2014

INVESTIGATION OF FILTERING METHODS FOR LARGE-EDDY SIMULATION

Weiyun Liu
University of Kentucky, wli234@g.uky.edu

INVESTIGATION OF FILTERING METHODS FOR LARGE-EDDY SIMULATION

THESIS

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Mechanical Engineering in the College of Engineering at the University of Kentucky

By

Weiyun Liu

Lexington, KY

Director: Dr. J. M. McDonough, Professor of Mechanical Engineering and Mathematics

Lexington, KY

2014

ABSTRACT OF THESIS

INVESTIGATION OF FILTERING METHODS FOR LARGE-EDDY SIMULATION

This thesis focuses on the phenomenon of aliasing and its mitigation with two explicit filters, i.e., Shuman and Padé filters. The Shuman filter is applied to velocity components of the Navier–Stokes equations. A derivation of this filter is presented as an approximation of a 1-D "pure math" mollifier and extend this to 2D and 3D. Analysis of the truncation error and wavenumber response is conducted with a range of grid spacings, Reynolds numbers and the filter parameter, β. Plots of the relationship between optimal filter parameter β and grid spacing, L^2-norm error and Reynolds number to suggest ways to predict β are also presented. In order to guarantee that the optimal β is obtained under various stationary flow conditions, the power spectral density analysis of velocity components to unequivocally identify steady, periodic and quasi-periodic behaviors in a range of Reynolds numbers between 100 and 2000 are constructed. Parameters in Padé filters need not be changed. The two filters are applied to velocities in this paper on perturbed sine waves and a lid-driven cavity. Comparison is based on execution time, error and experimental results.

KEYWORDS: Implicit filters, explicit filters, Shuman filters, Padé filters, lid-driven cavity

Weiyun Liu

11/05/2014

INVESTIGATION OF FILTERING METHODS FOR LARGE-EDDY SIMULATION

By

Weiyun Liu

Dr. J. M. McDonough
Director of Thesis

Dr. J. M. McDonough
Director of Graduate studies

11/05/2014
Date

To Lin and Chunling

ACKNOWLEDGMENTS

The thesis is a product of effort and commitment. I wish to thank my advisor and Thesis Chair, Dr. James M. McDonough, who teaches me so many helpful study tips, from edit tool Latex to plot tool Gnuplot, from programming language Fortran 77 to English grammar. I learn from him that research is a strict topic, and one should pay attention not only to subject but also details. I am fortunate that he could accept my application to be his student three years ago. Dr. McDonough patiently guided me to finish this paper and gave me good suggestions. It is an honor to be his student. What I learned from him is helpful in my future research career.

I also wish to acknowledge my parents, Lin Liu and Chunling Wei. In the days of working on my thesis, it is them that give me power, trust me, and encourage me. They know nothing about professional engineering research, but they always tell me to follow the guidance of advisors, step by step and finally I can make it. Without their support, I cannot survive in these tough days filled with thesis, jobs, applications, and school work. Thank them for supporting me to go abroad to study alone and supporting me in every depressing day in the past three years. Without them, I would never be able to pursue my dream. They are the power to me when I feel helpless, hopeless and get into trouble.

Last, I wish to thank all people in 216, RGAN Building. They are the person who support me, comfort me, and help me in the process of finishing this thesis. And I hope that all of them can have a bright future.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The phenomenon of aliasing is common in many engineering problems, especially in signal processing, and it occurs in many numerical simulations. Numerical simulation sometimes cannot represent high-wavenumber components (Fourier modes) in a signal (or solution), that is how aliasing arises by Olshausen [1]. The aliasing "problem" is ubiquitous in computational fluid dynamics (CFD), especially in direct numerical simulation (DNS) and large-eddy simulation (LES) of turbulence. In this thesis, large eddy simulation is employed (LES). LES is a numerical method where large scales are calculated and small scales are modeled. Its accuracy is between RANS and DNS. There are three main classes of methods to treat this problem: artificial dissipation by Pulliam [2], flux modification [3] and filtering [4], [5], all of which introduce additional diffusion (dissipation) to a solution of the differential equations, thus resulting in smoothing (hence, removal of high-wavenumber modes). Here only the third method is studied.

The first filter of the type studied in this thesis was possibly first used in a fluid dynamics setting by Shuman [6] but not in the context of CFD. Rather, Shuman was simply smoothing meteorological data to be utilized in weather prediction. The filter employed by Shuman was possibly first used in CFD as a method for discretizing advective fluxes, *viz.*, in a flux modification context [6]. Stephan [7] gave the derivation and introduction of its response function in details. It was later used as a post-processing filter by Yang and McDonough [8], and it is now receiving increasing attention in LES algorithms by Vasilyev [9], and Lund [10]; But there seems to be no formal derivation of this filter beyond the heuristics of a weighted average constructed so as to maintain consistency, and its formula contains an unknown parameter, denoted by β herein. Behavior of the filter depends on the value of β in a fairly strong way; yet, there seems to be no specific prescription for these values, say, in terms of grid spacing and Reynolds number, Re.

In the present research, effectiveness of the filter is demonstrated via the lid-driven cavity model problem depicted in Fig. 1.1. Investigations have been made of 3-D flows of an incompressible fluid in a square (aspect ratios equal unity) cubical cavity. The flows are driven by sliding the upper surface (the lid) of the cavity at a constant speed following an impulsive start. This model is attractive because of its simple geometry and easily implemented (no-slip) boundary conditions. The introduction of basic properties is presented by Ercan Erturk [11]: numerically, it is possible to obtain numerical solutions of 2-D incompressible cavity flow at high Reynolds numbers when fine grid meshes are used; 3-D DNS solutions for bifurcation Reynolds number differ an order from that obtained from 2-D DNS solutions of the driven cavity flow problem. Moreover, flow within the cavity exhibits a wide variety of behaviors at different locations, and from a mathematical perspective it displays singularities in the upper corner by Boppana and Gajjar [12]. Hence, in neighborhoods of these corners solutions are not classical.

The lid-driven cavity problem has been extensively investigated by numerical methods and laboratory experiments. However, most of the prior numerical work, until rather recently, has been confined to treating 2-D flow situations. Two-dimensional studies have disclosed that the global flow structure could be characterized by a primary eddy and secondary eddies that formed near corners of the lower solid walls by Bruneau and Saad [13]. The present research focuses on 3-D flow which can be considerably more complicated. The key method of the work of thesis is based on

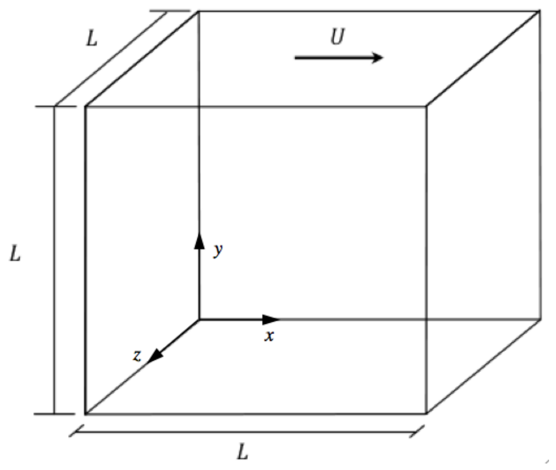


Figure 1.1: Lid-driven cavity

formal mathematical treatment of partial differential equations, namely, use of mollification. This is equivalent to the typical convolution filters widely used in LES, and the Shuman filter is derived via approximation of this mollification process. Then a truncation error analysis is presented to demonstrate that the Shuman filter is second-order accurate and dissipative at leading order. Furthermore, the normalization typically applied to mollifiers will be shown to lead to consistency of the discrete Shuman filter.

As already noted, the Shuman filter contains a single unknown parameter that regulates the number of high-wavenumber modes that are removed by its application, and this parameter must be prescribed, *a priori*, by the user. There currently is no theoretical method for predicting values of this parameter, and one of the goals of this research is to produce at least a semi-empirical prediction technique based on Reynolds number and finite-difference grid spacing.

Three-dimensional lid-driven cavity flow simulated at $Re = 1000$, $Re = 1500$ and

2000 using a standard projection method applied to a finite-volume discretization of the 3-D Navier–Stokes equations will be used to initiate this study. Employ grids of 11^3, 21^3, and 41^3 points and change the filter parameter value and Re separately. The goal of this work is to provide some guidance toward choosing a proper filter value when Re and number of grid points are known, leading, hopefully, to an automatic implementation. Finally, note that the Shuman filter is not necessarily an appropriate one in some contexts, but it is computationally very inexpensive.

The Padé filter is another explicit filter that is becoming more and more widely used. Derivations and parameters of Padé filters are introduced by Vasilyev [14]. In Liu's paper [15], one set of Padé parameters is also presented. In the present research, the same Padé parameters are used and the filtered results with those of Shuman filters for a range of β are compared. The comparison is based on two parts. One is a perturbed sine wave without consideration of the LES problem, in order to isolate fundamental behaviors of the fitlers, and the other is the lid-driven cavity problem, that is more complicated.

Besides Shuman and Padé filters, Kalman filters, described by Grewal [16], for example, Gaussian filters and top-hat filters are all used in various applications. The Gaussian filter is used to blur images and remove noise and details. In recent research, Gaussian filters, computationally efficient, are still widely used, and the coefficients by Liu [15] employed for the Padé filter cause it to have the properties similar to those of a sharp cutoff Gaussian filter. Gaussian filters are linear low-pass filters, so in some cases where high-pass filters are needed, they cannot be effective enough. Kalman filters are usually applied to electrical signals [17]. They first produce estimates of the current state with uncertainties, then update the estimation according to the next measurement with higher certainties by Ribeiro [18]. Kalman filters can be used to track objects and they also have some computer vision applications, e.g., feature tracking and cluster tracking; Kalman filters, on one hand, take full advantage of all the information stored in the covariance matrix, so it is lot smarter than a low pass filter can be. On the other hand, it is computational complicated. Top-hat filters are a class of nonlinear signal processing algorithms which have been applied extensively in computer vision, image processing and, more recently, for target detection by Wang [19]. When transformed from Fourier space to physical space, top-hat filters present the possibility of producing Gibbs-like oscillations Nathan [20], which of course is not desirable. According to the discussions above and the wide use of Shuman and Padé filters, decision is made to do research on the latter two filters in this thesis. Shuman filters only have one filtering parameter, therefore, it is not complicated to implement them. For the Padé filters, the parameters are already fixed, and consider the high accuracy of Padé filters, it only needs to investigate the effectiveness of them.

Derivations of the Shuman filter, both in 1D and higher dimensions, are presented in the next chapter. Chapter 2 also presents the derivation and parameters of the Padé filter used in this thesis. Comparisons of Shuman and Padé filters are made in Chapter 3 for one-dimensional signals in the framework of error removal and run time, and this is repeated for the more relevant 3-D case. The results lead to the conclusion that Shuman filters are more effective than Padé filters when applied to a simple perturbed sine wave. Then the two filters are applied to the lid-driven cavity problem where the flow movement is more complicated. In this case, the Reynolds number is set to 1500, 2000, and 10000 separately. In previous research, flow is laminar at *Re* 1500 and 2000, and turbulent at *Re* 10000. In this way, the research includes both the laminar and turbulent phenomena. In Chapter 4, conclusions and future work are presented. According to the comparison, Shuman filters work better than Padé filters on perturbed sine waves, but on the more complicated lid-driven cavity problem, Padé filters are more effective on high *Re* situations, e.g., turbulence.

Chapter 2

Analysis

In the world, most fluid flows are turbulent. The turbulent behavior is one of the most important but the most challenging problems in all the classical physical. Even though fluid flow is widespread, the problem of turbulence remains to this day the last unsolved problem. In the 21^{st} Century, most analysis of fluid flow could be performed via CFD. CFD saves money and time. In some special occasions such as high temperature where experiments cannot easily be carried out, CFD is a helpful tool to make simulations or predictions. But it also has its own issues. In this section, issues of CFD are discussed, and treatment is also provided. In this chapter, the mathematical explanation of aliasing is first presented. In this way, the root reason of aliasing is known. Then two filtering methods Shuman and Padé filters are introduced. Their derivations and how they work are presented from the mathematical viewpoint.

Aliasing is ubiquitous in CFD, especially in the research of direct numerical simulation (DNS) and large-eddy simulation (LES) of turbulence. However, aliasing is not the only issue in CFD. The reason is that no matter which numerical method is chosen, it cannot represent-wavenumber components in its solution. Cell-Re problem is another issue. The cell-Re problem easily causes grid-point to grid-point oscillations in numerical computations, and this oscillation is nonphysical. If the solution magnitude increases, the magnitude of these oscillations tends to increase, corresponding to increasing cell Re [21]. Cell-Re problems arise due to centered differencing of first-order derivatives. Suggested remedies involve replacing centered differencing with some other differencing. It has been suggested that a difference approximation using only information that is carried in the flow direction would be more accurate in Pathankar [22]. The most widely-used approach is 1^{st}-order upwinding. Even though the symptoms of cell-Re problem are similar to aliasing, it should be noted that the root cause is rather different.

2.1 Aliasing

In order to have a better understanding of aliasing, it can be explained from a mathematical viewpoint. Ames [23] gives the details as described below. Let $f(x)$ be a function in $L^2(-1,1)$ and consider its Fourier representation:

$$f(x) = \sum_{k=-\infty}^{\infty} a_k e^{ik\pi x} \tag{2.1}$$

with

$$a_k = \frac{1}{2} \int_{-1}^{1} f(x) e^{ik\pi x} dx. \tag{2.2}$$

Partition the interval $[-1,1]$ with $2N$ uniformly-spaced points; therefore, $x_j = j/N$, where $-N \leq j \leq N$. In this way, it can be constructed the Fourier polynomial for the value f_j at $x = x_j$

$$f_j = \sum_{m=-N}^{N-1} A_m e^{im\pi j/N}, \tag{2.3}$$

where

$$A_m = \frac{1}{2N} \sum_{j=-N}^{N-1} f_j e^{-im\pi j/N}. \tag{2.4}$$

In order to find how the $A_m s$ which are obtained from the discrete approximation, are related to the actual Fourier coefficients, the $a_k s$, $f(x)$ can be evaluated at the discrete point $x = x_j = j/N$:

$$f(x_j) = \sum_{k=-\infty}^{\infty} a_k e^{ik\pi x_j} = \sum_{k=-\infty}^{\infty} a_k e^{ik\pi j/N}. \tag{2.5}$$

There is an very important property of the complex exponential that it is periodic, and there can be only $2N$ distinct values of $e^{ik\pi j/N}$. For any finite N, rewrite the infinite sum as

$$\sum_{k=-\infty}^{\infty} a_k e^{ik\pi j/N} = \sum_{k=-\infty}^{\infty} \sum_{n=-N}^{N-1} a_{n+2N} e^{i(n+2Nk)\pi j/N}. \tag{2.6}$$

Now substitute the right-hand side of Eq. 2.6 instead of f_j into Eq. 2.4 to obtain

$$\begin{aligned} A_m &= \frac{1}{2N} \sum_{j=-N}^{N-1} \sum_{k=-\infty}^{\infty} a_k e^{ik\pi j/N} e^{-im\pi j/N} \\ &= \sum_{k=-\infty}^{\infty} a_{m+2Nk} \\ &= a_m + \sum_{|k|>0} a_{m+2Nk}. \end{aligned} \tag{2.7}$$

If $a_k = 0$, $\forall\ k$, $\exists \mid k \mid > N$, then there is no contribution to aliasing from the series. Also suppose N is sufficiently large that a_{m+2Nk} is small $\forall\ k$, then contributions are negligible. If $f \in L^2$, but not much better, then a_{m+2Nk} can be fairly large, even for very large N. This is the aliasing effect: A_m then is not close a_m.

In order to treat the aliasing, additional diffusion needs to be introduced to a solution of the differential equations. In this way, it results in smoothing, i.e., removal of high wavenumber modes. In McDonough and Yang [24], three main classes of methods to treat this aliasing are presented: flux modification, artificial dissipation and filtering. Flux modification is relatively expensive, which essentially doubles run time; artificial dissipation is not as expensive or as effective and usually contains unknown scaling constants; filters are inexpensive and fairly effective. In this thesis, the author focuses attention on the third method in this thesis.

2.1.1 Filtering Models: implicit and explicit filtering

From a mathematical viewpoint, it can be known that aliasing occurs automatically in nonlinear evolution problems since, e.g., the square of a discrete Fourier series, corresponding to the numerical solution method, contains unresolved modes due to nonlinearities in the differential equations, even with formally well-resolved discretizations, as noted by Shapiro [25]. Moreover, aliasing can occur due to under resolution in essentially any circumstance, including linear and/or non-evolving situations. Two approaches to filtering have been distinguished in the large-eddy simulation (LES) context by Vasilyev *et al.* [14]: use of implicit and explicit filters. Implicit filtering refers to formally applying a filter to governing equations and until recently has been the usual practice in construction of LES methods. As is well known, no specific filter is applied; but the formalism contains subgrid-scale (SGS) stresses, the modeling of which results in dissipation analogous to what occurs in Reynolds-averaged Navier–Stokes (RANS) methods, and very much like use of artificial dissipation for shock capturing. Nevertheless, new aliasing problems can appear after every time step in an implicit filtering method because of the nonlinear terms in the Navier–Stokes, or similar, equations as noted above. Use of implicit filters will not be the subject of the current studies.

Explicit filtering is a solution filtering technique wherein the governing equations are not filtered. Governing equations are solved directly on a grid (or otherwise), and this grid can be coarser than required by a fully-resolved direct numerical simulation; then the solution is filtered (formally, mollification) at each time step to mitigate effects of aliasing. Since filtered solutions are used in subsequent time steps, the aliasing phenomenon can be well controlled. In principle, this is no different than employing filters for image and signal processing. Implicit filtering can cause simulation results to be sensitive to the mesh resolution for several technical reasons, while explicit filters alleviate grid sensitivities to a significant extent, as described by You *et al.* [26].

In using explicit filtering, the filtering operation and the differentiation need to commute. This is not the case in inhomogeneous flow fields. The required smallest resolved length scales vary throughout the flow fields in inhomogeneous flows. The varying filter width, Δ, introduces a commutation error of $\mathcal{O}(\Delta^2)$ [27][28]. Most of the explicit filters are usually used in homogeneous flow fields or in homogeneous directions of more general flows by Gullbrand [29]. Vasilyev *et al.* [14] proposed a set of rules for constructing discrete filters and a general theory of discrete filtering for LES in complex geometries.

2.2 Introduction to Shuman filter

What is now termed the Shuman filter was first devised by Shuman [5], and has been successfully employed in operational practice to eliminate short-wavelength components from fields of meteorological variables. Numerical weather prediction, making use of finite-difference approximation of the equations of motion and computers, has invariably suffered from amplification of high-frequency components often beyond physical reality in computed solutions. Furthermore, attendant alteration of short-wavelength components detracts from the appearance of results, is annoying to analysis, and can be misleading to the uninitiated. A simple, weighted-average method of constructing filtering, or "smoothing," operators was devised by Shuman to mitigate these difficulties; and this has been analyzed in Shapiro [25]. Harten and Zwas [30] employed the Shuman filter in one of their early shock-capturing schemes, and in recent years it has begun to see application for removal of aliasing in LES.

In this section first derive a 1-D version of the Shuman filter, analyze its truncation error and obtain its wavenumber response (transfer function). Then extend the resulting formulas to higher dimensions.

2.2.1 Shuman filter in one dimension

The use of mollification is one of the modern analytical tools in PDE theory. It converts non-smooth (non-classical) solutions to ones that are in C^∞ in a well-defined way that permits control of error induced by this smoothing procedure (see, e.g., Gustafson [31]). Discrete implementations of such mollifiers can be used to treat aliasing of numerical solutions. An early example of such a filter was presented by Majda *et al.* [32]. Mollification significantly reduces the number of terms needed to represent the solution by a Fourier series and thus, in principle, to the ability to approximate solutions with rather coarse discretizations without concern for effects of aliasing.

In McDonough [33], (pp: 90-96, some words are from the reference directly) the derivation of Shuman filters is presented as similar to the following, which corrects this derivation.

Suppose $u(x)$ is not smooth; it can be mollified as follows:

$$u_\epsilon(x) = \int_{-\epsilon}^{\epsilon} u(\xi)\delta_\epsilon(x-\xi)\ d\xi \tag{2.8}$$

where δ_ϵ is a normalized C_0^∞ function with support $\to 0$ as $\epsilon \to 0$. This is clearly a filter expressed as a convolution analogous to the formal representation of filters employed in LES. The approximate solution u_ϵ can now be represented by a finite number of terms in a Fourier series—or even a Taylor series, and correspondingly only a finite number of grid points will be needed for a numerical simulation. In 1D, the formula presented by Shuman is

$$\tilde{u}_i = \frac{u_{i-1} + \beta u_i + u_{i+1}}{2+\beta}\,, \tag{2.9}$$

where u_i are grid-point values; and $\beta = 2$ is used. Derive this formula as an approximation to the above convolution, Eq. (2.8).

To begin, consider Fig. 2.1 which compares the graph of a typical mollifier kernel, say,

$$\delta_\epsilon(x) = c_\epsilon e^{-1/(\epsilon^2 - x^2)} \qquad |x| < \epsilon\,,$$

with the numerical approximation embodied in Eq. (2.9). Formally, it can be written this as

$$\tilde{u}_h(x_i) = \int_{x_i-h}^{x_i+h} u_h(\xi)\delta_h(x_i - \xi)d\xi \equiv F(\delta_h)u_h, \tag{2.10}$$

and define a function δ_h (which is not C^∞ but which does have compact support) as indicated in Fig. 2.1. Notice, in particular, that the support of δ_h is $2h$ about any particular point x_i, and that δ_h is constructed from two straight lines sitting above a rectangle of unit height.

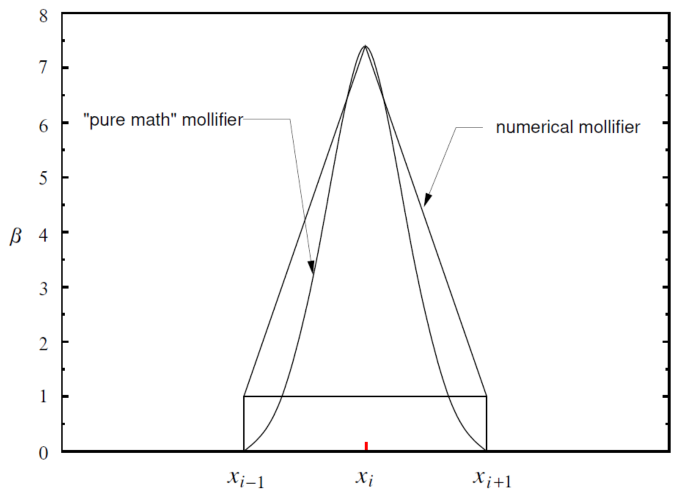


Figure 2.1: Discrete and pure math mollifiers.

The normalization of δ_ϵ is

$$c_\epsilon \equiv \left[\int_{-\epsilon}^{\epsilon} e^{-1/(\epsilon^2-x^2)}\ dx\right]^{-1},$$

and normalization δ_h is based on the geometry Fig. 2.1 and the definition of δ_h by formally employing trapezoidal quadrature:

$$\begin{aligned}\int_{x_i-h}^{x_i+h} \delta_h(x)dx &= 2h + (\beta^\star - 1)h \\ &= (\beta^\star + 1)h.\end{aligned}$$

This is exact for the geometry of Fig. 2.1 and leads to a normalization constant

$$C_h = \frac{1}{(\beta^\star + 1)h}$$

Now apply δ_h to a grid function, again employing trapezoidal quadrature. Consider the grid function values u_{i-1}, u_i, u_{i+1}; then

$$\begin{aligned}\tilde{u}_h(x_i) &= \int_{x_i-h}^{x_i+h} u_h(\xi)\delta_h(x_i-\xi)d\xi \\ &= \frac{h\left[\frac{1}{2}(u(x_i-h)+u(x_i+h))+\beta^\star u(x_i)\right]}{(\beta^\star+1)h} \\ &= \frac{u(x_i-h)+2\beta^\star u(x_i)+u(x_i+h)}{2+2\beta^\star}.\end{aligned}$$

Now define $\beta \equiv 2\beta^\star$ to obtain the Shuman filter:

$$\tilde{u}(x_i) = \frac{u(x_i-h)+\beta u(x_i)+u(x_i+h)}{2+\beta}, \tag{2.11}$$

where the value of β is arbitrary except that $\beta > -2+\epsilon$, $\epsilon > 0$, must hold with ϵ not the same as in Eq. (2.8) and the following.

Observe that if $u_i \equiv$ *const* in a neighborhood of x_i, then $\tilde{u}_i = u_i$ in that neighborhood.

This is known as "consistency" of the filter.

