HIGHER-ORDER UNSTRUCTURED FINITE VOLUME METHOD FOR COMPUTATIONAL FLUID DYNAMICS

A Dissertation

by

RAYMOND LEE FONTENOT

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Paul Cizmas
Committee Members,	Diego Donzis
	Kentaro Hara
	Dara Childs
Head of Department,	Rodney Bowersox

May 2019

Major Subject: Aerospace Engineering

ABSTRACT

This dissertation presents a higher-order finite volume method (FVM) for computational fluid dynamics (CFD) for unstructured mesh topologies using Moving Least-Squares (MLS) as the backbone of the method. The MLS method is improved in several ways. First, the local stencil is weighed using a minimum volume enclosing ellipsoid (MVEE), which better encapsulates the local nodal topology than traditional spherical descriptions. Furthermore, a novel approach, known herein as Affine MLS, uses a spherical transformation of the ellipsoidal weights to map to the unit ball, where direct application of orthogonal polynomial bases can be used. This approach dramatically reduces the condition number of the MLS Moment/Gram matrix, especially on stretched grids which are commonly used for viscous flows and where traditional methods fail. All the MLS methods are also extended to use the Pivoting QR method for matrix inversion. The MLS method and improvements are extensively tested with several analytical functions for the full MLS reconstruction and fully diffuse derivatives. Optimal scaling parameters are also determined for the MLS method.

Additionally, from work with MLS, the boundary conditions of the higher-order method are enforced with ghost nodes, an approach more commonly used in Immersed Boundary Methods. These boundary conditions do a better job of enforcing the boundary states, since they are included directly into the fluxes and gradients. Non-reflecting ghost nodes are implemented using the Navier–Stokes Characteristic Boundary Condition (NSCBC) for the inlet, outlet, and freestream boundary conditions for the first time in a finite volume ghost boundary node context. A higher-order viscous state reconstruction is presented as well wherein the MLS method is used to determine the state and derivatives at the quadrature location. Some simple test cases are presented that highlight the benefits of the ghost node boundary conditions and viscous flux reconstruction. Finally, the higher-order CFD method is applied to the Taylor-Green Vortex (TGV) problem, a benchmark Large-Eddy Simulation (LES) case.

DEDICATION

To my loving wife, my family, and all those along the way.

ACKNOWLEDGMENTS

This dissertation brings to a close a decade of post-graduate work at Texas A&M. The work presented herein and for my Master of Science degree program are the result of the generosity of many people who I now wish to thank and make note of.

First, I would like to thank my advisor, Dr. Paul Cizmas. Dr. Cizmas made my graduate studies possible and served as my advisor for my Master of Science degree as well. Throughout my entire program of study, Dr. Cizmas shared his time and insight in addition his patience. He continuously challenged me to think both inside and outside the box to overcome problems.

Next, I would like to thank my committee members Dr. Diego Donzis and Dr. Kentaro Hara from the Department of Aerospace Engineering and Dr. Dara Childs from the Department of Mechanical Engineering. Dr. Donzis was my faculty advisor for the Graduate Teaching Fellowship and helped me help hone my teaching and presentation skills, for which I am extremely grateful. Dr. Hara was gracious enough to join my committee near the end of my doctoral work to replace a former member. Finally, I would like to thank Dr. Childs for helping fund parts of this research through the Turbomachinery Research Consortium as well as helping me learn that being relaxed and authentic are often more important than just knowing the facts.

Several fellow graduate students have also been instrumental in my graduate studies. First, I would like to thank Thomas Brenner, who I forgot to thank in my thesis, but was integral to that work and for helping to guide me in the initial years of my post-graduate career. Robert Brown was extremely helpful with ensuring my math made sense in addition to being a great officemate for several years. Elizabeth Krath helped me when I was teaching and allowed bounce ideas off her when we were officemates, in addition to furthering our hobbies. Brian Freno gave excellent advice for figures, formatting, and general topics. Forrest Carpenter and Neil Matula not only were helpful in so far as waking up my computer when it would go down, but also let me bounce ideas off them to improve our CFD code. David Liliedahl was also a great officemate, and I appreciated the sometimes random discussions we would have.

Finally, I would like to thank my friends and family. First, my roommates helped keep me grounded during the inevitable ups and downs of graduate school and helped get me out of the house when I needed a break. All of their continued friendships are something I will always cherish, even when they crush me at Catan or rub in bad football picks. I will always be grateful to the advice and guidance of my spiritual directors as well, particularly Fr. Will Straten and Fr. Ryan Higdon, who helped me become the man I am today. My family has always been supportive of my endeavors, and my youngest sister might have finally forgiven me for going to graduate school in the first place. I am especially grateful for my parents for teaching me the benefits of hard work, the strive to do what is right, and for their sacrifices to help me and my siblings have the lives we do. Finally, I would like to thank my loving wife, Loralee, who has as much claim to my doctoral work as myself. She has always given me encouragement for my studies, patience in my long days to finish, and comfort when I ran into road blocks or when I was sick and unable to work for several months. My love, we are finally done, and I cannot think of anyone else I would rather enjoy this with than you.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professor Paul Cizmas, my advisor, and Professors Diego Donzis and Kentaro Hara of the Department of Aerospace Engineering and Professor Emeritus Dara Childs of the Department of Mechanical Engineering.

