$

Here, \mathbf{Q} is the property that is conserved, \mathbf{F} is the flux of properties and \mathbf{S} is a source. The cells are so small that \mathbf{Q} is assumed to be constant over a cell and dV is the cell volume. Also, the total flux from cell surfaces can be calculated by summation of fluxes on all areas. It can be seen that FVM is an approximation just like other numerical methods and smaller the cell size, better is the accuracy of FVM. FVM is inherently conservative, which makes it more suitable for fluid problems.

1.2.2 Finite Difference Method

In the Finite Difference Method (FDM), the domain is discretized by considering various points at discrete intervals inside the domain, the values of properties at those points being known[5]. Governing equations, which are partial differential equations are solved numerically using finite

difference approximation. The underlying idea of FDM is the Taylor series.

$$f(x+a) = f(x) + \frac{f'(x)}{1!}a + \frac{f''(x)}{2!}a^2 + \frac{f'''(x)}{3!}a^3 + \dots$$

Based on Taylor series about point x and $f(x)$, the values of derivatives of $f(x)$ with respect to x can be approximated. The accuracy of the approximation reduces with an increase in the value of a . Finite difference method requires a structured grid.

1.2.3 Finite Element Method

In Finite Element Method (FEM), the domain is divided into a finite number of subdomains (finite elements)[6]. Inside each subdomain, the dependent variable $f(x, t)$ is approximated as a linear combination of basis functions.

$$f(x, t) \approx f_N(x, t) = \sum_k c_k(t)\phi_k(x)$$

Here, f_N is the approximation of the dependent variable f , $\phi_k(x)$ are the basis functions and $c_k(t)$ are their coefficients at a time t . Smoothness and continuity needs to be ensured at the boundary of each subdomain. Selection of basis function is influenced by several factors, like boundary conditions, accuracy requirements, computational cost etc. The computation of coefficients c_k is done by minimizing the following:

$$R(x) = f_N(x) - f(x) \rightarrow \min$$

$R(x)$ is called as residual. It can be seen that for exact interpolation, the residual is zero. In theory, coefficients c_k can be found by solving a system of N linear equations in N variables.

The basis functions $\phi_k(x)$ may be algebraic expressions in x like x^k , so that $f(x, t)$ is an algebraic polynomial in x ; or they may be chosen from one of the following[7]:

1. Fourier Series
2. Chebyshev Polynomials
3. Legendre Polynomials
4. Spherical Harmonics
5. Laguerre Functions

Accuracy of the interpolation depends mainly on three metrics and it can be increased (also called as refinement) by altering one or more of those metrics, which are as follows[7]:

1. Increase the number of discrete points, called h -type refinement
2. Increase the number of discrete points in the regions of steep gradients, called r -type refinement. This is the topic of our interest in this thesis.
3. Increase the degree of interpolating polynomial, called p -type refinement. The degree of interpolating polynomial is typically denoted by p , hence the name p -type refinement.

The error is introduced due to interpolating the data in FEM is called as truncation error and this introduces numerical diffusion and dispersion in the problem. For turbulence simulation, the discretization scheme is desired to be as free from these errors as possible[8]. Increasing the number of points in entire domain or in some parts of the domain (h or r -type refinements) reduce the truncation errors. However, the downside of these is the requirement of more number of points and a corresponding increase in memory requirement. The p -type refinement, which uses a higher order interpolation, is a potential alternative to reduce truncation error without increasing memory requirement.

The degree of interpolating polynomial $c_k\phi_k$, denoted by p determines the accuracy of the discretization. Higher the value of p , higher is the accuracy and vice versa. FEM is called as Spectral Element Method (SEM) if higher order functions are used in discretization, typically $p > 6$. SEM is thus, a special case of FEM.

In SEM or FEM, the partial differential equations that needs to be solved is given as follows:

$$\frac{\partial f}{\partial t} = \mathcal{L}f(x, t)$$

where \mathcal{L} is an operator containing partial differentiation in the spatial domain. When $f(x, t)$ is interpolated as a sum of basis functions, above equation can be written as follows:

$$\frac{\partial f}{\partial t} \approx \frac{\partial f_N}{\partial t} = \sum_{k=0}^N c_k(t)\mathcal{L}\phi_k(x)$$

This is an ordinary differential equation in time, since $\phi_k(x)$ is known, and can be solved numerically by finite differencing in time.

1.2.4 Spectral Method

Spectral method is a special case of SEM where the number of subdomains is 1, i.e., the domain is not divided into smaller domains. Spectral method takes a global approach to interpolate the function and defines a single high order interpolating function in the entire domain. The basis function can be any of the functions listed above.

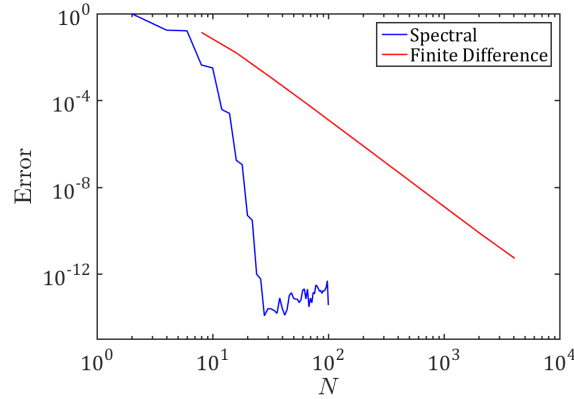


Figure 1.1: Comparison of Error in Spectral Derivative (taken using Fourier series) and Finite Difference Derivative. A demonstration of h -type and p -type convergence.

Fig 1.1 shows the comparison between error in derivative of $e^{\sin x}$ taken over interval $(0, 2\pi)$ by Fourier spectral method and fourth order finite difference order method. It can be observed that the spectral method can achieve the same level of accuracy with a lesser number of points as compared to the finite difference method thereby requiring lesser memory.

1.3 Fourier Series

The Fourier series is an example of spectral expansion of a function where the basis function is a trigonometric function which can be indicated as

$$f(x) = \sum_{k=-\infty}^{\infty} c_k e^{ikx}$$

Fourier series can be characterized by rapid convergence, which is faster than algebraic, that is, $k^p c_k \rightarrow 0$ as $k \rightarrow \infty$ for all $p > 0$ [9]. Thus, a Fourier transformable function $f(x)$ can be accurately interpolated by a Fourier series truncated at $k = \pm N$.

$$f(x) \approx f_N(x) = \sum_{k=-N}^N c_k e^{ikx}$$

The rapid convergence can also be visualized from fig. 1.1. Derivative calculated by Fourier spectral method quickly approaches the roundoff error, the least possible error. This also indicates that the truncation errors of Fourier series are limited by the roundoff error, given that the number of points N is sufficiently large. This low truncation error makes Fourier series interpolation suitable for turbulence simulations.

The process of calculating values of coefficients c_k is called Fourier transform. Also, if the values of coefficients c_k are known, then the function $f(x)$ can be reconstructed by a process called Inverse Fourier Transform. However, Fourier series gives accurate representations only for periodic and continuous function. If the function is not periodic, then there appears a jump discontinuity at the ends of the domain and the representation is inaccurate at the discontinuity, due to Gibbs phenomenon[10].

The constants are often represented as \hat{f} in Fourier analysis, by convention. Fourier transform can be taken analytically as follows:

$$\hat{f}(k) = \int_{-\infty}^{\infty} f(x)e^{-ikx} dx \quad (1.1)$$

The inverse Fourier transform, which reconstructs the function f from known values of Fourier coefficients \hat{f} can be taken analytically as

$$f(x) = \frac{1}{P} \int_{-\infty}^{\infty} \hat{f}(k)e^{ikx} dk \quad (1.2)$$

Here, P is the time period of the function.

Spectral method where Fourier series is used to interpolate the data is called as Fourier spectral method. In Fourier spectral method, error in the interpolation reduces as $\mathcal{O}(N^{-N})$ where N is the number of gridpoints. This is called as “infinite order convergence” or “exponential convergence”.

In finite difference or finite volume method, the error in spatial derivative can be given as Δx^c where $\Delta x \ll 1$, $\Delta x = 1/N$ and c is a positive integer that depends on the discretization scheme used but not on the number of gridpoints. Usually, in compact finite difference method or in finite volume method, c lies between 4 and 6[11, 12, 13]. In Fourier spectral method, the error in spatial derivative scales as Δx^N due to exponential convergence. Thus, increasing the number of points increases not only reduces the Δx but also increases the order of accuracy of the discretization scheme with the same computational cost as other methods.

Another factor that makes a difference in the quality of CFD solutions is the numerical dispersion.

This is the error in the phase of a wave whose propagation is simulated numerically. Numerical dispersion is also called as phase error. The turbulent eddies are transported in spatial coordinates and numerical dispersion can play an important role in the overall accuracy of the simulation. The spatial transport is modelled by equations of type $(c \partial f / \partial x)$, where c is the wave speed and $f(x, t)$ is the field that travels in space.[3] The spatial derivative $\partial f / \partial x$ is approximated by the discretization methods. Higher the error in spatial derivative, higher is the numerical dispersion error in the final solution. As explained earlier, the Fourier method gives least error in the spatial derivative and will thus lead to minimal dispersion errors. The same can be said about numerical dissipation which depends upon the second derivative of the field, $(d \partial^2 f / \partial x^2)$.

A combination of lower truncation, dispersion and diffusion errors explains the incentive to use Fourier spectral method in CFD problems.

1.4 Fast Fourier Transform

In many cases the function $f(x)$ is known at discrete points x_j where $j = 0, 1, 2, \dots, M - 1$. The Fourier transform of such a function can be obtained by the process of discrete Fourier transform (DFT). Fast Fourier Transform (FFT) is an algorithm that computes the discrete Fourier transform and has been classified in the top ten algorithms of the 20th century[14]. The standard notation for discretized Fourier series is as follows

$$f(x_j) = \sum_{k=-\infty}^{\infty} \hat{f}_k e^{ikx_j} \quad (1.3)$$

where $j = 0, 1, 2, \dots, M - 1$.

To take the forward Fourier transform of the discrete data, the values of coefficients \hat{f}_k for all $k \in (-\infty, \infty)$ need to be found. $f(x)$ can then be approximated by an interpolating polynomial I .

It is impossible to store and calculate infinite values of \hat{f} . Thus, we limit the number of \hat{f} to some number n and assume that $\hat{f}_k = 0$ for $|k| > n/2$. This assumption is justified due to the exponential convergence discussed earlier. This yields a truncated Fourier series where the summation in equation (1.3) is taken from $-n/2 + 1$ to $n/2$.

The most commonly used FFT algorithm is the Cooley-Tuckey algorithm that calculates the FFT in $\mathcal{O}(N \log N)$ operations[15]. For any discrete Fourier transform, the length of Fourier transform is limited by the number of the function values m , i.e., the length of Fourier transform, $n = m$ [16]. Thus, we get a truncated Fourier series, which makes the interpolating polynomial of order n as

opposed to the infinite order polynomial as expected by theory. This interpolating polynomial aims to approximate the function in order to take its derivatives. As explained in the previous section values of \hat{f} converge rapidly to zero due to exponential convergence and truncation of the series to $n = m$ terms is an accurate approximation.

In the standard FFT, when it is required that $n > m$, output of FFT algorithm is padded with zeros for values of $k > m$. If it is required that $n < m$, then the FFT output is chopped.

Discretized forms of equations (1.1) and (1.2) for calculation of forward FFT and inverse FFT respectively are given as:

$$\hat{f}_k = \sum_{j=0}^{m-1} f(x_j) e^{-ikx_j}$$

$$f(x_j) = \frac{1}{nP} \sum_{k=-n/2+1}^{n/2} \hat{f}_k e^{ikx_j}$$

Here, n is the number of discrete function values ($n = m$ for FFT) and P is the period of the function.

In spectral method, Fourier series or other high order methods are used to interpolate the function from its discrete values and the spatial derivative can then be taken. For a discretely defined function f with Fourier coefficients \hat{f}_k , the Fourier coefficients of its spatial derivative $\partial f/\partial x$ are given as $ik\hat{f}_k$. It can be easily derived by differentiating equation (1.3). Thus the spatial derivative can be calculated by taking inverse DFT of $ik\hat{f}_k$.

1.5 Other Approaches

Fourier spectral methods is one of the approaches of high order approximations, also called as p -type refinement. Although Fourier series interpolation is highly accurate, it leads to a loss of accuracy in the regions of discontinuous gradients, e.g. boundaries, due to the well known Gibbs phenomenon[10]. Thus, it is suitable only for simulations with periodic boundary conditions.

Other methods to interpolate a polynomial with a high order accuracy have been used in turbulence simulations involving sharp gradients. Chebyshev spectral method, B-spline interpolation and compact finite difference schemes[17] are some of the examples.

Chebyshev collocation method uses Chebyshev polynomials instead of Fourier series to approximate the function from discrete given values. It has been used by Moser *et. al.* in 1999 [18] to simulate channel flow between two infinite horizontal plates. Wengle and Seinfeld(1978)[19] have proven that the differentiation matrix in Chebyshev spectral method may be ill conditioned and

errors in the smallest coefficients of Chebyshev series may cause a large error in even the largest coefficients of derivative of Chebyshev series. For higher Reynolds numbers, we need to use a larger number of points thereby increasing the possibility of errors in derivatives. Moreover, the grid-points can't be chosen arbitrarily for a Chebyshev polynomial interpolation. The grid-points (collocation points) must be chosen from either the Gauss-Chebyshev quadrature or Gauss-Lobatto quadrature points[7], as in other cases the higher order differentiation becomes ill conditioned.

The compact finite difference scheme(Lele 1992[17], Fadel 2011 *et. al.*[11], Shukla *et. al.* 2007[20]) has been implemented in channel flow by Avsarkisov *et. al.* [21]. Compact finite difference schemes are generalizations of Padé schemes. As shown by Lele(1992)[17], the first derivative calculated using compact finite difference schemes has some error compared to derivative calculated by Fourier spectral method. The errors increase as the order of derivative increases, i.e., the errors are higher for second derivative as compared to the first derivative and so on. These errors are in the high wavenumber components of the derivative, and they accumulate over the time. Thus filters are applied to mitigate them, as discussed by Lele (1992)[17] and Zhang *et. al.* (2004) [22]. The overall accuracy of the scheme is limited by the accuracy of the filters which are usually third or fourth order. It has also been discussed that the error increases with increasing order of the derivative. We, thus, look for another scheme to interpolate the discretized data.

Myoungkyu and Moser(2015)[23] have solved turbulent channel flow using B-spline collocation method. As explained by Kwok *et. al.*(2001)[24], B-spline collocation method suffers from similar issues as compact finite difference scheme discussed above, that the error increases with each derivative. B-spline collocation, however, is more accurate than the compact finite difference schemes, but still less accurate than spectral methods.

The errors in the spatial differentiation due to other methods will lead to numerical diffusion and dispersion, as discussed earlier. Limitations of other methods using high order polynomials indicate that the Fourier series is the best high order interpolation method for periodic boundary conditions and smooth functions.

1.6 Non-Equispaced Grid

The r -type refinement approach discussed in sec. 1.2.3 increases the grid resolution only in the places where we need a finer resolution. As discussed earlier, the r -type refinement is precisely what we need to simulate a number of turbulent flows with increased accuracy. h -type refinement leads to an increased number of gridpoints even where they are not required. This implies that the memory

requirement for h -type refinement is higher than the r -type approach.

Chebyshev methods described above is one of the ways to solve the problem with non-equispaced grid. Another way to take a derivative of a function on non equispaced grid, using compact finite differences, is to map the grid first onto an equispaced grid. Let the original nonequispaced grid be denoted by x and its mapping on uniform grid be denoted by x' [25]. When the function f is defined on grid x , its derivative can be taken as follows:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial x'} \frac{dx'}{dx}$$

Above two derivatives can then be calculated with compact finite difference method.

However, we want to get r -type refinement without the limitations of Chebyshev methods on selection of gridpoints or finite difference method on the overall accuracy due to involvement of filters. Thus, we want to be able to use Fourier spectral method on a non-equispaced grid. The FFT library based on the Cooley Tuckey algorithm discussed in sec. 1.4 requires the function to be sampled on equally spaced points. Thus, a different technique is needed to solve the problems with non-equispaced grid. The Non-Equispaced Fast Fourier Transform (NFFT) library computes the Fourier transform of data sampled on non-equispaced grid, which will be explained in section 1.7 and chapter 2.

1.7 Non-Equispaced FFT

Non Equispaced FFT (also called NFFT) is the calculation of FFT when the function $f(x_j)$ is known at non equispaced discrete gridpoints. In principle, NFFT can be done by solving a linear system of equations. The system of equations can be arranged in such a way that the residual $f - f_N$ is minimized at a finite number of points x_j where f and f_N denote the original and the interpolated functions respectively. In this case, x_j are called as collocation points and this approach is called pseudospectral method or collocation method. Mathematically,

$$\|f(x_j) - f_N(x_j)\| \rightarrow 0$$

Here, $\|\cdot\|$ indicates a L_2 norm or Euclidean norm. In this case, if $N = M$, the system of equations is a well determined system and it has a solution. If $N < M$, the system is an over-determined system and it does not have a solution. Thus, we use the lease square approach to get an interpolating

polynomial I_N such that the residual is minimized. If $N > M$, the system is an under-determined system of equations and it has infinite number of solutions. An additional constraint is required in order to achieve a particular solution. These additional constraints will be explained in subsequent chapters.

1.8 NFFT in Turbulence

In numerical simulation of turbulent flows, truncation of Fourier series at a larger wavenumber is always desirable in order to be able to resolve the smallest dissipative motions (characterized by Kolmogorov lengthscales)[26]. It needs to be understood that when we truncate the Fourier series at a index N , we put the values of wavenumbers greater than N equal to zero. In turbulence, we are interested in amplitudes of Fourier coefficients at higher wavenumbers. In turbulence, the amplitudes at high wavenumbers are fairly small as explained by Pope (2000) (section 6.5.4)[26]. More discussion on this will be done in section 3.3.

The amplitude decreases exponentially as wavenumber increases, becoming zero as $k \rightarrow \infty$. Higher wavenumber Fourier coefficients are of importance in turbulence simulation to resolve the smallest scales[26, 27]. If equispaced grid and FFT are used to simulate turbulence which is resolved at higher wavenumbers, we need a larger number of gridpoints as FFT requires a corresponding increase in the number of gridpoints. This increases the memory requirements of the simulation.