Furthermore, it is clear from the definition of the discrete mollifier, that $\tilde{u}_h(x_i) \to u_h(x_i)$ as $h \to 0$. While this is not obvious from the final form of the Shuman filter given in Eq. (2.9), it can be demonstrated via a simple truncation error analysis which now can be carried out. First return to the grid point notation of Eq. (2.9) and expand u_{i-1} and u_{i+1} in Taylor series:

$$u_{i-1} = u_i - hu_x\Big|_i + \frac{h^2}{2}u_{xx}\Big|_i - \frac{h^3}{6}u_{xxx}\Big|_i \pm \cdots , \tag{2.12a}$$

$$u_{i+1} = u_i + hu_x\Big|_i + \frac{h^2}{2}u_{xx}\Big|_i + \frac{h^3}{6}u_{xxx}\Big|_i + \cdots . \tag{2.12b}$$

Here, subscripts of the spatial independent variables denote partial differentiation. Then substitution of Eq. (2.12a) and Eq. (2.12b) into Eq. (2.9) yields

$$\tilde{u}_i = u_i + \frac{h^2}{2+\beta}u_{xx}\Big|_i + \mathcal{O}(h^4). \tag{2.13}$$

This representation displays two important features of this filter. As discussed before, one of the main requirements for successful treatments of aliasing is adding dissipation. Here, u_i is being replaced with a quantity containing this property. In particular, it can be seen that the dominant truncation error is diffusive, corresponding to addition of a Laplacian (with diffusion coefficient $h^2/(2+\beta)$). At the same time, the parameter β and the grid spacing h control the actual amount of added diffusion. Thus, even though a modified equation would contain extra diffusion at the level of the physical second-order operators, this goes to zero with h^2 rather than only with h as in the first-order upwinding. Moreover, it can be shown (by carrying more terms in the above Taylor expansions) that the $\mathcal{O}(h^4)$ term is anti-diffusive, leading to some cancellation of the effects at second order [33].

In the simulations relevant to time-dependent solutions in CFD, where the filter must be applied at each time step, the cumulative dominant error due to filtering after n discrete time steps can be shown to be

$$n\frac{h^2}{2+\beta}u_{xx}\Big|^n \tag{2.14}$$

for a 1-D Burgers' equation. It is clear from this that if the number of time steps becomes excessive, this term could potentially completely damp all aspects of the computed solution; this is a major disadvantage in this approach. But it can be seen from this term that it can be controlled, at least to some extent, by the choice of β. In addition, remark that Eq. (2.14) extends to multidimensions in the expected way, viz., $\partial^2/\partial x^2 \to \Delta$, where Δ is the Laplace operator in the appropriate space dimension.

There are additional items that should be investigated for the filter given in Eq. (2.9). One of these is "frequency response" in the context of signal processing (in which the signals are typically functions of time), and which will here, more appropriately for our purposes, be called wavenumber response since usually the filter is applied spatially. Wavenumber response, Yang and McDonough [8], shows the effect of β on magnitudes of all Fourier coefficients. Increasing β results in retaining more high-wavenumber effects, hence reducing the ability to control aliasing; decreasing β increases control of aliasing, but also increases diffusive truncation error: small β implies large dissipation and truncation error; and large β leads to insufficient control of aliasing—usually resulting in instability in nonlinear problems. Therefore choosing a proper value of β is important, and this is to be investigated below.

To determine the wavenumber response in 1D, start with a more detailed Taylor expansion of the Shuman filtered quantity $\tilde{u}$:

$$\tilde{u} = u + \frac{1}{2+\beta}\left[h^2u_{xx} + \frac{h^4}{12}u_{xxxx} + \frac{h^6}{360}u_{xxxxxx} + \cdots\right]. \tag{2.15}$$

Fourier transforming the above expansion, and considering only a single term from the corresponding Fourier series, leads to

$$\begin{aligned}\tilde{a}_m &= a_m + \frac{1}{2+\beta}\left[-m^2h^2 + \frac{m^4h^4}{12} + \cdots\right] a_m, \\ &= \left[1 - \frac{1}{2+\beta}(m^2h^2 - \frac{m^4h^4}{12} \pm \cdots)\right] a_m.\end{aligned} \tag{2.16}$$

Using the Taylor expansion of the cosine function, it is obtained

$$\tilde{a}_m = \left[1 - \frac{2}{2+\beta}(1 - \cos mh)\right] a_m, \tag{2.17}$$

where h is π/N; and N is the number of "grid" points. This shows that if $h \to 0$, $\tilde{a}_m = a_m, \forall\ m$.

Wavenumber response curves, $\tilde{a}_m/a_m$ *vs.* wavenumber, are shown for various values of β in Fig. 2.2. From this figure, it can be seen that different values of β result in different filtering effects. Small values of β, e.g., 2, completely remove the high-wavenumber components of a Fourier representation, which is what is needed to treat aliasing. If β is increased, filtering is less effective and possibly not good enough, and it can be seen that aliasing still exists at higher wavenumbers.

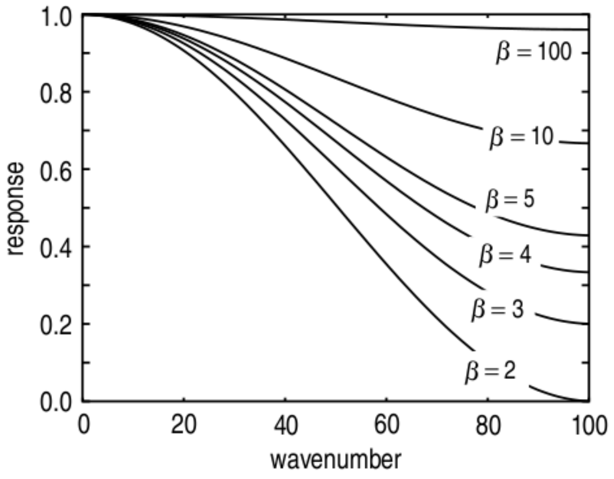


Figure 2.2: Wavenumber response with different filter parameter values [34].

2.2.2 Shuman filter in higher space dimensions

As is true for the 1-D filter, this 2-D case can be derived from a mathematical mollifier via the same procedure described above.

$$\begin{aligned}
\tilde{u}_h(x_i, y_j) &= \int_{y_j-h}^{y_j+h} \int_{x_i-h}^{x_i+h} u_h(\xi, \eta)\delta_h(x_i - \xi, y_j - \eta) d\xi d\eta \\
&= \frac{h\left[\frac{1}{4}(u_{i-1,j} + u_{i,j-1} + u_{i,j+1} + u_{i+1,j}) + \beta^\star u_{i,j}\right]}{(\beta^\star + 1)h} \\
&= \frac{u_{i-1,j} + u_{i,j-1} + 4\beta^\star u_{i,j} + u_{i,j+1} + u_{i+1,j}}{4 + 4\beta^\star}.
\end{aligned}$$

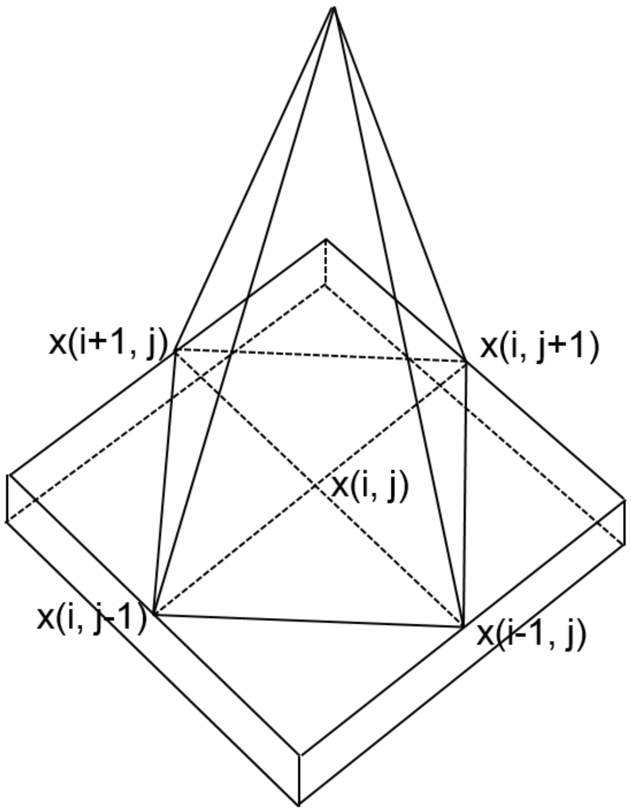


Figure 2.3: Discrete mollifier in 2D.

The discrete mollifier is the volume of the pyramid composed by $x(i, j-1), x(i, j+$

1), $x(i+1,j), x(i-1,j)$ and the top point. The edge length of every element is h. Now define $\beta \equiv 4\beta^{\star}$. In 2D, the formula for the Shuman filter is, by analogy with Eq. (2.9),

$$\tilde{u}_{i,j} = \frac{u_{i-1,j} + u_{i,j-1} + \beta u_{i,j} + u_{i,j+1} + u_{i+1,j}}{4+\beta}. \tag{2.18}$$

Note, however, that Eq. (2.18) is not the only possibility; in particular, there are no terms such as $u_{i-1,j-1}$. Including such terms results in additional filter parameters, and it is preferred to avoid this here. Observe that the complete formula, used in multigrid restriction operators, Briggs [35], includes these terms, and is highly diffusive.

Then expand $u_{i-1,j}, u_{i,j-1}, u_{i,j+1}$, and $u_{i+1,j}$ in Taylor series:

$$u_{i-1,j} = u_{i,j} - hu_x \mid_{i,j} + \frac{h^2}{2} u_{xx} \mid_{i,j} - \frac{h^3}{6} u_{xxx} \mid_{i,j} \pm \cdots \tag{2.19a}$$

$$u_{i,j-1} = u_{i,j} - hu_y \mid_{i,j} + \frac{h^2}{2} u_{yy} \mid_{i,j} - \frac{h^3}{6} u_{yyy} \mid_{i,j} \pm \cdots \tag{2.19b}$$

$$u_{i+1,j} = u_{i,j} + hu_x \mid_{i,j} + \frac{h^2}{2} u_{xx} \mid_{i,j} + \frac{h^3}{6} u_{xxx} \mid_{i,j} + \cdots \tag{2.19c}$$

$$u_{i,j+1} = u_{i,j} + hu_x \mid_{i,j} + \frac{h^2}{2} u_{xx} \mid_{i,j} + \frac{h^3}{6} u_{xxx} \mid_{i,j} + \cdots \tag{2.19d}$$

Substitution of Eq. (2.19) into the Eq. (2.18) yields

$$\tilde{u}_{i,j} = u_{i,j} + \frac{h^2}{4+\beta} (u_{xx} + u_{yy}) \Big|_{i,j} + \mathcal{O}(h^4). \tag{2.20}$$

Note that Eq. (2.20) is not the form typically used in smoothing meteorological data. In particular, in higher dimensions, more points than nearest neighbors are typically used, as in multigrid restriction operators. This results in a more complicated—as discussed above—and generally a more dissipative filter.

The Shuman filter studied here can be extended to 3D as

$$\tilde{u}_{i,j,k} = \frac{u_{i-1,j,k} + u_{i,j-1,k} + \beta u_{i,j,k} + u_{i,j+1,k} + u_{i+1,j,k} + u_{i,j,k-1} + u_{i,j,k+1}}{6+\beta}, \tag{2.21}$$

again, using only nearest-neighbor data. Expansion in Taylor series results in:

$$\tilde{u}_{i,j,k} = u_{i,j,k} + \frac{h^2}{6+\beta}(u_{xx} + u_{yy} + u_{zz})\Big|_{i,j,k} + \mathcal{O}(h^4).$$

The expressions in 2D and 3D involve, respectively, four and six nearest points, and they show that the Shuman filter is a weighted average of nearest-neighbor solution values. This property suggests that if grid-point spacing is coarse, filtered results may not accurately represent the data from which they were obtained. In fact, it is likely that such consideration—probably actual observations—led to use of more elaborate forms of the Shuman filter in higher dimensions: nine- and 27-point, respectively, filters in 2D and 3D. Nevertheless, as shown in the Taylor expansions, the leading truncation errors, $\frac{h^2}{4+\beta}(u_{xx} + u_{yy})$ in 2D, and $\frac{h^2}{6+\beta}(u_{xx} + u_{yy} + u_{zz})$ in 3D, are diffusive (as in 1D), as is required for mollification.

2.3 Introduction to Padé filter

2.3.1 Mathematical Description

Padé filters are considered as examples of discrete filters with vanishing moments. Wavelet analysis is performed using a single function called a wavelet, which can be regarded as a filter. Wavelet transform and Fourier transform are both approaches to signal analysis. In the analysis of wavelets, fine temporal analysis is done with high-frequency versions, while fine frequency analysis uses low-frequency versions. The essence of the wavelet transform is to view signals at different scales [36]. A wavelet [37] has p vanishing moments if and only if the wavelet scaling function can generate polynomials up to degree $p-1$. The "vanishing" part means that the wavelet coefficients are zero for polynomials of degree at most $p-1$. The wavelet analysis is not applicable to the Padé filter, which corresponds to Fourier transform.

The algorithm used in Padé filter construction is given in [14] and references therein. The derivation of Padé filters is cited from [38]. The basic form is

$$\sum_{m=-M_j}^{N_j} v_m^j \bar{\phi}_{j+m} = \sum_{l=-K_j}^{L_j} w_l^j \phi_{j+l}, \tag{2.22}$$

requiring solution of a linear system of equations, as noted earlier. ϕ is a 1-D field function, and $\bar{\phi}$ is the corresponding filtered function. v_m^j, w_l^j are constraints, and $M_j, L_j, K_j,\ N_j$ are numbers of constraints. The Fourier transform, $\widehat{G}_{(k)}$, associated with a Padé filter is given by

$$\widehat{G}_{(k)} = \frac{\sum_{l=-K_j}^{L_j} w_i^j e^{-i\Delta kl}}{\sum_{m=-M_j}^{N_j} v_m^j e^{-i\Delta km}}, \tag{2.23}$$

and in light of the filter definition, weight factors should satisfy the properties,

$$\sum_{l=-K_j}^{L_j} w_l^j = 1, \tag{2.24a}$$

$$\sum_{m=-M_j}^{N_j} v_m^j = 1, \tag{2.24b}$$

$$\sum_{m=-M_j}^{N_j} m^i v_m^j = \sum_{l=-K_j}^{L_j} l^i w_l^j, \quad i = 1, 2, ..., n-1. \tag{2.24c}$$

Note that the first two of constraints provide consistency as discussed earlier for the Shuman filter. It is straightforward to constrain Padé filters to a specific frequency (wavenumber) range; and use of Padé filters gives more flexibility in constructing filters which are closer to spectral cutoff filters. In this thesis, a symmetric Padé filter is employed, namely, $M_j = N_j$ and $K_j = L_j$.

In the present work, explicit filtering was performed using an optimized high-accuracy and maximum-resolution (HAMR) scheme [15]. To obtain $\bar{\phi}$, by filtering a variable ϕ, the HAMR formula employed is given by

$$\bar{\phi}_i + \alpha(\bar{\phi}_{i-2} + \bar{\phi}_{i+2}) + \beta(\bar{\phi}_{i-1} + \bar{\phi}_{i+1}) = \sum_{l=0}^{3} \frac{P_l}{2}(\phi_{i+l} + \phi_{i-l})$$

for interior points. Values for the filter coefficients are given in Table 2.1.

Table 2.1: Interior-point Padé filter coefficients

α	β	P_0
0.5673952755	0.1209216774	0.9931634217
P_1	P_2	P_3
1.2890384701	0.2965587062	0.0006836578

Near the boundary, it is impossible to maintain Padé filter symmetry (unlike the Shuman filter case) due to higher-order accuracy; an asymmetric scheme of the form is $\boldsymbol{a}_i \cdot [\bar{\phi}_1, \ldots, \bar{\phi}_5] = \boldsymbol{b}_i \cdot [\phi_1, \ldots, \phi_6], i = 2, 3$. The coefficients a_2, b_2 and a_3, b_3 for the second point and third points near the boundary are given respectively in Table 2.2.

Table 2.2: Padé filter coefficients near boundaries

a_2	b_2	a_3	b_3
0.3096256995	0.3084688023	0.1477868412	0.1470348738
1.0	1.0057844862	0.6357553622	0.6395151994
1.1380646293	1.1264956568	1.0	0.9924803256
0.4106696169	0.4222385894	0.6357553622	0.6432750366
0.0	-0.0057844862	0.1477868412	0.1440270040
-	0.0011568972	-	0.0007519674

This allows construction of filters that closely approximate spectral cutoff filters without the expense of transforming the solution to the spectral domain. Parameters in Padé filters need not be changed, unlike the Shuman filter where β can adjust the simulation accuracy. Which filter is more effective will be presented in the next section.

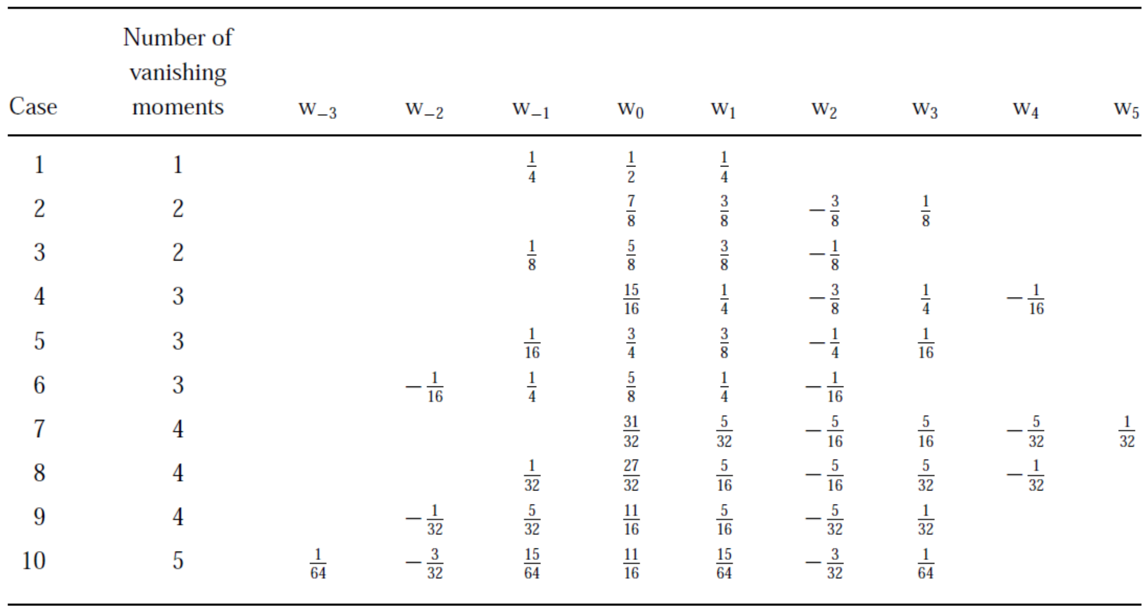
2.3.2 Relationship to Shuman filter

Eq. (2.24c) contains n constraints on w_l^j and is solvable if and only if $L_j + K_j + 1 \geq n$. If $L_j + K_j + 1 > n$ then additional constraints must be applied.

For derivative and filtering operations to commute to order n, the minimum number of degrees of freedom for a discrete filter is given by Eqs. (2.24). This condition gives the minimum filter support, which can be altered depending on the desired shape of the Fourier transform. If different shapes of the Fourier transform $\widehat{G}_{(k)}$ associated with filters are desired, the additional linear (or nonlinear) constraint(s) should be changed. A desirable constraint on a filter is that its Fourier transform be zero at the cutoff frequency, i.e.,

$$\sum_{l=-K_j}^{L_j} (-1) w_l^j = 0. \tag{2.25}$$

Eqs. (2.24) and Eq. (2.25) represent the minimum number of constraints which should be imposed on the filter. In [14], Vasilyev *et al.* show that with increase in the number of vanishing moments, a filter becomes a better approximation to the sharp cutoff filter.



Case	Number of vanishing moments	w_{-3}	w_{-2}	w_{-1}	w_0	w_1	w_2	w_3	w_4	w_5
1	1			$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{4}$				
2	2				$\frac{7}{8}$	$\frac{3}{8}$	$-\frac{3}{8}$	$\frac{1}{8}$		
3	2			$\frac{1}{8}$	$\frac{5}{8}$	$\frac{3}{8}$	$-\frac{1}{8}$			
4	3				$\frac{15}{16}$	$\frac{1}{4}$	$-\frac{3}{8}$	$\frac{1}{4}$	$-\frac{1}{16}$	
5	3			$\frac{1}{16}$	$\frac{3}{4}$	$\frac{3}{8}$	$-\frac{1}{4}$	$\frac{1}{16}$		
6	3		$-\frac{1}{16}$	$\frac{1}{4}$	$\frac{5}{8}$	$\frac{1}{4}$	$-\frac{1}{16}$			
7	4				$\frac{31}{32}$	$\frac{5}{32}$	$-\frac{5}{16}$	$\frac{5}{16}$	$-\frac{5}{32}$	$\frac{1}{32}$
8	4			$\frac{1}{32}$	$\frac{27}{32}$	$\frac{5}{16}$	$-\frac{5}{16}$	$\frac{5}{32}$	$-\frac{1}{32}$	
9	4		$-\frac{1}{32}$	$\frac{5}{32}$	$\frac{11}{16}$	$\frac{5}{16}$	$-\frac{5}{32}$	$\frac{1}{32}$		
10	5	$\frac{1}{64}$	$-\frac{3}{32}$	$\frac{15}{64}$	$\frac{11}{16}$	$\frac{15}{64}$	$-\frac{3}{32}$	$\frac{1}{64}$		

Figure 2.4: Values of Weight Factors and the Numbers of Vanishing Moments for Different Minimally Constrained Discrete Filters [14]

In case 1, only one vanishing moment is applied, and this is equivalent to the Shuman filter. It indicates that the Padé filter with only one vanishing moment is the same as the Shuman filter.

2.3.3 Implementation

The accuracy of the Padé filter depends on how many filter coefficients used. For interior points, the Padé filter has five vanishing moments, so the commutation error is $\mathcal{O}(\Delta^6)$; for boundary points, the Padé filter has four vanishing moments, so the commutation error is $\mathcal{O}(\Delta^5)$, where Δ is the associated filter width, and it is equal to the grid spacing in this thesis. This shows that the Padé filter can be a high-accuracy filter. If different accuracies are required, the number of vanishing moments can be changed. This is different from Shuman filter, which is second-order accurate.

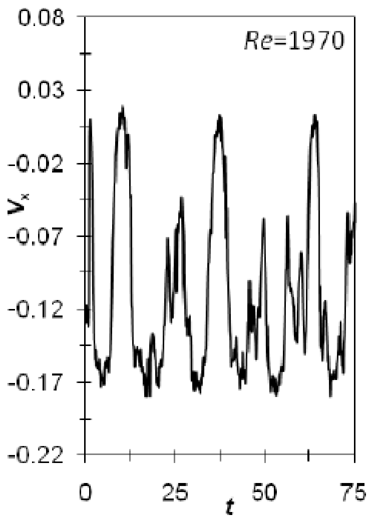
Chapter 3

Results and Discussions

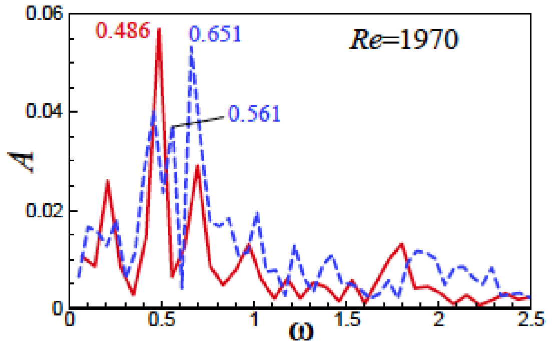
In this chapter, the experimental results at $Re = 1000$ and $Re = 2000$ are shown, and the comparison of Shuman and Padé filters is based on them. Cases with $Re = 1000$ and 2000 are used to prove that improper Shuman filter parameters will lead to wrong results. There are two parts of comparison of Shuman and Padé filters. One is based on perturbed sine waves and the other one is based on the lid-driven cavity problem.

3.1 Experimental Results

The lid-driven cavity problem is a fundamental problem in CFD, and many complicated engineering problems are based on it. Liberzon *et al.* performed experiments showing flow status with different Reynolds numbers in [39]. They report that flow is steady for $Re < 1700$ at least; in the range $1700 < Re < 1970$, steady-unsteady transition occurs; the flow becomes oscillatory when $Re \geq 1970$. However, in the stability analysis, flow is quasiperiodic at $Re = 1970$, and their statements regarding this do not agree; but their PSDs and time series do agree in [39]. Comparison between experimental results and numerical simulations are made in the next section to show that improper Shuman filter parameter values will result in inconsistency with physical experimental results.



(a)



(b)

Figure 3.1: Experimental results with $Re = 1970$; (a) Dimensionless time evolution of u velocity component, (b) Fourier transform of u velocity component, solid red and dash blue lines correspond to experiments with water and glycerin solution, respectively [39].