The work conducted for this dissertation was completed by the student under the advisement of Dr. Paul Cizmas of the Department of Aerospace Engineering.

Funding Sources

This work was made possible in part by the Graduate Teaching Fellowship from Texas A&M University, grants from the Turbomachinery Research Consortium through the Turbomachinery Laboratory at Texas A&M University under grant numbers 400124-00061, 400124-00069, 400124-00074, and 400124-00091, and various assistantships provided by the Department of Aerospace Engineering at Texas A&M University.

NOMENCLATURE

Roman

a, b, c	-	indexes for derivatives
c	-	speed of sound
CFL	-	Courant-Friedrichs-Levy stability condition
D	-	deviatoric stress tensor
E	-	total energy
$\mathbf{F}$	-	vector of fluxes
G	-	convolution filter
$\mathbf{G}$	-	vector of source terms
$\mathbf{I}$	-	identity matrix
i, j, k	-	indexes for derivatives
k	-	thermal conductivity, Moving Least-Squares smoothing parameter, or number of neighbors for implicit residual smoothing
ℓ	-	order of accuracy
M	-	Mach number
$\mathbf{n}$	-	normal vector for surface
p	-	pressure
$\mathbf{p}$	-	polynomial basis vector
$\mathbf{q}$	-	heat flux
$\mathbf{Q}$	-	state vector of conservative variables
r	-	Moving Least-Squares smoothing length or support radii
$\mathbf{R}$	-	residual vector

s	-	edge-based area
s	-	spline parameter for Moving Least-Squares
S	-	cell face surface area
t	-	time
T	-	temperature or total time
u	-	velocity component in x-direction
$\mathbf{v}$	-	free-stream flow velocity vector
v	-	velocity component in y-direction
w	-	velocity component in z-direction or quadrature weighting function
W	-	Moving Least-Squares weight function
$\mathbf{x}$	-	coordinate vector

Greek

β	-	vortex strength
γ	-	gas constant
Δ	-	filter length
Θ	-	work of viscous stresses and heat conduction
μ	-	viscosity
ξ	-	general integration variable
ρ	-	density
σ	-	sub-grid stress
τ	-	viscous stress
ϕ	-	general basis function
Φ	-	limiter function
ψ	-	Moving Least-Squares shape function

Ψ	-	Moving Least-Squares basis function
Ω	-	cell volume
$\Omega_{\mathbf{x}_I}$	-	Moving Least-Squares stencil

Subscripts

$contra$	-	contravariant
$conv$	-	convective
i	-	i^{th} node
i, j, k	-	component pertaining to the x, y, or z-direction
$visc$	-	viscous
x	-	component pertaining to x-direction
y	-	component pertaining to y-direction
z	-	component pertaining to z-direction

Abbreviations

ADER	-	Arbitrary Derivative Riemann Problem -
Aniso.	-	Anisotropic
CFD	-	Computational Fluid Dynamics
DGM	-	Discontinuous Galerkin method
DGSEM	-	Discontinuous Galerkin Spectral Element method
DNS	-	Direct Numerical Simulation
ENO	-	Essentially Non-Oscillatory
ILES	-	Implicit Large-Eddy Simulation
Iso.	-	Isotropic
LES	-	Large-Eddy Simulation

MILES - Monotonically Integrated Large-Eddy Simulation

MEA - Modified Equation Analysis

MLS - Moving Least-Squares

NASA - National Aeronautics and Space Administration

RANS - Reynolds-averaged Navier–Stokes

Re - Reynolds Number

RPM - rotations per minute

SGS - sub-grid stress

SST - Shear Stress Transport

SPD - Symmetric Positive Definite

SVD - Singular Value Decomposition

UNS3D - Unstructured 3D flow solver

WENO - Weighted Essentially Non-Oscillatory

TABLE OF CONTENTS

Page

Page

Page

LIST OF FIGURES

FIGURE Page

Page

Page

Page

Page

Page

Page

LIST OF TABLES

TABLE Page

Page

Page

Page

Page

Page

1. INTRODUCTION

1.1 Statement of the Problem

With a push toward increasing efficiencies in turbomachinery, accurate simulation of complex flow geometries with Computational Fluid Dynamics (CFD) has become critical in the design process. Flows through these machines are generally turbulent, highly unsteady, and compressible, increasing the difficulty of performing numerical simulations. While there are several approaches to solve this difficult problem, not all of them are well suited.