In turbulence simulation, time-stepping is done on entire velocity field. This requires the field to be loaded into physical memory before time-stepping. Thus, the requirement of memory increases with an increase in the field size or the number of gridpoints. We would like to be able to increase the resolution of the grid only in the regions of dynamically important scales, and achieve the same effect as increasing the resolution of entire grid by using NFFT. Let's assume that the initial grid had p equally spaced points. The grid-spacing, i.e. the spacing between adjacent points on the grid is $1/p$. Using FFT, we can get p Fourier coefficients and compute the derivative. Now, the number of equispaced points and the number of Fourier coefficients that can be obtained is increased to $P > p$. Smaller lengthscales are responsible for producing a non-zero amplitude in the higher wavenumbers of Fourier series. With increased gridpoints, the smaller lengthscales are not ignored thereby increasing the accuracy. The grid-spacing in this case is $1/P$. Note that $\frac{1}{P} < \frac{1}{p}$. This approach, however, results in an increased memory usage as P values need to be stored. Using NFFT, we redistribute p points in the grid so that the grid-spacing in the regions containing smaller dynamically important scales is $1/P$ while that of other regions is greater than $1/p$. If P Fourier

coefficients can be obtained from this arrangement, the same accuracy can be achieved with lesser gridpoints, thereby reducing the memory requirement.

Aim of this thesis is to use the NFFT algorithm developed by Kunis *et. al.* [1] to evaluate Fourier transform of a function known at non equispaced discrete gridpoints. NFFT algorithm inherently does not calculate Fourier transform where the number of Fourier coefficients, $N > M$ where M is the number of discrete data points. We modify the algorithm in order to make it able to calculate N Fourier coefficients where $N > M$.

NFFT algorithm has been explained in detail in chapter 2 for inverse Fourier transform, i.e. calculation of Fourier coefficients \hat{f}_k when discrete function values $f(x_j)$ are known. Pseudo codes for the algorithm used have been presented. Theory of inversion of non square matrices, called Moore-Penrose pseudoinverse [28] has also been discussed along with its implementation to solve systems of linear equations. The problem of solving an under-determined system of linear equations is also discussed, which is what we need when $N > M$.

The method to get correct Fourier transform when $N > M$, called FOCUSS algorithm (Gorodnitsky and Rao 1997)[2] has been discussed in chapter 3. Implementation of NFFT and FOCUSS to get Fourier transform of test function has also been discussed.

It is observed that the computational speed of the test case of computational simulations using NFFT and FOCUSS algorithms is too slow to be acceptable in large sized simulations. Mathematical aspects related to the slow speed have been discussed in Chapter 4.

The combination of NFFT and FOCUSS algorithms was later tested on slices of large sized wake of realistic resolution $4096 \times 2048 \times 2048$. Non-equispaced grid was used in z direction with a size of 2048 equispaced gridpoints, which was mapped onto a grid of 1024 non-equispaced points and NFFT and FOCUSS algorithm was tested on it. The results related to accuracy of the speed have been analyzed in Chapter 5.

Finally, Chapter 6 discusses the conclusions. It was observed that the combination of NFFT and FOCUSS algorithms can not be used in our case, for direct numerical simulations of turbulent flows.

CHAPTER 2

NFFT AS LINEAR SYSTEM OF EQUATIONS

Non Equispaced Fast Fourier Transform (NFFT) is a library to calculate the forward and inverse Fourier transforms of discretized function values at non-equispaced points. When values of function $f(x_j)$ are known at M non-equispaced nodes x_j where $j = 0, 1, 2, \dots, M - 1$, calculation of N Fourier coefficients \hat{f}_k where $k = -N/2 + 1, \dots, 0, \dots, N/2$ is called as forward Fourier transform. Forward Fourier transform implementation by Kunis *et. al.* [1] involves solving a system of linear equations to calculate the coefficients of truncated Fourier series. The system can be categorized as under-determined ($N > M$), well-determined ($N = M$) and over-determined ($N < M$). Under-determined and over-determined system solution involves inversion of non-square coefficient matrix. A system of linear equations involving non square coefficient matrix can be solved by taking Moore-Penrose pseudoinverse of the matrix as discussed in section 2.1. Section 2.2 discusses the adaption of conjugate gradient method used to invert the matrix in NFFT. Section 2.3 discusses the problem of solving under-determined system of linear equations which is of our interest.

2.1 Moore-Penrose Pseudoinverse

Moore Penrose pseudoinverse[28], also called as generalized inverse, was developed by E. H. Moore and Roger Penrose. Pseudoinverse of a matrix A is denoted by A^+ . Pseudoinverse can be defined as a matrix that satisfies the following properties:

1. $AA^+A = A$

Matrix AA^+ need not be an identity matrix, but it maps the columns of matrix A to themselves.

2. $A^+AA^+ = A^+$

Matrix A^+A need not be an identity matrix, it just maps the columns of A^+ to themselves.

3. $(AA^+)^* = AA^+$

Here, $*$ denotes the conjugate transpose. Matrix AA^+ is a hermitian matrix.

$$4. (A^+A)^* = A^+A$$

Matrix A^+A is also a hermitian matrix

Moreover, for any matrix $A \in \mathbb{C}^{m \times n}$, the pseudoinverse A^+ always exists and is unique. Some other properties of the pseudoinverse are given as follows:

$$1. \text{ If } A \text{ is square and invertible, then } A^+ = A^{-1}$$

$$2. (A^+)^+ = A$$

$$3. (\alpha A)^+ = \alpha^{-1}A^+, \text{ where } \alpha \text{ is a scalar.}$$

$$4. (AB)^+ = B^+A^+$$

2.1.1 Linear Systems of Equations

Systems of linear equations that have multiple variables and multiple linear equations are called as linear systems. The equations can be solved simultaneously to get a solution. A linear system with m equations and n unknowns can be represented as follows:

$$\begin{array}{cccccc} a_{11}x_1 & +a_{12}x_2 & + \dots & +a_{1n}x_n & = & b_1 \\ a_{21}x_1 & +a_{22}x_2 & + \dots & +a_{2n}x_n & = & b_2 \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ a_{m1}x_1 & +a_{m2}x_2 & + \dots & +a_{mn}x_n & = & b_m \end{array}$$

Above system can also be written in matrix form as follows:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

Let's use the following notation

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^T \text{ and } \mathbf{b} = \begin{bmatrix} b_1 & b_2 & \dots & b_m \end{bmatrix}^T$$

The system can then be represented as

$$A\mathbf{x} = \mathbf{b} \quad (2.1)$$

Here, $A \in \mathbb{C}^{m \times n}$ is called as the coefficient matrix. $x \in \mathbb{C}^{n \times 1}$ is the vector containing the unknowns. $b \in \mathbb{C}^{m \times 1}$ is the vector containing all the scalar constants. Coefficient matrix A can be a square matrix ($m = n$), a vertical matrix ($m > n$) or a horizontal matrix ($n > m$). Based on this, the system can be divided into following three classes.

1. Well-determined, when $m = n$. Well-determined systems have equal number of unknowns and equations. They have exactly one solution.
2. Over-determined, when $m > n$. Over-determined systems have more equations than unknowns. Over-determined systems of equations have no solution. The residual can however be minimized as discussed above.
3. Under-determined, when $n > m$. Underdetermined systems have more unknowns than equations. Under-determined systems have infinite number of solutions.

2.1.2 Solution of Linear Systems

A system of linear equations can be solved by multiple methods, analytically by following approaches[29]:

1. Elimination of Variables.
 - Solve first equation for one variable in terms of other
 - Substitute the expression in remaining equations, which reduces one variable from the system
 - Go on reducing one variable at a time until only one variable remains.

- Value of other variables can then be calculated by back substitution.
2. Row reduction: Above approach can be simplified in matrix form calculations by using row reduction form. An augmented matrix is created by augmenting scalar constants matrix b as an additional column to the coefficient matrix A . This is then reduced by row transformations into an upper triangular matrix. This approach is also called as Gauss elimination.
 3. Matrix Inversion: A system of linear equations can also be solved by matrix inversion. Both the sides of equation (2.1) can be multiplied by A^+ , the pseudoinverse of coefficient matrix A . The solution is then given as:

$$A^+Ax = A^+b$$

$$x = A^+b$$

This approach can be validated from the fact that A^+A maps column vectors to themselves.

When the system is large, i.e. m and n are large, it is often solved numerically with the help of computers. Elimination of variables and row reduction are extremely expensive operations, thus are not commonly used for large m and n . Also, matrix inversion is an expensive operation, thus not used for large systems of equations. Most commonly used methods to solve large linear systems numerically are conjugate gradient, which will be discussed in the subsequent sections.

2.1.3 Fourier Transform as Linear System

Discrete Fourier transform can be done by solving a system of linear equations, so that the values of Fourier series evaluated at given points matches the discrete function values at those points, using collocation approach as explained in chapter 1. Kunis (2006 Ph.D. Dissertation)[30] has discussed this approach when the period of the function is unity and it is defined on the interval $[0.5, 0.5)$ such that $f(0.5) = f(-0.5)$. Coefficient matrix for Discrete Fourier transform (DFT) can be written as follows:

$$A = \begin{pmatrix} e^{2\pi i k_1 x_1} & e^{2\pi i k_2 x_1} & \dots & e^{2\pi i k_n x_1} \\ e^{2\pi i k_1 x_2} & e^{2\pi i k_2 x_2} & \dots & e^{2\pi i k_n x_2} \\ \vdots & \vdots & \vdots & \vdots \\ e^{2\pi i k_1 x_m} & e^{2\pi i k_2 x_m} & \dots & e^{2\pi i k_n x_m} \end{pmatrix}$$

The unknown and constant vectors can be written as:

$$\hat{\mathbf{f}} = \begin{bmatrix} \hat{f}_1 & \hat{f}_2 & \dots & \hat{f}_n \end{bmatrix}^T \quad \text{and} \quad \mathbf{f} = \begin{bmatrix} f_1 & f_2 & \dots & f_m \end{bmatrix}^T$$

Here, k_i is the i -th wavenumber, \hat{f}_i is the value of Fourier coefficient associated i -th wavenumber where $i = 0, 1, 2, \dots, n-1$. f_j is the value of function at point x_j where $j = 0, 1, 2, \dots, m-1$. Here, $x_j \in [-0.5, 0.5)$ for all j . The system of linear equations can be written as

$$A\hat{\mathbf{f}} = \mathbf{f} \quad (2.2)$$

We can see that equation (2.2) is analogous to equation (2.1) representing a system of linear equations. Matrix A is called the DFT matrix. The solution of this system, $\hat{\mathbf{f}}$ is also the Fourier transform and the corresponding Fourier series can be written as follows:

$$f_j = \sum_{i=0}^N \hat{f}_i e^{2\pi i k_i x_j} \quad \text{for all } j$$

Thus we have interpolated the given function into a sum of orthogonal basis functions, and calculated the coefficients of individual terms by solving a system of linear equations. This approach has been used in the development of NFFT library[1] to calculate Fourier transform of non function f_j sampled over m non-equispaced data points x_j . We see that $A \in \mathbb{C}^{m \times n}$, $\hat{\mathbf{f}} \in \mathbb{C}^{n \times 1}$ and $\mathbf{f} \in \mathbb{C}^{m \times 1}$. Thus A is a $m \times n$ matrix. The system of equations is well-determined if $m = n$, over-determined if $m > n$ and under-determined if $n > m$.

In practice, the values of number of equations, m and the number of unknowns, n , are large, at about $\mathcal{O}(10^2) - \mathcal{O}(10^4)$. This makes elimination of variables and matrix inversion prohibitively expensive. Thus, NFFT library uses an adapted version of conjugate gradient method to solve the system, which will be explained in the later section.

2.2 Iterative Method for Linear Systems

Solutions to systems of linear equations can be obtained by two kinds of methods, viz. direct methods and iterative methods[31]. Direct methods (e.g. Gaussian elimination) give an exact answer in absence of rounding errors. Direct methods are easier and computationally efficient for smaller systems, but they become prohibitively expensive for large systems and the computational complexity of direct methods increases rapidly with increasing number of equations. Iterative methods (e.g. Gauss Seidel, Krylov Subspace Methods etc.), on the other hand, are computationally more efficient for systems with a large number of equations [31]. Iterative methods begins with an initial guess of the solution and iterates it until a solution with acceptable accuracy is reached. The convergence

and applicability of iterative methods is expected to be ascertained before they are applied to a particular application[32].

2.2.1 Conjugate Gradient

Conjugate gradient (CG) method is a Krylov subspace iterative method to solve linear systems[33, Page 288]. It uses an adaptation of conjugate directions method, where the search directions are defined by residual of the previous iteration. CG gives an exact solution to the linear system provided that the arithmetic is exact. CG requires the coefficient matrix A to be positive definite, i.e., for any vector x , $x^T Ax > 0$; as well as symmetric. CG uses the fact that residual at any point is the direction of steepest descend of the error, and uses the residual at a given iteration $x^{(k)}$ to minimize the error and calculates the next iteration $x^{(k+1)}$. Iterations are continued until a predetermined acceptable level of error is reached. The requirement of symmetric coefficient matrix A means that CG can only be used when the system is well determined i.e. the coefficient matrix A is a square matrix.

2.2.2 Normal Equations

Since CG requires the coefficient matrix of a linear system to be symmetric and positive definite, it can't be applied to non square coefficient matrices. In order to modify the non square matrices, we multiply the system by the conjugate transpose of its coefficient matrix to get the following system.[34]

$$A^H A \hat{\mathbf{f}} = A^H \mathbf{f} \quad (2.3)$$

The matrix $A^H A$ is a positive definite and symmetric matrix. CG can then be applied to this system.

Linear system (2.2) can be converted to a system of normal equations in one more way, by factorizing the unknown vector.

$$AA^H \mathbf{y} = \mathbf{f} \quad \text{where} \quad \hat{\mathbf{f}} = A^H \mathbf{y} \quad (2.4)$$

Matrix AA^H is positive definite and symmetric and can be solved by conjugate gradient method. Equations (2.3) and (2.4) are called normal equations of (2.2). The solution of these equations by CG is then called conjugate gradient on Normal equations.

2.2.3 Conjugate Gradient on Normal Equations

The normal equations can be solved by conjugate gradient. There are two approaches for solving normal equations called CGNR and CGNE[34] based on the quantity that is minimized. CGNR stands for “Conjugate Gradient on Normal equations with residual minimization”. CGNR and CGNE are two sub-types of conjugate gradient methods on normal equations. CGNR stands for Conjugate gradient on Normal equations with Residual minimization and CGNE stands for Conjugate gradient on Normal equations with Error minimization.

CGNR

For over-determined systems, there is no solution and thus the Fourier coefficients vector $\hat{\mathbf{f}}$ can only be approximated up-to a certain residual.

$$\|\mathbf{f} - A\hat{\mathbf{f}}\| \rightarrow \min$$

Thus, the NFFT library uses CGNR[30] which minimizes the residual is the preferred choice for over-determined case.

CGNE

CGNE is used in case of under-determined systems which are consistent. It minimizes the error in solution with each iteration.

$$\|\hat{\mathbf{f}}_{TS} - \hat{\mathbf{f}}^k\| \rightarrow \min$$

Here, $\hat{\mathbf{f}}_{TS}$ is the true solution and $\hat{\mathbf{f}}^k$ is the solution after k^{th} iteration. Algebraically, true solution can be obtained as:

$$\hat{\mathbf{f}}_{TS} = A^+ \mathbf{f}$$

As the error is minimized with each iteration, the iterative solution comes closer and closer to the true solution and iterations are stopped at a predetermined level of acceptable error. Adaptation of CGNE for NFFT can be summarized[35] as follows:

Algorithm 2: CGNE

Input:

$$m, n \in \mathbb{N}, \hat{\mathbf{f}}_0 \in \mathbb{C}^{n \times 1}$$

$$x_j \in [-0.5, 0.5) \text{ for } j = 0, 1, 2, \dots, m-1$$

$$\mathbf{f} \in \mathbb{C}^{m \times 1}$$

$$w_j \text{ for } j = 0, 1, 2, \dots, m-1$$

$$W = \text{diag}(w_j)$$

$$\mathbf{r}_0 = \mathbf{f} - A\hat{\mathbf{f}}_0$$

$$\mathbf{p}_0 = A^H \mathbf{r}_0$$

for $l=0, 1, 2, \dots$ do

$$\alpha_l = \mathbf{r}_l^H \mathbf{r}_l / \mathbf{p}_l^H W \mathbf{p}_l$$

$$\hat{\mathbf{f}}_{l+1} = \hat{\mathbf{f}}_l + \alpha_l W \mathbf{p}_l$$

$$\mathbf{r}_{l+1} = \mathbf{r}_l - \alpha_l A W \mathbf{p}_l$$

$$\beta_l = \mathbf{r}_{l+1}^H \mathbf{r}_{l+1} / \mathbf{r}_l^H \mathbf{r}_l$$

$$\mathbf{p}_{l+1} = \beta_l \mathbf{p}_l + A^H \mathbf{r}_{l+1}$$

end for

Output: vector of Fourier coefficients $\hat{\mathbf{f}}$

Above algorithm solves the system (2.4), while applying weights w_j and mathematically, it can be represented as follows:

$$A W A^H \mathbf{y} = \mathbf{f} \quad \text{where} \quad W A^H \mathbf{y} = \hat{\mathbf{f}}$$

Kunis and Potts [35] have estimated the cost of above algorithm as $n \log n + m$. CGNE has been used to get the NFFT when $n > m$, i.e., when the system (2.2) is under-determined.

2.3 Minimum Norm Solution

As discussed in section 2.1.1, under-determined systems (referred to as under-constrained systems by some others) have an infinite number of solutions and a specific solution needs to be selected.

For well-determined consistent systems similar to (2.2), the solution is given by:

$$\hat{\mathbf{f}} = A^{-1} \mathbf{f}$$

Here, A^{-1} is the inverse to the coefficient matrix A . In case of under-determined or over-determined systems, the iterative numerical methods try to take their solution closer and closer to the pseudo-inverse solution $\hat{\mathbf{f}}_*$.

$$\hat{\mathbf{f}}_* = A^+ \mathbf{f}$$

Hearon (1968)[36] has proven that for an over-determined system, the pseudo-inverse solution $\hat{\mathbf{f}}_*$ is same as the least squares solution that minimizes the euclidean norm (also called as $L2$ norm) of the residual as discussed previously in CGNR.