In [39], $Re = 1970$ is employed, which is close to the value 2000 used in the present research. The physical experimental location of measurements is $(-0.325, -0.378, 0)$ in meters within a $1m$ cube [39]. Flow oscillations with a significantly large amplitude are shown in Fig. 3.1. The corresponding power spectral density shows that the flow is quasiperiodic, although the authors describe it as periodic at $Re = 1970$. It was pointed out in [39] that some oscillations were caused by finite-amplitude experimental noise. This is not true from what can be seen in Fig. 3.1. It can be seen from Fig. 3.1 (a) that some peaks are two thirds of the highest magnitude. If they are noises, then the experiment in [39] does not make any sense. Besides, it can be seen from Fig. 3.1 (b) that the flow is quasiperiodic since there are some incommensurate harmonics.

3.2 Simulations

In previous research, the Shuman filter parameter is usually set to be 2. In this section, different Shuman filter values are explicitly applied to velocities computed for the lid-driven cavity. Reynolds numbers employed are $500, 1000, 1500$ and 2000. The physical location (0.175, 0.122, 0.5) is analyzed in this thesis, which is corresponding to the experimental location $(-0.325, -0.378, 0)$ in [39], left bottom corner of the lid driven cavity. In order to match this location on a discrete computational grid, the nearest grid point to the physical location is chosen to get the most accurate results. The following two sections take two examples of all simulation work, $Re = 1000$ with grid points 11^3 and $Re = 2000$ with grid points 81^3. In the first case, $\beta = 1800$, 2000, 3000 and 5000 are applied to each velocity component. In the second case, β is set to 2, 25, 50 and 100. Time series, power spectral densities and phase portraits are analyzed for the u velocity component in every case. Applying power spectral analysis to a data set (usually a time series, but sometimes a spatial distribution) yields the power spectral density (PSD), a Fourier-space representation of energy (in the L^2 sense) of a signal as a function of frequency (or wavenumber). PSDs can provide good information of flow status.

The program used in this thesis solves the 3-D incompressible Navier–Stokes equations of fluid flow. The pressure-velocity coupling is treated via Gresho's Projection 1 method [40]. Spatial derivatives are approximated using a 2nd-order, centered finite-volume discretization with staggered indexing. Time integration is performed with the trapezoidal method, with delta-form Douglas and Gunn time splitting [41]. Nonlinearities are handled with Delta-form Newton-Kanorovich (quasilinearization) implemented in a "block-Jacob" diagonal fashion [33].

3.2.1 Simulation results with $Re = 1000$, 11^3 grid points

In this case, different Shuman filter parameter values are applied to velocities, and through trying various filter parameters, it is found that near $\beta = 100$, 500, 1000, 3000 and 5000, the flow status changes.

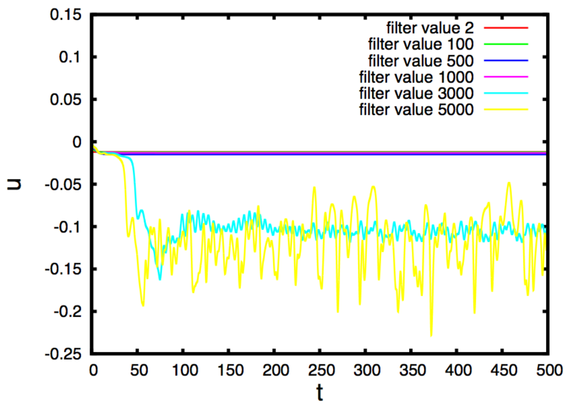
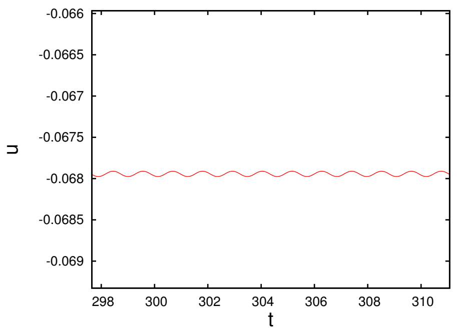
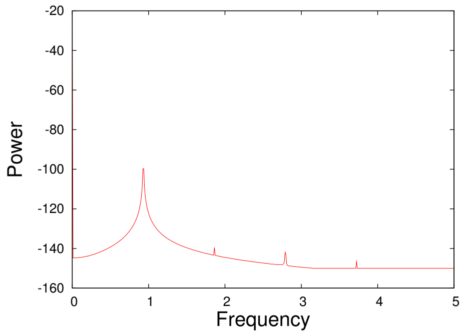


Figure 3.2: u component velocity with different Shuman filter values for 11^3 grid points, $Re = 1000$

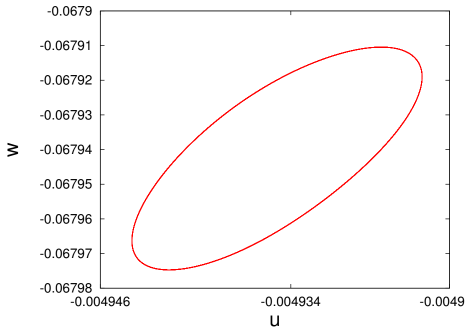
In Fig. 3.2, it is obvious that under different filter parameter values, the flow status is different. From $\beta = 2$ to 1000, flow is steady. When $\beta = 3000$, the flow behaves in a turbulence fashion. Flow usually does not change directly from steady to a turbulent status, therefore, there must be transition steps. One sequence of transitions that a flow will undergo as Re is increased to arrive at a chaotic state is steady, periodic, quasiperiodic and turbulent. This theoretical sequence is presented by Ruelle and Takens [42]. Although Re is fixed, the change of filter parameter value causes the computed flow to undergo this same sequence, showing the importance of choosing a proper filter parameter. An improper filter parameter value can result in qualitatively incorrect flow behavior predictions. Transitions (bifurcations) of flow states with different filter parameters are investigated in the following.



(a)



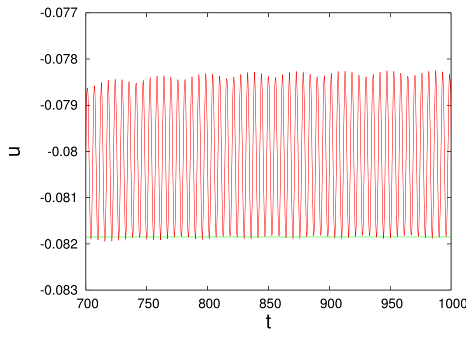
(b)

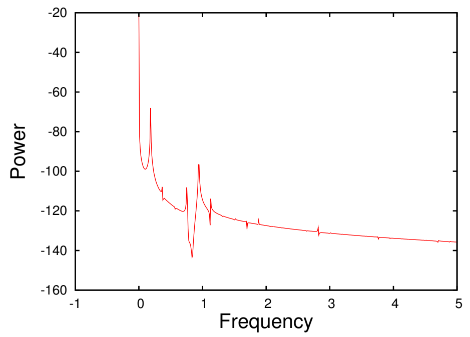


(c)

Figure 3.3: Velocity, PSD and phase-portrait with 11^3 grid points; $Re = 1000$, $\beta = 1800$

The time series of Fig. 3.3, corresponding to $\beta = 1800$, appears to be periodic. In order to prove this, the power spectral density and phase portrait are analyzed. A phase portrait is a plot of two components of a dynamical system against one another as time evolves by Alligood [43]. It is a way quantitatively to assess the flow state. If a flow is steady, trajectories of the initial point (u_0, v_0) will ultimately end at one point (u_s, v_s), and this ending point is termed the attractor; if a flow is periodic, the structure of attractor takes the form of a "limit cycle"; if a flow is subharmonic, the trajectory is two overlapping circles [43]. Part(b) of the figure shows that the interval between harmonics is the same, implying that the flow is periodic.


(a)


(b)

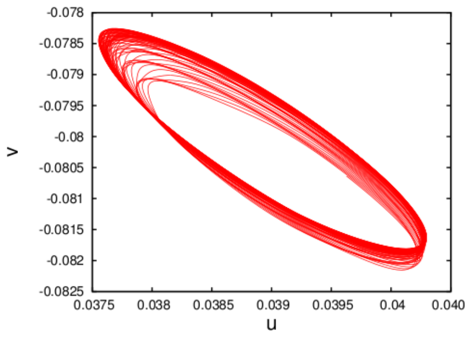
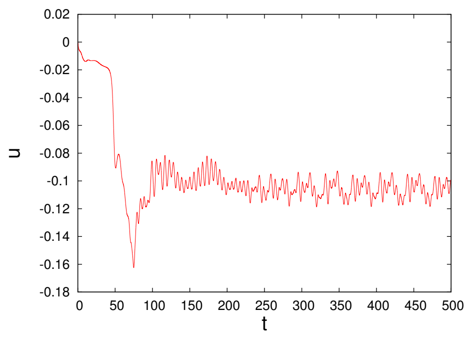
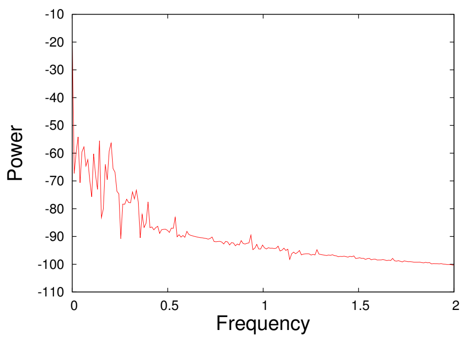

(c)

Figure 3.4: Velocity, PSD and phase-portrait with 11^3 grid points; $Re = 1000$, $\beta = 2000$

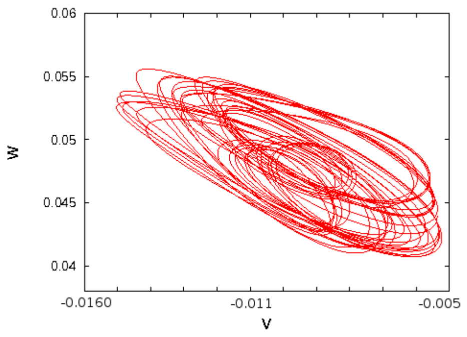
With $\beta = 2000$, from Fig. 3.4(a), it is seen that the flow behaves somewhat periodically, but the magnitude of velocity does not exactly repeat. Velocities at some points are over the green line provided to detect perfect periodicity. From the time series, it is expected that the flow is probably quasiperiodic. From Fig. 3.4(b), the power spectral density, it can be seen that the intervals of harmonics are not the same with a suggestion of an incommensurate frequency. In (c), phase portrait shows more than one circle; however, they are interweaved. Therefore, the flow is quasiperiodic at $Re = 1000, \beta = 2000$. The flow should be steady at $Re = 1000$ [39]; therefore, the simulation results are inconsistent with physical phenomena.



(a)



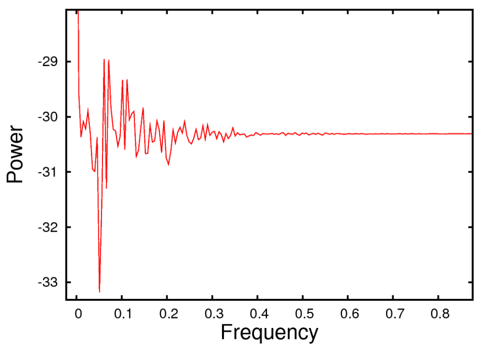
(b)



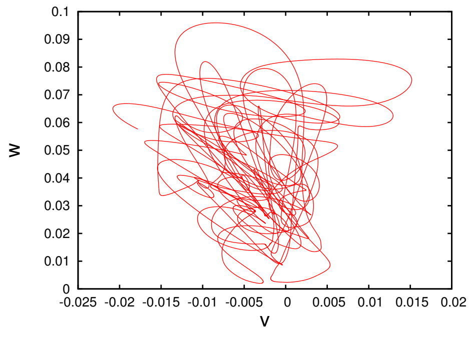
(c)

Figure 3.5: Velocity, PSD and Phase-portrait with 11^3 grid points; $Re = 1000$, $\beta = 3000$

With $\beta = 3000$, from Fig. 3.5 (a), the flow seems to be both quasiperiodically and turbulently. The same structure does repeat except during the initial transient. In the Fortran code, the time step size, time steps and total time are set to be 0.0125, 40000 and 500 *sec*, respectively. The PSD is calculated using data between 400 and 500 *sec* because velocity time series has passed the transitive state and appears stationary. The PSD shows that the intervals of harmonics are not the same. The phase portrait shows one big cycle and one small cycle. Therefore, the flow is quasiperiodic with noise at $\beta = 3000$, which is again inconsistent with physical phenomena.



(a)



(b)

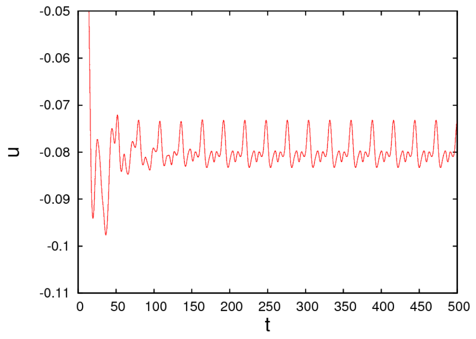
Figure 3.6: PSD and Phase-portrait with 11^3 grid points; $Re = 1000$, $\beta = 5000$

With $\beta = 5000$, from Fig. 3.2 above, the time series shows the flow is turbulence. The time step size, time steps and total time are 0.0125, 40000 and 500 *sec.*, respectively, for the simulation. The PSD is calculated as in the preceding case. In Fig. 3.6 (a), the power spectral density obviously shows that the intervals are not the same. There is no obvious cycle in the phase portrait, and lines are interweaved with each other. The power spectral density and the phase portrait suggest the flow is turbulent.

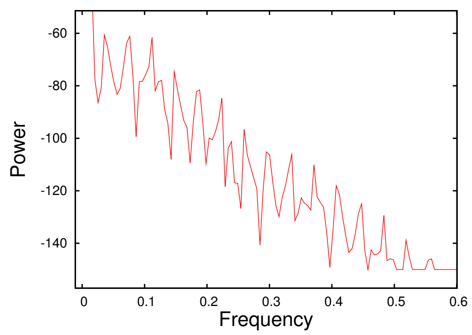
In this case, it is obvious that different Shuman filtering values result in different flow states, proving the importance of choosing a proper filter parameter value.

3.2.2 Simulation results with $Re = 2000$, 81^3 grid points

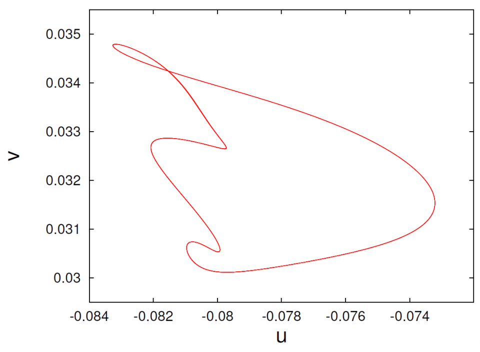
In order to investigate the effect of Shuman filtering values more deeply, 81^3 grid points are employed with $Re = 500$, 1000, 1500 and 2000 shown in Table 3.1 below.



(a)



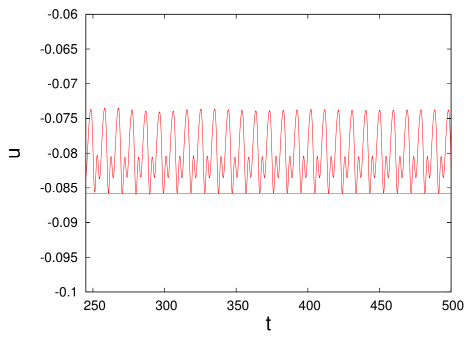
(b)



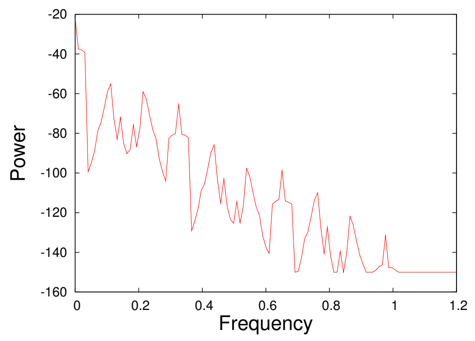
(c)

Figure 3.7: Velocity, PSD and phase portrait with 81^3 grid points, $Re = 2000$, $\beta = 25$; (a) Velocity, (b) Power spectral density, (c) Phase portrait.

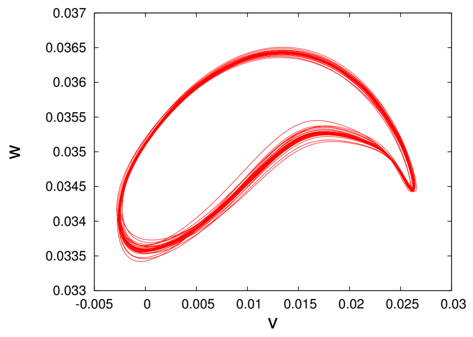
For $Re = 2000$, flow is periodic. In order to demonstrate this, the power spectral density is shown in the Fig. 3.7(b). The corresponding time series and phase portrait, Fig. 3.7(a) and (c), respectively are in agreement with this. However, this is not consistent with the experimental results. In [39], it is pointed out that flow exhibits oscillations and has subsequent multiple harmonics in the power spectral density analysis shown in Fig. 3.1.



(a)



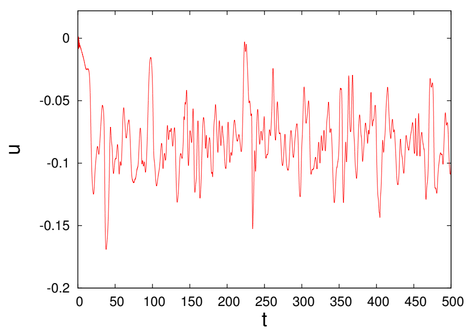
(b)



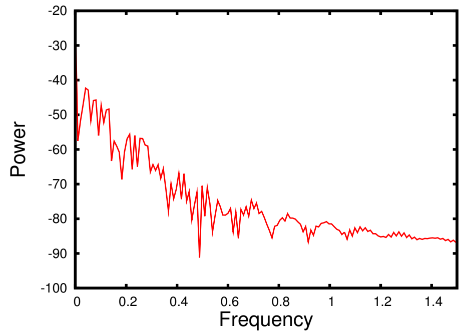
(c)

Figure 3.8: Velocity, PSD and phase portrait with 81^3 grid points, $Re = 2000$, $\beta = 50$; (a) Velocity, (b) Power spectral density, (c) Phase portrait.

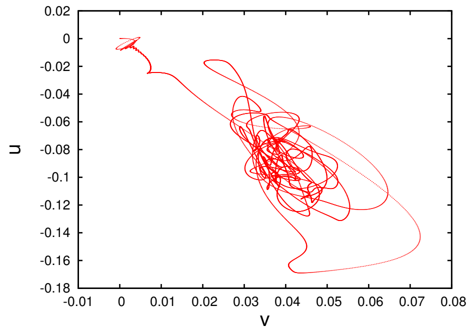
In Fig. 3.8, the flow seems periodic from the time series alone; but some peaks go over the green line indicating that the magnitude is not a constant. The power spectral density shows harmonics with different intervals, and the phase portrait also shows the flow is "mildly" quasiperiodic, consistent with experimental results of Liberzon *et al.* [39].



(a)



(b)



(c)

Figure 3.9: Velocity, PSD and phase portrait with 81^3 grid points, $Re = 2000$, $\beta = 100$; (a) Velocity, (b) Power spectral density, (c) Phase portrait.

There is no repeating structure in the time series shown in Fig. 3.9; the intervals of harmonics are obviously not the same in Fig. 3.9(b); and no clear-cut cycles exist in Fig. 3.9(c), the phase portrait. All indicate that the computed flow is turbulent, which is inconsistent with experimental results.

Through Fig. 3.1 to Fig. 3.9, it is obvious that flow status changed with different filter parameter values. It is easy to get a wrong flow status conclusion with improper β.

3.2.3 Additional simulation results

In the preceding sections, simulation results are analyzed at $Re = 1000$ with 11^3 grid points and $Re = 2000$ with 81^3 grid points. In this section further simulation results are presented to prove that improper filter parameter values result in incorrect prediction of flow status.

Table 3.1: Flow status at different Re with different β

Grid=11^3	$Re = 500$	$Re = 1000$	$Re = 1500$	$Re = 2000$
β=2	steady	steady	steady	steady
β=100	steady	steady	steady	steady
β=500	steady	steady	steady	steady
β=1000	steady	steady	steady	steady
β=1800	steady	periodic	-	
β=2000	steady	quasiperiodic	-	
β=3000	steady	quasiperiodic with noise	turbulent	turbulent
β=5000	steady	turbulent	turbulent	turbulent
Grid=81^3	$Re = 500$	$Re = 1000$	$Re = 1500$	$Re = 2000$
β=2	steady	steady	steady	steady
β=25	steady	steady	steady	periodic
β=50	steady	steady	steady	quasiperiodic
β=100	steady	steady	periodic	turbulent
β=500	steady	steady	-	turbulent
β=1000	steady	steady	quasiperiodic	turbulent
β=3000	steady	steady	-	turbulent
β=5000	steady	steady	-	turbulent

From the above table, it can be seen that with 11^3 grid points, the simulation results are not as accurate as results computed with 81^3 grid points. The way to gage accuracy is the physical flow status. For example, when $Re = 1000$, flow should be steady, but the simulation results with 11^3 grid points show some incorrect behaviors; when $Re = 2000$ with 81^3 grid points, incorrect behaviors are also showed. The flow experienced steady, period, quasiperiodic and turbulent, which indicates that β can be regarded as a bifurcation parameter. On one hand, how many grid points employed affects the results; on the other hand, an improper filter parameter value will produce wrong results.

3.2.4 Selection of optimal β

It is clear from the mathematics of the Navier–Stokes equations that solutions become less smooth as *Re* increases (see, e.g., Foias *et al.* [44]). This implies that in order to get an adequate discretization, more terms are needed in Fourier representations, and hence, more grid points are required. In turn, this suggests that for if grid is fixed, increasing *Re* will result in increased aliasing, so more filtering (smaller values of β) will be necessary. Conversely, for fixed *Re*, as the number of grid points is decreased, aliasing can be expected to become more prevalent.

Discretization errors—simple truncation errors—are also present, and it is not straightforward to establish to what extent these are interacting with aliasing. Moreover, the Shuman filter produces its own truncation error, which depends on β, as already described. In order to minimize the total error, an optimal β needs to be selected.

The N.–S. equations are nonlinear; therefore, with each new time step of a numerical solution, they may generate new, higher Fourier modes. Except for highly-resolved DNS calculations this will result in aliasing if *Re* is sufficiently high. It is difficult to predict the degree to which this occurs in detail. Moreover, aliasing of sufficiently high magnitude is sufficient to destabilize evolution of a numerical solution. On the other hand, if it is controlled, but not entirely removed, a robust numerical scheme may remain stable but produce completely wrong solutions. The method used in this thesis exhibits this property (as shown in the preceding sections), which means that the optimal values of β are needed to lead to at least qualitatively correct solutions with respect to experimental results and those of relatively low-*Re* DNS.

Analysis of truncation error, however, is fairly straightforward since computed solutions have been mollified. Thus, our approach to finding optimal β values is to set the grid spacing and Re. Then calculate solutions for a range of β so as to find the one that produces the smallest solution residual in the L^2 norm, that is, $R(h, Re, \beta)$, which is related to grid space, the Reynolds number and the filter parameter value. Then seek the value of β for which this is smallest (see Appendix A to obtain L^2 norm). Observe that this residual is discrete. Hence, its value arises from iteration error arising from iteratively (at each time step) solving the N.–S. equations and from the truncation error of the filter. An absolute iteration tolerance of 10^{-6} which should be smaller than the filter truncation error is employed. There is an additional truncation error arising from numerical approximation of the above integral, but for fixed grid spacing and Re this is expected to be relatively constant. Thus, the residuals are expected to depend almost entirely on truncation error induced by the Shuman filter.

When an optimal value of β is found in this manner, the corresponding solution is compared with experimental and/or DNS results to verify correct qualitative behavior.

Generation of numerical data

In this section, it is tried to find the optimal β using the approach suggested above in conjunction with experimental results. The Reynolds number is global in this research, e.g., $Re = UL/\nu$, where U is the global velocity, and L is the length of lid driven cavity. The errors are calculated using trapezoidal method. The standard to judge the best β is the minimal error. In this thesis the best β values for grid sizes 11^3, 21^3, 41^3, 51^3 and 81^3 at Reynolds numbers 1500, 1700 and 2000 are found. Part of these results is shown as in Tables 3.2 to 3.4. Define $\|x\|$ to measure the size of a vector x; here we use the L^2 norm, $||x||_2 = \sqrt{{x_1}^2 + {x_2}^2 + \cdots {x_N}^2}$, where x_1 to x_N are the components of x. In the analysis of L^2-norm error here, $\|x\|$ represents the residual error. The residual error should be the value of the left side minus right side of Eq. (3.6b) since the values on both sides are not equal anymore.