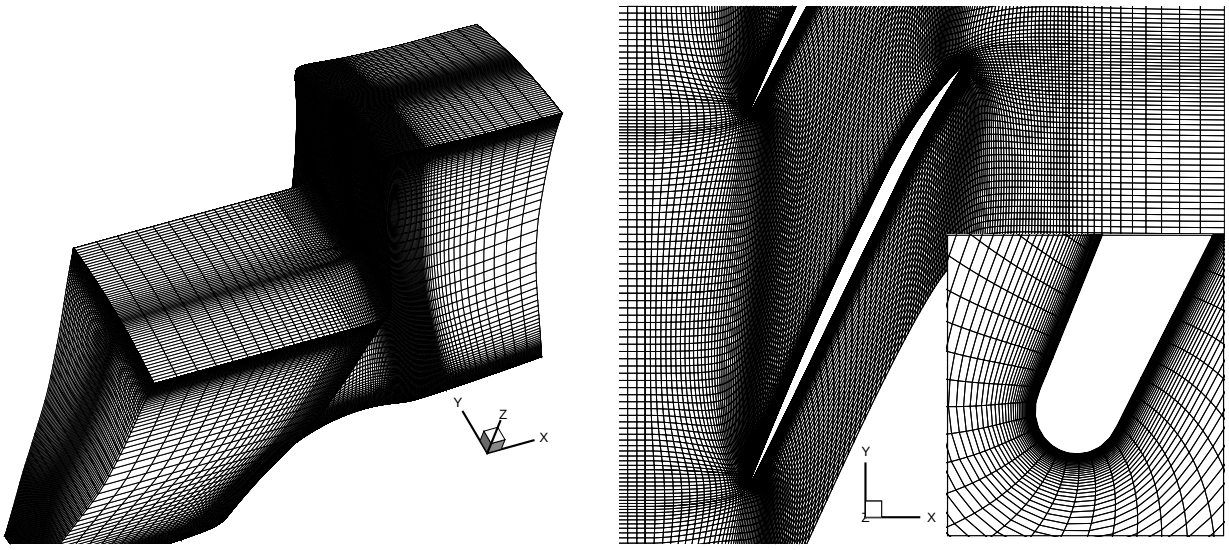
The most accurate method for computing turbulent flows is Direct Numerical Simulation (DNS), which directly solves the Navier–Stokes equations. However, the fidelity of DNS does come at a steep price: as one increases the Reynolds number, Re, the computational cost associated with solving the Navier–Stokes scales with $Re^{9/4}$ and the number of operations increases as $Re^{11/4}$ [131, pgs. 347-348]. In addition to the large computational cost associated with DNS, DNS utilizes high-order schemes, such as Weighted and Essentially Non-Oscillatory (WENO/ENO), compact, and spectral, that are either extremely difficult or impossible to extend to complex geometries [129]. Furthermore, recent DNS simulations [99] solved the flow in a channel at $Re = 5200$, which is a Re that is well below what is normally seen in typical turbomachinery applications. Additionally, while DNS has been performed on the T106A Low Pressure Turbine Cascade at more applicable Reynolds numbers ($Re = 60000 - 148,000$) [171], DNS is still generally limited to a tool for understanding the underlying nature of turbulence.

Less accurate but more efficient than DNS [131, pg. 560], Large-Eddy Simulation (LES) decomposes the Navier–Stokes equations into large and small scale components, utilizing the small scale universality of turbulence [114]. LES solves the large scales and models the small universal scales. The goals of LES are "to compute the three-dimensional, time-dependent details of the largest scales of motion using a simple model for the smaller scales" [137]. LES has been successfully applied to a wide range of complex flows [143, 56, 142], including flows in turboma-

chinery [158].

The most common method for computing turbulent flows is the Reynolds-averaged Navier–Stokes (RANS) method. RANS decomposes the Navier–Stokes equations into a time-averaged and temporally fluctuating component, where only the time-averaged, but still unsteady, equations are solved [131, pgs. 83-85]. This method has been successfully used to solve a wide range of flows; however, there are drawbacks. The resulting decomposition produces an additional term in the governing equations known as Reynolds stresses. These stresses require modeling as the Reynolds decomposition creates a closure problem [131, pg. 87]. There is a wide range of turbulence models which address the closure problem, see [168, Chapters 3-5] and [131, Chapters 10-11]. All turbulence models use the Boussinesq eddy viscosity approximation in some form to compute the Reynolds stress tensor [168, pg. 53]. However, eddy viscosity is flow dependent [168, pg. 53], and turbulence models are specifically tuned for certain flows [169]. This *ad hoc* nature of turbulence models means that there is no 'universal' model. Furthermore, the very nature of turbulence would suggest that a time-averaged approach, especially for unsteady flows with large non-local effects, will not yield accurate results [65]. Tyacke et al. [159] showed that for labyrinth seals, RANS solutions had higher error in the results by as much as 20% between models when compared to LES solutions.