For an under-determined system, which is of our interest, there are infinite number of solutions and the pseudo-inverse solution $\hat{\mathbf{f}}_*$ is the one with minimum euclidean norm[36]. If the set of all the solutions to (2.2) is denoted as $F = \hat{\mathbf{f}}_1, \hat{\mathbf{f}}_2, \dots, \hat{\mathbf{f}}_\infty$, then $\hat{\mathbf{f}}_* \in F$ and

$$\|\hat{\mathbf{f}}_*\| \leq \|\hat{\mathbf{f}}_i\| \text{ for every } i \in \mathbb{N}$$

Vectors in set F can be given by a vector sum of the minimum norm solution (or pseudo-inverse solution) and a vector \mathbf{g} in the null space of coefficient matrix A .

$$\hat{\mathbf{f}}_i = \hat{\mathbf{f}}_* + \mathbf{g} \text{ where } \mathbf{g} \in \mathcal{N}(A)$$

If $\hat{\mathbf{f}}_*$ is a solution to the system (2.2) and $\mathbf{g} \in \mathcal{N}(A)$, then

$$\begin{aligned} A\hat{\mathbf{f}}_i &= A(\hat{\mathbf{f}}_* + \mathbf{g}) \\ &= A\hat{\mathbf{f}}_* + A\mathbf{g} \\ &= A\hat{\mathbf{f}}_* \quad \text{since } \mathbf{g} \in \mathcal{N}(A) \\ A\hat{\mathbf{f}}_i &= \mathbf{f} \end{aligned}$$

This proves that $\hat{\mathbf{f}}_i$ satisfies (2.2). Geometrically, this can be described in 2 dimensions as in Fig. 2.1.

The minimum norm solution $\hat{\mathbf{f}}_*$ obtained by methods discussed above tends to spread the energy

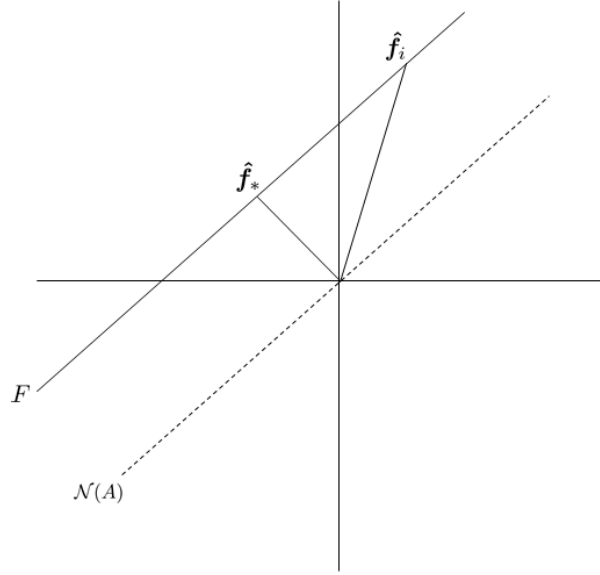


Figure 2.1: Graphical illustration of minimum-norm solution

along multiple Fourier coefficients, as discussed by Gorodnitsky and Rao (1997)[2]. The Fourier transform \hat{f} , on the other hand, has a tendency of rapid decay as discussed in section 1.4. This leads to the vector of Fourier coefficients having few low wavenumbers with large amplitudes and rest high wavenumbers of magnitudes several orders smaller. This indicates that the Fourier transform is the sparsest solution to the under-determined system of linear equations. Focal Under-determined equation Solver (FOCUSS)[2] has been proposed by Gorodnitsky and Rao (1997) to get the sparse solution to the system of under-determined linear equations. Chapter 3 discusses the FOCUSS algorithm, its implementation to NFFT discussed above and the test cases including application to the simulation of Taylor Green vortex in 3 dimensions.

CHAPTER 3

FOCUSS ALGORITHM AND TESTING

NFFT calculates the Fourier transform by solving a system of linear equation where the coefficient matrix is also called as DFT matrix. Our interest is in solving the under-determined systems of equation, where the number of unknowns (Fourier coefficients) is greater than the number of variables (discrete function values). As discussed in section 2.3, an under-determined system has infinite number of solutions, and solving it gives the solution with the minimum $L2$ norm. The Fourier transform needed is the sparsest solution as described previously.

The idea of computing the discrete Fourier transform where the number of function values is lesser than the number of Fourier coefficients wanted by iterative weighted norm solution was put forward by Cabrera and Parks[37]. The idea was further developed and analysed by Gorodnitsky and Rao[2] which led to development of Focal Under-determined equation solver (FOCUSS) algorithm.

Section 3.1 and 3.2 respectively discuss the limitations of the minimum norm solution pertaining to computation of Fourier transform and how FOCUSS algorithm can be used to overcome them. Section 3.3 discusses the characteristics of Fourier series in turbulent flows. Section 3.4 discusses the application of FOCUSS algorithm to a fluid dynamics problem with known solution.

3.1 NFFT with Minimum Norm Solution

This algorithm solves the problem of computing n Fourier coefficients of a function known at m discrete points where $n > m$. The computation is done by solving the under-determined system of linear equations in n unknowns and m equations. Minimum norm is the additional required constraint in order to solve the solution. Formulation of the system is similar to eq. (2.2) where $\hat{\mathbf{f}} \in \mathbb{C}^{n \times 1}$ is the vector of Fourier coefficients, $\mathbf{f} \in \mathbb{C}^{m \times 1}$ is the vector of discrete function values sampled at non-equispaced nodes and $A \in \mathbb{C}^{m \times n}$ is the DFT matrix. Moreover, the system has infinite number of solutions as discussed previously.

As discussed in [37, 2], the direct solution to (2.2) gives the minimum norm solution $\hat{\mathbf{f}}_*$ such that $\hat{\mathbf{f}}_*$ has the $L2$ norm less than that of any other solution to system (2.2). The minimum norm

solution is not the accurate estimation of the Fourier transform, as discussed earlier and can be visualized from Fig. 3.1. The figure shows Fourier transform of a sinewave. The values of Fourier coefficients should be $\mp 0.5j$ for wavenumbers of ± 1 and zero elsewhere. The minimum norm solution, as shown in fig 3.1(a), tends to distribute the energy among all the wavenumbers while reducing the $L2$ norm. For taking the spectral derivative, each element in the vector of Fourier coefficients $\hat{\mathbf{f}}$ is multiplied by the corresponding element of the vector of wavenumbers \mathbf{k} and j , where $j = \sqrt{(-1)^k}$. As can be seen from fig. 3.1(a), the Fourier coefficients for higher wavenumbers are not close to zero when FOCUSS is not implemented, thus giving higher values of amplitudes for high wavenumbers. This is reflected by the spectral derivative in fig. 3.1(b), which clearly shows the presence of high wavenumber components with smaller amplitudes on top of a cosine profile, which is the theoretical derivative of the sine function considered here.

Moreover, the FFT of a sinewave is expected to be conjugate symmetric about the zeroth wavenumber, as the sinewave is a real function.[38]. It can be observed from Fig. 3.1(a) that the Fourier transform does not show a perfect conjugate symmetry.

On the other hand, when the FOCUSS algorithm is used (fig. 3.1(c)), the Fourier transform value shows the amplitudes for wavenumbers ± 1 as 0.5 and rest of the wavenumbers are closer to zero, as compared to the case without FOCUSS. Also, the Fourier transform of a sinewave is expected to be conjugate symmetric about the zeroth wavenumber as it is a real function. Thus it can be expected that the absolute value of the Fourier transform plotted below is symmetric about $k = 0$. The symmetry can be observed in fig. 3.1(c) when the FOCUSS algorithm is applied. When spectral derivative is taken using the FFT coefficients obtained from NFFT with FOCUSS, we get a perfect cosine function which is expected from the theory.

3.2 FOCUSS Algorithm

The minimum norm solution in fig. 3.1(a) can be refined iteratively to obtain the correct solution in fig. 3.1(c). $L2$ norm of the vector $\hat{\mathbf{f}}$ can be given as follows:

$$\|\hat{\mathbf{f}}\| = \sqrt{\hat{f}_{-n/2+1}^2 + \hat{f}_{-n/2+2}^2 + \dots + \hat{f}_0^2 + \dots + \hat{f}_{n/2}^2}$$

As can be seen from the above equation, the norm $\|\hat{\mathbf{f}}\|$ is minimum when the energy tends to be more equally distributed in all the wavenumbers. In case of a sinewave, the solution tends to increase the higher wavenumber coefficients, i.e., values of $\hat{f}_{\pm i}$ where i is higher. This means that to

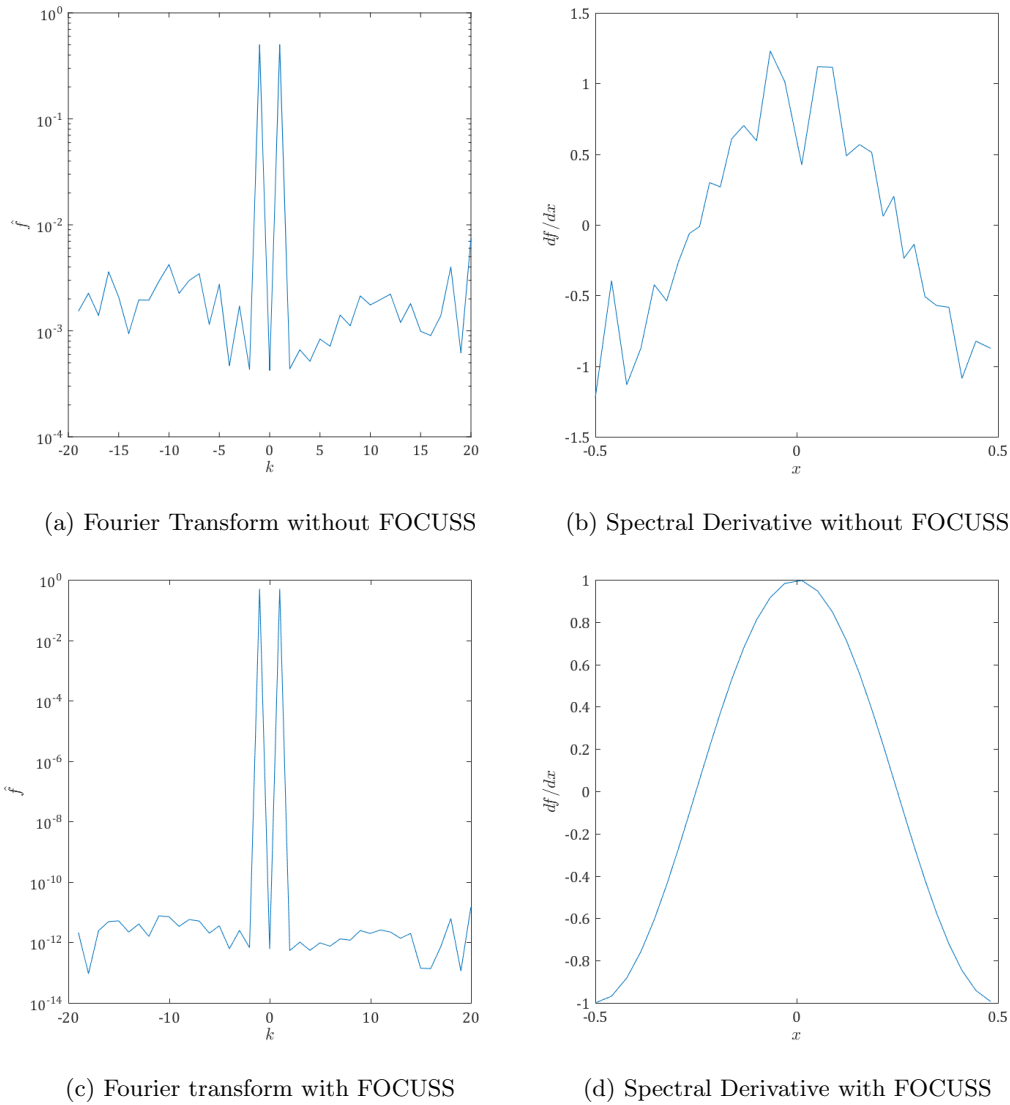


Figure 3.1: (a,c): Absolute values of the Fourier coefficients against the wavenumbers for function $\sin(2\pi x)$ and $x \in [-0.5, 0.5]$; (b,d): Spectral derivative of the function with respect to $2\pi x$

achieve the minimum norm, the energy needs to be distributed more equally in all the coefficients of a vector. This can also be understood from the concept of euclidean norm, which is the magnitude of the vector $\hat{\mathbf{f}}$.

The Fourier transform required by us has a behaviour opposite to this. It has a low wavenumbers with high amplitude and the amplitude rapidly diminishes as wavenumber increases. This will be discussed in more detail in section 3.3. This can be achieved if we calculate the weighted norm, as follows:

$$\|\hat{\mathbf{f}}\| = \sqrt{\sum \hat{w}_i \hat{f}_i^2} \text{ for } i \in (-n/2, n/2] \text{ and } i \in \mathbb{Z} \quad (3.1)$$

Here, the values of \hat{w}_i can be selected to be large for larger $|i|$ to increase the contribution of higher wavenumber Fourier coefficients to the calculation of norm. This will increase the contribution of higher wavenumber coefficients to calculation of norm, and will force the value of higher wavenumber coefficients to reduce in order to achieve the minimum norm. Eq. (3.1) can be rewritten as follows by replacing \hat{w}_i by $1/\hat{w}_i$.

$$\|\hat{\mathbf{f}}\| = \sqrt{\sum \frac{\hat{f}_i^2}{\hat{w}_i}} \quad (3.2)$$

It can be seen from fig. 3.1 (a), it can be seen that the value of higher wavenumbers is smaller than that of the lower wavenumbers, but still larger than the correct Fourier transform. Thus, we need to increase the contribution of the higher wavenumbers in calculation of norm. This can be achieved by using the minimum norm solution $\hat{\mathbf{f}}_*$ itself as the vector of weights $\hat{\mathbf{w}}$. This will make sure that $1/\hat{w}_i$ is larger for higher wavenumbers and vice versa. This approach has been discussed in more details by Gorodnitsky and Rao[2].

Let \hat{W} be the diagonal matrix with each diagonal element as \hat{w}_i . Considering the weights, the norm to be minimized can be given as $\|\hat{W}^{-1}\hat{\mathbf{f}}\|$. Analytically, this can be achieved as follows for equation (2.2).

$$A\hat{\mathbf{f}} = \mathbf{f}$$

Let

$$\hat{\mathbf{f}} = \hat{W}\mathbf{q}$$

Thus the above equations becomes

$$A\hat{W}\mathbf{q} = \mathbf{f}$$

where $\mathbf{q} = \hat{W}^{-1}\hat{\mathbf{f}}$

$$\mathbf{q} = (A\hat{W})^+ \mathbf{f}$$

This gives a solution where the norm of \mathbf{q} is minimized. Thus,

$$\|\mathbf{q}\| = \|\hat{W}^{-1}\hat{\mathbf{f}}\| \rightarrow \min$$

The initial values of \hat{w}_i can be considered to be 1 for all i . It can be seen that the solution with $\hat{w}_i = 1$ for all i is the minimum norm solution, as shown in fig. 3.1(a). Also, by default, the values of \hat{w}_i are taken equal to 1 in the NFFT library used here.

After the first iteration, the minimum norm solution $\hat{\mathbf{f}}_*$ is obtained as discussed earlier. This solution is then used as weights for the next iteration, so that $\hat{\mathbf{w}} = \hat{\mathbf{f}}_*$. The procedure is repeated until the answers of successive iterations no longer vary beyond a predetermined threshold value. This solution is the correct Fourier transform as shown in fig. 3.1(c) and (d).

Algorithm 3 summarizes the procedure of computation of Fourier transform with FOCUS.

Algorithm 3: CGNE with FOCUS

Input:

$m, n \in \mathbb{N}, \hat{\mathbf{f}} \in \mathbb{C}^{n \times 1}$

$x_j \in [-0.5, 0.5)$ for $j = 0, 1, 2, \dots, m-1$

$\hat{\mathbf{f}} \in \mathbb{C}^{m \times 1}$

$\hat{\mathbf{k}}_0 \in \mathbb{C}^{n \times 1}$ where $\hat{\mathbf{k}}_0(i) = 1 \forall i$, this is vector of initial weights

$\hat{\mathbf{k}}_1 \in \mathbb{C}^{n \times 1}$ where $\hat{\mathbf{k}}_1(i) = 0 \forall i$

t , the predetermined threshold for allowable difference between two successive iterations of FOCUS

while $\|\hat{\mathbf{k}}_0 - \hat{\mathbf{k}}_1\| > t$ do

$\hat{W} = \text{diag}(\hat{\mathbf{k}}_0)$

$A_1 = A\hat{W}$

$\mathbf{q} = \hat{W}^{-1}\hat{\mathbf{f}}$, the initial guess for conjugate gradient iterations

do conjugate gradient iterations on $A_1\mathbf{q} = \hat{\mathbf{f}}$ to find \mathbf{q} such that $\|\mathbf{q}\| \rightarrow \min$

$\hat{\mathbf{f}} = \hat{W}\mathbf{q}$

$\mathbf{k}_1 = \mathbf{k}_0$

$\mathbf{k}_0 = |\hat{\mathbf{f}}|$

end while

Output:

vector of Fourier coefficients $\hat{\mathbf{f}}$

Here, the matrix A is the DFT matrix as discussed in section 2.2. Also, the weights for norm calculation, \hat{w}_i and their matrix form \hat{W} are entirely independent of the weights to compensate for clusters in the sampling points, denoted by w_j and W in chapter 2. It should be noted that although the algorithm 3 starts with initial weights as a vector of all ones, we may use some other values in order to reduce the number of iterations required and speed up the algorithm. This has been illustrated in the next section.

The example illustrated in the previous section (fig 3.1) is the Fourier transform of a sinewave which has energy in only one wavenumber. Fig. 3.2 below shows the implementation of FOCUSS algorithm for a Gaussian function. The Fourier transform of a Gaussian has energy in all its wavenumbers. It can be observed that Gaussian shows similar behaviour as a sinewave with and without FOCUSS.