Table 3.2: L^2 error with different β values

Grid	Re	β	L^2 norm error
11	1500	875	5.644E-05
11	1500	895	5.614E-05
11	1500	900	5.607E-05
11	1500	905	5.600E-05
11	1500	<u>910</u>	<u>5.591E-05</u>
11	1500	915	5.632E-05
11	1500	925	6.244E-05
11	1500	1000	7.300E-05
21	1500	2500	2.491E-05
21	1500	2550	2.489E-05
21	1500	2600	2.48740E-05
21	1500	2610	2.48719E-05
21	1500	<u>2620</u>	<u>2.48716E-05</u>
21	1500	2630	2.48744E-05
21	1500	2650	2.490E-05
21	1500	2700	2.499E-05
41	1500	500	1.273E-05
41	1500	700	1.259E-05
41	1500	900	1.254E-05
41	1500	2000	1.245E-05
41	1500	3000	1.239E-05
41	1500	4000	1.235E-05
41	1500	5000	1.233E-05
41	1500	8000	1.231E-05

Table 3.3: L^2 error with different β values

Grid	*Re*	β	L^2 norm error
11	1700	810	5.653E-05
11	1700	830	5.618E-05
11	1700	<u>840</u>	<u>5.612E-05</u>
11	1700	850	6.563E-05
11	1700	875	7.032E-05
21	1700	1290	2.4971E-05
21	1700	1295	2.4966E-05
21	1700	<u>1300</u>	<u>2.4964E-05</u>
21	1700	1305	2.4969E-05
21	1700	1310	2.4986E-05
41	1700	1000	1.230E-05
41	1700	2000	1.221E-05
41	1700	3000	1.217E-05
41	1700	4000	1.214E-05
41	1700	5000	1.214E-05
41	1700	8000	1.208E-05

In order to show simulated flow is steady, plots of velocity in the x direction in terms of time (shown in next section) are made. In Tables 3.2 and 3.3, Reynolds numbers are 1500 and 1700, respectively. The minimal L^2-norm error which corresponds to the best β for different grid spacing from these tables is shown underlined.

Whenever solutions to a problem are obtained via numerical approximation, it is necessary to investigate the accuracy of the solutions. The theoretical ratio of the errors for two different step sizes is known to be simply [45]

$$\frac{e_i^{rh}}{e_i^h} = r^{q_1}. \tag{3.1}$$

where e_i is the dominant error, h and rh are the step sizes, and r is the step size ratio. In our research, $r = 1/2$ which means a reduction in the step size by a factor of two, and q_1, the order of accuracy, needs to be known.

All the error in this research is calculated through the equation

$$e = \sqrt{\sum_{i=1}^{i=N} e_i^2}. \tag{3.2}$$

The sum of error at all points needs to be known. So the accurate equation should be

$$e = \sqrt{\sum_{i=1}^{i=N} e_i^2 h} = \sqrt{\sum e_i^2}\sqrt{h}. \tag{3.3}$$

From Table 3.4, choose $\beta = 1000$, grid points 21^3 ($h = 0.05$) and 41^3 ($h = 0.025$) as an example. We get $e_h/e_{h/2}$ is around 3.45, where e_h is the summation of $R(0.05, 2000, 1000)$ at all points, and $e_{h/2}$ is the summation of $R(0.025, 2000, 1000)$ at all points. Through $(1/r)^{q_1} = 1/3.45$, where $r = 2$, we get $q_1 \simeq 2$, so the accuracy of the scheme is second order. This indicates that Shuman filter is second order accuracy, and if higher-order accuracy is required, then Shuman filter is not a good option.

Table 3.4: L^2-norm error with different β

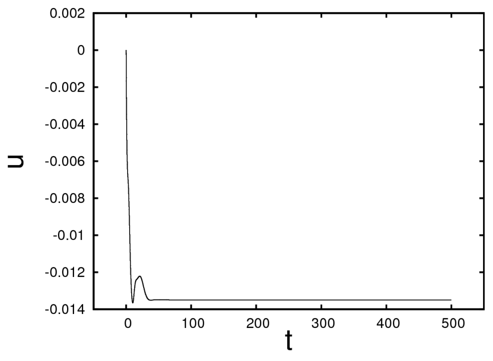
11	2000	1000	7.2450093E-05
11	2000	1100	7.4679097E-05
11	2000	1150	6.9222246E-05
11	2000	1170	6.4424952E-05
11	2000	1175	6.0191185E-05
11	2000	1170	6.5623688E-05
11	2000	1200	7.0535665E-05
21	2000	600	2.6201233E-05
21	2000	700	2.5623269E-05

21	2000	800	2.5174551E-05
21	2000	810	2.5134406E-05
21	2000	820	2.5097004E-05
21	2000	<u>830</u>	<u>2.5064319E-05</u>
21	2000	840	2.5235737E-05
21	2000	900	2.8475570E-05
21	2000	1000	2.9091771E-05
21	2000	1100	2.9451430E-05
21	2000	1300	2.9747631E-05
21	2000	1400	3.0740241E-05
41	2000	1000	1.1939765E-05
41	2000	1500	1.1838957E-05
41	2000	2000	1.1753959E-05
41	2000	2100	1.1748787E-05
41	2000	2130	1.1748333E-05
41	2000	<u>2140</u>	<u>1.1748234E-05</u>
41	2000	2150	1.1748326E-05
41	2000	2150	1.1748508E-05
41	2000	2250	1.1753145E-05
41	2000	2500	1.1791923E-05
41	2000	3000	1.1891751E-05
41	2000	4000	1.1883767E-05
81	2000	5000	5.4794937E-06
81	2000	4000	5.3762151E-06
81	2000	3000	8.1433909E-06
81	2000	2500	5.3700232E-06
81	2000	1700	7.1109876E-06
81	2000	1600	5.5168016E-06
81	2000	<u>1500</u>	<u>5.2395458E-06</u>
81	2000	1450	7.1159921E-06
81	2000	1400	5.5401320E-06
81	2000	1350	7.2279677E-06

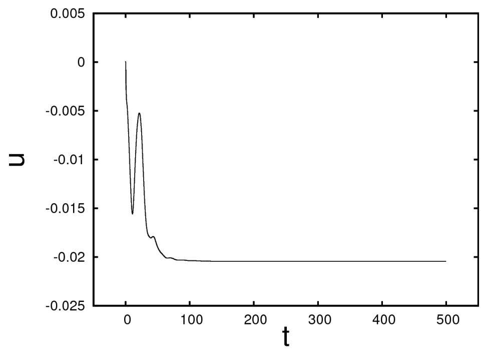
81	2000	1200	6.3525194E-06
81	2000	1000	5.2403352E-06
81	2000	800	6.9271482E-06

Time series and PSDs

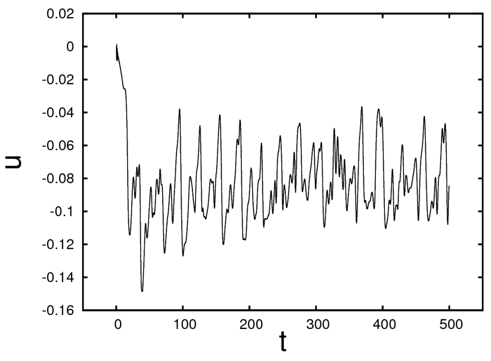
Figs. 3.10 and 3.11 show the dimensionless time evolution of the u velocity component *vs* optimal filter parameter β.



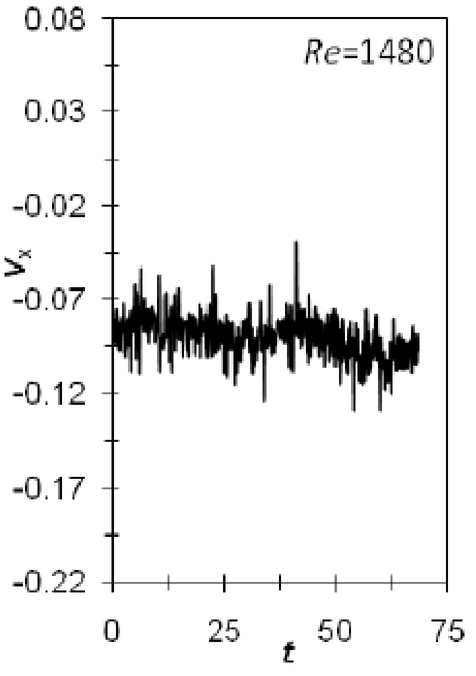
(a)



(b)



(c)

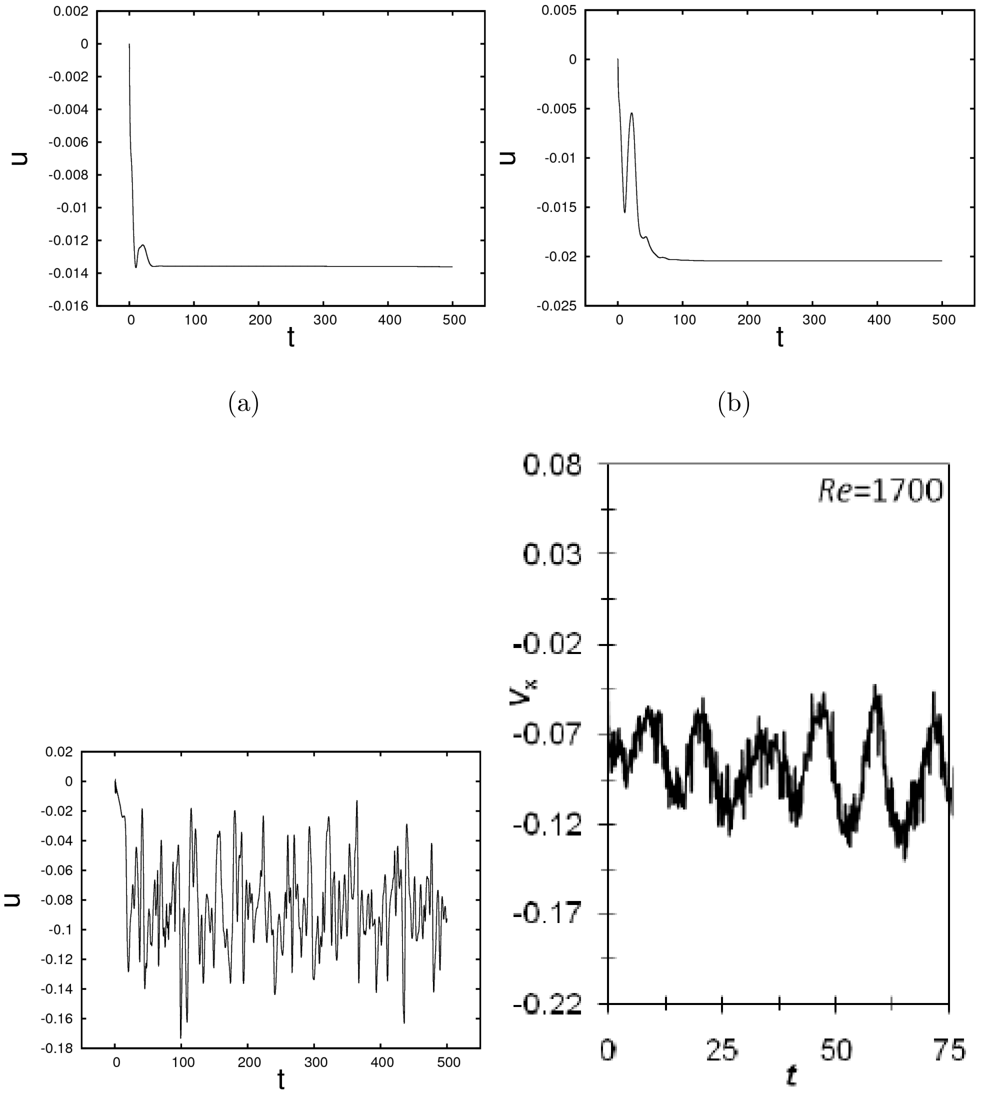


(d)

Figure 3.10: Dimensionless time evolution of the u velocity component with optimal β, $Re = 1500$; (a) grid spacing=0.1, $\beta = 910$, (b) grid spacing=0.05, $\beta = 2620$, (c) grid spacing=0.0125, $\beta = 1700$, (d) experimental results [39].

From the experimental results [39], the state for $Re = 1500$ should be steady, and this is also supported by theory [46]. This also holds in Figs. 3.10 (a) and (b). However, it is not the case in (c). In the experimental results (d), there are some small oscillations and they are noises from the experiment device proved in [39].

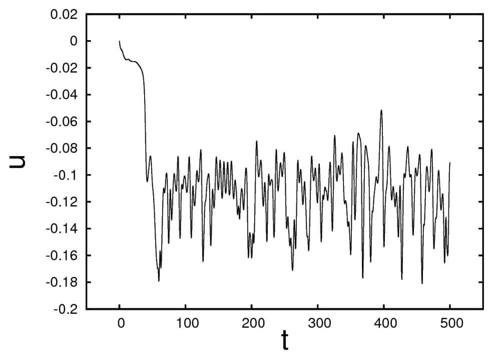
The calculation of L^2 norm-error is not the only standard needed to choose the optimal value of β. In order to choose the best β, experimental results must be used as reference.



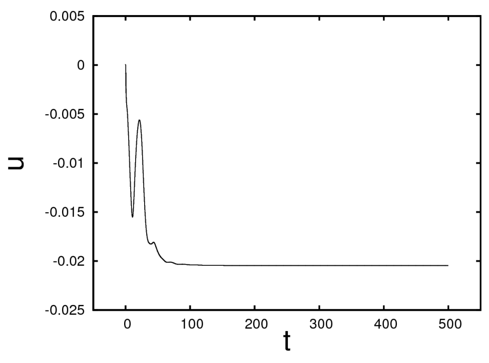
(a) (b) (c) (d)

Figure 3.11: Dimensionless time evolution of the u velocity component with optimal β, Re = 1700; (a) grid spacing=0.1, $\beta = 840$, (b) grid spacing=0.05, $\beta = 1300$, (c) grid spacing=0.0125, $\beta = 1700$, (d) experimental results. [39]

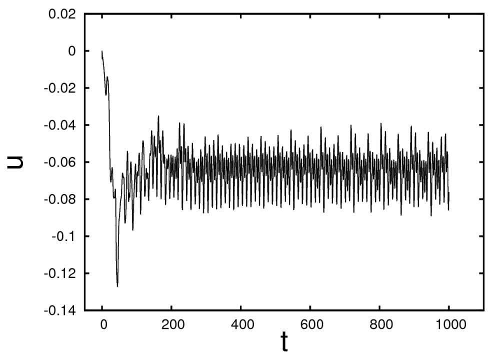
From Fig. 3.11, in cases (a) and (b), i.e. grid points 11^3 and 21^3, the flow is steady [39]. This indicates that the velocities have been smoothed too much.



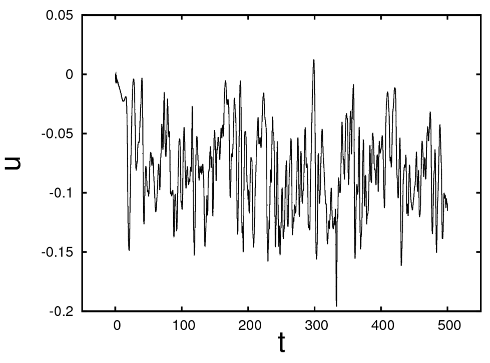
(a)



(b)



(c)

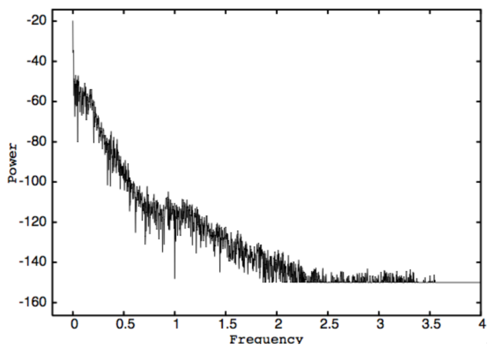


(d)

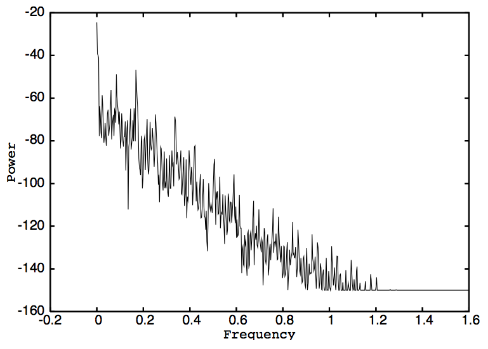
Figure 3.12: Dimensionless time evolution of the u velocity component with optimal β, Re = 2000; at the cavity middle plane (a) grid spacing=0.1, $\beta = 1175$ (b) grid spacing=0.05, $\beta = 830$ (c) grid spacing=0.025, $\beta = 2140$ (d) grid spacing=0.0125, $\beta = 1500$

The location used in above figures still is $(0.175, 0.122, 0.5)$, corresponding to the experimental location $(-0.325, -0.378, 0)$ in [39]. Note that in the case $Re = 1500$, no matter which spacing size is used, the velocity is steady except at beginning of time. But in case $Re = 2000$, when grid spacing is 0.1, it can be seen the solution is quasiperiodic (oscillations that appear to follow a regular pattern but which do not have a fixed period); for grid spacing 0.05, it is steady; for grid spacing 0.025, it is noisy periodic.

From previous research, further increase of the Reynolds number up to $Re = 2000$ leads to the flow oscillations with a significantly larger amplitude. Solution behavior is the same as before: when 21^3 grid points are used, and the optimal β is chosen according to the L^2-norm error, the flow is steady; this is not in accordance with experimental data.



(a)



(b)

Figure 3.13: Power *vs.* frequency; (a) $Re = 2000$, h=0.1, optimal β=1175, (b) $Re = 2000$, h=0.025, β=2140

Next, use power spectral density to make a more detailed analysis. Fig. 3.13 shows power *vs.* frequency for grid spacings 0.1 and 0.025. Compute the same time series length, and the same number of points (8192) for the PSD for both cases. β=1175 and β=2140 are chosen here because they produced the minimal L^2-norm error respectively. The PSD analysis further proves that the flow status is consistent with the times series result.

Relation among Re, grid spacing, optimal β

In this analysis, two cases are studied, $Re = 1500$ and $Re = 2000$. In each case, consider four different grid sizes, 11^3, 21^3, 41^3 and 81^3. In the preceding subsection, optimal β under various conditions is found. In Fig. 3.14 the optimal filter parameter value is plotted (based on minimum L^2 norm of error) versus grid spacing. Grid points 11^3, 21^3, 41^3 and 81^3 correspond to grid spacings 0.1, 0.05, 0.025 and 0.0125, respectively.

It can be seen that for the case $Re = 1500$, optimal β decreases monotonically with increasing grid spacing. In Fig. 3.14 it can be seen that when grid spacing is 0.025 optimal β is very large indicating almost no need to filter. In case 2 ($Re = 2000$), the optimal parameter decreases monotonically until a minimum is reached, and then rises monotonically. This possibly can be explained by viewing discrete solutions as if they were weak analytical solutions. Initially, as grid spacing is decreased the solutions appear to be more irregular; thus a smaller filter parameter corresponding to stronger filtering is required. When nearly complete resolution is achieved, further decrease in grid spacing leads to more regular behavior and an attendant ability to employ less filtering indicating a higher filter parameter value.

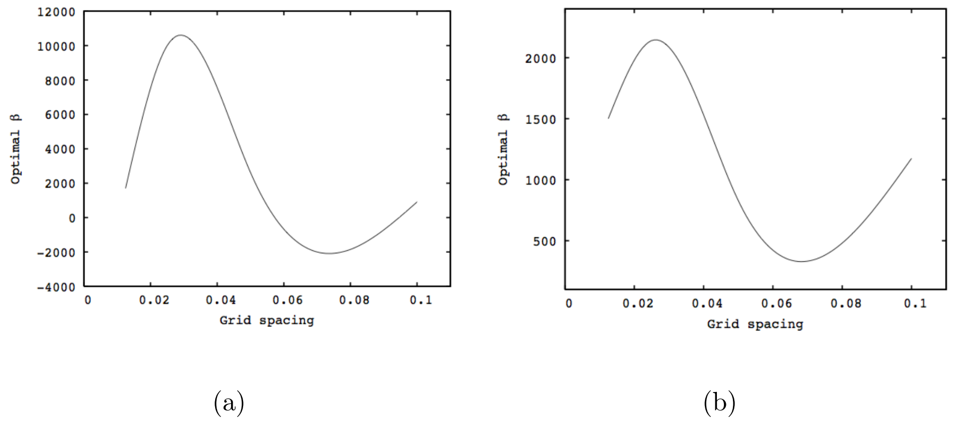
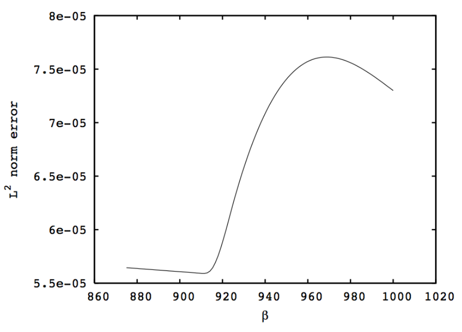
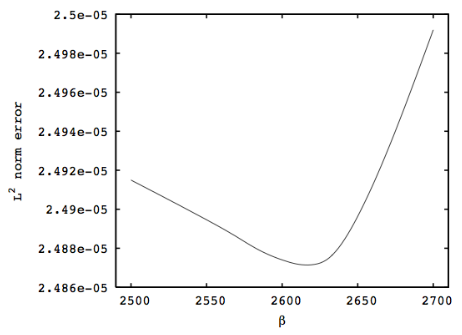


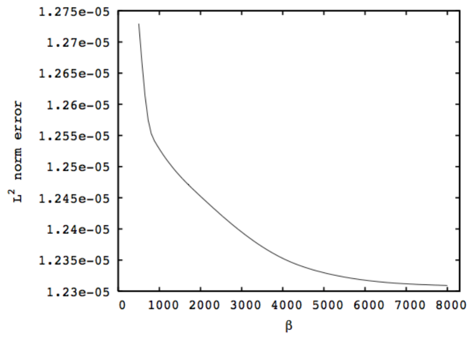
Figure 3.14: Optimal filter parameter value *vs.* grid spacing in two cases; (a) $Re = 1500$, (b) $Re = 2000$.



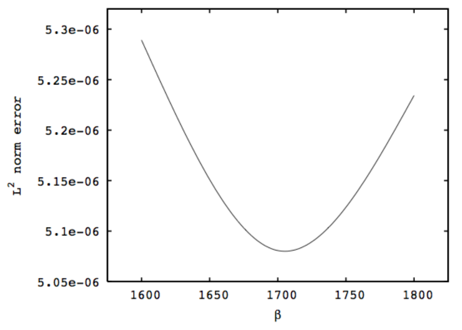
(a)



(b)



(c)



(d)

Figure 3.15: L^2-norm error *vs.* filter parameter, $Re = 1500$; (a) $h = 0.1$, (b) $h = 0.05$, (c) $h = 0.025$, (d) $h = 0.0125$.

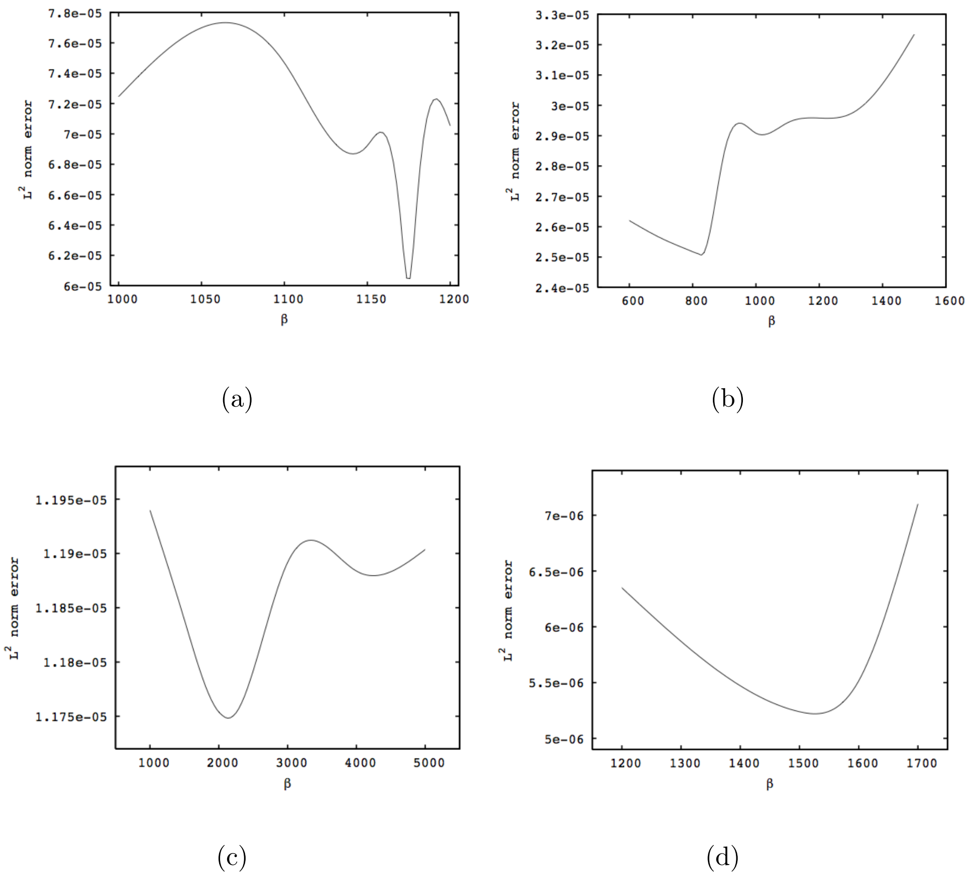


Figure 3.16: L^2-norm error *vs.* filter parameter, $Re = 2000$; (a) $h = 0.1$, (b) $h = 0.05$, (c) $h = 0.025$, (d) $h = 0.0125$.