Experience shows that RANS might not be well suited for vortex dominated flows in turbomachinery. The NASA Rotor 67 and preswirl vane geometry are just two examples. The NASA Rotor 67 is a standard turbomachinery test case. Carpenter [18] simulated a single vane of the 22-bladed cascade at 16,000 RPM with a total pressure ratio of 1.6 and mass flow of 34 kg/s, with $Re = 2.92 \cdot 10^6$ based on the relative Mach number at tip of the rotor. The case was run second-order accurate scheme and steady, with the Reynolds stresses modeled using the Shear Stress Transport (SST) model [115] in the RANS solver UNS3D [77]. The mesh, shown in Figures 1.1(a) and 1.1(b), contains approximately 4.5 million nodes.



(a) Full Mesh (b) Mesh near rotor blade

Figure 1.1: Mesh for Rotor 67, adapted with permission [18][1].

Results are shown in Figure 1.2. Toward the hub, the flow follows the blade passages nicely; however, as the tip is approached the flow becomes highly turbulent and vortex dominated. Flow near the tip of the blade is complex. Flow travels up and down the blade surface as well as over and around the tip of the blade, generating vortices that spanned multiple blades. To resolve any of the flow without having the solver diverge, significant refinement was needed in this region as seen in Figure 1.1(b). Not only would using higher-order methods improve the accuracy of the results, it would also come with a reduction in the number of nodes.

[1]*Adapted with permission from *Practical Aspects of Computational Fluid Dynamics for Turbomachinery*, by Forrest Carpenter, 2016, Ph.D. Thesis. College Station, Texas. Copyright 2016 by Forrest Carpenter.

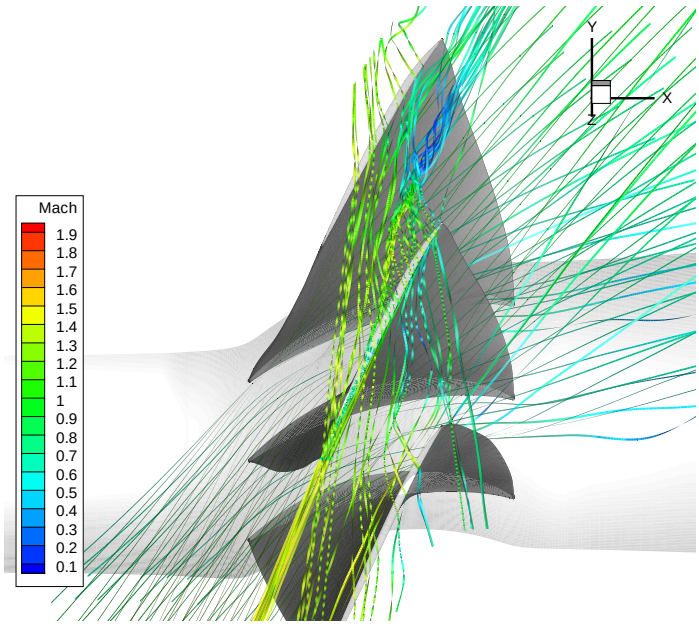


Figure 1.2: Mach Ribbons for Rotor 67, adapted with permission [18].

The preswirl vane geometry is also a complex problem with large, strong vortices. Simulating a single 0^o vane from a cascade of 72 with a mesh of 5.6 million nodes (see Figure 1.3),

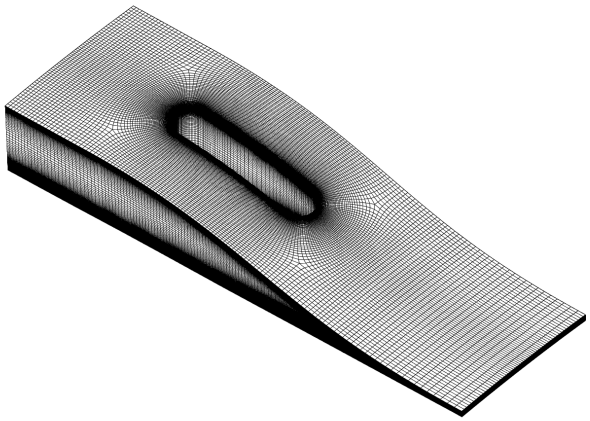


Figure 1.3: Mesh for Preswirl Vane, Courtesy of Neil Matula[2].

[2]*Obtained from personal communication with Neil Matula while discussing flow characteristics of pre-swirl vanes.

the preswirl vane is run second-order accurate scheme and steady with turbulence modeled using the SST model in the RANS solver UNS3D [77]. The flow simulated with a stagnation pressure of 7 MPa, a mass flow of 0.7 kg/s, and a rotor speed of 10,200 RPM, with $Re = 1.2 \times 10^6$ based on the inlet gap. Results for the preswirl vane are given in Figure 1.4.