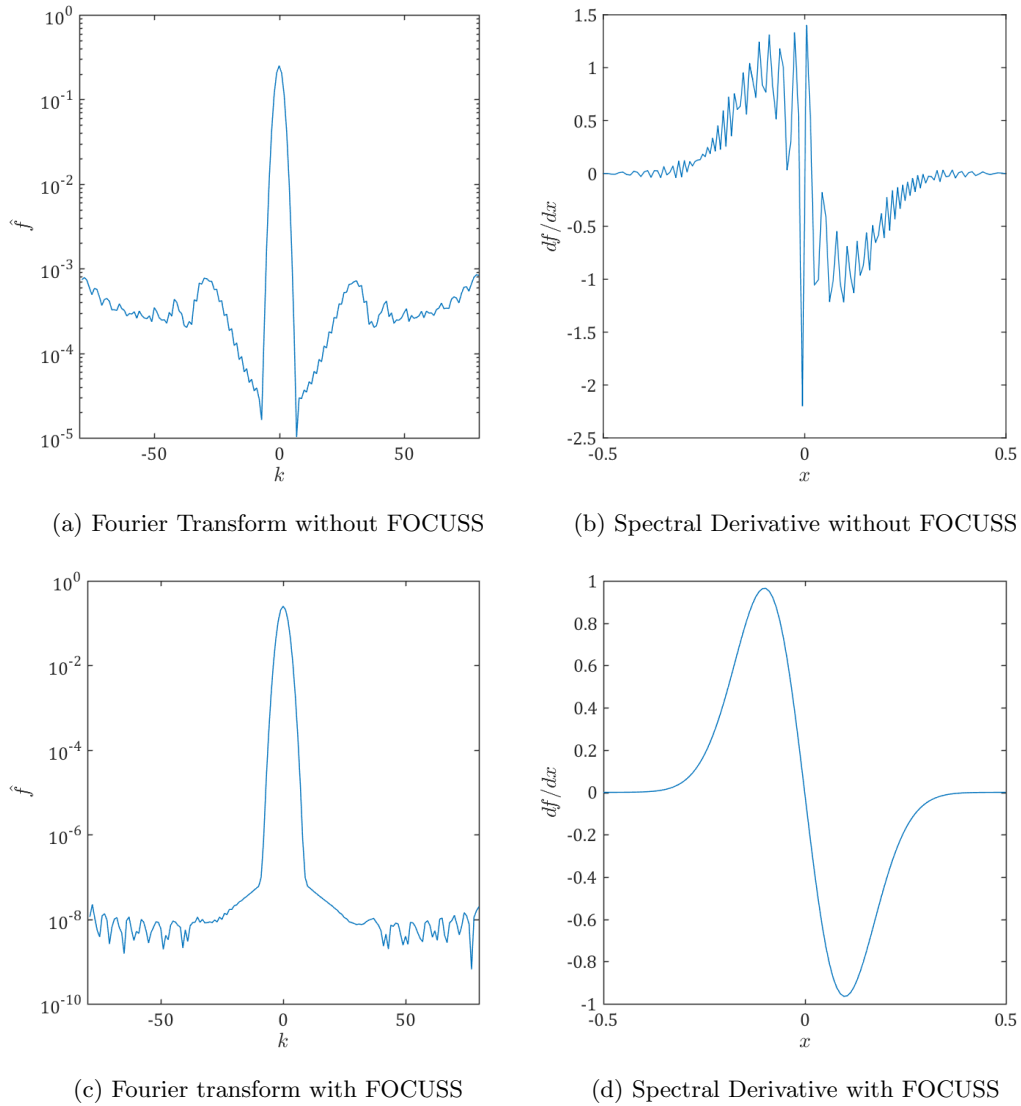


Figure 3.2: (a,c): Absolute values of the Fourier coefficients against the wavenumbers for the Gaussian $\exp(-50x^2)$ and $x \in [-0.5, 0.5]$; (b,d): Spectral derivative of the function with respect to $2\pi x$

In the simulations this algorithm is expected to be applied, e.g. turbulent wake, the dynamically important lengthscales are situated in the center of the domain. The example in Fig. 3.2 uses a Gaussian sampled on a non-equispaced grid, similar to what could be used for the actual simula-

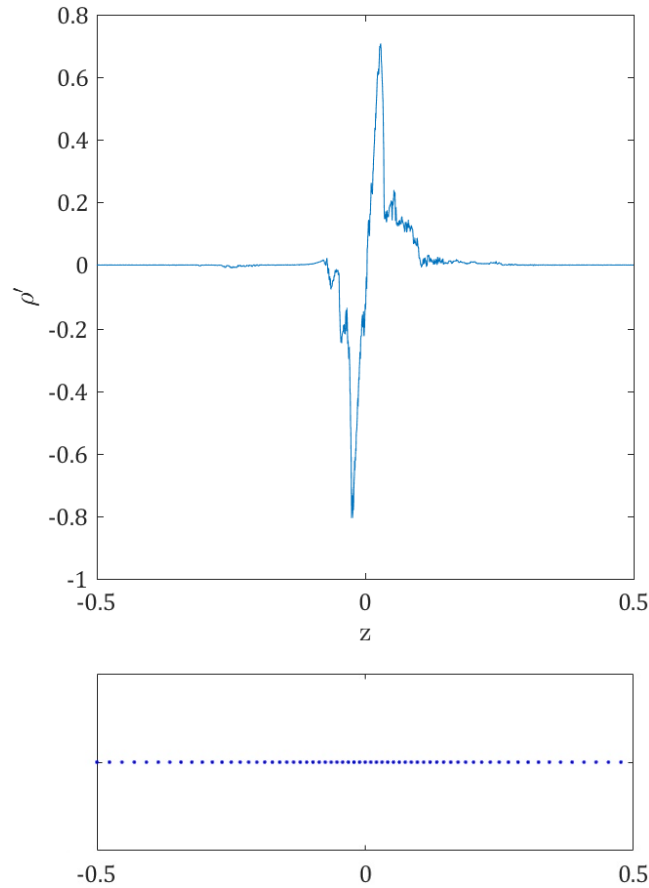


Figure 3.3: The grid and a typical function observed in turbulence

tions. Fig. 3.3 shows the grid used and a line on a slice of an actual turbulent wake. Mathematical properties of the grid and the details about turbulent wake will be discussed subsequently.

3.3 Introduction to Turbulence and its Numerical Simulation

Turbulence can be thought of a superposition of various eddies of different sizes, or length scales[26]. The larger ones are called integral length scales. The maximum length scale is constrained by the size of the flow domain. The eddies corresponding to these contain most of the energy of the flow. They have large flow velocity fluctuation which is low in frequency. The velocity of the eddies corresponding to the integral length scales is comparable to the characteristic velocity of the flow. The range of integral length scales is called as energy containing range.

Often in three dimensional turbulence, eddies produce even smaller eddies until molecular viscosity is effective in dissipating the kinetic energy. The smallest length scales are called as Kolmogorov

length scales[39]. The range of dissipative eddies is called as dissipation range. The eddies in dissipation range have low velocity and high frequency.

In a Fourier series, the high wavenumbers represent high frequency components in the interpolated function and low wavenumbers represent the low frequency components. It can thus be concluded that the low wavenumbers will represent the eddies corresponding to integral lengthscale and higher wavenumbers will represent the eddies corresponding to Kolmogorov lengthscale (c. f. Pope Section 6.5.3[26]).

Thus, in a Fourier series representing a turbulent flow velocity field, the low wavenumbers have high amplitude, which decays exponentially in the dissipation range. Thus in Fourier transforms of turbulent flows, the energy is usually biased to lower values of wavenumbers k . As it is observed that very few of the total wavenumbers account for most of the energy in a turbulent flow, we propose to apply FOCUSS with NFFT to obtain Fourier transforms in the simulation of turbulent flows.

There are mainly three approaches of numerical simulation of turbulent flows, of which Direct Numerical Simulation (DNS) is under consideration in this thesis. DNS is the most accurate approach where all the dynamically relevant length scales and time scales of the flow are resolved[40]. The first DNS was performed by Orszag and Patterson (1972)[41] on a Reynolds number (based on Taylor Microscale) of 35. This work demonstrated the use of Fourier spectral methods in turbulence simulation. Periodic boundary conditions were imposed in all the dimensions in this simulation. Rogallo (1981)[42] applied DNS to homogeneous turbulence and examined the effects of mean shear, irrotational strain and rotation. DNS of turbulent flows in plane channel and curved channel were performed in 1987 by Kim *et. al.*[43] and Moser and Moin[44] respectively. In DNS, Fourier spectral method is used in the directions of homogeneity and finite difference or compact finite difference schemes in other directions. In recent years, the major improvement in DNS is the expansion of grid size, due to increase in the computing power available. As of today, one of the largest DNS of homogeneous turbulence contain roughly 0.5 trillion gridpoints[45].

3.4 Implementation on a test case

Algorithm 3 was applied to solve 3D Navier Stokes equations using DNS with spectral method for spatial differentiation. The equation was solved with Taylor Green initial conditions[46]. Three dimensional Navier Stokes equation has an asymptotic solution for early time in this case. Pressure projection method developed by A. J. Chorin[47] was used to decouple the velocity and pressure field, combined with third order Adams bashforth method for timestepping.

The Navier Stokes equation in its convective form for incompressible flow in absence of any external force is given by:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho} \nabla P + \nu \nabla^2 \mathbf{u}$$

Here, \mathbf{u} is the velocity field, P is the pressure field, ρ is the density, ν is the viscosity and t is the time. For x dimension, the above equation can be given as:

$$\frac{\partial u_x}{\partial t} + u_x \frac{\partial u_x}{\partial x} + u_y \frac{\partial u_x}{\partial y} + u_z \frac{\partial u_x}{\partial z} = -\frac{1}{\rho} \left(\frac{\partial P}{\partial x} + \frac{\partial P}{\partial y} + \frac{\partial P}{\partial z} \right) + \nu \left(\frac{\partial^2 u_x}{\partial x^2} + \frac{\partial^2 u_x}{\partial y^2} + \frac{\partial^2 u_x}{\partial z^2} \right) \quad (3.3)$$

Here, u_x, u_y, u_z are the velocities in x, y and z directions, respectively. In a similar manner, the equation can be written for y and z dimensions. The Navier Stokes equation needs to be solved in such a way that it satisfies the continuity equation at any time. The continuity equation for incompressible fluids is as follows:

$$\nabla \cdot \mathbf{u} = 0$$

or

$$\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} + \frac{\partial u_z}{\partial z} = 0$$

The simulation of Taylor Green vortex assumes a periodic boundary conditions in space, and thus can be solved by spectral method for spatial domain. Initial conditions as given by Taylor and Green [46] are:

$$\begin{aligned} u_x &= \cos(2\pi x) \sin(2\pi y) \sin(2\pi z) \\ u_y &= -\sin(2\pi x) \cos(2\pi y) \sin(2\pi z) \\ u_z &= 0 \end{aligned}$$

It can be verified that the initial conditions satisfy the continuity equation. For the test case, a grid of 64^3 points was used. The grid spacing in y and z directions was taken to be uniform while that in x direction was taken to be non uniform. The Fourier transform was done to give 32 Fourier coefficients in both y and z directions and 40 Fourier coefficients in x direction. Fourier coefficients of partial spatial derivative in any direction were calculated by multiplying the Fourier transform of the original field by the wavenumbers for that direction and $1i$ where $i = \sqrt{-1}$. The time stepping was done in Fourier space, i.e., the values of Fourier coefficients of velocity field $\hat{\mathbf{u}}^{n+1}$ at time t^{n+1} were calculated while knowing the value of Fourier coefficients at earlier timestep. The actual velocity

field at time t^{n+1} was calculated by taking the inverse fast Fourier transform of $\hat{\mathbf{u}}^{n+1}$.

It was observed that a significantly large portion of the total time required to run the simulation was required for taking the NFFT with FOCUSS algorithm. To reduce the time requirement, the solution $\hat{\mathbf{u}}^{n-1}$ of $(n-1)^{\text{th}}$ timestep was stored in memory and used as weights to minimize the norm while calculating the NFFT of the velocity field at time t^n , i.e., while taking the Fourier transform $\hat{\mathbf{u}}^n$ from \mathbf{u}^n . Computing Fourier transform in each timestep was necessary due to the non-linear term $u_i \frac{\partial u_j}{\partial x_i}$. The non linear term was calculated by computing the partial derivative $\partial u_j / \partial x_i$ by spectral method, and multiplying it with u_i . Fourier transform of the product $u_i \partial u_j / \partial x_i$ was again calculated using the NFFT algorithm, where the initial weights were considered to be the Fourier transform from the previous timestep. This made sure that the initial weights are sufficiently close to the actual solution, as the timestep was small, and reduced the number of iterations of algorithm 3 required to give the true solution, thereby reducing the time required. The reduction in time requirement has been summarized in Table 3.1.

Initialization weights	CPU Time (sec)
All weights equal one	160
Previous timestep solution	58.74

Table 3.1: CPU time requirement for Taylor Green vortex simulation for 32^3 grid-points in real space, $32^2 \times 40$ Fourier coefficients, carried out for 50 timesteps.

Taylor and Green [46] have analytically calculated the mean dissipation rate of energy of a viscous fluid, which gives us a basis to compare our solution. The dissipation rate of energy is given as:

$$\overline{W} = \mu \overline{\omega^2}$$

Here, \overline{W} denotes the mean of energy dissipation rate over entire volume and $\overline{\omega^2}$ represents the mean of the square of vorticity of the velocity field over entire volume and μ represents the dynamic viscosity. The vorticity is given by:

$$\boldsymbol{\omega} = \nabla \times \mathbf{u}$$

The rate of dissipation of energy was calculated for every timestep and plotted against the time in Fig. 3.4. The numerical simulation was found to be in excellent agreement with the analytical solution initially. As the time goes on increasing, the analytical solution is not accurate anymore [46]. Taylor and Green have also given the approximate time upto which the analytical solution is not reasonably accurate. W' and T represent the dimensionless values of energy dissipation rate and time respectively. According to Taylor and Green [46], the analytical solution for $\text{Re} = 20$ is

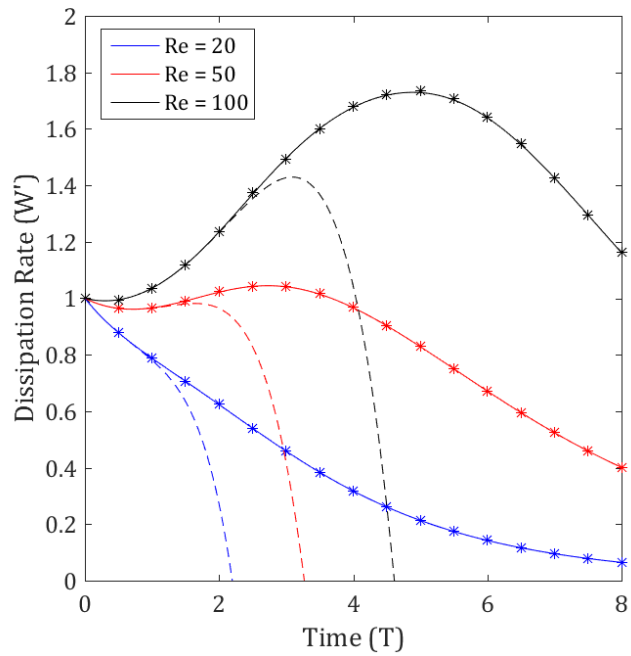


Figure 3.4: The Solution of 3D Navier Stokes equation with Taylor Green initial conditions. The solid lines give the results of DNS using NFFT. Stars (*) denote the solution obtained independently using an equispaced grid. Dashed lines give the analytical solution. W' and T represent the dimensionless forms of dissipation rate and time, respectively.

accurate upto approximately $T=1.5$, $Re = 50$ upto $T=2$ and that for $Re = 100$ upto $T=2.5$. Our solution matches the analytical solution initially as expected.

Slow computational speed of this approach was observed to be the biggest drawback of this approach when applied to the test case. In Chapter 4, the mathematical aspects related to computational performance of CGNE algorithms used in this approach are discussed.

CHAPTER 4

CONDITIONING OF NFFT

In the initial tests, when NFFT and FOCUSS were applied for DNS, it was observed that the speed of DNS drops down considerably as the simulation proceeds, compared to the conventional equispaced grid approach. The time required to perform DNS with NFFT and FOCUSS was observed to be several times greater than that required to perform DNS on an equispaced grid. It was observed that the slowdown in computational speed is due to the large number of conjugate gradient iterations required for the computation of Fourier transform by NFFT. In this chapter, the condition number and its effect on the convergence of conjugate gradient system and the observations of condition number of DFT matrix in NFFT will be discussed.

4.1 Condition Number

For a matrix A , condition number[48] is defined as

$$\begin{aligned}\text{cond}(A) = \kappa(A) &= \frac{\lambda_{\max}}{\lambda_{\min}} \\ &= \|A\| \cdot \|A^{-1}\|\end{aligned}$$

where λ_{\max} and λ_{\min} are the maximum and the minimum eigenvalues of matrix A . The matrix A is called as “well-conditioned” if it has a low condition number and ill-conditioned if it has a high condition number. For a system of linear equations $Ax = b$, condition number is a measure of how much the error in the constant vector b affects the solution x . If A is ill-conditioned, small errors in b result in large errors in the solution x and vice versa.

Condition number also affects the convergence of conjugate gradient iterations[31]. The error in k^{th} iteration of a conjugate gradient method e^k is related to the error in the initial guess e^0 as follows:

$$e^k \leq 2C^k e^0$$

where

$$\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} = C$$

$\kappa(A)$ is always positive, thus it can be proven that $C < 1$. It can be seen that if C is close to 1, higher is the number of iterations required for convergence and vice versa. If the condition number $\kappa(A)$ is higher, C is close to 1. Thus, systems of linear equations with high condition number coefficient matrices tend to require more iterations for convergence.

In our case, the coefficient matrix $A_{m \times n}$ is not a square matrix, but has more columns than rows, i.e. $n > m$. In this case, we use the conjugate gradient method for normal equations (CGNE) as discussed in Sec. 2.2.3. We solve the following system of equations.

$$A\hat{W}A^H \mathbf{y} = \mathbf{f} \quad \text{where} \quad \hat{W}A^H \mathbf{y} = \hat{\mathbf{f}}$$

Let $A\hat{W}A^H = K$. In this case the convergence of this system depends upon the condition number of K [33].

4.2 Condition Number in NFFT

As NFFT uses conjugate gradient method to solve a system of linear equations, condition number proves to be an important criteria to be investigated. The condition number of matrix K changes with each FOCUSS loop, when new weights \hat{W} are applied. It was observed that the condition number of K increases with each FOCUSS iteration, thereby affecting the number of conjugate gradient iterations required for convergence. The increase in condition number is steep and results in a sharp increase in the number of conjugate gradient iterations required within each FOCUSS algorithm, as shown in Fig. 4.1.

The non-equispaced grid used here is given as follows:

$$\mathbf{x}_n = \tan^{-1} \left[\frac{m}{n} \tan(\mathbf{x}_e) \right] \quad (4.1)$$

where \mathbf{x}_n is the non-equispaced grid and \mathbf{x}_e is the equispaced grid of m points. n is the number of Fourier coefficients required. It can be proven that $\Delta x_{\min} = 1/n$ where Δx_{\min} is the minimum grid-spacing on the non-equispaced grid. This observation is in agreement with the requirement that the entire grid should be resolved on the finest grid-spacing.