Figures 3.15 and 3.16 display the L^2-norm error plotted against the filter parameter value β. It is readily observed that filtering error decreases monotonically with grid spacing by comparing parts (a) through (d). But once the grid spacing is sufficiently small to provide nearly full resolution, the optimal filter value begins to increase slightly. The L^2 norm of error here is total error including all grid points. This also shows one disadvantage of the Shuman filter: it is hard to choose an optimal filter parameter value and hard to give an equation to get it.

3.3 Comparison between Shuman and Padé filters

This section makes direct comparisons between Shuman and Padé filters. It is based on two parts; one is a simple perturbed sine wave in 1D and 3D; the other one is the 3-D lid-driven cavity.

3.3.1 Applications of filters to perturbed sine wave

Shuman and Padé filters in 1D

In this section, both Shuman and Padé filters are applied to a perturbed sine wave. In order to investigate the two filters' effect, three different types of noise are added to the this function.

The first perturbation is produced by adding terms analogous to Fourier aliasing. In mathematics, a Fourier series represents functions, or signals, as the sum of a set of (usually) simple oscillating functions, namely sines and cosines. Therefore, the Fourier aliasing can be constructed as

$$0.1\cos\left(\frac{2N}{3}t(i)\right) + 0.05\sin\left(\frac{3N}{5}t(i)\right), \quad i = 1, 2, \ldots, n_s,$$

where n_s is the sample size (taken to be 101), and N is a number larger than the sample size used for a discrete reconstruction. The perturbed sine wave becomes,

$$u(i) = s(i) + 0.1\cos\left(\frac{2N}{3}t(i)\right) + 0.05\sin\left(\frac{3N}{5}t(i)\right), \tag{3.4}$$

where $N = 101$ will be used herein, and s is the pure sine wave. Constants 0.1 and 0.05 are used to adjust the relative amplitudes of added noise. This form was chosen to represent the fact that modes leading to aliasing have wavenumber starting at $2N$, but they affect all lower wavenumbers; see, e.g., Ames [23].

The second noise is produced randomly by averaging (pseudo) random numbers r_1

and r_2 obtained from the Fortran 90 intrinsic subroutine $RANDOM\_NUMBER$, [47]. Then the perturbed sine wave is,

$$u(i) = s(i) + 0.5(r_1(i) + r_2(i)), \quad i = 1, 2, \ldots, 101. \tag{3.5}$$

The third perturbation is generated by a linear combination of the former two.

Shuman and Padé filters are applied separately to filter these noises. Filters are applied 100000 times to the same signal in order to produce better timing results. In particular, the Fortran timer employed is the intrinsic subroutine SECNDS which has only millisecond resolution. In a Gflops context, the filters are studied must be run many times to produce reliable timing results. But in a sense, this is not unrealistic because a typical CFD calculation will be run for many thousands of time steps, with filtering required for each of these.

In most research, the Shuman filter parameter value is usually set to $\beta = 2$ for reasons that are obvious from Fig. 2.2; it is adhered to this in the present work. As can be seen from Eq. (2.8) and Fig. 2.2, if a large filter parameter is employed, the filtering effect would be very small; use of $\beta = 2$ ensures that sufficient filtering is applied to the perturbed sine wave to remove all wavenumber contributions beyond those actually computed.

Table 3.5: Error and execution time for the two filters

Fourier aliasing only	
Induced error	7.8482×10^{-3}
Error after Shuman filtering	4.0897×10^{-3}
Error after Padé filtering	4.8700×10^{-3}
Shuman filter execution time	0.2969
Padé filter execution time	0.9648
Random number generator only	
Induced error	2.1978×10^{-3}
Error after Shuman filtering	2.2969×10^{-3}
Error after Padé filtering	1.8594×10^{-3}
Shuman filter execution time	0.3125
Padé filter execution time	0.8789
Combination	
Induced error	7.8730×10^{-3}
Error after Shuman filtering	4.1631×10^{-3}
Error after Padé filtering	4.8608×10^{-3}
Shuman filter execution time	0.2891
Padé filter execution time	0.8750

Two standards used in this analysis to assess filter performance are execution time and error after application of the filter. Results are summarized in Table 3.5 if only Fourier noise exists, error added to the pure sine wave (termed "Induced error" in the above table) is 7.8482×10^{-3}. The Shuman filter removes approximately half of this error; and the Padé filter also decreases the error, but not as effectively. If only random error exists, error added to the pure sine wave is 2.1978×10^{-3}. The

Shuman filter does not decrease this error; but the Padé filter does, and this is the only case where the Shuman filter works worse than the Padé filter, as can be seen in Table 3.5. It is easily seen by comparing arithmetic operations that the Shuman filter is significantly more efficient than the Padé filter, even when several passes of the former are used. The third case included in the table is a combination of the two forms of noise; similar to the first case, the Shuman filter decreases the error and is more efficient than the Padé filter. Again, the Shuman filters require significantly less execution time than does the Padé filter, approximately a factor of three.

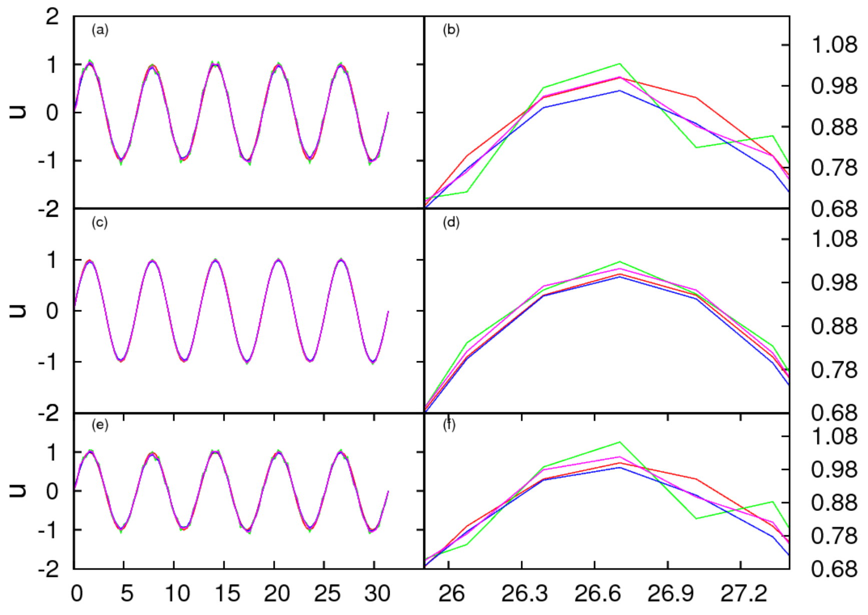


Figure 3.17: Comparison of u *vs.* t with only Fourier aliasing, only random and both; (a) only Fourier aliasing, (b) zoomed in, (c) only random, (d) zoomed in, (e) combination, (f) zoomed in. Red line, pure sine wave; green line, aliasing; blue line, Shuman filter; purple line, Padé filter.

As shown in Fig. 3.17, no matter which type of error is chosen, both Shuman and Padé filters work reasonably well for this 1-D problem. Figures 3.17 (a)(c)(e) show that after application of either Shuman or Padé filters, the filtered perturbed waves nearly overlap the original (noise-free) pure sine wave. Nevertheless, the perturbation is still obvious in the zoom ins; but both Shuman and Padé filters smooth the results. Remark that the Shuman filter seems to underestimate the pure sine wave, while Padé filter leads to overshot. One should expect that somewhat different choices of filter parameters would result in different overall behaviors of both filters.

Application of Shuman and Padé filters in 3D

In this section, the same three types of noise previously added to the pure sine wave in 1D are used, in three directions as shown in the following

$$u(i) = s(i) + 0.1\cos\left(\frac{2N}{3}t(i)\right) + 0.05\sin\left(\frac{3N}{5}t(i)\right)$$

$$v(i) = s(i) + 0.1\cos\left(\frac{2N}{3}t(i)\right) + 0.05\sin\left(\frac{3N}{5}t(i)\right)$$

$$w(i) = s(i) + 0.1\cos\left(\frac{2N}{3}t(i)\right) + 0.05\sin\left(\frac{3N}{5}t(i)\right)$$

in the first case;

$$u(i) = s(i) + 0.5(r_1(i) + r_2(i))$$

$$v(i) = s(i) + 0.5(r_1(i) + r_2(i))$$

$$w(i) = s(i) + 0.5(r_1(i) + r_2(i))$$

in the second case;

$$u(i) = s(i)0.1\cos\left(\frac{2N}{3}t(i)\right) + 0.05\sin\left(\frac{3N}{5}t(i)\right) + 0.5(r_1(i) + r_2(i))$$

$$v(i) = s(i)0.1\cos\left(\frac{2N}{3}t(i)\right) + 0.05\sin\left(\frac{3N}{5}t(i)\right) + 0.5(r_1(i) + r_2(i))$$

$$w(i) = s(i)0.1\cos\left(\frac{2N}{3}t(i)\right) + 0.05\sin\left(\frac{3N}{5}t(i)\right) + 0.5(r_1(i) + r_2(i))$$

in the third case. Shuman and Padé filters are separately applied to these noise types. Both filters decrease the effect of aliasing, but from the error shown in the tables below, it is evident that Padé filters are less efficient in 3D; the Shuman filter still works well in 3D. The advantage of the Shuman filter on execution time is more pronounced in 3D. Its execution time is approximately one ninth that of the Padé filter's, as can be seen in Table 3.6, and as should be expected based on 1-D timings. In general, pentadiagonal banded-matrix systems must be solved, line by line, in each of the three directions for the Padé filter being used here. Although the required arithmetic is $\mathcal{O}(N)$, where N is the total number of points, it is still significantly greater than required by explicit applications of the Shuman filter.

Table 3.6: Error and execution time for 3-D signal

Fourier aliasing only	
Induced error	0.6789
Error after Shuman filtering	0.4386
Error after Padé filtering	0.6053
Shuman filter execution time	0.0547
Padé filter execution time	0.4102
Random number generator only	
Induced error	0.1815
Error after Shuman filtering	0.1392
Error after Padé filtering	0.1678
Shuman filter execution time	0.0547
Padé filter execution time	0.4258
Combination	
Induced error	0.7120
Error after Shuman filtering	0.4672
Error after Padé filtering	0.6327
Shuman filter execution time	0.0547
Padé filter execution time	0.4219

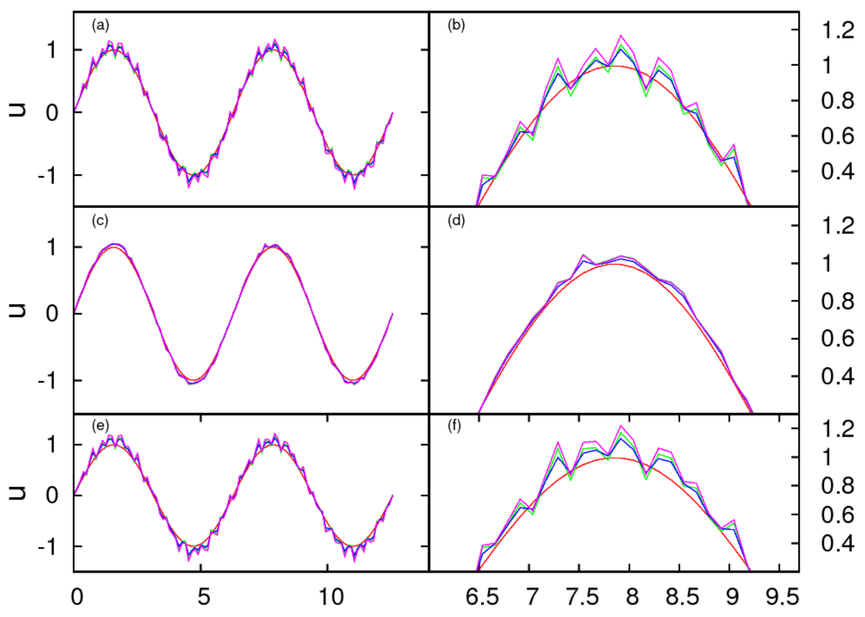


Figure 3.18: Comparison of u $vs.$ t with only Fourier aliasing, only random and both; (a) only Fourier aliasing, (b) zoomed in, (c) only random number perturbation, (d) zoomed in, (e) combination, (f) zoomed in. Red line, pure sine wave; purple line, aliasing; blue line, Shuman filter; green line, Padé filter.

As shown in Fig. 3.18, in the left column, noise is more obvious, and filters are not as effective as in 1D. The right column is the zoomed-in second peak in the left column. The Padé filter tries to follow noise, while the Shuman filter tries to smooth it. The filtering performance is not as effective as in 1D for either filter; but generally, the Shuman filter is superior to the Padé filter on these perturbed sine waves.

3.3.2 Comparison of Shuman and Padé filters: laminar LDC flow

In this section, Shuman and Padé filters are applied to laminar flows in a lid-driven cavity. If a filter is effective, it is supposed to produce flow states consistent with experimental results; besides, it can make the simulation results smooth. The flow in the lid-driven cavity is described by the continuity and momentum equations,

$$\nabla \cdot \boldsymbol{u} = 0 \tag{3.6a}$$

$$\partial \boldsymbol{u}/\partial t + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = -\nabla p + \frac{1}{Re}\Delta \boldsymbol{u}. \tag{3.6b}$$

Define the domain $\Omega \equiv (0, L_x) \times (0, L_y) \times (0, L_z)$ as shown in Fig. 3.19 and let

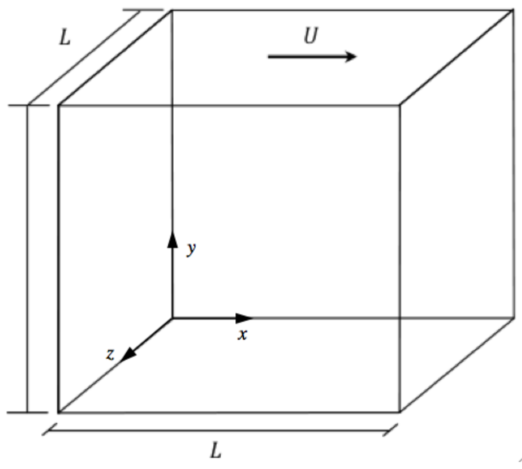


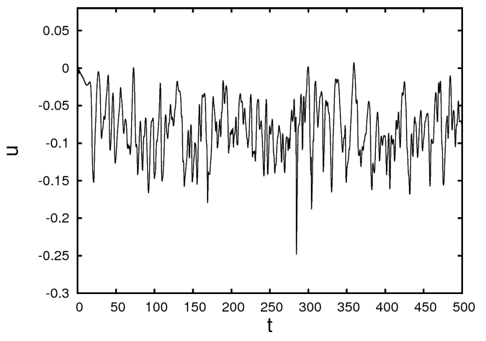
Figure 3.19: Lid-driven cavity

$L_x = L_y = L_z = 1$. Initial conditions are $u = 1$ at $y = 1, u \equiv 0$ in $\bar{\Omega}$ $(x, 1, z)$. Boundary conditions can be described as $u \equiv 0$ at five walls except the top one (no-slip boundary condition).

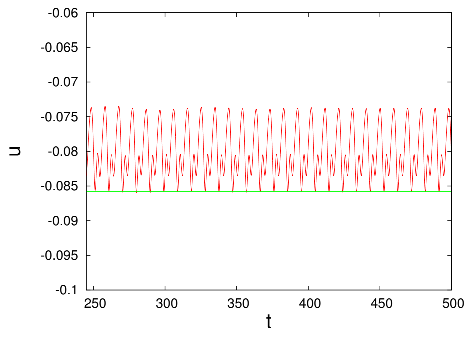
Results with Shuman filter, Re=2000

According to experimental results [39], flow is oscillatory at $Re = 1970$. It is pointed out in [39] that some high-frequency vibrations are caused by the experimental device—the motor driving the lid. But from the time series and power spectral density from Fig. 3.1 at $Re = 1970$, it can be seen that flow is quasiperiodic.

In order to investigate effects of Shuman filter, results without filtering are provided. Plots shown below are at different time steps (total time steps are 40000), e.g., time steps 20000 and 40000. For the present simulations, the time interval, times steps and total time are set to be 0.0125, 40000 and 500 *sec.*, respectively. The power spectral density is calculated from u-velocity time series using data between 400 and 500 *sec* with 81^3 grid points. There are 40000 time points totally and the PSDs use points from 31808 to 39999. First, investigate the time series to see whether fluid appears to be quasi-periodic. In order to have the same physical location investigated in the experiment [39], the point studied here is $(0.175, 0.122, 0.5)$, toward the left bottom of the cavity in the central z plane and with 81^3 grid points. Magnitude of vorticity shown below corresponds to this middle section.



(a)

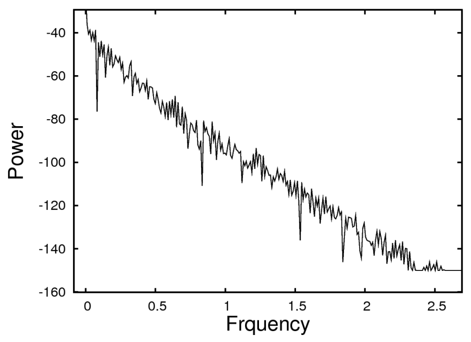


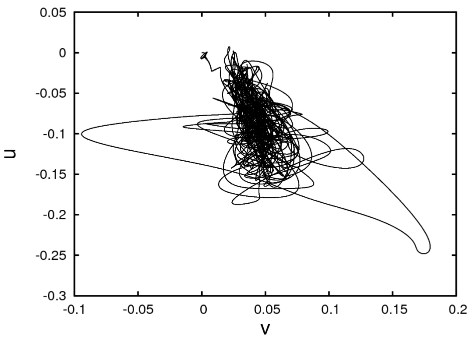
(b)

Figure 3.20: Time series (a) without filtering, (b) with Shuman filter

Figure 3.20 (a) shows solutions are in chaotic oscillation without filtering, which is inconsistent with the physical measurements [39]. In Fig. 3.20 (b), flow is quasi-periodic since the peaks are not exactly repeating, (Shuman filter parameter value is 50) indicating that the Shuman filter can make the solutions close to real phenomena. Making filtered results consistent with the physical phenomenon is only one expected effect; how much Shuman filter can smooth results must also be studied.

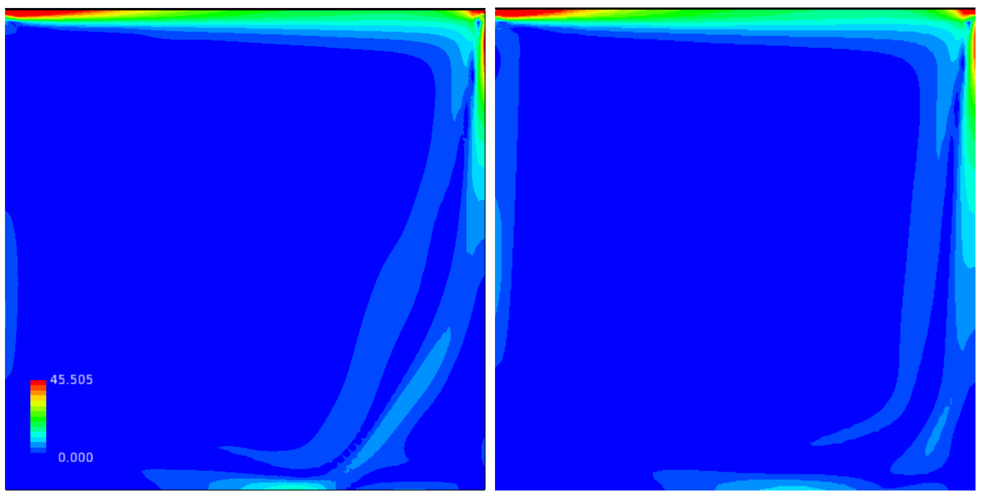
In order to check the status without filters, the power spectral density and phase portrait are analyzed. It is obvious that intervals of harmonics are not the same; the phase portrait also does not show any identifiable cycles, and what can be observed is an interweaved mixture.





(a) (b)

Figure 3.21: $Re = 2000$ without filters; (a) Power spectral density, (b) Phase-portrait

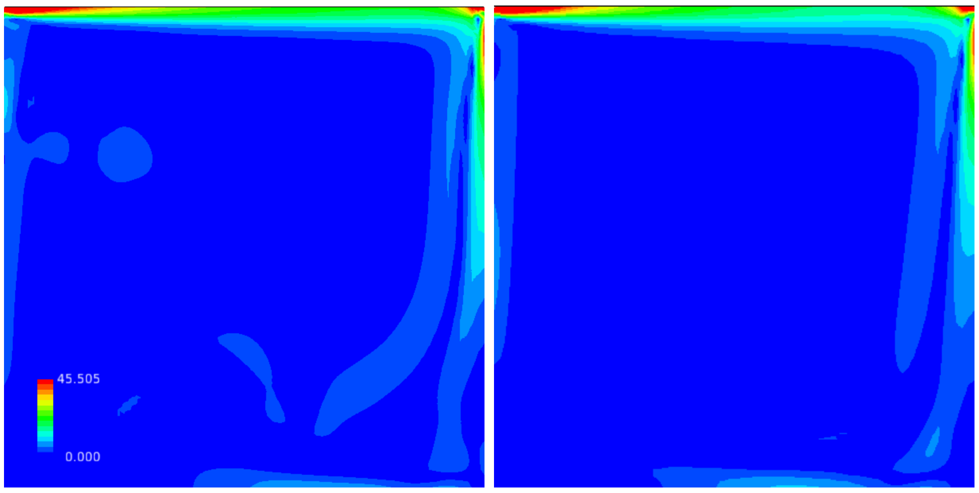


(a) (b)

Figure 3.22: Magnitude of vorticity at time step 20000; (a) without filtering, (b) with Shuman filter.

From Figure 3.22 (a), it can be seen that without filtering, the vorticity near the moving top is large, especially at the left and right corners. When fluids with velocity reach and hit the right wall, large vorticity is produced. At the right bottom corner of the cavity, the vorticity also decreases after application of Shuman filter.

As seen in Figure 3.22 (b), at the early time steps, Shuman filter does make positive effects on smoothness of velocities.



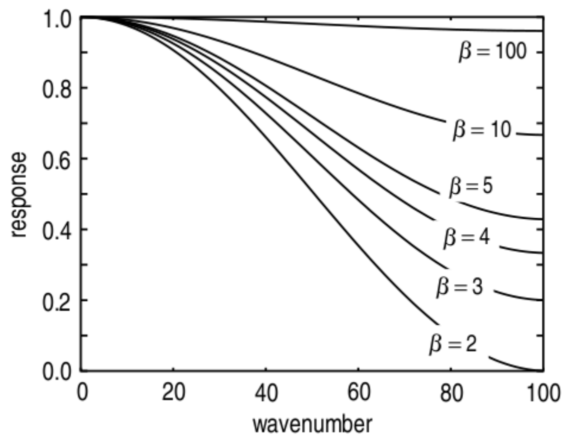
(a) (b)

Figure 3.23: Magnitude of vorticity at time steps 40000; (a) without filtering, (b) with Shuman filter.