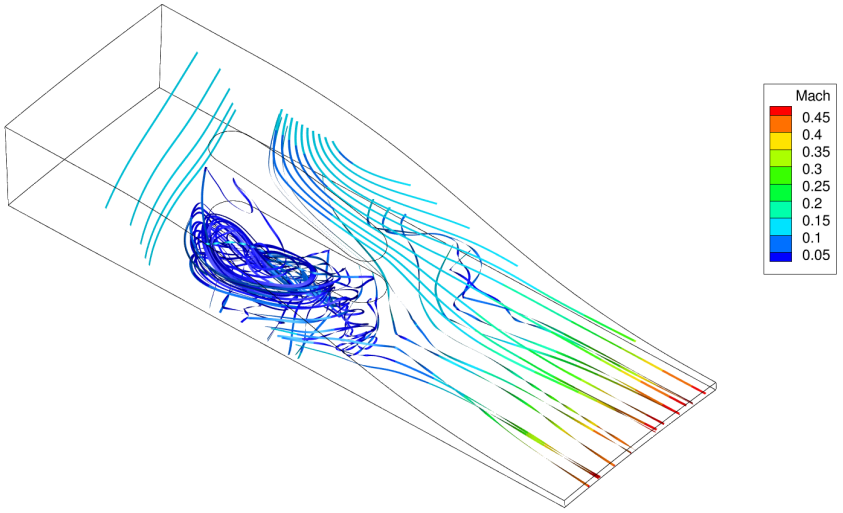


Figure 1.4: Mach Ribbons for Preswirl Vane, Courtesy of Neil Matula.[3]

The flow enters the domain at nearly 90^o and turns sharply almost immediately parallel with the blade. A large, full blade vortical structure is on the suction side, with spillage of the structure onto the pressure side. This large vortical structure is very hard to initialize and requires extreme refinement to capture. As with the Rotor 67, using higher-order methods would allow both a reduction in the total mesh refinement and improvement in overall accuracy of the results.

Additionally, using higher-order methods could provide marked improvement flows with strong dependence on fluid-structure interaction, such as the aeroelastic response of the General Trans-

[3]*Obtained from personal communication with Neil Matula while discussing flow characteristics of pre-swirl vanes.

port Wing [82]. The results generated in [82] were simulated on a unstructured mesh of 'medium' refinement to reduce the computational costs for long, unsteady simulations. The structure model is a plate model, which does not include thermal effects. The flow regime in the stability/instability envelope is very complex, with strong, moving shocks and separation, wing-tip vortices, and transition to turbulence. Using RANS models in this regime might not be appropriate, due to the known inadequacies of the RANS in these regimes [36]. The application of LES in this class of flows would be more suitable [36]. Aeroelastic simulations of these flow regimes have shown great promise [67, 68]. The usage of higher-order methods helps facilitate these simulations, since Implicit LES comes from the usage of higher-order methods. Additionally, higher-order methods may lend to a better fluid/structure model coupling method, depending on how the boundary conditions are applied linking the two methods.

To accurately solve complex flows, such as those previously discussed, it would appear a different approach than the one currently employed is necessary. Currently, the problems discussed have been solved using an unstructured finite volume RANS solver, UNS3D [77], which is second-order accurate in both space and time. Since DNS is still out of reach for these class of flows, and RANS as previously mentioned may be inadequate, the method required would appear to be LES. Also, since the current second-order accuracy of the current approaches may also be the cause of the inability to accurately model complex flows, a higher-order method would also appear to be required. The next section will review current approaches in both LES and higher-order methods, and how they both can solve the current problem.

1.2 Literature Review

Large-Eddy Simulations were originally developed to predict global meteorological patterns [56, Chapter 1], where the scales of turbulence range from kilometers to centimeters, and the computational domain is highly under-resolved [46, Chapter 3, pg. 51]. The first LES model was developed by Smagorinsky [148], and later expanded by Lilly [103] and Deardorff [34]. The first engineering flow of interest simulated with LES was channel flow by Deardorff [33]. Incompressible pipe flow results were first obtained by Unger & Friedrich [161]. The first compressible LES

simulations were carried out by the research group at NASA Langley on compressible isotropic turbulence [42]. Xu et al. [175] generated compressible second-order results for isothermal and minimally convective turbulent pipe flow. Subbareddy & Candler [150] developed a kinetic energy based method with shock-capturing ability to aid in solving compressible flows with LES.

LES has seen wide application to various components of turbomachinery, from compressors to low pressure turbines [158]. Kato et al. [87] used LES to simulate a mixed-flow pump stage. Larchevêque et al. [98, 97, 96] performed simulations of deep cavity flows which matched well with experimental data. Mahesh et al. [108] simulated flow through swirl combustors. Tyacke et al. [160] performed simulations on two labyrinth seal geometries. O'Mahoney et al. [127] performed a comprehensive study comparing Unsteady RANS to LES for a turbine rim seal, showing much better agreement of the LES results to experimental data. McMullan & Page [112] outline the steps necessary to simulate axial compressor cascades and show the benefits of running LES simulations on these configurations, especially in off-design configurations, for the Monterey cascade, Cranfield BRR axial compressor, and Cambridge axial compressor rig. Wang et al. [163] outlined some of the approaches and difficulties in simulating complete engines. Wang et al. [163] also introduced an overlapping grid solver for rotor-stator meshes using LES with the Smagorinsky model [148] producing reasonable results.