Fig. 4.1 indicates how the number of conjugate gradient iterations increases with an increase in

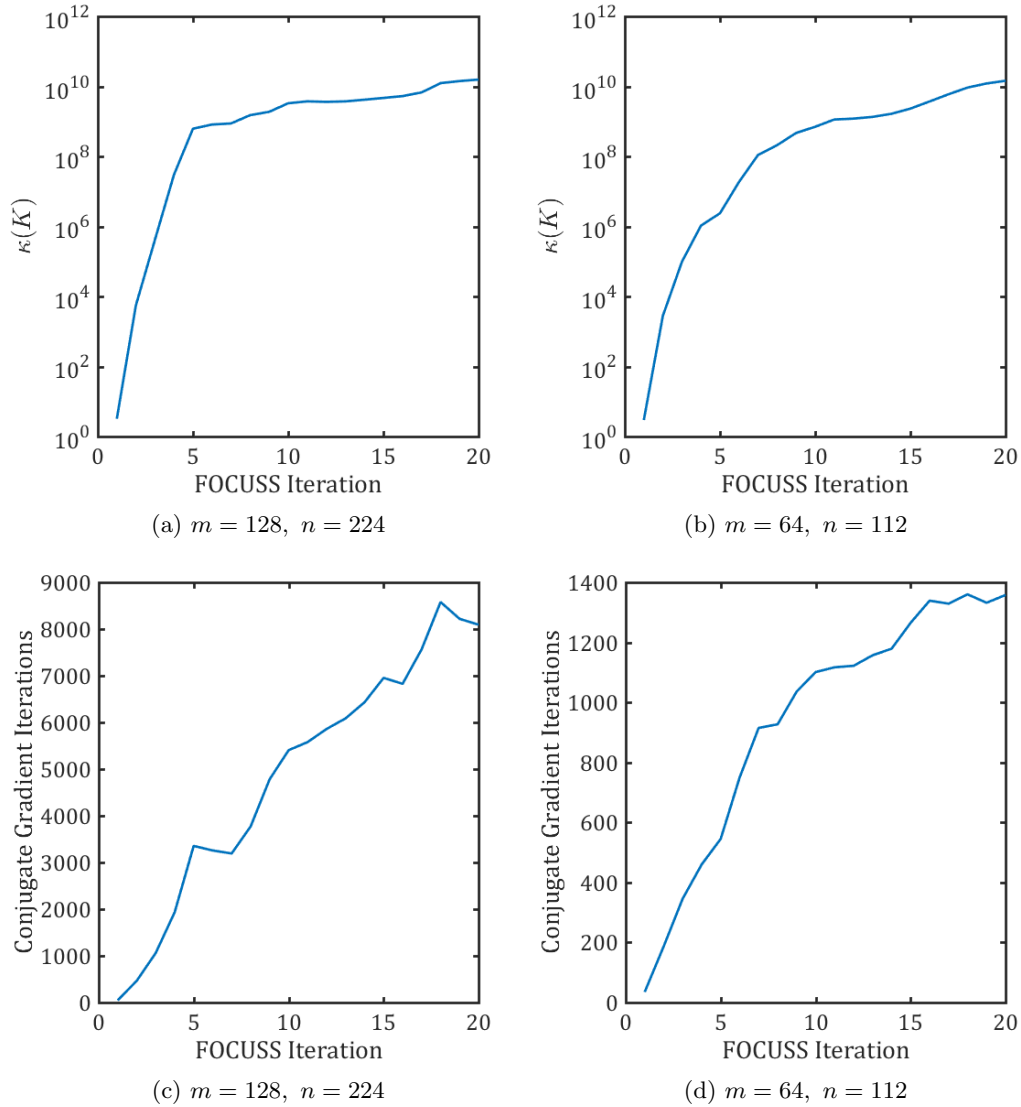


Figure 4.1: (a,b): Condition number of K ; (c,d): Number of conjugate gradient iterations required for convergence for $n/m = 1.75$

condition number with each FOCUSS loop. It can also be seen that the number of conjugate gradient iterations required for $m = 128$ is much larger than that for $m = 64$.

A similar trend can be observed for slightly less perturbed grid ($n/m = 1.5$) as can be seen from Fig. 4.2. However, in this case, it can be seen that both the condition numbers and the number of FOCUSS iterations required is less than those in case of $n/m = 1.75$. It can be observed that a smaller number of FOCUSS loops is required for convergence in case of the less perturbed grid, thereby reducing the overall number of conjugate gradient iterations considerably. This can be related to the fact that the grid used in Fig. 4.2 is closer to an equispaced grid than that in Fig. 4.1. Both Fig. 4.1 and 4.2 indicate the observations related to condition number and conjugate gradient

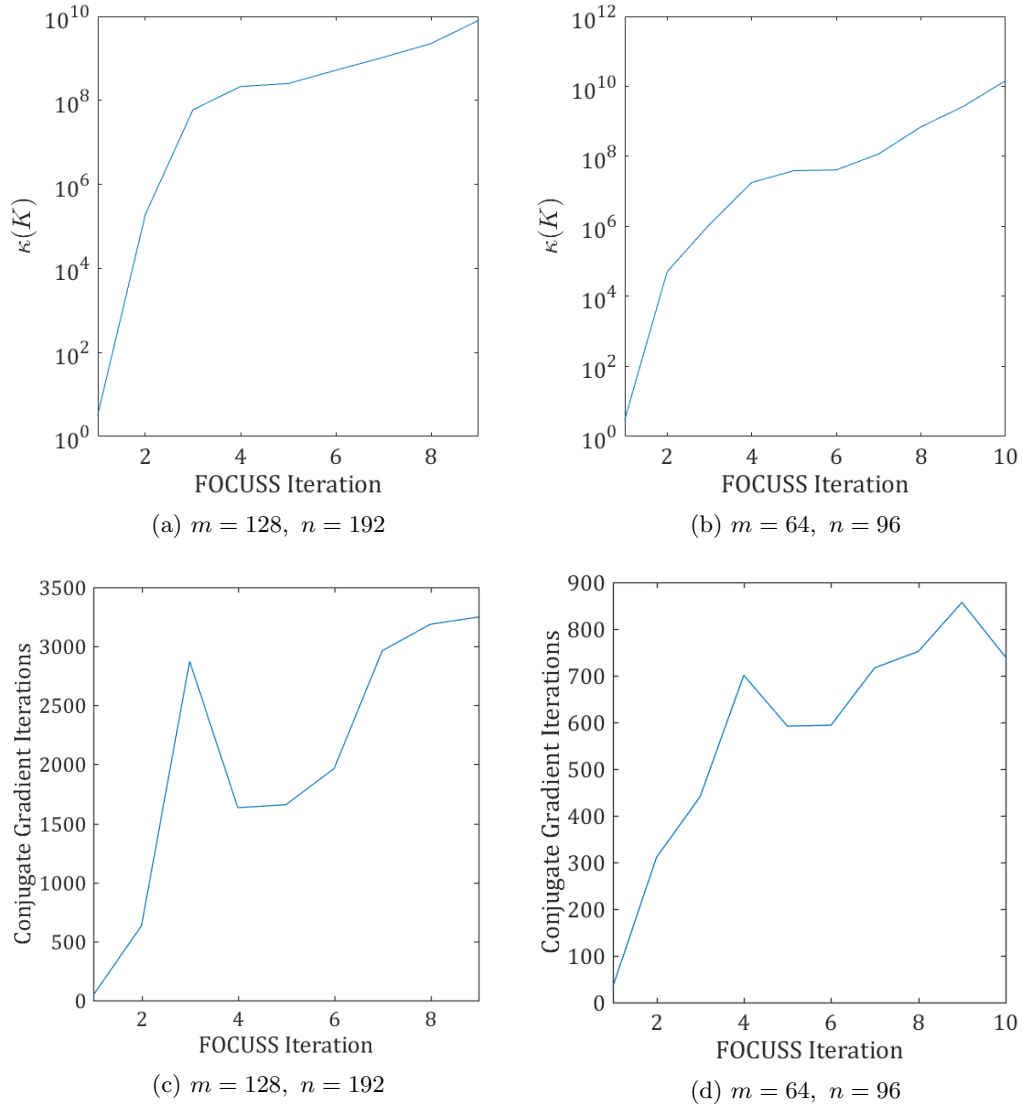


Figure 4.2: (a,b): Condition number of K ; (c,d): Number of conjugate gradient iterations required for convergence for $n/m = 1.5$

iterations while taking the Fourier transform of a Gaussian function as a test case.

4.3 Analysis

The condition number of a matrix is a measure of how far its rows are from being orthogonal. More skewed the rows of the matrix are from being orthogonal, higher is the condition number of the matrix and vice versa.

It can be observed that matrices with pairwise orthogonal rows are easier to solve. Diagonal

matrix is an example whose rows are perfectly orthogonal to each other, and is the easiest matrix to solve. As rows become less orthogonal, the matrix becomes more and more difficult to solve. Intuitively, a matrix with pairwise orthogonal rows (e.g. diagonal matrix) leads to each equation in only one variable, which makes it easier to solve.

It was observed that for the modified coefficient matrix K , the rows are more orthogonal without application of the weights and they become less orthogonal as the weights are applied. This causes the condition number of K to grow. This behaviour has been explained in Fig. 4.3 and 4.4.

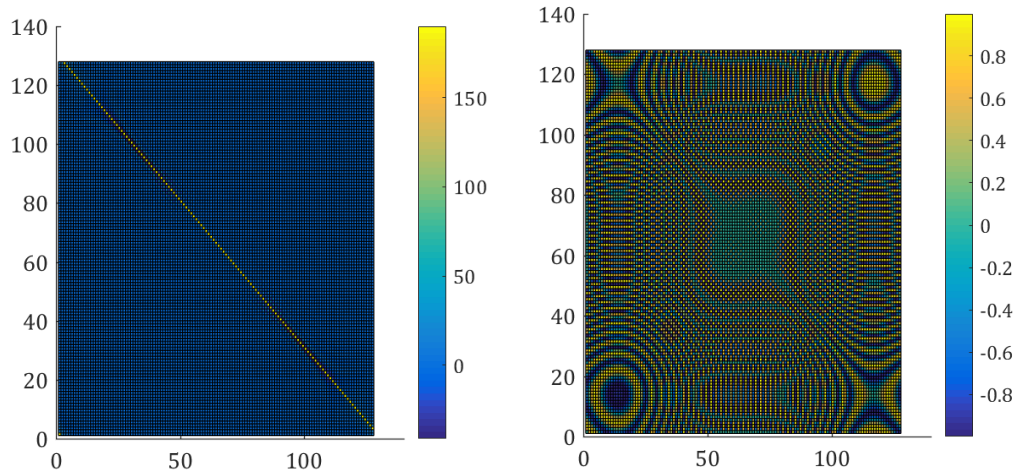


Figure 4.3: Left: Real part, Right: Imaginary part of the modified coefficient matrix K before application of weights. This matrix is independent of the function but depends only on the grid.

As can be seen in Fig. 4.3, the real part of matrix K has its rows perfectly orthogonal to each other. The imaginary part is not orthogonal, but the magnitudes of the imaginary components are small compared to those in the real parts and the orthogonality does not get affected significantly.

Figure 4.4 shows the real and imaginary component of K after weights are applied. It can be seen that the real component dominates in the total magnitude. The rows of real part of K are less orthogonal as compared to Fig. 4.3, resulting in a higher condition number.

Condition number, as discussed earlier, is the ratio of maximum and minimum eigenvalues. The plots of eigenvalue distribution before and after weighting are shown in Fig. 4.5. A matrix is singular if any of its eigenvalues is zero. The weighted coefficient matrix K has a cluster of eigenvalues of magnitudes $\mathcal{O}(10)$ while other cluster of eigenvalues is small, close to zero. Matrices with these characteristics are nearly singular[49], i.e., they are difficult to solve. This property is also evident from the result regarding condition number described in Sec. 4.2.

Fig 4.6 and 4.7 show the modified coefficient matrix K for NFFT of a slice of a turbulent wake,

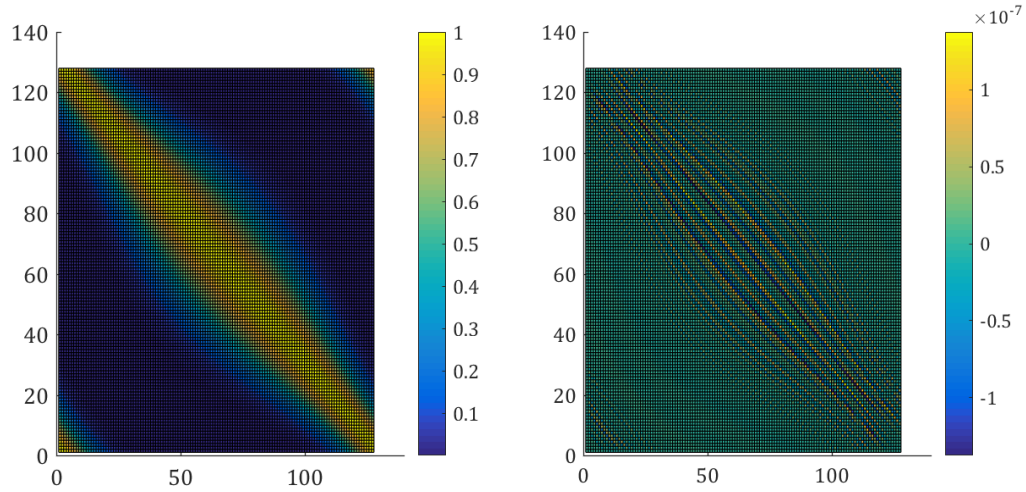


Figure 4.4: Left: Real part, Right: Imaginary part of the modified coefficient matrix K after application of weights in computation of Fourier transform of a Gaussian

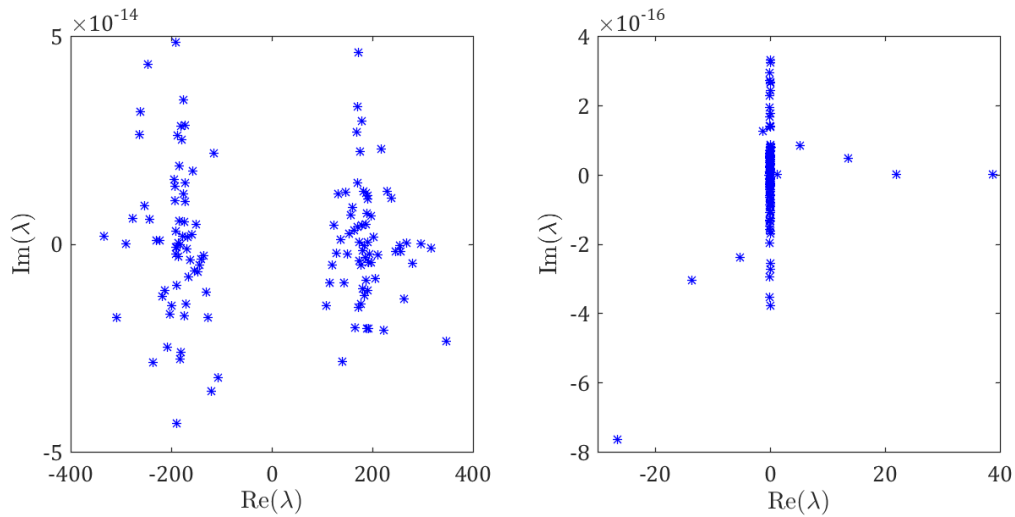


Figure 4.5: Eigenvalue distribution of matrix K ; Left: before application of weights (independent of the function), right: After application of weights for computation of Fourier transform of a Gaussian

sampled on 1024 non-equispaced points for computation of 2048 Fourier coefficients. The matrix K before the application of weights has rows that are more orthogonal to each other and thus has a smaller condition number. After application of weights for one time, as shown in Fig. 4.7, the rows become less orthogonal, in a pattern similar to Fig. 4.4. A similar observation can be made about the eigenvalue distribution in Fig. 4.8 for this case, before and after the application of weights. The smallest eigenvalues, after the application of weights can be observed to move closer to zero, thereby increasing the condition number.

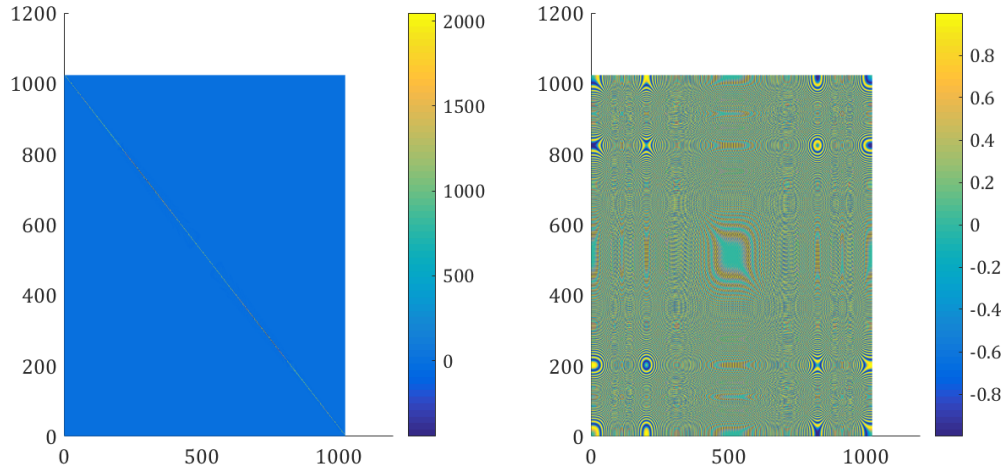


Figure 4.6: Left: Real part, Right: Imaginary part of the modified coefficient matrix for $m = 1024$ non-equispaced gridpoints, $n = 2048$ Fourier coefficients before application of weights

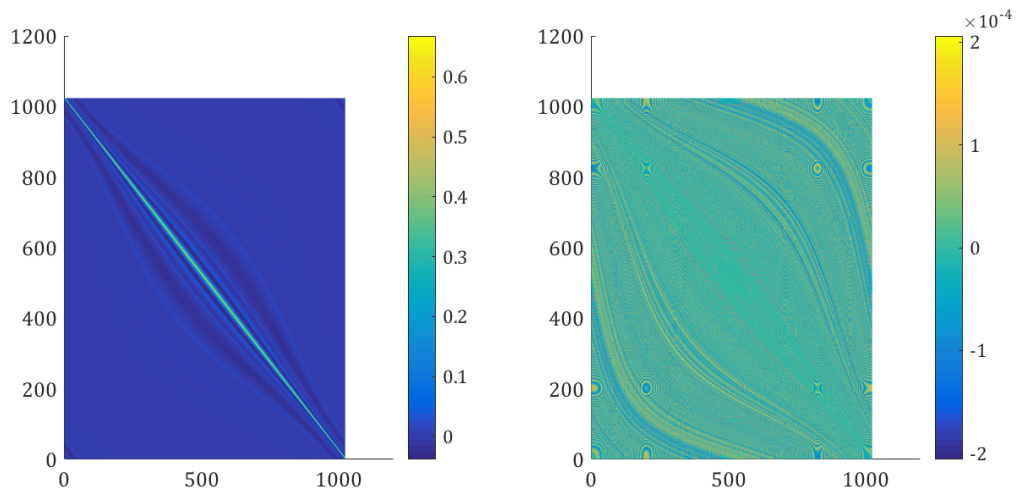


Figure 4.7: Left: Real part, Right: Imaginary part of the modified coefficient matrix for $m = 1024$, $n = 2048$ after application of weights for NFFT of a slice of turbulent wake

4.4 Methods for Improving Speed

The speed of computation of Fourier transform can be potentially improved by primarily by two means, preconditioning and opting for optimum number of Fourier coefficients. Use of preconditioned conjugate gradient is one of the means to achieve this, by reducing the condition number of the modified coefficient matrix K . Benzi (2002)[50] has done a comprehensive survey of preconditioning techniques for the linear systems.