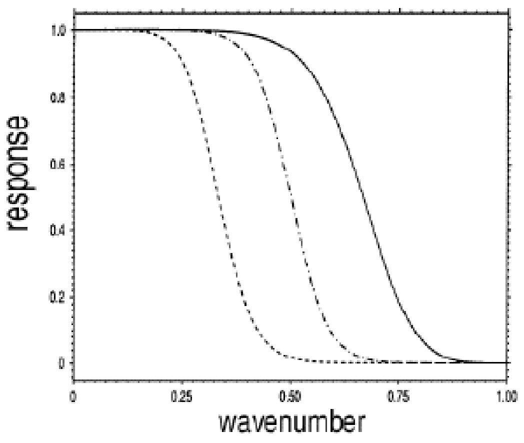
As time evolves, effects of the Shuman filter are obvious. In general, it is observed easily that the whole vorticity decreases (more blue color) with Shuman filter. Secondly, without filtering, there are ‘vorticity spots’ (red area in the figure) at which vorticity is high. Thirdly, at top, bottom and left side of lid-driven cavity, vorticity decreases noticeably. There are also some new ‘weak spots’ appearing after application of Shuman filter. In general, the total vorticity in lid-driven cavity decreases; hence, the Shuman filter does smooth velocities as expected.

Comparison between Shuman and Padé filter

Before the comparison of vorticities with the implementation of the two filters, some main differences between them need be demonstrated. The Shuman filter has only one parameter that needs to be adjusted to change filter’s effect, while Padé filter has several parameters which makes it a bit more difficult to set. The Shuman filter represents a simple linear operator, derivable form formal mathematical mollification, and is a weighted average of nearest neighbor solution values. The Padé filter on the other hand, requires the solution of linear systems of equations (more computational work), but Padé filters give more flexibility in constructing filters which are closer to approximations of sharp cutoff filters (see, e.g., Liu *et al.* [15]). Their wavenumber response is demonstrated by Fig. 3.24 below ((a) McDonough [24] and (b) Vasilyev *et al.* [14]).



(a)

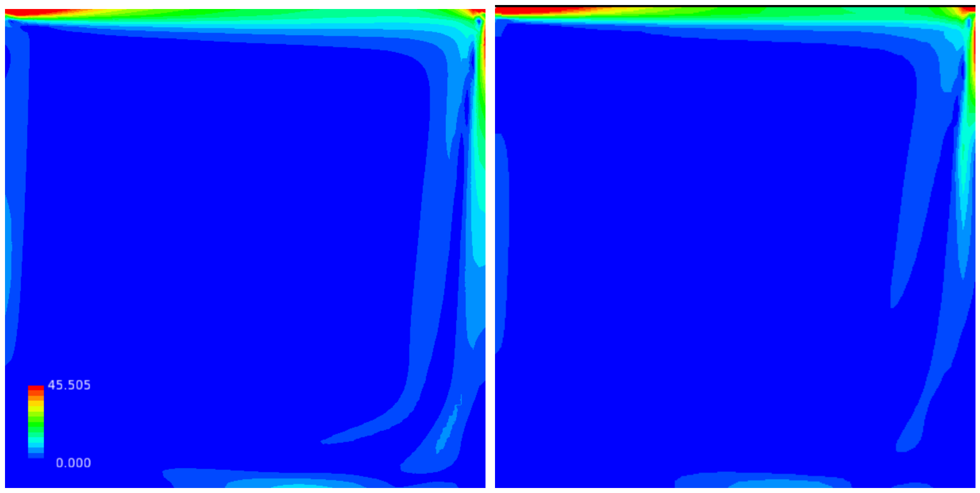


(b)

Figure 3.24: Wavenumber response (a) Shuman filter, (b) Padé filter

Figure 3.24 (b) indicates the Padé filter's good approximation to sharp cutoff. Figure 3.24 (b) provides results for a symmetric Padé filters with five vanishing moments and different linear constraints [14]. From Fig. 3.24 (a) it is clear that the filter parameter in the Shuman filter influences its effectiveness. Previous research, presented earlier in this thesis, has shown that the optimal filter parameter is related to Reynolds number and number of grid points, but not in a straightforward way.

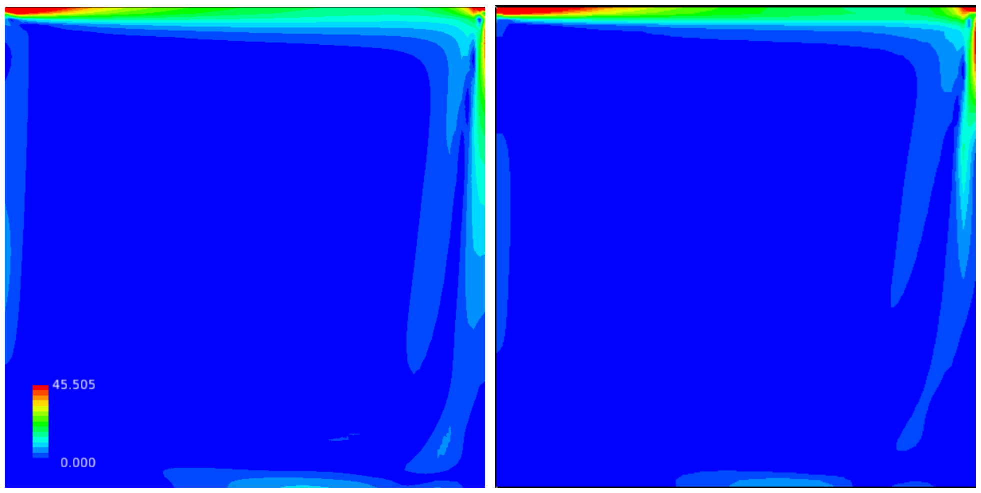
Application of the two filters separately at each time step leads to results shown below.



(a) (b)

Figure 3.25: Magnitude of vorticity at time step 20000; (a) with Shuman filter, (b) with Padé filter.

From Fig. 3.25, it is easy to see that the Padé filter does not improve the smoothness of velocity at the top of lid-driven cavity. However, the Padé filter smooths more than the Shuman filter in other areas, especially at the right bottom corner of the cavity. Compared with the results without application of filters in Fig. 3.22, it can be seen that there is not much improvement with the application of filters.

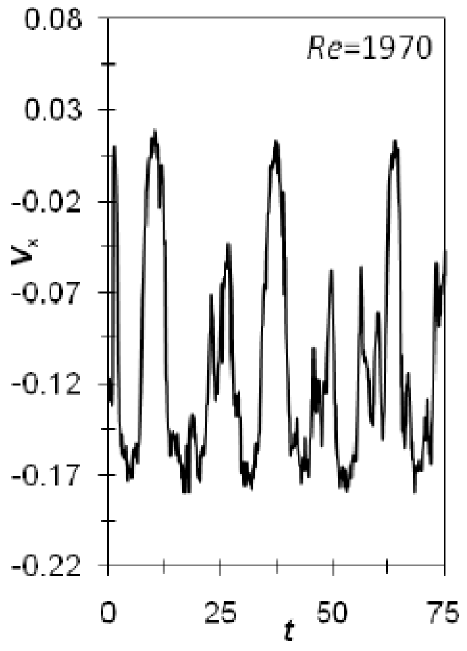


(a) (b)

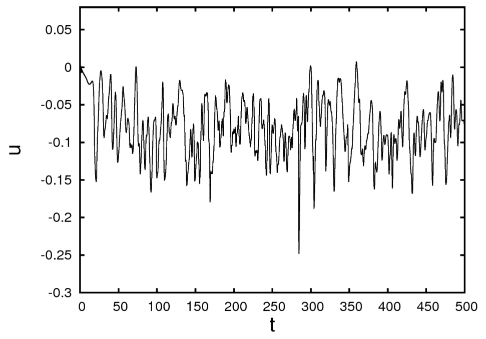
Figure 3.26: Magnitude of vorticity at time step 40000; (a) with Shuman filter, (b) with Padé filter.

At later time steps, e.g, time step 40000, the Padé filter seems not to work better than the Shuman filter generally. At the four corners of the cavity, circulations are obvious and the Padé filter does not work as well as the Shuman filter. But it does reduce the 'vorticity spot' (red area).

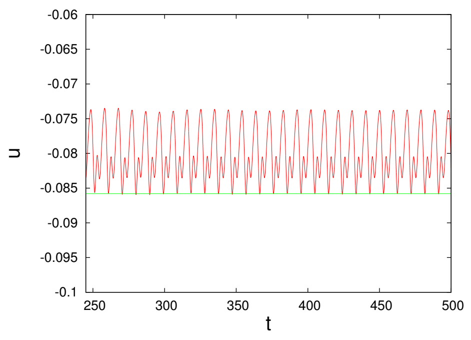
In order to make a further comparison of Shuman and Padé filters, time series of velocities with the Shuman filter, Padé filters and without filters are investigated at $Re = 2000$. From Fig. 3.21 (a) (shown earlier as Fig. 3.1 (a)) experimental results, the time series showing multiple frequencies with noise, indicates that flow is quasiperiodic; if no filter is applied, flow behaves like turbulence or quasiperiodic with noise, as shown in (b); with application of the Shuman filter with $\beta = 50$, flow is quasiperiodic. In part (d), it can be seen that with the Padé filter, the flow is totally steady, which is inconsistent with experimental results. From the above comparison, Shuman and Padé filters both have an effect on the solutions; a proper Shuman filter value can produce good results consistent with experiments. Padé filters in this case filter too strongly, leading to wrong qualitative predictions of flow status.



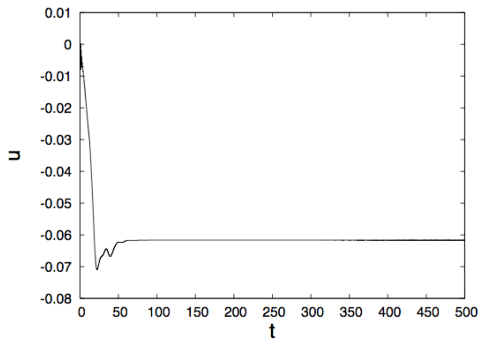
(a)



(b)



(c)



(d)

Figure 3.27: Time series at $Re = 2000$; (a) experimental results [39], (b) without filters, (c) with Shuman filter, (d) with Padé filter.

In order to make a further comparison of Shuman and Padé filters, v component velocities at the mid-plane along the x axis are shown below. It seems that the two filters both work reasonably well, and results are almost consistent with the experimental results. However, it should be noticed that, Padé filters do better than Shuman filters near the maximal velocities, due mainly to higher formal accuracy of the Padé filter.

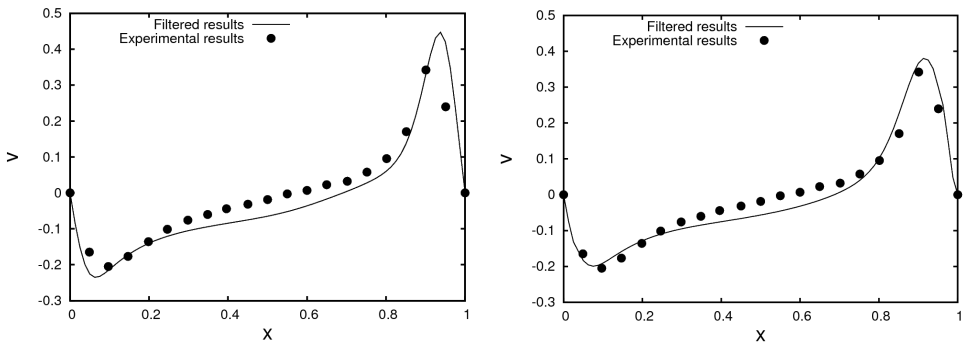


Figure 3.28: Velocity at the mid-plane with filters at $Re = 1500$; (a) Shuman filter, (b) Padé filter.

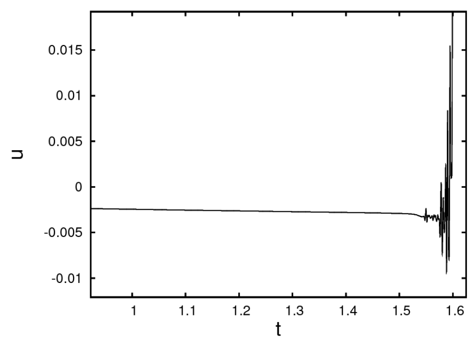
3.3.3 Comparison of Shuman and Padé filters: turbulent LDC flow

In this section, both Shuman and Padé filters are applied to the lid-driven cavity problem for turbulent flow. *Re* is 10000 in this research; obviously the flow is turbulent [39]. Spatial resolution consists of 81^3 grid points, so the grid spacing is 0.0125 *m*, which is rather coarse. The turbulence model employed in this thesis is deconvolution. If deconvolution is applied to the filtered solution, an accurate representation of the filtered nonlinear combination of solution components with discontinuity can be obtained [48]. Deconvolution methods are techniques employed to build sub-grid scale models in large-eddy simulation; the mechanism is extracting information from the highest resolved wavenumber parts of a solution and using this to infer behavior of the lowest wavenumber unresolved parts. This method is based on the assumption "scale similarit", which presumes that the behavior at the lowest wavenumbers of the unresolved part is similar to that of the highest wavenumbers of the resolved scale.

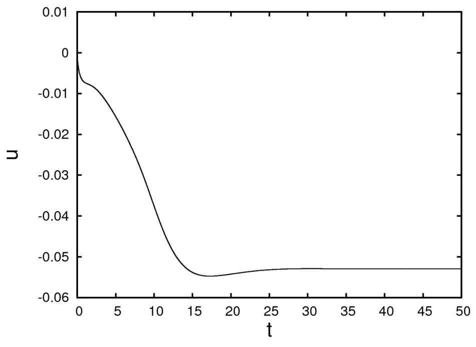
Results with Shuman filter

Exercises of the preceding section are now repeated with a turbulent *Re*; e.g., vorticities are shown without filter, and with Shuman and Padé filtering.

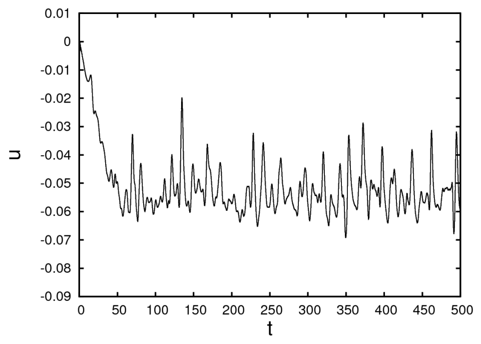
Because of the change of flow status, from laminar to turbulent flow, the application of Shuman filter also changes correspondingly. In order to solve the Navier–Stokes equation, large eddy simulation (LES) is applied in this research. For LES, larger scales are solved directly while small scales are modeled.



(a)



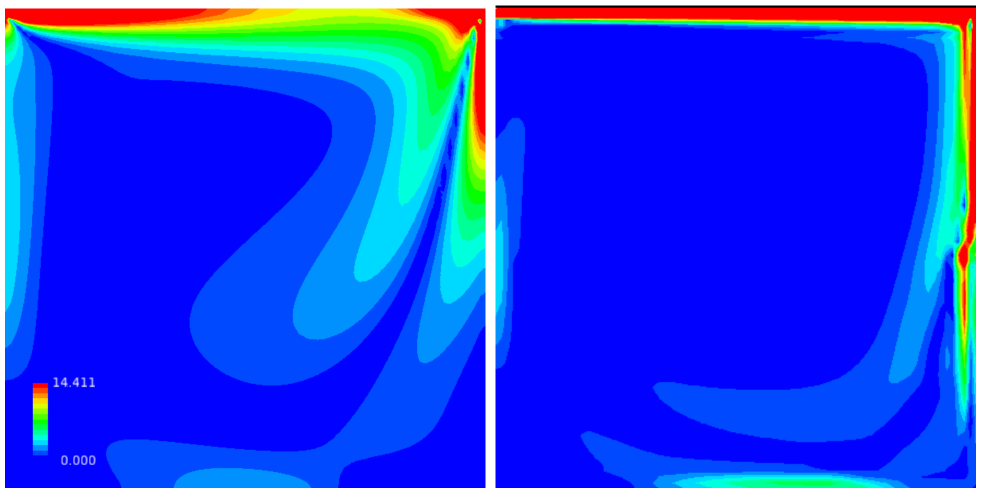
(b)



(c)

Figure 3.29: Time series; (a) without filters, (b) with Shuman filter, (c) with Padé filter.

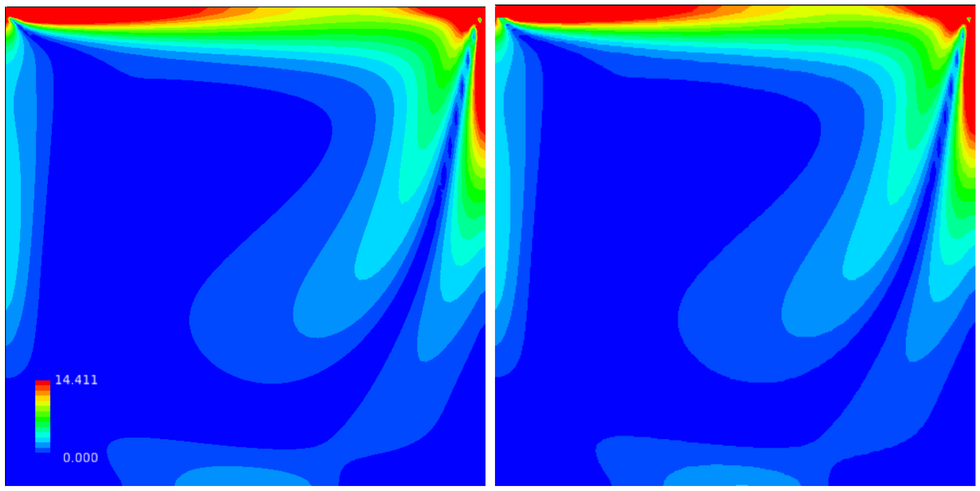
As in laminar flow, time series are first investigated. The spatial point chosen here is the center point of the cavity. Spatial discretization consisted of 81^3 uniformly-spaced points; $\beta = 500$ for low-pass filtering, and a second set of filter values for small scales is 150; time step size is 0.0125, and total time is 500 s. It is clear that without the application of filtering, the time series is nearly steady early in time, and it then blows up at later time. With application of the Shuman filter, the time series is steady, which is inconsistent with properties of turbulence. In order to get high-pass filtering, β needs to be increased. However, the flow is still steady at $\beta = 10000$.



(a) (b)

Figure 3.30: Magnitude of vorticity at time 250 s; (a) with Shuman filter (b) with Padé filter.

From the Fig. 3.30, it can be seen that the intensity of vorticity on the top of the cavity is decreased to some extent. However, compared with Padé filters, the vorticity increased in the right side of the cavity when Shuman filter is applied. This indicates that Shuman filter cannot filter too well, making the solution too smooth.



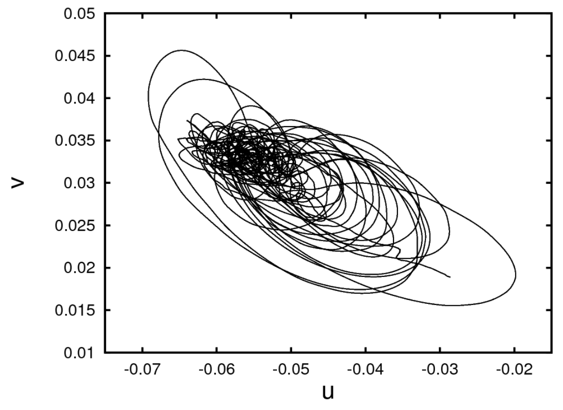
(a) (b)

Figure 3.31: Magnitude of vorticity at time steps at time 500 s; (a) with Shuman filter (b) with Padé filter.

Solutions without filters blow up finally. Compared with Fig. 3.30(b), the vorticity magnitude in Fig. 3.31(b) increased. This is because turbulence has not fully developed at the beginning, when at later time, the properties of turbulence appear.

In the above figure, it is obvious that vorticity decreases significantly when the Padé filter is employed. This indicates that the Padé filter smooths solutions less than does the Shuman filter in turbulent flow. It should be noticed here that no matter what β is, the flow is steady with application of Shuman filter, which is inconsistent with experimental results.

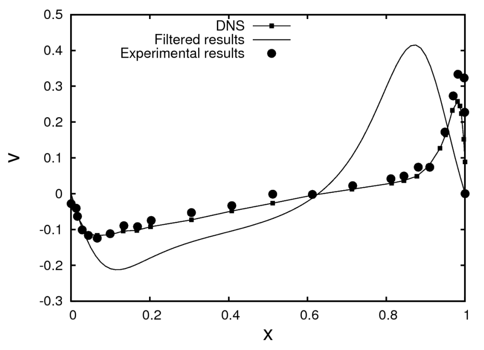
Through the time series, it can be seen that the flow is turbulent. In further analysis of the phase portrait, it also shows that the flow is turbulent, and that is consistent with experimental results.



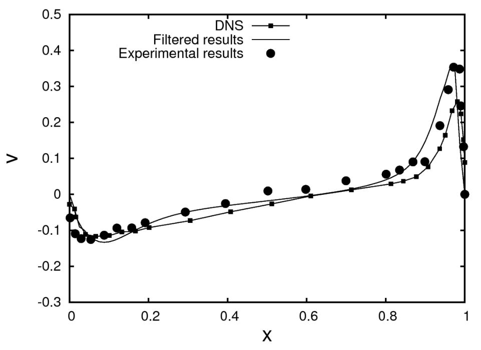
(a)

Figure 3.32: Phase portrait with Padé filter

In order to make a better comparison between Shuman and Padé filters, the v component of velocity along the centerline in the middle plane is compared. First, it is obvious that the results of DNS [46] [51] are good, and almost overlap with experimental results. After application of the Shuman filter, the results are not as expected. It cannot produce a good prediction of maximal and minimal velocity magnitudes. Compared with Shuman filters, Padé filters work better at other places along the middle plane. Velocities after application of Padé filters are almost the same as experimental and DNS results. On the other hand, it proves that if a proper filtering method is applied, LES can produce results nearly as good as DNS, while it saves time and arithmetic calculations.



(a)



(b)

Figure 3.33: Velocity at the mid-plane with filters (a) with Shuman filter, (b) with Padé filter.

One of the characterizations of a turbulent flow is its wide range of length and time scales [49], as is well known. In fact, if such scales did not cover wide ranges, the "turbulence problem" would have been solved long ago. Begin by noting that there are, in general, four main sets of scales in a turbulent flow (there may be more if other physical phenomena, e.g., heat transfer and/or combustion are important); these are large scale, integral scale, Taylor microscale, and Kolmogorov scale [50]. It is worthwhile to compare these scales in terms of (spatial) wavenumbers.

Based on this, there is a widely-quoted Kolmogorov $k^{-5/3}$ inertial-range scaling of the turbulent energy spectrum. This theory provides a way to show the effectiveness of Padé filters. It can be see below that in part of the power spectral density, results with application of Padé filters are consistent with what is expected.

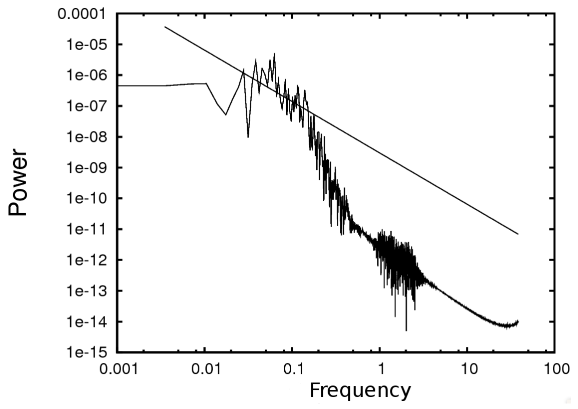


Figure 3.34: Power spectral density when using Padé filter.

The wavenumber corresponding to beginning of the dissipation scales is strongly influenced by Re, and the wavenumber range covered by the inertial scales must increase with increasing Re. In our case, the Re is not high enough, so the length of Kolmogorov $k^{-5/3}$ is not long. It should be noted that the Kolmogorov $k^{-5/3}$ is expected in homogeneous and isotropic turbulence, and little of the LDC flow would correspond to this.

Chapter 4

Conclusion

In this chapter, summary of the present study and conclusions are introduced in Sec. 1. Section 2 presents recommendations for future work.

4.1 Conclusions

In this investigation, a very common problem in computational fluid dynamics, termed aliasing, is studied. Its mathematical explanation and treatment via filtering are provided. Two different explicit filters are introduced. In the case of the Shuman filter, derivations are made in 1D and 2D with obvious extensions to 3D. Simulation work is made with different Reynolds numbers, number of grid points and filter parameter values. Following this, two cases $Re = 1500$ and 2000 are shown. In each case, different grid-spacing sizes 0.1, 0.05, 0.025 are used, and optimal β is found for different Re and grid spacing. For the latter of these, power spectral density analysis was made. In order to prove the importance of Shuman filter parameter values, experimental results are chosen as a reference to check the simulation results. The results show that flow states could be inconsistent with experimental results when inappropriate filter parameters are used. In the case of the Padé filter, its mathematical description is provided.