LES is generally divided into two camps: explicit sub-grid models and implicit sub-grid models. Explicit sub-grid models seek to model the forward energy cascade explicitly, and derivations of these models can be divided into two additional categories: Fourier and physical space [141, pg. 105]. Fourier sub-grid models are used for spectral finite difference methods. A list and description of the various spectral sub-grid models is given in [141, pgs. 105-109]. Sub-grid models based on physical space fall into three categories: (1) based on the resolved scales, (2) based on the energy at the energy cutoff, and (3) based on the sub-grid scales themselves [141, pg. 112]. Additionally, sub-grid models can be combinations of each of the three types; these are the so-called mixed models [141, pg. 123]. Physical space sub-grid models owe their existence to Smagorinsky [148], who developed the now famous Smagorinsky model [148]. Though still widely used, the Smagorinsky

model suffers from poor wall treatment and poor performance in flows with substantial shear [100], in addition to being global in nature and deficient for high Reynolds numbers [176]. Various modifications to the original Smagorinsky model have been developed over the years, including the Dynamic Smagorinsky model [61, 63] and the shear-improved Smagorinsky [100]. For a comprehensive review of physical space sub-grid models and strategies for both incompressible and compressible flows for explicit LES, see [141, Chapters 5-7], [58, Chapters 4-5], [131, Chapter 13], and [114].

Implicit sub-grid modeling in LES, rather than explicitly computing the sub-grid contributions, relies on the numerical discretization to mimic the sub-grid terms. Boris [14] was the first to postulate and utilize this approach in MILES, or Monotonically Integrated LES. Implicit LES (ILES), an extension of MILES, typically employs high-resolution finite volume schemes to generate the sub-grid models [143] [72, Chapter 4a]. Not every high-resolution finite volume scheme is suitable for ILES; a method known as Modified Equation Analysis (MEA) provides a framework for determining the explicit sub-grid model counterpart that the scheme will generate [72, Chapter 5]. The implicit sub-grid models in ILES tend to mimic mixed models in explicit LES [110]. ILES is not widely accepted in the turbulence community due to the lack of a rigorous justification for ILES [110]. However, ILES has been successfully used to model a variety of complex flows, such as a swept wing [75], internal hill flows [74], cavity flows [96], low pressure turbine blades [135], swirl combustors [71], and labyrinth seals [159]. For a comprehensive review on ILES, see [72].

Modern commercial and many research codes model the LES equations second-order accurate in space and time [129]. This level of accuracy, however, is not acceptable for LES. Ghosal [62] showed that the LES equations must be solved with schemes greater than two because the numerical discretization produces errors larger than the sub-grid terms. Dairay et al. [30] has performed extensive work on tuning ILES for higher-order schemes to be spectral in nature and shows the sensitivity of explicit sub-grid models to numerical errors even at higher-orders. While strides have been made in improving the accuracy of LES, especially for flows of engineering interest, many challenges remain [37]. An often overlooked consideration for higher-order finite volume

methods is temporal accuracy. Balsara & Shu [7] have shown that higher-order temporal schemes are required to obtain accurate and non-oscillatory (stable) solutions when using higher-order spatial schemes. High-order temporal schemes tend to be memory intensive; therefore, low-storage schemes are desirable [41]. The low storage Runge–Kutta schemes produced by Kennedy et al. [89], for example, are accurate up to sixth-order and have been successfully used for DNS simulations. The major challenge for implementing higher-order finite volume schemes, though, is the employment and implementation of higher-order schemes on unstructured topologies [129]. There are several methodologies to achieve higher-order accuracy, including compact schemes, WENO/ENO, and Discontinuous Galerkin, and meshless methods. For a comprehensive review on the subject, see [41].

Pouangué et al. [133] have developed a sixth-order LES solver using compact schemes. Compact schemes, however, are only valid for structured and curvilinear meshes, as grid distortion common with unstructured grids automatically reduces the formal accuracy to third order at best [149]. Biswas [11] used a fourth/fifth-order WENO finite difference scheme for transition flows in a low pressure turbine. WENO/ENO schemes are extremely difficult to construct for unstructured topologies in three dimensions [179]. WENO/ENO schemes of at least fifth order have been constructed for finite volume schemes, but only for structured meshes [155]. Combined Arbitrary Derivative Riemann Problem - Weighted Essentially Non-Oscillatory (ADER-WENO) schemes have achieved up to seventh order, but again these are only for structured meshes [38]. Currently, WENO schemes up to only third order for tetrahedral meshes have been constructed [179, 39].

Discontinuous Galerkin methods (DGM) are another way to achieve high-order accuracy in finite volume methods, where the accuracy is achieved through higher-order finite element basis functions [25]. DGM does not suffer like the previous two methods on arbitrary topologies: as a finite element based method, elements of arbitrary shape are permitted and easily accommodated [147]. While DGM has excellent performance for linear problems, nonlinear problems, such as the equations solved for LES, are not provably convergent [25]. Additionally, DGM may not be stable with explicit time stepping methods when strong shocks are present [25]. Shock and

discontinuity handling in DGM is still an open research topic, behind that of more traditional finite volume methods [125]. However, some recent advances for DGM, such as inclusion of spectral methods (or DGSEM) [80], have allowed for DNS simulation of flow past a sphere. DGSEM have been successfully applied to the canonical Taylor–Green Vortex flow [59]. While DGM techniques have been developed for LES [83] and used for simple flows [86, 15], the outlined issues with DGM has prevented it from gaining wide usage.