The condition number depends both upon the grid and the number of Fourier coefficients expected, which is another factor affecting computational performance. For a given grid, the number

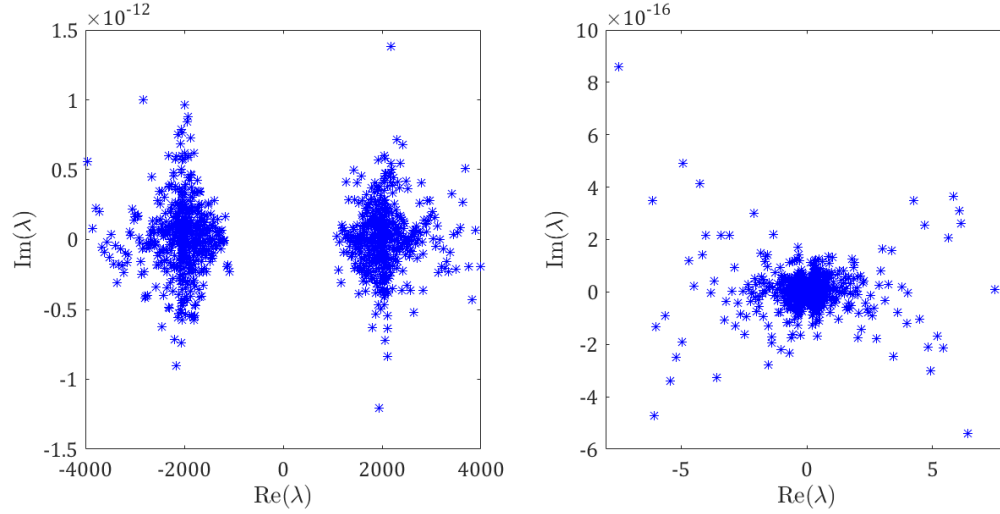


Figure 4.8: Eigenvalue distribution of modified coefficient matrix K for NFFT of slice of turbulent wake before application of weights (Left) and after application of weights for 4 FOCUSS iterations(Right)

of Fourier coefficients can impact the condition number considerably.

The problem can be split into two well-determined or a well-determined and an over-determined problem for lower and higher wavenumber Fourier coefficients. The feasibility of this approach has been discussed in Sec. 4.4.3.

4.4.1 Preconditioning

Preconditioning[50] of a linear system involves multiplying the coefficient matrix K of a linear system by another matrix M , such that the product $M^{-1}K$ is well-conditioned. Preconditioning is of the following three types:

1. Left-side preconditioning
2. Right-side preconditioning
3. Split preconditioning

The left side preconditioning solves the following system.

$$M^{-1}K\hat{f} = f$$

Right-side preconditioning, on the other hand, solves the following system.

$$KM^{-1}\hat{f}_1 = f \quad \text{where} \quad M^{-1}\hat{f}_1 = \hat{f}$$

Split preconditioning employs preconditioners on both the sides.

$$M^{-1}K(M^{-1})^H \hat{\mathbf{f}}_1 = \mathbf{f} \quad \text{where} \quad M^{-1} \hat{\mathbf{f}}_1 = \hat{\mathbf{f}}$$

Computation of the preconditioner matrix needs to be done such that the cost of estimating the entries of preconditioner matrix is less, as well as the condition number of the linear system is reduced. Incomplete Cholesky preconditioner[51] is the most commonly used type. Incomplete Cholesky factorization is the sparse approximation to the Cholesky factorization which is given as follows.

$$A = LL^H$$

Here, L is a lower triangular matrix. L is easy to invert as it is a lower triangular matrix, and is used as a preconditioner. The algorithm for computation of Incomplete Cholesky preconditioner can be found in [52, section 10.3.2]. For NFFT using FOCUSS, the condition number was found to reduce significantly if a preconditioner is used, thereby reducing the number of conjugate gradient iterations required. The preconditioner can be computed either in every FOCUSS loops, or it can be calculated in the first few FOCUSS loops and the same preconditioner can be used throughout.

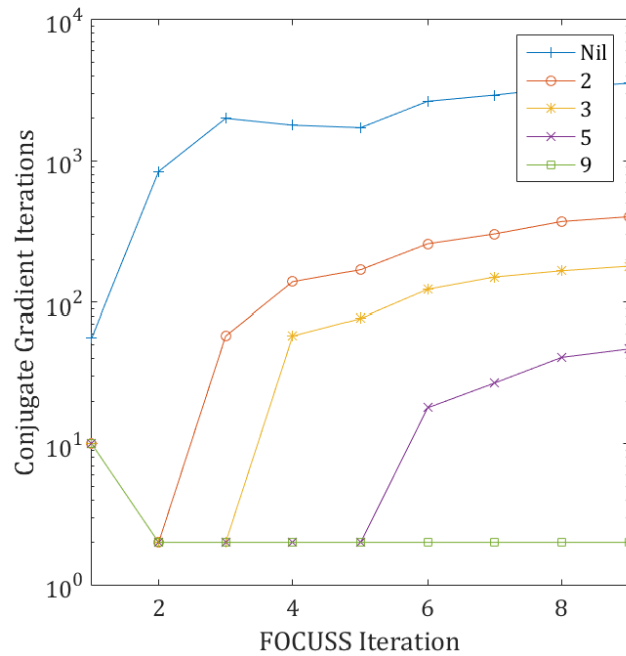


Figure 4.9: The reduction in conjugate gradient iterations required with preconditioner applied. Numbers in legend indicate the number of FOCUSS loops for which preconditioner was calculated, for the test case of a Gaussian.

This approach saves the cost required for computation of the incomplete Cholesky factorization. Fig. 4.9 shows how preconditioning impacts the number of conjugate gradient iterations required for convergence. It can be seen that the number of CG iterations is lesser if the preconditioner is applied in more FOCUSS loops and vice versa.

A similar trend can be observed in the condition number of modified coefficient matrix K , as shown in Fig. 4.10. The condition number of K can be observed to reduce when the preconditioner

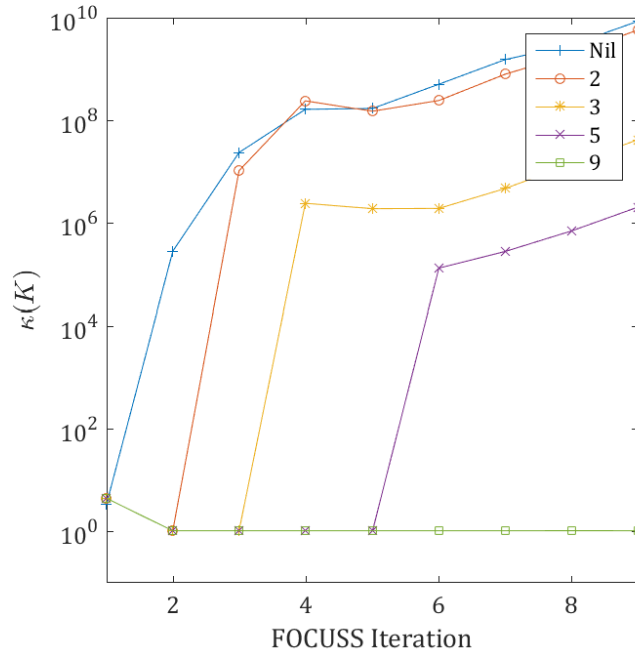


Figure 4.10: The reduction in condition number of modified coefficient matrix K with application of preconditioner. Numbers in legend indicate the number of FOCUSS loops for which preconditioner was calculated, for the test case of a Gaussian.

is applied. In application of the preconditioner, the computational effort saved by reduced number of CG iterations is offset by cost computation of preconditioner. Thus, it needs to be considered that it may not be feasible to apply the preconditioner for all FOCUSS loops. The number of FOCUSS loops for which preconditioner should be calculated for the least computational cost depends on the problem.

4.4.2 Selection of Number of Fourier Coefficients

It was also observed that the condition number of the modified coefficient matrix K also changes with the number of Fourier coefficients n , for a given grid. It was observed that the condition number of K is varies for different n for a given grid, i.e., with fixed m . Note that the condition number, in

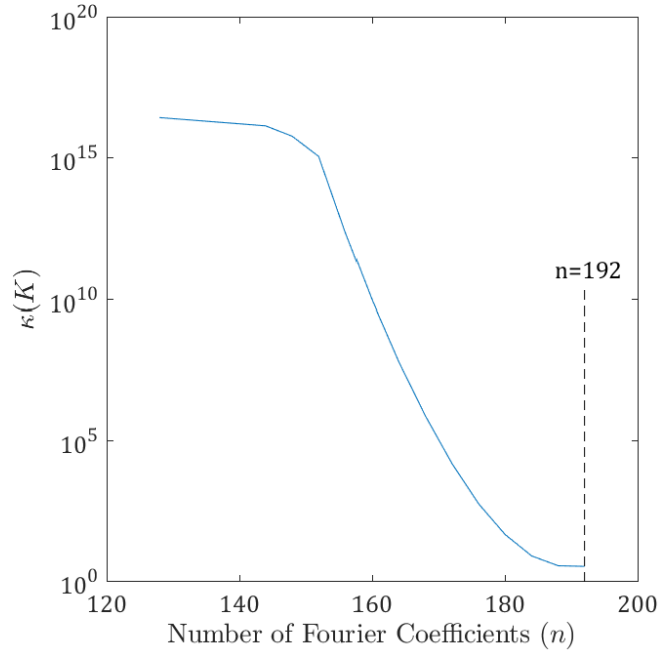


Figure 4.11: The variation of $\kappa(K)$ with varying n . The grid has $m = 128$ non-equispaced points and $1/\Delta x_{\min} = 192$. The condition number depends only on the grid before application of weights.

this case, does not depend on the function. It was observed that when $n = 1/\Delta x_{\min}$, the condition number of the coefficient matrix was minimum. The condition number increases rapidly as n moves away from $\frac{1}{\Delta x_{\min}}$. In other words, the number of Fourier coefficients is also the reciprocal of the minimum grid-spacing in order to get a well-conditioned problem. Fig. 4.11 shows the condition number of K without applying any weights, for various values of n and a fixed number of points $m = 128$. It can be observed that the problem is not optimal if n is selected arbitrarily for a given grid. Similar results were observed on grids of different sizes and with different n/m ratios.

4.4.3 Splitting the Problem

As explained in Sec. 2.1.3, the DFT matrix can be given as:

$$A = \begin{pmatrix} e^{2\pi i k_1 x_1} & e^{2\pi i k_2 x_1} & \dots & e^{2\pi i k_n x_1} \\ e^{2\pi i k_1 x_2} & e^{2\pi i k_2 x_2} & \dots & e^{2\pi i k_n x_2} \\ \vdots & \vdots & \vdots & \vdots \\ e^{2\pi i k_1 x_m} & e^{2\pi i k_2 x_m} & \dots & e^{2\pi i k_n x_m} \end{pmatrix} \quad (4.2)$$

Here, $A \in \mathbb{C}^{m \times n}$ and $n > m$. In NFFT, the wavenumbers corresponding to n Fourier coefficients are given as $\{k_1, k_2, \dots, k_n\} = \{-n/2 + 1, -n/2 + 2, \dots, 0, \dots, n/2\}$. DFT matrix in equation 4.2 can then be written as:

$$A = \begin{pmatrix} e^{2\pi i(-n/2+1)x_1} & \dots & e^{2\pi i(-m/2)x_1} & \dots & e^{2\pi i(0)x_1} & \dots & e^{2\pi i(m/2)x_1} & \dots & e^{2\pi i(n/2)x_1} \\ e^{2\pi i(-n/2+1)x_2} & \dots & e^{2\pi i(-m/2)x_2} & \dots & e^{2\pi i(0)x_2} & \dots & e^{2\pi i(m/2)x_2} & \dots & e^{2\pi i(n/2)x_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ e^{2\pi i(-n/2+1)x_m} & \dots & e^{2\pi i(-m/2)x_m} & \dots & e^{2\pi i(0)x_m} & \dots & e^{2\pi i(m/2)x_m} & \dots & e^{2\pi i(n/2)x_m} \end{pmatrix}$$

The under-determined problem with this matrix can be divided into two well-determined problems (if $n = 2m$) using following two matrices.

$$A_1 = \begin{pmatrix} e^{2\pi i(-m/2)x_1} & \dots & e^{2\pi i(0)x_1} & \dots & e^{2\pi i(m/2)x_1} \\ e^{2\pi i(-m/2)x_2} & \dots & e^{2\pi i(0)x_2} & \dots & e^{2\pi i(m/2)x_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ e^{2\pi i(-m/2)x_m} & \dots & e^{2\pi i(0)x_m} & \dots & e^{2\pi i(m/2)x_m} \end{pmatrix}$$

$$A_2 = \begin{pmatrix} e^{2\pi i(-n/2+1)x_1} & \dots & e^{2\pi i(-m/2-1)x_1} & e^{2\pi i(m/2+1)x_1} & \dots & e^{2\pi i(n/2)x_1} \\ e^{2\pi i(-n/2+1)x_2} & \dots & e^{2\pi i(-m/2-1)x_2} & e^{2\pi i(m/2+1)x_2} & \dots & e^{2\pi i(n/2)x_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ e^{2\pi i(-n/2+1)x_m} & \dots & e^{2\pi i(-m/2-1)x_m} & e^{2\pi i(m/2+1)x_m} & \dots & e^{2\pi i(n/2)x_m} \end{pmatrix}$$

The matrices A_1 and A_2 solve the system for Fourier coefficients corresponding to lower and higher wavenumbers respectively. If $n < 2m$, the system corresponding to higher wavenumbers (matrix A_2) is an over-determined system. These two well-determined systems do not require FOCUSS algorithms, thus only two conjugate gradient solvers are sufficient.

This approach gave unsatisfactory results when tried for Gaussian function e^{-50x^2} sampled on $m = 128$ non-equispaced gridpoints and the output was $n = 256$ Fourier coefficients. It was observed that the system of equations representing low wavenumber matrix A_1 gave an accurate answer for the 128 lower wavenumber Fourier coefficients for wavenumbers ranging from $[-63, 64]$, as can be expected from the analysis in Kunis (2006 PhD Dissertation)[30]. However, the system representing the higher wavenumber matrix A_2 was observed to be unstable, and return the values which are several orders of magnitude higher than the correct answer, as shown in Fig. 4.12. The FFT on equispaced grid can be considered as the accurate solution, and it can be observed that the answer

given by conjugate gradient solver for the higher wavenumbers using split matrices is not useful.

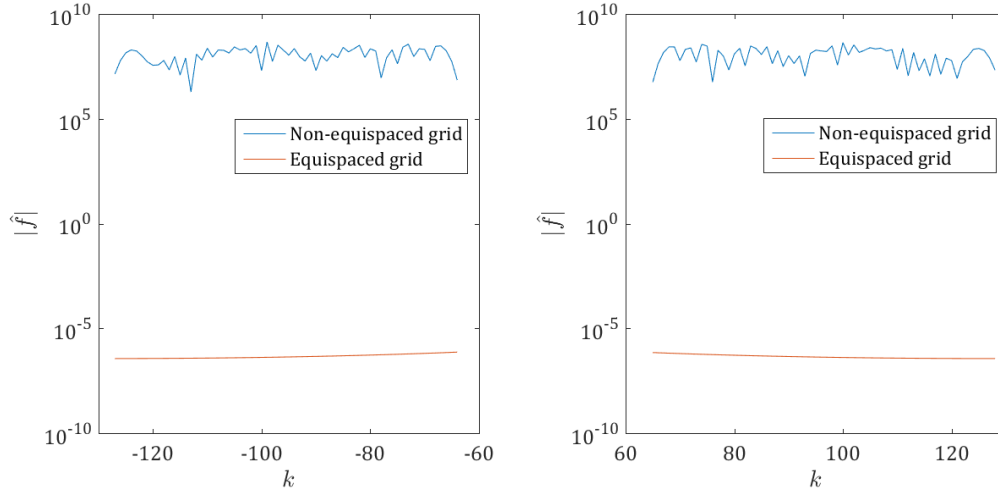


Figure 4.12: Fourier coefficients for higher wavenumbers computed by splitting the matrix, compared with the FFT on equispaced grid. Left: k from -127 to -64, Right: k from 65 to 128

The spectrum shown in Fig. 4.12 is that of a real function (a Gaussian test case). According to theory, the two plots should be mirror images of each other, as the Fourier transform of a real function is conjugate symmetric about the zeroth wavenumber. The solution on equispaced grid demonstrates the conjugate symmetry. However, the solution obtained using splitting the DFT matrix for non-equispaced grid, apart from not matching the equispaced grid solution, is also not conjugate symmetric thus disagreeing with the theory.

A combination of all or some of these approaches can be used to improve the computational performance when this approach is applied to turbulent wake, given that the results of Fourier transform of turbulent wake are accurate. A test of this approach was done on a slice of turbulent wake, considering the observations listed above, to check the accuracy. The number of Fourier coefficients was selected such that $n = \frac{1}{\Delta x_{min}}$, and the solution was compared to the one obtained using equispaced grid. Chapter 5 discusses the observations related to computational cost and accuracy of computation of NFFT of a slice of turbulent wake.