Comparison of Shuman and Padé filters on sine waves with perturbations in 1D and 3D is made. Noise is generated by Fourier noise, a random number generator, and the combination of them. Two aspects are taken into consideration to assess the effectiveness of two filters: error after application and execution time. Results show that the Shuman filter saves time and more effectively reduces error than does the Padé filter in both 1D and 3D. The Padé filter seems to follow noise rather than remove it, while the Shuman filter tries to smooth noise.

In an industrial setting, problems are not as simple as the perturbed sine wave studied here. We apply the same filters to a lid-driven cavity (LDC) problem to investigate the effectiveness in more complicated situations. In the study of the LDC problem, laminar and turbulent flows are both investigated. By the analysis of time series, magnitude of vorticity and comparison with experimental results, it can be seen at the conclusion that the Padé filter treats the aliasing problem better than the Shuman filter with filter parameter value 500 in turbulence but does less well in the laminar case.

4.2 What still is needed

In this thesis, both Shuman filters and Padé filers are analyzed based on perturbed sine waves and the lid-driven cavity problem. In the case of Shuman filters, different filter parameter values are applied in an attempt to find the optimal one. This proved to be unsuccessful. The best filter parameter value is related to number of grid points, Reynolds number and other possible parameters. In further studies, an equation for optimal Shuman filtering value should be sought. Besides, in this work, both Shuman and Padé filters are applied to the LDC problem only with uniform grid-point geometry, which means grid points are spaced the same in all three directions. More research needs to be done for nonuniform grid-point geometry, where optimal Shuman filter values and Padé filter moments may be different from uniform cases. Moreover, both Shuman filter and Padé filters can be applied together for deconvolution subgrid-scale models. In the comparison of Shuman and Padé filters on the LDC problem in turbulent flows, more filter parameter values of the Shuman filter can be tried. In this thesis, only filter parameter value 500 is used, and if other values are used, there is a possibility that the Shuman filter works better than the Padé filter.

Appendix A

L^2-norm error code

```
      PROGRAM NS3DLMNR
      PARAMETER (NX=101,NY=101,NZ=101)
      IMPLICIT REAL*4 (A-H,O-Z)

      character*14 filexyz ,fileqqq

      DIMENSION E(NX,NY,NZ),E1(NX,NY,NZ),E2(NX,NY,NZ),E3(NX,NY,NZ)

      DIMENSION X(NX,NY,NZ),Y(NX,NY,NZ),Z(NX,NY,NZ),U(NX,NY,NZ),
     1          V(NX,NY,NZ),W(NX,NY,NZ),P(NX,NY,NZ),PX(NX,NY,NZ),
     2          PY(NX,NY,NZ),PZ(NX,NY,NZ)

      DIMENSION UX(NX,NY,NZ),UY(NX,NY,NZ),UZ(NX,NY,NZ),
     1 VX(NX,NY,NZ),
     2       VY(NX,NY,NZ),VZ(NX,NY,NZ),WX(NX,NY,NZ),WY(NX,NY,NZ),
     3       WZ(NX,NY,NZ),UXX(NX,NY,NZ),UYY(NX,NY,NZ),
     4             UZZ(NX,NY,NZ),
     5             VXX(NX,NY,NZ),VYY(NX,NY,NZ),VZZ(NX,NY,NZ),
     6             WXX(NX,NY,NZ),WYY(NX,NY,NZ),WZZ(NX,NY,NZ)

      BETA = 1.e3

      write(*,*)'Enter value of beta'
```

```
          read(*,*)beta

          RE = 2.D2
          write(*,*)'enter value of Re'
          read(*,*) Re

          UREF = 1.D0
          LREF = 1.D0

*  READ GEOMETRY

ccc          write(*,*)'Input geometry filename'
ccc          read(*,*)filexyz

          CALL GEOMETRY(U,V,W,P,X,Y,Z,NX0,NY0,NZ0)


*  GET THE DERIVATIVE OF VELOCITY AND PRESSURE
          CALL DERIVATIVE(U,V,W,P,UX,UY,UZ,UXX,UYY,UZZ,
     1              VX,VY,VZ,VXX,VYY,VZZ,WX,WY,WZ,WXX,WYY,
     2              WZZ,PX,PY,PZ,X,Y,Z,DX,DY,DZ,NX0,NY0,NZ0)



          RMU = UREF*LREF/RE

          CALL NS_EQUATION(E1,E2,E3,E,U,V,W,UX,UY,UZ,UXX,UYY,UZZ,
     1             VX,VY,VZ,VXX,VYY,VZZ,WX,WY,WZ,WXX,WYY,
     2             WZZ,PX,PY,PZ,RMU,NX0,NY0,NZ0)


          OPEN(12,FILE='ETOTAL.DAT',ACCESS = 'APPEND',
          STATUS='UNKNOWN')

          ETOTAL1=0.D0
          ETOTAL2=0.D0

          DO K=2,NZ0-1
           DO J=2,NY0-1
            DO I=2,NX0-1
              ETOTAL1=ETOTAL1+E(I,J,K)

              ETOTAL2=ETOTAL2+E(i,j,k)**2
            END DO
           END DO
```

```
      END DO
      ETOTAL3=E(2,2,2)
      DO K=2,NZ0-1
       DO J=2,NY0-1
        DO I=2,NX0-1
         IF (E(i,j,k).gt.ETOTAL3) THEN
            ETOTAL3=E(i,j,k)
         end if
        end do
       end do
      end do

      ETOTAL1 = ETOTAL1*dx*dy*dz
      ETOTAL2 = (sqrt(ETOTAL2))*dx*dy*dz
      ETOTAL3 = ETOTAL3*DX*DY*DZ


      WRITE (*,*) nx0,ny0,nz0,Re,beta,ETOTAL1,ETOTAL2,ETOTAL3
      WRITE (12,*) nx0,ny0,nz0,Re,beta,ETOTAL1,etotal2,etotal3
      CLOSE (12)
      STOP
      END

      SUBROUTINE GEOMETRY(U,V,W,P,X,Y,Z,NX0,NY0,NZ0)
      IMPLICIT REAL*4(A-H,O-Z)

ccc         character*14 filexyz,fileqqq

      PARAMETER (NX=101,NY=101,NZ=101)
      DIMENSION  X(NX,NY,NZ),Y(NX,NY,NZ),Z(NX,NY,NZ)
      DIMENSION U(NX,NY,NZ),V(NX,NY,NZ),
     1  W(NX,NY,NZ),P(NX,NY,NZ)
* grid file is to be read

ccc         write(*,*)filexyz

      OPEN (7,file='output3ldc.xyz',status='unknown')
      READ (7,*) NX0,NY0,NZ0
      READ (7,*) (((X(I,J,K),I=1,NX0),J=1,NY0),K=1,NZ0),
     1            (((Y(I,J,K),I=1,NX0),J=1,NY0),K=1,NZ0),
     2            (((Z(I,J,K),I=1,NX0),J=1,NY0),K=1,NZ0)
      CLOSE (7)

      OPEN (7,FILE='ldcl050.qqq',STATUS='UNKNOWN')
      READ (7,*) NX0,NY0,NZ0
      READ (7,*)  (((P(I,J,K),I=1,NX0),J=1,NY0),K=1,NZ0),
     1            (((U(I,J,K),I=1,NX0),J=1,NY0),K=1,NZ0),
     2            (((V(I,J,K),I=1,NX0),J=1,NY0),K=1,NZ0),
     3            (((W(I,J,K),I=1,NX0),J=1,NY0),K=1,NZ0)
      CLOSE (7)


      RETURN
      END




      SUBROUTINE DERIVATIVE(U,V,W,P,UX,UY,UZ,UXX,UYY,UZZ,
     1           VX,VY,VZ,VXX,VYY,VZZ,WX,WY,WZ,WXX,WYY,
     2           WZZ,PX,PY,PZ,X,Y,Z,DX,DY,DZ,NX0,NY0,NZ0)

      IMPLICIT REAL*4(A-H,O-Z)
      PARAMETER (NX=101,NY=101,NZ=101)
      DIMENSION    X(NX,NY,NZ),Y(NX,NY,NZ),Z(NX,NY,NZ),
     1  U(NX,NY,NZ),
     2      V(NX,NY,NZ),W(NX,NY,NZ),P(NX,NY,NZ),PX(NX,NY,NZ),
     3       PY(NX,NY,NZ),PZ(NX,NY,NZ)

      DIMENSION UX(NX,NY,NZ),UY(NX,NY,NZ),UZ(NX,NY,NZ),
     1 VX(NX,NY,NZ),
     2  VY(NX,NY,NZ),VZ(NX,NY,NZ),WX(NX,NY,NZ),WY(NX,NY,NZ),
     3      WZ(NX,NY,NZ),UXX(NX,NY,NZ),UYY(NX,NY,NZ),
     4      UZZ(NX,NY,NZ),
     5       VXX(NX,NY,NZ),VYY(NX,NY,NZ),VZZ(NX,NY,NZ),
     6        WXX(NX,NY,NZ),WYY(NX,NY,NZ),WZZ(NX,NY,NZ)



       DX=X(2,1,1)-X(1,1,1)
       DY=Y(1,2,1)-Y(1,1,1)
       DZ=Z(1,1,2)-Y(1,1,1)
       HX2I=0.5D0/DX
       HY2I=0.5D0/DY
       HZ2I=0.5D0/DZ
       HX5Q=1.D0/((DX)**2)
       HY5Q=1.D0/((DY)**2)
       HZ5Q=1.D0/((DZ)**2)


*  DEFINE THE DERIVATIVE OF VELOCITY AND PRESSURE
```

```
  DO K=2,NZ0-1
     DO J=2,NY0-1
        DO I=2,NX0-1
          UX(I,J,K)=(U(I+1,J,K)-U(I-1,J,K))*HX2I
          UY(I,J,K)=(U(I,J+1,K)-U(I,J-1,K))*HY2I
          UZ(I,J,K)=(U(I,J,K+1)-U(I,J,K-1))*Hz2I
          UXX(I,J,K)=(U(I+1,J,K)-2*U(I,J,K)+U(I-1,J,K))*HX5Q
          UYY(I,J,K)=(U(I,J+1,K)-2*U(I,J,K)+U(I,J-1,K))*HY5Q
          UZZ(I,J,K)=(U(I,J,K+1)-2*U(I,J,K)+U(I,J,K-1))*HZ5Q
          PX(I,J,K)=(P(I+1,J,K)-P(I-1,J,K))*HX2I
          PY(I,J,K)=(P(I,J+1,K)-p(I,J-1,K))*HY2I
          PZ(I,J,K)=(P(I,J,K+1)-P(I,J,K-1))*HZ2I
        END DO
     END DO
  END DO
  RETURN
  END


  SUBROUTINE NS_EQUATION(E1,E2,E3,E,U,V,W,UX,UY,UZ,
1   UXX,UYY,UZZ,
2                            VX,VY,VZ,VXX,VYY,VZZ,WX,WY,WZ,WXX,WYY,
3                            WZZ,PX,PY,PZ,RMU,NX0,NY0,NZ0)

  PARAMETER (NX=101,NY=101,NZ=101)
  DIMENSION E1(NX,NY,NZ),E2(NX,NY,NZ),E3(NX,NY,NZ),
1   E(NX,NY,NZ),U(NX,NY,NZ),V(NX,NY,NZ),W(NX,NY,NZ),UX(NX,NY,NZ),
2        UY(NX,NY,NZ),UZ(NX,NY,NZ),VX(NX,NY,NZ),VY(NX,NY,NZ),
3           VZ(NX,NY,NZ),WX(NX,NY,NZ),WY(NX,NY,NZ),WZ(NX,NY,NZ),
4              uXX(NX,NY,NZ),uYY(NX,NY,NZ),uZZ(NX,NY,NZ),
5             vXX(NX,NY,NZ),vYY(NX,NY,NZ),vZZ(NX,NY,NZ),
6         wXX(NX,NY,NZ),wYY(NX,NY,NZ),wZZ(NX,NY,NZ),
7           PX(NX,NY,NZ),PY(NX,NY,NZ),PZ(NX,NY,NZ)

  DO K=2,NZ0-1
     DO J=2,NY0-1
        DO I=2,NX0-1
          E1(I,J,K)=U(I,J,K)*UX(I,J,K)+V(I,J,K)*UY(I,J,K)+
1                    W(I,J,K)*UZ(I,J,K)+PX(I,J,K)-RMU*(UXX(I,J,K)
2                   +UYY(I,J,K)+UZZ(I,J,K))
          E2(I,J,K)=U(I,J,K)*VX(I,J,K)+V(I,J,K)*VY(I,J,K)+
1                   W(I,J,K)*VZ(I,J,K)+PY(I,J,K)-RMU*(VXX(I,J,K)
2                   +VYY(I,J,K)+VZZ(I,J,K))
          E3(I,J,K)=U(I,J,K)*WX(I,J,K)+V(I,J,K)*WY(I,J,K)+
1                   W(I,J,K)*WZ(I,J,K)+PZ(I,J,K)-RMU*(WXX(I,J,K)
     2                  +WYY(I,J,K)+WZZ(I,J,K))
            E(I,J,K)=ABS(E1(I,J,K))+ABS(E2(I,J,K))+ABS(E3(I,J,K))
          END DO
        END DO
      END DO
      RETURN
      END
```

Appendix B

Filters applied to perturbed sine waves in 1D code

```
      program padetest

      implicit real*8 (a-h,o-z)
      real*4 rtsec,rtsecp,rtsecs

      parameter(nmx=1001)
      parameter(alpha=0.5673952755,beta=0.1209216774)

      dimension q2(nmx),f(nmx),u(nmx),q(nmx),r(nmx),
     1   r2(nmx),s(nmx)
      dimension a(nmx),b(nmx),c(nmx),d(nmx),e(nmx)
      dimension pa(4),a2(5),a3(5),pp(5),b2(6),b3(6),p(7)
      dimension u1(nmx),s1(nmx),s2(nmx),t(nmx)

********     build the original sin wave function, the period
********     is 2*pi,
********     and the sin wave velocity is u1, store results
********     in file
********     output, the first column is t, the second is u1.

      nx = 101
      npass = 1
      amp = 5.d-2
      betas = 2.d0
      pi = dacos(-1.d0)
```

```
        rt2 = 2*nx/3.d0   !remove sqrt to get linear behavior
        *sqrt(5.d0)
        exp3 = 3*nx/5.d0   !remove exp to get linear behavior
        *exp(1.d0)
        dt = 5.d0/(nx-1)

        do i=1,nx
         t(i) = 2.d0*(i-1)*dt*pi
         s(i) = sin(t(i))
         call random_number(r(i))
         call random_number(r2(i))
         r(i) = 2.d0*r(i) - 1.d0
         r2(i) = 2.d0*r2(i) - 1.d0
         if(i.gt.1.and.i.lt.nx)then
           u1(i) = s(i)
c    1      + 0.1d0*cos(rt2*t(i)) +0.05*sin(exp3*t(i))
     1             + 0.5d0*amp*(r(i)+r2(i))   !random noise
         else
           u1(i) = s(i)
         end if
        end do

        call l2err(s,u1,errr,nx)

        write(*,*)'   '
        write(*,*)'******************************************'
        write(*,*)'random number induced error ', errr
        rtsec = secnds(0.0)
        call shuman(u1,s1,betas,nx,npass)
        rtsecs = secnds(rtsec)

        call l2err(s,s1,errs,nx)

        write(*,*)'error after Shuman filtering', errs

        OPEN(8,FILE='output-r.s',STATUS='unknown')

        do i=1,nx
         WRITE(8,*)t(i),s(i),u1(i),s1(i),0.5*amp*(r(i)+r2(i))
        end do

        CLOSE(8)
        rtsec = secnds(0.0)
        call padefltr(u1,s2,nx)
        rtsecp = secnds(rtsec)
```

```
call l2err(s,s2,errp,nx)

write(*,*)'error after Pade filtering  ', errp

write(*,*)'  '
write(*,*)'Shuman filter execution time:',rtsecs,' seconds'
write(*,*)'Pade filter execution time:  ',rtsecp,' seconds'
write(*,*)'*****************************************'
write(*,*)'   '

OPEN(12,FILE='output-r.p',STATUS='unknown')

do i=1,nx
WRITE(12,*)t(i),s(i),u1(i),s2(i),0.5*amp*(r(i)+r2(i))
end do

CLOSE(12)

stop
end


subroutine shuman(u1,s1,betas,nx,npass)

implicit real*8 (a-h,o-z)

parameter(nmx=1001)

dimension u1(nmx),s1(nmx),tmp(nmx)

nexe = 100000

do ii=1,nexe

frac = 1.d0/(2.d0+betas)
tmp = u1
s1 = u1

do k=1,npass
 do i=2,nx-1
  s1(i) = frac*(u1(i-1)+betas*u1(i)+u1(i+1))
 end do
 u1 = s1
end do
```

```
      u1 = tmp

      end do

      return

      end


      subroutine padefltr(u1,s2,nx)



      implicit real*8 (a-h,o-z)

      parameter(nmx=1001)
      parameter(alpha=0.5673952755,beta=0.1209216774)

      dimension u1(nmx),s2(nmx)
      dimension q(nmx),q2(nmx)
c       dimension a(nmx),b(nmx),c(nmx),d(nmx),e(nmx)
      dimension pa(4),a2(5),a3(5),pp(5),b2(6),b3(6),p(7)

*  Load Pade filter coefficients

      pa = (/ 0.9931634217d0,1.2890384701d0,0.2965587062d0,
     1       0.0006836578d0 /)
      p = (/ pa(4),pa(3),pa(2),2.d0*pa(1),pa(2),pa(3),pa(4) /)
      pp = (/ beta,alpha,1.d0,alpha,beta /)

      a2 = (/ 0.3096256995d0,1d0,1.1380646293d0,
     10.4106696169d0, 0.d0 /)
      a3 = (/ 0.1477868412d0,0.6357553622d0,1.d0,
     1 0.6357553622d0,0.1477868412d0 /)
      b2 = (/ 0.3084688023d0,1.0057844862d0,1.1264956568d0,
     1      0.4222385894d0, -0.0057844862d0, 0.0011568972d0 /)
      b3 = (/ 0.1470348738d0,0.6395151994d0,0.9924803256d0,
     1      0.6432750366d0,0.1440270040d0,0.0007519674d0 /)

      a2 = a2/sum(a2)
      a3 = a3/sum(a3)
      b2 = b2/sum(b2)
      b3 = b3/sum(b3)
      p = p/sum(p)
      pp = pp/sum(pp)
```

```
*   Store i-direction rows of u3d, v3d, w3d in q and call line
*   filter

      nexe = 100000

      do ii=1,nexe

      do i=1,nx
       q(i) = u1(i)
      end do
      call padeline(q,q2,a2,a3,b2,b3,p,pp,nx)
      do i=1,nx
       s2(i) = q2(i)
      end do

      end do

      return

      end


      subroutine padeline(q,q2,a2,a3,b2,b3,p,pp,nd)



      implicit real*8 (a-h,o-z)

      parameter(nmx=1001)

      dimension q(nmx),q2(nmx),f(nmx)
      dimension a(nmx),b(nmx),c(nmx),d(nmx),e(nmx)
      dimension pa(4),a2(5),a3(5),pp(5),b2(6),b3(6),p(7)

      a = 0.d0
      b = 0.d0
      c = 1.d0
      d = 0.d0
      e = 0.d0

      a(3:nd) = pp(1)
      b(2:nd) =  pp(2)
      c(2:nd-1) = pp(3)
      d(1:nd-1) = pp(4)
      e(1:nd-2) = pp(5)

*     boundary conditions
      d(1) = 0.d0
      e(1) = 0.d0

      b(2) = a2(1)
      c(2) = a2(2)
      d(2) = a2(3)
      e(2) = a2(4)

ccc      write(*,*)nd

      a(3) = a3(1)
      b(3) = a3(2)
      c(3) = a3(3)
      d(3) = a3(4)
      e(3) = a3(5)

      a(nd) = 0.d0
      b(nd) = 0.d0


      d(nd-1) = a2(1)
      c(nd-1) = a2(2)
      b(nd-1) = a2(3)
      a(nd-1) = a2(4)

      e(nd-2) = a3(1)
      d(nd-2) = a3(2)
      c(nd-2) = a3(3)
      b(nd-2) = a3(4)
      a(nd-2) = a3(5)


       f(1) = q(1)
       f(2) = dot_product(q(1:6),b2)
       f(3) = dot_product(q(1:6),b3)
       do j=4,nd-3
        f(j) = dot_product(p,q(j-3:j+3))
       end do
       f(nd-2) = dot_product(q(nd:nd-5:-1),b3)
       f(nd-1) = dot_product(q(nd:nd-5:-1),b2)
       f(nd) = q(nd)
       call pentdiag(a,b,c,d,e,f,q2,nd)
```

```
c         do m=1,3
c           f(1,m) = q(1,m)
c           f(2,m) = dot_product(q(1:6,m),b2)
c           f(3,m) = dot_product(q(1:6,m),b3)
c           do j=4,nd-3
c             f(j,m) = dot_product(p,q(j-3:j+3,m))
c           end do
c           f(nd-2,m) = dot_product(q(nd:nd-5:-1,m),b3)
c           f(nd-1,m) = dot_product(q(nd:nd-5:-1,m),b2)
c           f(nd,m) = q(nd,m)
c           call pentdiag(a,b,c,d,e,f(:,m),q2(:,m),nd)
c         end do

        return

        end


        subroutine pentdiag(a,b,c,d,e,f,u,n)



        implicit real*8 (a-h,o-z)

        parameter (nmx=1001)

        dimension a(nmx),b(nmx),c(nmx),d(nmx),e(nmx),f(nmx),
     1     u(nmx),p(nmx),q(nmx)

        save

*   Initialize elimination and back substitution arrays
        if(c(1).eq.0.d0)stop        ! eliminate u2 trivially
        bet = 1.d0/c(1)
        p(1) = -d(1)*bet
        q(1) = -e(1)*bet
        u(1) = f(1)*bet

        bet = c(2) + b(2)*p(1)
        if(bet.eq.0.d0)stop
        bet = -1.0d0/bet
```

```
      p(2) = (d(2)+b(2)*q(1))*bet
      q(2) = e(2)*bet
      u(2) = (b(2)*u(1)- f(2))*bet

*   Construct upper-triangular matrix
      do i=3,n
       bet = b(i) + a(i)*p(i-2)
       den = c(i) + a(i)*q(i-2) + bet*p(i-1)
       if(den.eq.0.d0)then
         write(*,*)'  singularity in pentdiagonal matrix'
         stop
       end if
       den = -1.d0/den
       p(i) = (d(i)+bet*q(i-1))*den
       q(i) = e(i)*den
       u(i) = (a(i)*u(i-2)+bet*u(i-1)-f(i))*den
      end do

*   Perform back substitution
      u(n-1) = u(n-1) + p(n-1) * u(n)
      do i=n-2,1,-1
       u(i) = u(i) + p(i)*u(i+1) + q(i)*u(i+2)
      end do

      return
      end


      subroutine l2err(exct,aprx,err,nx)

      implicit real*8 (a-h,o-z)

      parameter (nmx=1001)

      dimension exct(nmx),aprx(nmx)

      err = 0.d0

      do i=1,nx
       err = err + ((exct(i)-aprx(i))/(nx-1))**2
      end do

      err = sqrt(err)

      return
```

```
end
```