The final method for achieving higher-order finite volumes schemes utilizes meshless methods. Initially used for smooth particle hydrodynamics [16, pg. 1], meshless methods offer promising advantages over the other techniques. First, since the methods are meshless by design, the mesh topology is handled as a collection of point clouds such that any elements generated by the meshing algorithm can be used. Additionally, meshless methods, in the context of finite volume methods, allow for easy extension of current finite volume solvers to higher order. Meshless methods were first used for finite volume discretizations of the shallow water equations, utilizing Moving Least Squares (MLS) for gradient determination by Cueto-Felgueroso [26]. A shock-capturing technique, which also can be used as an explicit filter for LES, has been developed by Nogueira et al. [125]. Chassaing et al. [22] have used the third-order Moving Kriging meshless method on decaying compressible turbulence. Simulations of decaying compressible turbulence have also been performed with a fourth-order MLS method [126]. Nogueira et al. [123] have successfully used the meshless method with MLS for ILES simulations of non-wall-bounded turbulent flows.

1.3 Original Contributions of Present Work

Building upon the previous work of the second-order accurate unstructured finite volume solver UNS3D [77], in addition to the works of Kim [91], Gargoloff [57], and Carpenter [18], this dissertation presents the development and implementation of a fourth-order spatial and temporal accurate, unstructured finite volume method.

The primary focus of this dissertation is on the implementation of the meshless method of Moving Least-Squares, first used in finite volume schemes by Cueto-Felgueroso [26], to compute the numerical gradients of an unstructured mesh topology. MLS is used herein to achieve fourth-

order spatial accuracy, overcoming the deficiencies of compact, WENO/ENO, and Discontinuous Galerkin higher-order methods by being easily extensible to unstructured topologies. MLS, unlike the other gradient methods, maintains the shock-capturing and limiting abilities of second-order methods. To increase the overall stability of MLS, several improvements to the original algorithm are made. First, a pivoting QR algorithm is applied to MLS, which delays deteriorating effects of rounding errors and improves decomposition of rank-deficient problems. A new, user-independent anisotropic weighting method is developed. This method uses minimum volume enclosing ellipsoids (MVEE) [156] built from the natural mesh topology, rather than ad-hoc rectangular anisotropic weights. For poorly conditioned systems, which arise in most three-dimensional problems, a novel affine transformation is applied to the MLS weights to dramatically improve the condition number of the MLS moment matrix, making derivatives computed via the method more reliable. A parametric study is performed to determine the bounds on the scaling parameter used with MLS. Additionally, a study is conducted on the effect of using orthogonal polynomials as the basis set for MLS. A new recursive form for the MLS derivatives is proposed, along with a study on the effect of diffusive derivatives on the accuracy of the computed MLS derivatives.

The work of dissertation then shifts to boundary conditions applied to both inviscid and viscous flows. First, the periodic boundaries are improved by properly accounting for the geometry and gradients on the master and slave sides of the domain. Next, using the concept of image/intercept nodes, non-periodic boundary conditions are implemented using a novel MLS-finite difference ghost node method. Furthermore, to improve the wave characteristics of the overall simulation, Navier-Stokes Characteristic Boundary Condition (NSCBC)-based ghost nodes are used for the first time in a finite-volume method to define the inlet, outlet, and farfield boundaries, as an extension to the work of Motheau et al. [120] who applied the NSCBC to finite difference schemes using ghost nodes. The various boundary conditions are tested with some simple flows. Finally, the combined method is used to simulate the Taylor–Green Vortex problem, an LES benchmarking case..

1.4 Dissertation Outline

Chapter 2 presents the governing flow equations, the Navier–Stokes equations. Chapter 3 presents the flow models, with focus on Large-Eddy Simulation (LES) and Reynolds-Averaged Navier–Stokes (RANS) and how both models can be implemented together in the same flow solver. Chapter 4 presents the standard numerical method used for second-order accurate finite volume methods. Chapter 5 presents the numerical methods necessary for higher-order flow solvers, specifically Moving Least-Squares (MLS), and improvements to MLS, and boundary conditions necessary for higher-order methods. Chapter 6 presents two-dimensional results for the Moving Least-Squares method. The method is tested with several functions showing the validation of the implementation and improvements to the method, particularly the use of anisotropic weights and Affine MLS. Chapter 7 presents results for the higher-order method, first showing the improvements to the conditioning of the MLS system for typical mesh topologies. Higher-order boundary conditions are then tested to show the improvement over the standard boundary conditions. Finally, the chapter concludes with results for a benchmark LES case with structured and unstructured grids, the Taylor–Green Vortex. Conclusions and recommendations for future work are given in Chapter 8.