CHAPTER 5

TESTS ON TURBULENT WAKE

Turbulent wake is an example of localized turbulence, others being jets, plumes, mixing layers etc; where the use of non-equispaced grid in DNS can potentially lead to reduced memory usage in the simulation. In these kinds of flows, there exists an interface between the turbulent and non-turbulent flows[53]. Mass, momentum and energy exchanges occur mainly in these regions which need to be simulated numerically. The use of denser grid is required only in these regions, thereby explaining the use of non-equispaced grid. The ultimate aim of this thesis is to be able to simulate such flows with non-equispaced grid, using NFFT and FOCUSS.

The reason behind taking Fourier transform is to compute the spatial derivatives. This approach was applied to compute the derivative of the density fluctuations of the field of a turbulent wake. A slice was extracted from the three dimensional field turbulent wake, and a spectral derivative was taken using NFFT and FOCUSS algorithm. This chapter discusses the results about the accuracy of Fourier transform computed using this approach for a slice of turbulent wake.

5.1 Slicing the Wake Field

The 3-dimensional field of fluctuating density ρ' in the high resolution DNS of von Kármán Vortex street is available, which has the appropriate shape to be sampled on non-equispaced grid in the z-axis. The field is sampled on $4096 \times 2048 \times 2048$ equispaced points in x , y and z directions respectively. For more information, refer to [54, Hebert (2007) PhD Dissertation].

The field can be sampled and various z-slices can be extracted. The field is initially sampled on equispaced grid as it is computed using a conventional FFT. This field can be mapped onto non-equispaced grid by taking the forward FFT on equispaced grid, and transforming it backward using NFFT on a non-equispaced grid. This non-equispaced grid is defined beforehand. The field is initially sampled on 2048 equispaced points, which can be transformed back onto less than 2048 gridpoints (e.g. 1024). One of the slices of the field is shown in Fig. 5.1. It can be observed that a finer grid-spacing is needed only at the center while a coarse gridspace is sufficient towards the

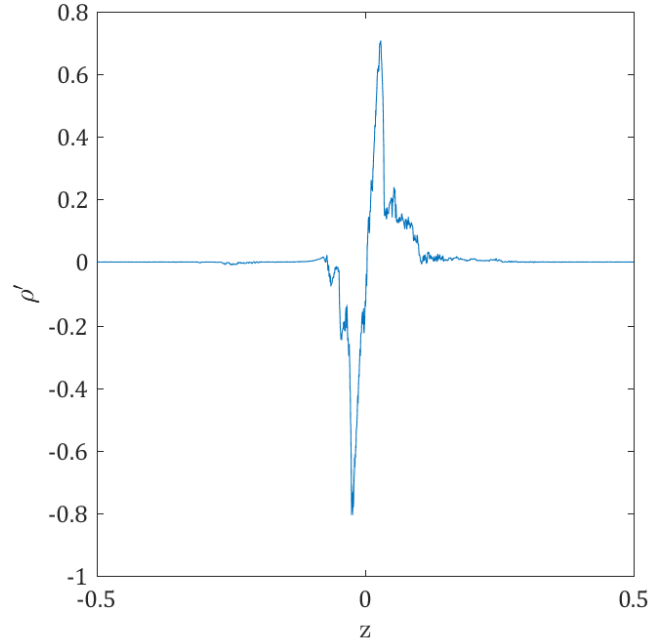


Figure 5.1: The line on a slice of the fluctuating density field in a turbulent wake. Y-axis shows the density variations.

ends. This slice was sampled from 2048 equispaced gridpoints, which was subsequently mapped onto 1024 non-equispaced points according to equation 4.1 for obtaining further observations about accuracy and computational performance.

5.2 Accuracy

Our interest is in computation of accurate derivatives of the periodic field, using Fourier spectral method. The comparison can be made between the derivative calculated using FFT on 2048 equispaced gridpoints and that calculated using NFFT on non-equispaced grid having less than 2048 points. In addition to the derivative, following parameters can also be used for comparison.

1. Fourier coefficients (\hat{f}): The vector of Fourier coefficients computed with NFFT should have 2048 coefficients, which can be compared with the vector of Fourier coefficients calculated with FFT for the field sampled on equispaced grid.
2. Fourier coefficients of Derivative ($ik\hat{f}$): The vector of Fourier coefficients of derivative can be compared to check whether all the frequencies can be accurately computed by the NFFT. A small error in the higher frequencies of Fourier transform of a function gets amplified while computing the derivative, when it is multiplied by the corresponding wavenumber.

Remainder part of this section describes the observations regarding accuracy of the derivative of the slice computed using NFFT.

Fig. 5.2 shows the two different slices of the field, and their derivatives taken using NFFT and FOCUSS algorithms. It can be seen that the mapping of the slice, originally sampled on 2048

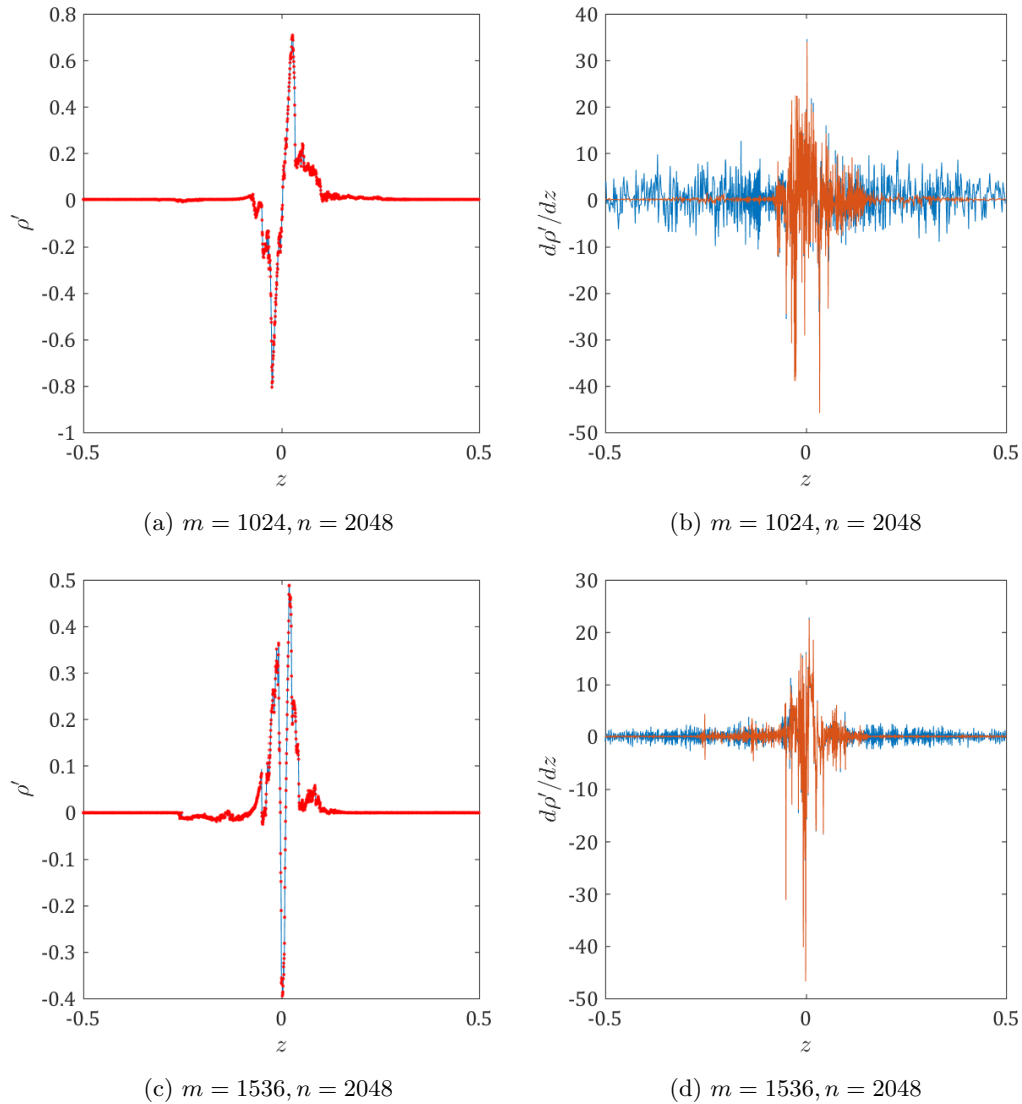


Figure 5.2: (a, c): Blue lines indicate a line on a slice of turbulent wake mapped onto non-equispaced grid, red dots indicate the original line on equispaced grid. (b, d): Blue lines indicate the derivative of lines (a) and (c) respectively sampled on non-equispaced grid, red lines indicate the derivative of original slice sampled on equispaced grid.

equispaced points, onto a non-equispaced grid is accurate. The derivatives computed with NFFT and FOCUSS are inaccurate, as can be seen from the figure. A presence of high frequency components can be seen in the derivatives, which will be explained subsequently.

5.3 Analysis

As can be seen from Fig. 5.2, the derivatives of two slices of turbulent wake using NFFT and FOCUSS algorithms do not have the desired accuracy. More insight about this can be gained by looking at the spectrum (magnitude of Fourier coefficients plotted against wavenumbers) of the slice computed using NFFT when it is sampled on non-equispaced grid and comparing it to the spectrum computed using FFT when it is sampled on equispaced grid. Fig. 5.3 shows the spectrum of the slice in Fig. 5.2 (a), sampled on both equispaced and non-equispaced grid. It can be seen from

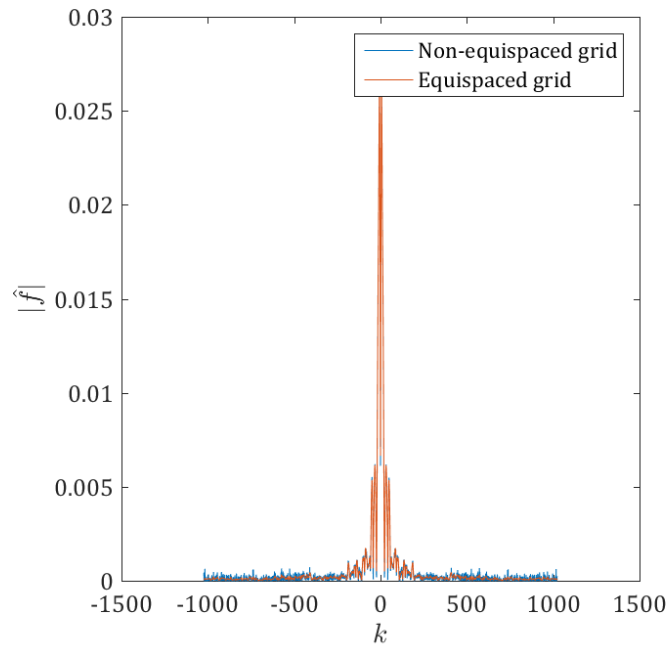


Figure 5.3: Spectrum of a line on a slice of turbulent wake, $m = 1024, n = 2048$

Fig. 5.3 that the spectrum shows more error in higher wavenumber components when sampled on non equispaced grid. The higher wavenumber part, from $k = 900$ to $k = 1024$ is shown in Fig. 5.4 separately. It can be seen from Fig. 5.4 that the spectrum of slice sampled on non-equispaced grid differs from that of the slice sampled on equispaced grid in following two aspects.

1. Peaks are higher and narrower in non-equispaced sampling
2. Troughs are lower in the non-equispaced sampling

However, it can be seen that the location of peaks and troughs in both the cases approximately coincide with each other.

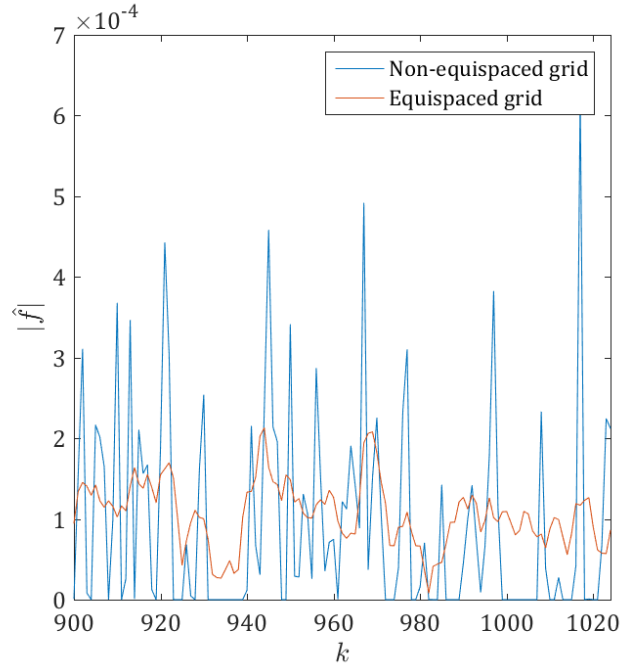


Figure 5.4: The spectrum of a line on a slice of turbulent wake, for wavenumbers from 900 to 1024.

When this spectrum is multiplied by the corresponding wavenumber to take derivative, the error in the spectrum gets amplified by the corresponding wavenumber. This can be observed in Fig. 5.5, where the difference between the accurate spectrum of derivative and the one computed using NFFT is shown.

Fig. 5.6 shows how first three FOCUSS loops affect the Fourier transform of a sine function. From analytical Fourier transform, we know that the only two wavenumbers having non-zero amplitude are ± 1 , where the amplitude is 0.5. Initially, minimum L_2 norm solution tends to distribute the energy more equally among all the wavenumbers. FOCUSS algorithm leads to increase in the magnitude of lower wavelength coefficients, while reducing the magnitude of higher wavenumber coefficients. It can be observed that the coefficients having lower amplitude become smaller and those with the higher value roughly remain the same. In other words, the algorithm favors a solution that is sparse. The initial solution tends to distribute the energy more equally among all coefficients, where the ‘shape’ of the spectrum is similar to the true solution, but higher wavenumbers have higher amplitudes than the actual solution. Application of weights refines the ‘shape’ and gets it closer to the actual solution. In other words, the peaks get narrower and sharper.

Fig. 5.7 shows how the FOCUSS loops affect the solution in case of a slice of Fourier transform. A similar phenomena, of peaks getting taller and narrower can be observed, finally leading to an

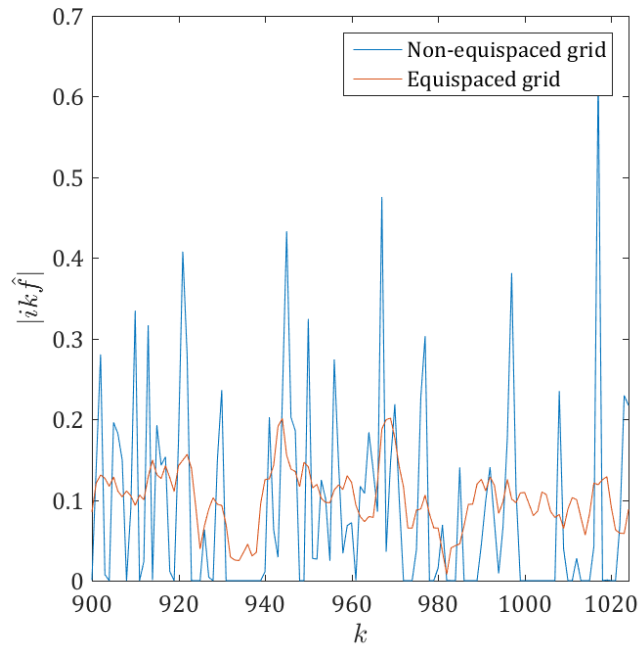


Figure 5.5: The spectrum of derivative of a line on a slice of turbulent wake, for wavenumbers from 900 to 1024

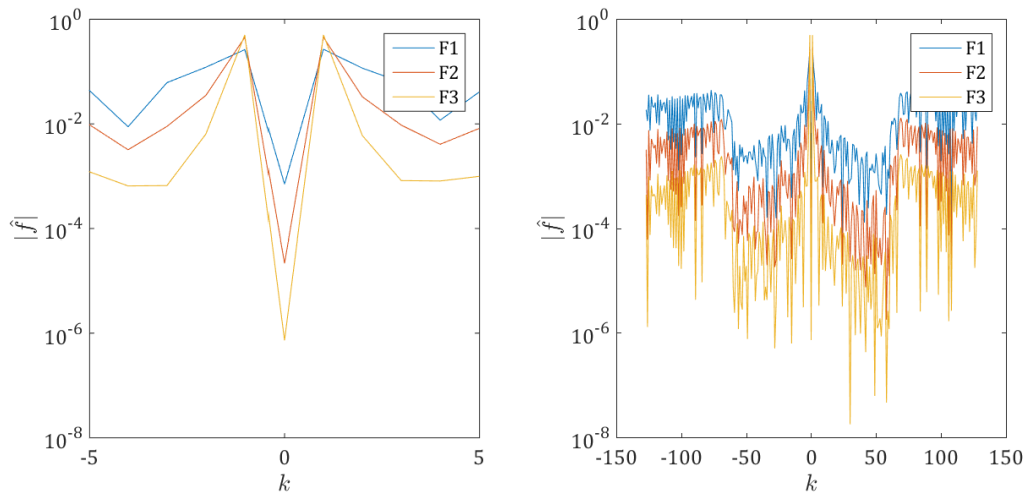


Figure 5.6: Fourier spectrum of a sinewave, sampled on a non-equispaced grid, for first three FOCUSS iterations. Legend indicates the number of FOCUSS iterations.

erroneous solution, which leads to inaccurate derivative.

Fig. 5.7 shows that the solution with one FOCUSS loop is more accurate for higher wavenumber coefficients. However, that solution still has errors beyond acceptable limits as the errors in derivative computed using NFFT with one FOCUSS loop are high. Fig. 5.8 compares the derivative computed with NFFT and one FOCUSS loop to that using an equispaced grid. It can be seen that the

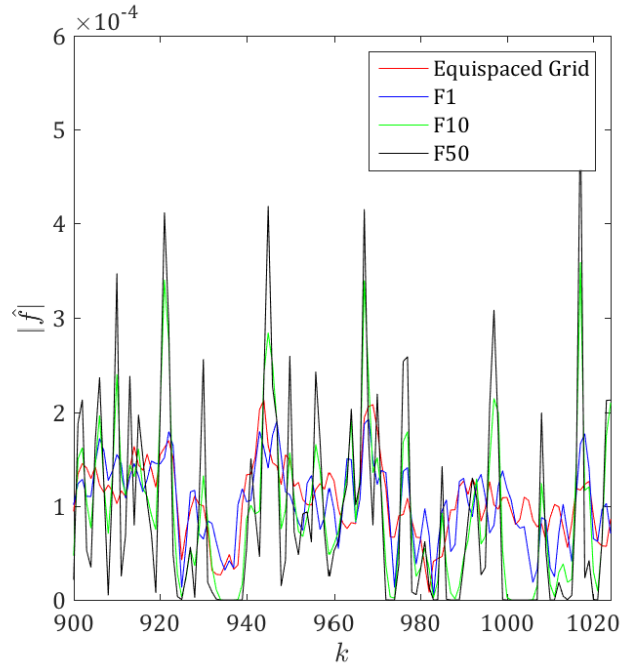


Figure 5.7: Effect of FOCUSS algorithm on the Fourier transform of a line on a slice of turbulent wakes for wavenumbers ranging from 900 to 1024.