Appendix C

Filters applied to perturbed sine waves in 3D code

```
      program padetest

       implicit real*8 (a-h,o-z)
       real*4 rtsec,rtsecp,rtsecs

       parameter(nmx=101)
       parameter(alpha=0.5673952755,beta=0.1209216774)

       dimension q2(nmx),f(nmx),u(nmx,nmx,nmx),q(nmx),r(nmx)
       dimension r2(nmx)
       dimension a(nmx),b(nmx),c(nmx),d(nmx),e(nmx)
       dimension u11(nmx),u12(nmx),u13(nmx),sx(nmx),
      1          sy(nmx),sz(nmx)
       dimension pa(4),a2(5),a3(5),pp(5),b2(6),b3(6),p(7)
       dimension tx(nmx),ty(nmx),tz(nmx)
       dimension u1(nmx,nmx,nmx),s2(nmx,nmx,nmx),
      1          s(nmx,nmx,nmx)
       dimension s1(nmx,nmx,nmx),x(nmx,nmx,nmx),
      1          y(nmx,nmx,nmx),z(nmx,nmx,nmx)

********    build the original sin wave function, the period
********    is 2*pi, and the sin wave velocity is u1, store
*******      results in file
********    output, the first column is t, the second is u1.
```

```
nx = 101
ny = 101
nz = 101
npass = 1
amp = 5.d-2
betas = 2.d0
pi = dacos(-1.d0)
rt2 = 2*nx/3.d0  !remove sqrt to get linear behavior
 *sqrt(5.d0)
exp3 = 3*nx/5.d0  !remove exp to get linear behavior
*exp(1.d0)
dt = 2.d0/(nx-1)
do k=1,nz
 tz(k)=2.d0*(k-1)*dt*pi
 do j=1,ny
  ty(j)=2.d0*(j-1)*dt*pi
  do i=1,nx
   tx(i)=2.d0*(i-1)*dt*pi
   x(i,j,k)=tx(i)
   y(i,j,k)=ty(j)
   z(i,j,k)=tz(k)
  end do
 end do
end  do

  do i=1,nx
  tx(i)=2.d0*(i-1)*dt*pi
  sx(i) = sin(tx(i))
  call random_number(r(i))
  call random_number(r2(i))
  r(i) = 2.d0*r(i) - 1.d0
  r2(i) = 2.d0*r2(i) - 1.d0

 if(i.gt.1.and.i.lt.nx)then

   u11(i) = sx(i)
1     + 0.1d0*cos(rt2*tx(i)) + 0.05*sin(exp3*tx(i))
1     + 0.5d0*amp*(r(i)+r2(i))  !random noise

 else
 u11(i) = sx(i)
 end if
end do
```

```
c         OPEN(12,FILE='u11.p',STATUS='unknown')
c         do i=1,nx
c         WRITE(12,*)  tx(i),u11(i)
c         end do
c         CLOSE(12)




        do j=1,ny
           ty(j)=2.d0*(j-1)*dt*pi
           sy(j) = sin(ty(j))
           call random_number(r(j))
           call random_number(r2(j))
           r(j) = 2.d0*r(j) - 1.d0
           r2(j) = 2.d0*r2(j) - 1.d0

          if(j.gt.1.and.j.lt.ny)then

             u12(j) = sy(j)
     1          + 0.1d0*cos(rt2*ty(j)) + 0.05*sin(exp3*ty(j))
     1          + 0.5d0*amp*(r(j)+r2(j))    !random noise

          else
          u12(j) = sy(j)
          end if
        end do

c         OPEN(12,FILE='u12.p',STATUS='unknown')
c         do j=1,ny
c         WRITE(12,*)  ty(j),u12(j)
c         end do
c         CLOSE(12)



        do k=1,nz
           tz(k)=2.d0*(k-1)*dt*pi
           sz(k) = sin(tz(k))
           call random_number(r(k))
           call random_number(r2(k))
           r(k) = 2.d0*r(k) - 1.d0
        r2(k) = 2.d0*r2(k) - 1.d0

       if(k.gt.1.and.k.lt.nz)then
         u13(k) = sz(k)
     1         + 0.1d0*cos(rt2*tz(k))+ 0.05*sin(exp3*tz(k))
     1         + 0.5d0*amp*(r(k)+r2(k))    !random noise
       else
       u13(k) = sz(k)
       end if
      end do


c      OPEN(12,FILE='u13.p',STATUS='unknown')
c      do k=1,nz
c      WRITE(12,*)  tz(k),u13(k)
c      end do

c      CLOSE(12)

      do k=1,nz
       do j=1,ny
        do i=1,nx
       if(i.gt.1.and.i.lt.nx.and.j.gt.1.and.j.lt.ny.and
     1   .k.gt.1.and.
     2     k.lt.nz)then
         u1(i,j,k) =u11(i)*u12(j)*u13(k)
       else
         u1(i,j,k) = sx(i)*sy(j)*sz(k)
       end if
      end do
      end do
      end do

      do k=1,nz
       do j=1,ny
        do i=1,nx
        s(i,j,k) = sx(i)*sy(j)*sz(k)
        end do
      end do
      end do
```

```
c        OPEN(12,FILE='nofilter.p',STATUS='unknown')
c        do k=1,nz
c          do j=1,ny
c            do i=1,nx

c        WRITE(12,*)  (((u1(i,j,k),i=1,nx),j=1,ny),k=1,nz)
c          write(12,*)tx(i),ty(j),tz(k),s(i,j,k),u1(i,j,k)
c        end do
c        end do
c        end do


c        CLOSE(12)






        call l2err(s,u1,errr,nx,ny,nz)

        write(*,*)'   '
        write(*,*)'****************************************'
        write(*,*)'random number induced error ', errr




        rtsec = secnds(0.0)
         call shuman(u1,s1,betas,nx,ny,nz,npass)
        rtsecs = secnds(rtsec)

        call l2err(s,s1,errs,nx,ny,nz)

        write(*,*)'error after Shuman filtering', errs
****** the list are correct code. but in order to use
******  fieldview, we need
*change it to plot3d format****
c        OPEN(8,FILE='output.s',STATUS='unknown')
c        do k=1,nz
c          do j=1,ny
c            do i=1,nx
c            end do
c          end do
c        end do
```

```
c        CLOSE(8)
****************

**********
         rtsec = secnds(0.0)

         call padefltr(u1,s2,nx,ny,nz)
         rtsecp = secnds(rtsec)
         call l2err(s,s2,errp,nx,ny,nz)

         write(*,*)'error␣after␣Pade␣filtering␣␣', errp

         write(*,*)'␣␣'
         write(*,*)'Shuman␣filter␣execution␣time:',
     1  rtsecs,'␣seconds'
         write(*,*)'Pade␣filter␣execution␣time:
␣␣␣␣␣1␣',rtsecp,'␣seconds'


***** Plot3d output format****
        OPEN(8,FILE='output.qqq',form='formatted',STATUS='unknown')
          write(8,'(4I5)')nx,ny,nz,4
         WRITE(8,'(5(e13.6,1x))')(((u1(i,j,k),i=1,nx),j=1,ny),
     1    k=1,nz),
     2            (((s1(i,j,k),i=1,nx),j=1,ny),k=1,nz),
     3            (((s2(i,j,k),i=1,nx),j=1,ny),k=1,nz),
     4            (((s(i,j,k),i=1,nx),j=1,ny),k=1,nz)
          close(8)

         OPEN(8,FILE='output.xyz',form='formatted',
     1      STATUS='unknown')
           write(8,'(3I5)')nx,ny,nz
         WRITE(8,'(5(e13.6,1x))')(((x(i,j,k),i=1,nx),j=1,ny),
     1      k=1,nz),
     2             (((y(i,j,k),i=1,nx),j=1,ny),k=1,nz),
     3             (((z(i,j,k),i=1,nx),j=1,ny),k=1,nz)

        CLOSE(8)

c        OPEN(12,FILE='output.p',STATUS='unknown')
c         do k=1,nz
c          do j=1,ny
c           do i=1,nx
c
```

```
c         end do
c         end do
c         end do
c         close(12)


c        OPEN(12,FILE='s2.p',STATUS='unknown')
c         do k=1,nz
c           do j=1,ny
c             do i=1,nx
c         write(12,*)tx(i),ty(j),tz(k),s2(i,j,k)
c        1 0.5*amp*(r(i)+r2(i))
c         end do
c         end do
c         end do

c         CLOSE(12)

        stop
        end

****shuman filter in 1D****
        subroutine shuman(u1,s1,betas,nx,ny,nz,npass)
        implicit real*8 (a-h,o-z)
        parameter(nmx=101)
    dimension u1(nmx,nmx,nmx),s1(nmx,nmx,nmx),tmp(nmx,nmx,nmx)
c         nexe = 100000
c         do ii=1,nexe
        frac = 1.d0/(6.d0+betas)
c         tmp = u1
        s1 = u1
c         do k=1,npass
          do k=2,ny-1
            do j=2,nz-1
              do i=2,nx-1
                s1(i,j,k) = frac*(u1(i-1,j,k)+u1(i,j-1,k)+
     1          u1(i,j,k-1)+
     2          betas*u1(i,j,k)+u1(i+1,j,k)+u1(i,j+1,k)+u1(i,j,k+1))
              end do
          end do
          end do
c           u1 = s1
c         end do
c         u1 = tmp
c         end do
```

```
      return
      end
*** pade filter ****

      subroutine padefltr(u1,s2,nx,ny,nz)
      implicit real*8 (a-h,o-z)
c       real*4 u3d,v3d,w3d,p3d
c       real*4 u3d
      parameter(mnx=101,mny=101,mnz=101,
     1     maxnxyz=max(mnx,mny,mnz))
      parameter(alpha=0.5673952755,beta=0.1209216774)

      dimension u1(mnx,mny,mnz),s2(mnx,mny,mnz)
      dimension q(maxnxyz),q2(maxnxyz),f(maxnxyz)
      dimension a(maxnxyz),b(maxnxyz),c(maxnxyz),d(maxnxyz),
     1          e(maxnxyz)
      dimension pa(4),a2(5),a3(5),pp(5),b2(6),b3(6),p(7)
*  Load Pade filter coefficients

      pa = (/ 0.9931634217d0,1.2890384701d0,0.2965587062d0,

     1        0.0006836578d0 /)

      p = (/ pa(4),pa(3),pa(2),2.d0*pa(1),pa(2),pa(3),pa(4) /)

      pp = (/ beta,alpha,1.d0,alpha,beta /)



      a2 = (/ 0.3096256995d0,1d0,1.1380646293d0,
     1 0.4106696169d0,
     2        0.d0 /)

      a3 = (/ 0.1477868412d0,0.6357553622d0,1.d0,
     1 0.6357553622d0,

     2        0.1477868412d0 /)

      b2 = (/ 0.3084688023d0,1.0057844862d0,1.1264956568d0,

     1   0.4222385894d0, -0.0057844862d0, 0.0011568972d0 /)

      b3 = (/ 0.1470348738d0,0.6395151994d0,0.9924803256d0,

     1   0.6432750366d0,0.1440270040d0,0.0007519674d0 /)
```

```
      a2 = a2/sum(a2)

      a3 = a3/sum(a3)

      b2 = b2/sum(b2)

      b3 = b3/sum(b3)

      p = p/sum(p)

      pp = pp/sum(pp)


*  Store i-direction rows of u3d, v3d, w3d in q and call
*   line filter



      do k=1,nz

       do j=1,ny

        do i=1,nx

         q(i) = u1(i,j,k)

c          q(i,2) = v3d(i,j,k)

c          q(i,3) = w3d(i,j,k)

        end do

        call padeline(q,q2,a2,a3,b2,b3,p,pp,nx)

        do i=1,nx

         s2(i,j,k) = q2(i)

c          v3d(i,j,k) = q2(i,2)

c          w3d(i,j,k) = q2(i,3)
```

```
          end do

         end do

        end do



*  Store j-direction rows of u3d, v3d, w3d in q and call
*   line filter



        do k=1,nz

         do i=1,nx

          do j=1,ny

           q(j) = u1(i,j,k)

c           q(j,2) = v3d(i,j,k)

c           q(j,3) = w3d(i,j,k)

          end do

ccc            write(*,201)(j,q(j,2),j=1,ny)

  201   format(1x,i3,2x,1pe13.6)

          call padeline(q,q2,a2,a3,b2,b3,p,pp,ny)

ccc            write(*,201)(j,q2(j,2),j=1,ny)

          do j=1,ny

           s2(i,j,k) = q2(j)

c           v3d(i,j,k) = q2(j,2)

c           w3d(i,j,k) = q2(j,3)

          end do
         end do
        end do


*   Store k-direction rows of u3d, v3d, w3d in q and call
*    line filter


        do i=1,nx
         do j=1,ny
          do k=1,nz
           q(k) = u1(i,j,k)
c          q(k,2) = v3d(i,j,k)
c          q(k,3) = w3d(i,j,k)
          end do
          call padeline(q,q2,a2,a3,b2,b3,p,pp,nz)
          do k=1,nz
            s2(i,j,k) = q2(k)
c          v3d(i,j,k) = q2(k,2)
c          w3d(i,j,k) = q2(k,3)
          end do
         end do
        end do

        return
```

```
      end




      subroutine padeline(q,q2,a2,a3,b2,b3,p,pp,nd)


      implicit real*8 (a-h,o-z)

      parameter(nmx=101)

      dimension q(nmx),q2(nmx),f(nmx)
      dimension a(nmx),b(nmx),c(nmx),d(nmx),e(nmx)


c       dimension q(maxnxyz),q2(maxnxyz),f(maxnxyz)

      dimension pa(4),a2(5),a3(5),pp(5),b2(6),b3(6),p(7)


      a = 0.d0

      b = 0.d0

      c = 1.d0

      d = 0.d0

      e = 0.d0


      a(3:nd) = pp(1)

      b(2:nd) =  pp(2)
```

```
        c(2:nd-1) = pp(3)

        d(1:nd-1) = pp(4)

        e(1:nd-2) = pp(5)


*       boundary conditions

        d(1) = 0.d0

        e(1) = 0.d0


        b(2) = a2(1)

        c(2) = a2(2)

        d(2) = a2(3)

        e(2) = a2(4)


ccc         write(*,*)nd


        a(3) = a3(1)

        b(3) = a3(2)

        c(3) = a3(3)

        d(3) = a3(4)

        e(3) = a3(5)


        a(nd) = 0.d0

        b(nd) = 0.d0
```

```
      d(nd-1) = a2(1)
      c(nd-1) = a2(2)
      b(nd-1) = a2(3)
      a(nd-1) = a2(4)

      e(nd-2) = a3(1)
      d(nd-2) = a3(2)
      c(nd-2) = a3(3)
      b(nd-2) = a3(4)
      a(nd-2) = a3(5)

c       do m=1,3
        f(1) = q(1)
        f(2) = dot_product(q(1:6),b2)
        f(3) = dot_product(q(1:6),b3)
        do j=4,nd-3
          f(j) = dot_product(p,q(j-3:j+3))
        end do
        f(nd-2) = dot_product(q(nd:nd-5:-1),b3)
        f(nd-1) = dot_product(q(nd:nd-5:-1),b2)
        f(nd) = q(nd)
        call pentdiag(a,b,c,d,e,f,q2,nd)
c       end do

      return

      end




      subroutine pentdiag(a,b,c,d,e,f,u,n)
      implicit real*8 (a-h,o-z)

      parameter (nmax=101)

      dimension a(nmax),b(nmax),c(nmax),d(nmax),
     1          e(nmax),f(nmax),u(nmax),p(nmax),q(nmax)

      save

*  Initialize elimination and back substitution arrays
      if(c(1).eq.0.d0)stop       ! eliminate u2 trivially
      bet = 1.d0/c(1)
      p(1) = -d(1)*bet
      q(1) = -e(1)*bet
      u(1) = f(1)*bet

      bet = c(2) + b(2)*p(1)

      if(bet.eq.0.d0)stop !singularity in
      pentdiagonal matrix

      bet = -1.0d0/bet

      p(2) = (d(2)+b(2)*q(1))*bet

      q(2) = e(2)*bet

      u(2) = (b(2)*u(1)- f(2))*bet


*   Construct upper-triangular matrix

      do i=3,n

       bet = b(i) + a(i)*p(i-2)

       den = c(i) + a(i)*q(i-2) + bet*p(i-1)

       if(den.eq.0.d0)then

       write(*,*)' singularity in
      1 pentdiagonal matrix'

         stop

       end if

       den = -1.d0/den
       p(i) = (d(i)+bet*q(i-1))*den
       q(i) = e(i)*den
       u(i) = (a(i)*u(i-2)+bet*u(i-1)-f(i))*den
       end do


*   Perform back substitution
```

```
u(n-1) = u(n-1) + p(n-1) * u(n)
do i=n-2,1,-1
u(i) = u(i) + p(i)*u(i+1) + q(i)*u(i+2)
end do
return
end

subroutine l2err(exct,aprx,err,nx,ny,nz)
implicit real*8 (a-h,o-z)
parameter (nmx=101)
dimension exct(nmx,nmx,nmx),aprx(nmx,nmx,nmx)

err = 0.d0
do k=1,nz
 do j=1,ny
  do i=1,nx
  err = err + ((exct(i,j,k)-aprx(i,j,k))/(nx-1))**2
end do
end do
end do
err = sqrt(err)

return

end
```

Bibliography

[1] Bruno A. Olshausen, Aliasing, PSC 129, Sensory Process, 2000.

[2] Thomas H. Pulliam, Artificial dissipation models for the Euler equations, *AIAA Journal*, 24, 12, 1986.

[3] G. de Vahl Davis and G. D. Mallinson. An evaluation of upwind and central difference approximations by a study of recirculating flow, *Computers and Fluids* 4, 29–43, 1976.

[4] J. M. McDonough, V. E. Garzon and D. Schulte. Effect of film-cooling hole location on turbulator heat transfer enhancement in turbine blade internal air-cooling circuits, presented at *ASME TURBO EXPO 99,* Indianapolis, IN, June 7–10, 1999.

[5] J. M. McDonough, T. Yang and M. Sheetz. Parallelization of a modern CFD incompressible turbulent flow code, presented at *Parallel CFD 2003, Moscow, May 13–15, 2003.*

[6] F. G. Shuman, Numerical methods in weather prediction: II Smoothing and filtering, *Monthly Weather Review*, 85, 357–361, 1957.

[7] Stephan P. Nelson, Michanel L. Weible, Three-dimensional Shuman filter, *Journal of applied meteorology*, 19, 464–469, 1998.

[8] T. Yang and J. M. McDonough, Solution filtering technique for solving Burgers' equation, *The Fourth International Conference on Dynamical Systems and Differential Equations,* University of North Carolina at Wilmington, U.S.A., May 23–27, 2002.

[9] Oleg V Vasilyev, A study of the effect of smooth filtering in LES, Department of mechanical and aerospace engineering, University of Missouri-Columbia.

[10] Lund, T.S. On the use of discrete filters for large eddy simulation, *Annual Research Briefs, Center for Turbulence Research, NASA Ames/Stanford Univ.*, 83-95,1997.

[11] Ercan Erturk, Discussions on driven cavity flow, *Int, J. Numer. Meth. Fluids,* 60, 275–294, 2009.

[12] V. B. L. Boppana and J. S. B. Gajjar, Global flow instability in a lid-driven cavity. *Int. J. for numer. meth. in fluids*, 2009.

[13] Charles-Henri Bruneau, Mazen Saad, The 2D lid-driven cavity problem revisited. *Computers and Fluids*, 35, 326–348, 2006.

[14] Oleg V. Vasilyev, Thomas S. Lund, and Parviz Moin, A general class of commutative filters for LES in complex geometries, *J. Comp. Phys.*, 146, 82–104, 1998.

[15] Liu, Z., Huang, Q., Zhao, Z., and Yuan, J., Optimized compact finite difference schemes with high accuracy and maximum resolution, *Int. J. Aeroacous.*, 7, 123–146, 2008.

[16] M. S. Grewal, *Kalman Filtering: Theory Practice,* Englewood Cliffs, NJ, Prentice-Hall, 1993.

[17] Vauhkonen, M, A Kalman filter approach to track fast impedance changes in electrical impedance tomography, *IEEE Transactions on Biomedical Engineering*, 45, Issue 4, 486–493, 1998.

[18] Maria Isabel Ribeiro, Kalman and extended Kalman filters: concept, derivation and properties, Institute for Systems and Robotics, 2004.

[19] Jiangang Wang, Deyuan Gao, Improved morphological top-hat filter optimized with genetic algorithm, *CISP 2nd International Congress on Image and Signal Processing*, 2009.

[20] N. E. Grube, M. Pino Martin, Assessment of subgrid-scale models and shock-confining filters in large-eddy simulation of highly compressible isotropic turbulence, *47th AIAA Aerospace Science Meeting*, 5–8, January 2009, Orlando, FL.

[21] Weinan E, Numerical methods for viscous incompressible flows: some recent advances, Princeton University, Available as a downloadable PDF file at*https://web.math.princeton.edu/ weinan/papers/cfd1.pdf.*

[22] S. V. Patankar. *Numerical heat transfer and fluid flow,* McGraw-Hill Book Co., New York, 1980.

[23] W. A. Ames, *Numerical methods for partial differential equations*, Academic Press, New York, 304-306, 1977.

[24] J. M. McDonough a T. Yang, A solution Filtering Technique for treating Under-Resolved Solutions to Partial Differential Equations: Burgers' equation model problem. Available as a downloadable PDF file at http://www.engr.uky.edu/ acfd/filter2.pdf.

[25] R. Shapiro, Smoothing, filtering and boundary effects, *Rev. Geophys. and Space Phys.*, 8, 359–387, 1970.

[26] D. You, S. T. Bose and P. Moin, Grid-independent large-eddy simulation of compressible turbulent flows using explicit filtering, *Center for Turbulence Research, Proceedings of the Summer Program,* 203–211, 2010.

[27] Ghosal, S. An analysis of numerical errors in large-eddy simulations of turbulence. *J. Comp. Phys.* 125, 187–206, 1995.

[28] Ghosal, S. Moin, P. The basic equations of the large eddy simulation of turbulent flows in complex geometry. *J. Comp. Phys.* 118, 24–37, 1995.

[29] Jessica Gullbrand, Explicit filtering and subgrid-scale models in turbulent channel flow, Center for Turbulence Research Annual Research Briefs, 2001.

[30] A. Harten and G. Zwas, Switched numerical Shuman filters for shock calculations, *J. Eng. Math.* 6, No. 2, 207–216, 1972.

[31] K. E. Gustafson, *Introduction to partial differential equations and Hilbert space methods,* John Wiley Sons, New York, 1980.

[32] A. Majda, J. Harlim and B. Gershgorin, *Dis. Cont. Dyn. Sys.*, 27, 441–486, 2010.

[33] J. M. McDonough, Lectures on computational fluid dynamics of incompressible flow: mathematics, algorithms and implementations,Available as a downloadable PDF at *http://www.engr.uky.edu/ acfd/me691-lctr-nts.pdf.*

[34] J. M. McDonough. Lectures on *Computational numerical analysis of partial differential equations, 2008.* Available as a downloadable PDF file at *http://www.engr.uky.edu/ acfd/me690-lctr-nts.pdf.*

[35] William L. Briggs, Van Emden Henson, and Steve F. McCormick, *A Multigrid Tutorial (2nd ed.)*, Philadelphia: Society for Industrial and Applied Mathematics, ISBN 0-89871-462-1, 2000.

[36] Martin Vetterli and Cormac Herley, Wavelets and Filter Banks: Theory and Design, *IEEE Transactions on Signal Processing.* 40, 1992.

[37] G. Beylkin, On the representation of operators in bases of compactly supported wavelets, *SIAM J. Numer. Anal.* 29, 1716 (1992).

[38] Joshua Strodtbeck, *A filter-forcing turbulence model for large eddy simulation incorporating the compressible "poor man's navier–stokes equations"*. Doctoral Dissertation, University of Kentucky, 2012.

[39] A. Liberzon, Yu. Feldman and A. Yu. Gelfgat, Experimental observation of the steady-oscillatory transition in a cubic lid-driven cavity, *Physics of fluids*, 23, 084106, 2011.

[40] P. Gresho, On the theory of semi-implicit projection methods for viscous incompressible flow and its implementation via a finite element method that also introduce a nearly consistent mass matrix, part 1: Theory, *Int. J. Numer. Meth. Fluids,* vol. 11, 587–620, 1990.

[41] J. Douglas, Jr. and J. E. Gunn. A general formulation of alternating direction methods, part I. parabolic and hyperbolic problems, *Numer. Math.*, 6, 428–453, 1964.

[42] David Ruelle and Floris Takens, On the nature of turbulence, *Comm. Math. phys.* 20, 167–192, 1971.

[43] Alligood, K.T., Sauer, T.D., Yorke, J.A., *Chaos, an introduction to dynamical systems*, Springer-Verlag, New York, 1996.

[44] Constantin, P. and Foias, C., Navier-Stokes Equations, University of Chicago Press, Chicago, 1988.

[45] J. M. McDonough. Lectures on *Basic computational numerical analysis, 2007.* Available as a downloadable PDF file at *http://www.engr.uky.edu/ acfd/lecturenotes1.html*

[46] P. N. Shankar and M. D. Deshpande, Fluid mechanics in the Driven Cavity, *Annu. Rev. Fluid Mech.*, 32:93–136, 2000.

[47] W. S. Brainerd, C. H. Goldberg, and J. C. Adams, *Programmer's Guide to Fortran 90,* Springer, New York, 1996.

[48] Adams, N. A. and Stolz, S., Deconvolution methods in subgrid-scale approximationsin LES, *Modern Simulation Strategies for Turbulent Flow,* edited by B. J. Geurts, Vol. 50, R. T. Edwards, Inc., pp. 21–44, 2001.

[49] Chanson, H. and Carosi, G., Turbulent Time and Length Scale Measurements in High-Velocity Open Channel Flows. *Experiments in Fluids*, 42 (3), 385–401, 2007.

[50] Kolmogorov, A. N., The local structure of turbulence in incompressible fluid for very large Reynolds number. Dokl. Akad. Nauk SSSR 30, 299–303, 1941.

[51] Srinivasan R. Accurate solutions for steady plane flow in the driven cavity. I.Stokes flow. *Z. Angew Math. Phys.* 46:524–45, 1995.

Vita

Weiyun Liu, was born in Jinan, China. After obtaining the Bachelor's degree in China University of Petroleum, she studied at the computational fluid dynamics research group in University of Kentucky. She did the research on the Shuman filter and the Padé filter on a lid-driven cavity problem. Her research areas include petrochemistry, petromechanics, large eddy simulation and turbulence.