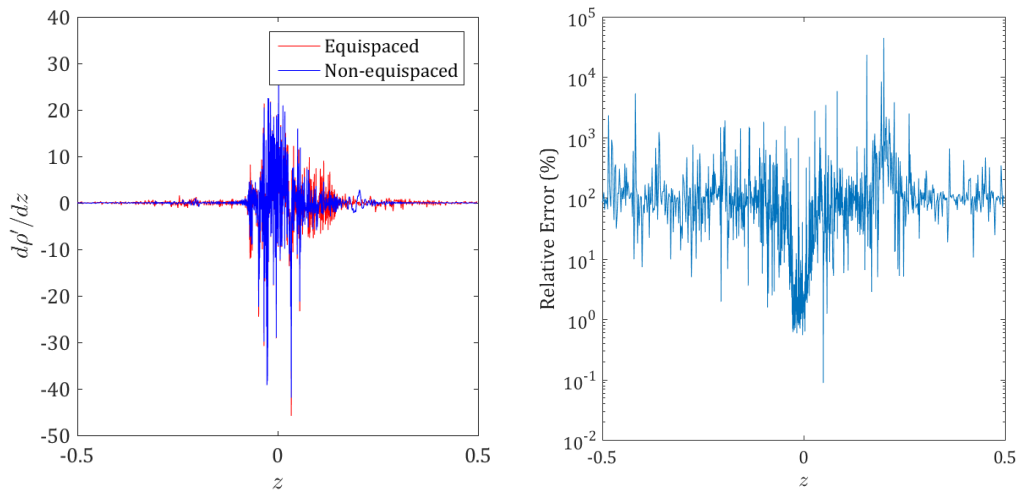


Figure 5.8: Left: The derivative of a line on a slice of turbulent wake, Right: Relative error in the derivative when it is computed using one FOCUSS loop and NFFT

derivative is inaccurate without using FOCUSS loop. However, as in case of functions where the Fourier transform is sparse, the FOCUSS algorithm does not lead to an accurate solution in this case.

For the case of turbulent wake, where high wavenumber Fourier coefficients are non zero, the FOCUSS algorithm can be observed to be inaccurate.

CHAPTER 6

CONCLUDING REMARKS

The purpose of this project was to implement the non-equispaced grids spacing in Direct Numerical Simulation of turbulent flows using Fourier spectral methods. The Non-Equispaced Fast Fourier Transform (NFFT) algorithm was used in order to take the forward and backward Fourier transform of a function sampled on non-equispaced grid. The forward Fourier transform on non-equispaced grid, from real to Fourier space, is the topic of study in this thesis. The Forward Fourier transform is done by solving a system of linear equations. In order to be able to reduce memory requirement of the DNS, it has been discussed that the number of Fourier coefficients required is higher than the number of non-equispaced gridpoints available. Thus, the system of linear equations becomes an under-determined system.

The characteristics of the solution of under-determined system of equations were studied. It was proven that the default minimum $L2$ norm solution obtained does not represent the Fourier transform for under-determined cases. It was observed that the minimum $L2$ norm solution distributes the energy more equally among all wavenumbers, resulting in high frequency components in the derivatives of test functions. Iterative reconstruction algorithm, FOCUSS was implemented along with NFFT to compute accurately the Fourier transform of test functions, by solving an under-determined system of equations, where more Fourier coefficients than the number of gridpoints were obtained. The combination of NFFT and FOCUSS has been used for a small test case of Direct Numerical Simulation on a grid size of 64^3 , using Taylor-Green initial conditions. The results are found to be in agreement with the analytical solution as well as a similar simulation using an equispaced grid.

The algorithm was found to be unacceptably slow for the test case of DNS. It has been found that the reason behind poor computational performance of this approach is the increase in condition number of the coefficient matrix in the system of linear equations after weights are applied in successive FOCUSS loops. Ill conditioned matrix requires a higher number of conjugate gradient iterations for solution, thereby increasing the computational cost. Following factors that may have

an impact on the condition number have been analyzed.

1. Preconditioning using incomplete Cholesky factorization was found to be effective in reducing the condition number of the matrix and subsequently the number of conjugate gradient iterations considerably.
2. For a given non-equispaced grid, it was found that the computational performance depends on the number of Fourier coefficients required. It was found that the condition number is the least when the number of Fourier coefficients corresponds to the minimum gridspace, i.e.,
$$n = \frac{1}{\Delta x_{\min}}.$$
3. Another approach, that would completely obviate the FOCUSS algorithm by splitting one under-determined problem into two well determined or over-determined problems was tested. It was found out that this approach could not compute the Fourier coefficients corresponding to the higher wavenumbers accurately.

The combination of NFFT and FOCUSS algorithms needs to satisfy two requirements to be used in turbulence simulation on large grids:

1. Accuracy: The derivative of the turbulent field needs to be computed accurately.
2. Computational Performance: The computation of derivative must be faster than a computation of similar accuracy on equispaced grid.

In order to test the accuracy, the NFFT and FOCUSS algorithms were tested on the slice of 3 dimensional field of fluctuating density in a turbulent wake and the results were analyzed. It was observed that the accuracy of the derivative of various slices of turbulent wake field computed using NFFT and FOCUSS approach was poor. The reason behind this is inaccurate computation of the Fourier transform, especially the coefficients representing the higher wavenumbers. The errors in higher wavenumber Fourier coefficients get amplified due to multiplication $ik\hat{f}$, resulting in an inaccurate derivative.

The reason behind why FOCUSS loops give inaccurate results for Fourier transform of a slice of turbulent field is the presence of local peaks in the high wavenumber regions of the spectrum. These local peaks, due to characteristics of the FOCUSS algorithm which biases the solution to put more energy into coefficients with higher amplitudes, get taller and narrower. This proves that FOCUSS algorithm is not useful to compute accurate under-determined Fourier transform on a non-equispaced grid.

Also, it has been proven that a direct computation of minimum $L2$ norm solution of the under-determined system still leads to inaccurate solution, both in case of test functions as well as a slice of turbulent wake field. Thus, there needs some other approach to refine the minimum $L2$ norm solution that gives the correct Fourier transform in under-determined case. The FOCUSS algorithm works in cases where only a few wavenumbers of the Fourier coefficient contain energy, but it fails when the Fourier transform is wideband, requiring some other approach.

APPENDIX

'C' CODE FOR NFFT

The C code for computing forward and backward Fourier transform of data sampled on non-equispaced grid is given in this appendix. The input array is a straightened 3 dimensional array in row-major order. The grid is non-equispaced in one dimension, and equispaced in the other two dimensions. The code here shows only the lines related to NFFT library. The C code for computation of Fourier coefficients from a data sampled on a non-equispaced grid is as follows:

```
nfft_plan pn; /*declear NFFT plan*/
solver_plan_complex pi /*declear iterative solver*/
nfft_init_1d(&pn,nz,mz); /*initialise 1D NFFT*/
for(i=0;i<mz;i++){ /*write values of grid for NFFT*/
pn.x[i]=z[i];
}
nfft_precompute_one_psi(&pn); /*precomputation for NFFT*/
solver_init_advanced_complex(&pi,(nfft_mv_plan_complex*)(&pn),CGNE|PRECOMPUTE_DAMP);
/*initialise iterative solver*/
complex double fhat0[nz],fhat1[nz]; /*intermediate arrays for NFFT*/
complex double diff_wt[nz]; /*difference in two successive solutions*/
for(i=0;i<nx*ny;i++){ /*extract 1D array for NFFT from the 3D array*/
for(j=0;j<mz;j++){
pi.y[j]=arri[i*mz+j]/(nx*ny);
}
for(j=0;j<nz;j++){ /*initialise intermediate array for weights*/
fhat1[j]=0+0*I;
fhat0[j]=1+0*I;
diff_wt[j]=fhat1[j]-fhat0[j];
}
}
```

```

while(l2norm(diff_wt,nz)>1e-12){          /*start FOCUSS algorithm*/
for(k=0;k<nz;k++){
pi.f_hat_iter[k]=0+0*I;    /*conjugate gradient initial guess*/
pi.w_hat[k]=cabs(fhat0[k]);          /*initialise weights*/
}
solver_before_loop_complex(&pi);
while(sqrt(pi.dot_r_iter)>1e-14){      /*conjugate gradient loops*/
solver_loop_one_step_complex(&pi);
}
}
}

```

The code for computation of inverse Fourier transform is as follows:

```

nfft_plan pn;          /*declear NFFT plan*/
nfft_init_1d(&pn,nz,mz);          /*initialise NFFT plan*/
for(i=0;i<mz;i++){          /*input the grid*/
pn.x[i]=z[i];
}
nfft_precompute_one_psi(&pn);          /*precompute NFFT plan*/
for(i=0;i<nx*ny;i++){          /*input the values of Fourier coefficients*/
for(j=0;j<nz;j++){
pn.f_hat[j]=arr1[i*nz+j];
}
nfft_trafo(&pn);          /*compute the inverse transform*/
for(j=0;j<mz;j++){          /*write the result on intermediate output array*/
arri[i*mz+j]=(pn.f[j]);
}
}
nfft_finalize(&pn);          /*finalise NFFT plan*/

```

These NFFT codes can be combined with FFTW to write a function of 3 dimensional forward and backward Fourier transforms with non-equispaced grid in one dimension and equispaced and two dimensions.

BIBLIOGRAPHY

- [1] J. Keiner, S. Kunis, D. Potts. Using nfft 3 :a software library for various nonequispaced fast fourier transforms. *ACM Transactions on Mathematical Software*, V(N):1–23, 2008.
- [2] I. Gorodnitsky, B. Rao. Sparse signal reconstruction from limited data using focuss: A re-weighted minimum norm algorithm. *IEEE Transactions on Signal Processing*, 45(3):600–616, 1997.
- [3] D. R. Durran. *Numerical methods for fluid dynamics: With applications to geophysics*, volume 32. Springer Science & Business Media, 2010.
- [4] F. Moukalled, L. Mangani, and M. Darwish. *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM and Matlab*. Springer Publishing Company, Incorporated, 1st edition, 2015.
- [5] Jiyuan Tu, Guan Heng Yeoh, and Chaoqun Liu. Preface. In J. Tu, G. H. Yeoh, and C. Liu, editors, *Computational Fluid Dynamics*. Butterworth-Heinemann, Burlington, 2008.
- [6] E. Dick. *Introduction to Finite Element Methods in Computational Fluid Dynamics*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
- [7] J. P. Boyd. *Chebyshev and Fourier spectral methods*. Dover Publicaions, Inc., 2001.
- [8] A.G. Kravchenko and P. Moin. On the effect of numerical errors in large eddy simulations of turbulent flows. *Journal of Computational Physics*, 131(2):310 – 322, 1997.
- [9] M. Y. Hussaini , T. A. Zang. Spectral methods in fluid dynamics. *Ann. Rev. Fluid Mech.*, 19:339–367, 1987.
- [10] R. Bhatia. *Fourier Series*. Mathematical Association of America, 1 edition, 2005.
- [11] H. Fadel, M. Agouzoul, and P. K. Jimack. High-order finite difference schemes for incompressible flows. *International Journal for Numerical Methods in Fluids*, 65(9):1050–1070, 2011.
- [12] P. McCorquodale and P. Colella. A high-order finite-volume method for conservation laws on locally refined grids. *Communications in Applied Mathematics and Computational Science*, 6(1):1–25, 2011.
- [13] P. Colella, M.R. Dorr, J.A.F. Hittinger, and D.F. Martin. High-order, finite-volume methods in mapped coordinates. *Journal of Computational Physics*, 230(8):2952 – 2976, 2011.
- [14] J. Dongarra and F. Sullivan. Guest editors’ introduction: The top 10 algorithms. *Computing in Science and Engg.*, 2(1):22–23, January 2000.
- [15] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.
- [16] P. Diniz, S. Netto, and E. D. Silva. *Digital Signal Processing: System Analysis and Design*. Cambridge University Press, New York, NY, USA, 2002.
- [17] S. K. Lele. Compact finite difference schemes with spectral-like resolution. *Journal of computational physics*, 103(1):16–42, 1992.

- [18] R. D. Moser, J. Kim, and N. N. Mansour. Direct numerical simulation of turbulent channel flow up to $re=590$. *Physics of Fluids*, 11(4), 1999.
- [19] H. Wengle and J. H. Seinfeld. Pseudospectral solution of atmospheric diffusion problems. *Journal of Computational Physics*, 26(1):87 – 106, 1978.
- [20] R. K. Shukla, M. Tatineni, and Xiaolin Zhong. Very high-order compact finite difference schemes on non-uniform grids for incompressible navierstokes equations. *Journal of Computational Physics*, 224(2):1064 – 1094, 2007.
- [21] V. Avsarkisov, S. Hoyas, M. Oberlack, and J.P. García-Galache. Turbulent plane couette flow at moderately high reynolds number. *Journal of Fluid Mechanics*, 751, 007 2014.
- [22] X. Zhang, G. A. Blaisdell, and A. S. Lyrintzis. High-order compact schemes with filters on multi-block domains. *Journal of Scientific Computing*, 21(3):321–339, 2004.
- [23] M. Lee and R. D. Moser. Direct numerical simulation of turbulent channel flow up to $re = 5200$. *Journal of Fluid Mechanics*, 774:395–415, 6 2015.
- [24] W. Y. Kwok, R. D. Moser, and J. Jimnez. A critical evaluation of the resolution properties of b-spline and compact finite difference methods. *Journal of Computational Physics*, 174(2):510 – 551, 2001.
- [25] D. Liu, Q. Chen, and Y. Wang. A sixth order accuracy solution to a system of nonlinear differential equations with coupled compact method. *Journal of Computational Engineering*, 2013, 2013.
- [26] S. B. Pope. *Turbulent flows*. Cambridge Univ. Press, Cambridge, 2011.
- [27] P. K. Yeung and S. B. Pope. Lagrangian statistics from direct numerical simulations of isotropic turbulence. *Journal of Fluid Mechanics*, 207:531–586, 10 1989.
- [28] R. Penrose. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51(3):406–413, 1955.
- [29] A. S. Householder. Some numerical methods for solving systems of linear equations. *The American Mathematical Monthly*, 57(7):453–459, 1950.
- [30] S. Kunis. *Nonequispaced FFT Generalisation and Inversion*. PhD thesis, Universität zu Lübeck, 2006.
- [31] O. Axelsson. *Iterative Solution Methods*. Cambridge University Press, New York, NY, USA, 1994.
- [32] W. B. Fritz. Numerical analysis (kaiser s. kunz). *SIAM Review*, 1(2):175–176, 1959.
- [33] Á. Björk. *Numerical Methods for Least Squares Problems*. Society for Industrial and Applied Mathematics, 1996.
- [34] R. Freund, G. H. Golub, and N. M. Nachtigal. Iterative solution of linear systems. *Acta Numerica*, 1:57–100, 1992.
- [35] S. Kunis and D. Potts. Stability results for scattered data interpolation by trigonometric polynomials. *SIAM Journal on Scientific Computing*, 29(4):1403–1419, 2007.
- [36] J. Z. Hearon. Generalized inverses and solutions of linear systems. *Journal of Research of the National Bureau of Standards-B. Mathematical Sciences*, 72B(4):303–308, 1968.
- [37] S. D. Cabrera and T. W. Parks. Extrapolation and spectral estimation with iterative weighted norm modification. *IEEE Transactions on Signal Processing*, 39(4):842–851, Apr 1991.

- [38] K. R. Rao, D. N. Kim, and J.-J. Hwang. *Fast Fourier Transform - Algorithms and Applications*. Springer Publishing Company, Incorporated, 1st edition, 2010.
- [39] A. N. Kolmogorov. Dissipation of energy in the locally isotropic turbulence. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 434(1890):15–17, 1991.
- [40] P. Moin and K. Mahesh. Direct numerical simulation: A tool in turbulence research. *Annual Review of Fluid Mechanics*, 30(1):539–578, 1998.
- [41] S. A. Orszag and G. S. Patterson. Numerical simulation of three-dimensional homogeneous isotropic turbulence. *Phys. Rev. Lett.*, 28:76–79, Jan 1972.
- [42] R.S Rogallo. Numerical experiments in homogeneous turbulence. Technical Report N81-31508/NASA-TM-81315, National Aeronautics and Space Administration - NASA. (VA US), 1981.
- [43] J. Kim, P. Moin, and R. Moser. Turbulence statistics in fully developed channel flow at low reynolds number. *Journal of Fluid Mechanics*, 177:133–166, 04 1987.
- [44] R. D. Moser and P. Moin. The effects of curvature in wall-bounded turbulent flows. *Journal of Fluid Mechanics*, 175:479–510, 02 1987.
- [45] S. M. de Bruyn Kops. Classical scaling and intermittency in strongly stratified boussinesq turbulence. *Journal of Fluid Mechanics*, 775:436–463, 007 2015.
- [46] G. I. Taylor and A. E. Green. Mechanism of the Production of Small Eddies from Large Ones. *Proceedings of the Royal Society of London Series A*, 158:499–521, February 1937.
- [47] A. J. Chorin. A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics*, 135(2):118 – 125, 1997.
- [48] E. Cheney and D. Kincaid. *Numerical Mathematics and Computing*. International student edition. Cengage Learning, 2007.
- [49] L. Eldén and V. Simoncini. Solving ill-posed linear systems with gmres and a singular preconditioner. *SIAM Journal on Matrix Analysis and Applications*, 33(4):1369–1394, 2012.
- [50] M. Benzi. Preconditioning techniques for large linear systems: A survey. *Journal of Computational Physics*, 182(2):418 – 477, 2002.
- [51] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, second edition, 2003.
- [52] G. H. Golub and C. F. Van Loan. *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.
- [53] T. Watanabe, J. J. Riley, S. M. de Bruyn Kops, P. J. Diamessis, and Qi Zhou. Turbulent/non-turbulent interfaces in wakes in stably stratified fluids. *Journal of Fluid Mechanics*, 797, 2016.
- [54] D. A. Hebert. *Turbulent mixing in stably stratified flows*. PhD thesis, University of Massachusetts Amherst, 2007.