2. GOVERNING EQUATIONS

Most flows of engineering interest are highly unsteady, compressible, and viscous. These flows are modeled using the mass, momentum and energy conservation equations collectively known as the the Navier-Stokes equations. They are:

$$\frac{\partial}{\partial t}\int_{\Omega} \mathbf{Q}\cdot d\Omega + \oint_{\partial\Omega} \left(\mathbf{F}_{conv} - \mathbf{F}_{visc}\right)\mathbf{n}\, dS = \int_{\Omega} \mathbf{G}\cdot d\Omega. \tag{2.1}$$

Ω is the fluid element over which we are integrating and $\mathbf{n}$ is the outward-pointing normal of Ω.

The state vector of conservative variables is:

$$\mathbf{Q} \equiv \begin{Bmatrix} \rho \\ \rho\mathbf{v} \\ \rho E \end{Bmatrix} \tag{2.2}$$

where

$$\mathbf{v} \equiv \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} \tag{2.3}$$

are the x, y, and z-velocity components, ρ is the density, and E is the total energy of a fluid particle. The convective, or inviscid, and viscous flux vectors are

$$\mathbf{F}_{conv} \equiv \begin{bmatrix} \rho\mathbf{v}^T \\ \rho\mathbf{v}\mathbf{v}^T + p\mathbf{I} \\ \rho E\mathbf{v}^T + p\mathbf{v}^T \end{bmatrix} \tag{2.4}$$

and

$$\mathbf{F}_{visc} \equiv \begin{bmatrix} 0 \\ \boldsymbol{\tau} \\ (\boldsymbol{\tau}\mathbf{v} + k\nabla T)^T \end{bmatrix} \tag{2.5}$$

respectively. k is the thermal conductivity of the fluid, and T is the temperature. $\boldsymbol{\tau}$ is the viscous stress tensor, defined as:

$$\boldsymbol{\tau} \equiv 2\mu\mathbf{D} - \frac{2}{3}\mu\left(\nabla \cdot \mathbf{v}\right)\mathbf{I} \tag{2.6}$$

where

$$\mathbf{D} \equiv \frac{1}{2}\left(\nabla\mathbf{v} + \nabla\mathbf{v}^T\right) \tag{2.7}$$

$\boldsymbol{\tau}$ can also be expressed as:

$$\begin{aligned} \tau_{xx} &= \tfrac{2}{3}\mu\left(2\tfrac{\partial u}{\partial x} - \tfrac{\partial v}{\partial y} - \tfrac{\partial w}{\partial z}\right) \\ \tau_{yy} &= \tfrac{2}{3}\mu\left(2\tfrac{\partial v}{\partial y} - \tfrac{\partial w}{\partial z} - \tfrac{\partial u}{\partial x}\right) \\ \tau_{zz} &= \tfrac{2}{3}\mu\left(2\tfrac{\partial w}{\partial z} - \tfrac{\partial u}{\partial x} - \tfrac{\partial v}{\partial y}\right) \\ \tau_{xy} &= \tau_{yx} = \mu\left(\tfrac{\partial u}{\partial y} + \tfrac{\partial v}{\partial x}\right) \\ \tau_{yz} &= \tau_{zy} = \mu\left(\tfrac{\partial v}{\partial z} + \tfrac{\partial w}{\partial y}\right) \\ \tau_{zx} &= \tau_{xz} = \mu\left(\tfrac{\partial w}{\partial x} + \tfrac{\partial u}{\partial z}\right) \end{aligned} \tag{2.8}$$

where μ is the viscosity of the fluid.

The vector of source terms, $\mathbf{G}$ is equal to zero in the absence of body forces or lack of internal sources.

3. FLOW MODEL

For the Navier–Stokes equations, there is no closed-form solution and very few exact solutions. This comes from the non-linearity of the equations, specifically due to viscous stresses τ (2.6) $(\mathbf{v}\nabla\mathbf{v})$. Mathematically, turbulence arises from these terms, and modeling must be applied to achieve solutions to the Navier–Stokes equations were turbulence is present, which is in most flows of importance. There are three main approaches available when modeling turbulence: Direct Numerical Simulation (DNS), Large-Eddy Simulation (LES), and Reynolds-averaged Navier–Stokes (RANS).

3.1 Direct Numerical Simulation

Direct Numerical Simulation takes the Navier–Stokes equations (2.1) and attempts to solve the equations directly, using the appropriate discretizations in time and space. This method is typically only feasible for low Reynolds Number flows [131, p. 347-348] and is therefore not yet applicable to turbomachinery or other complex flows. It will not be further discussed in this dissertation.

3.2 Large-Eddy Simulation

Large-Eddy Simulation strives to "compute the three-dimensional, time-dependent details of the largest scales of motion using a simple model for the smaller scales" [137]. The viscous stresses (2.6) are calculated by spatially averaging the flow variables and modeling the smaller scales, which are assumed to be universal.

To derive the LES model, filtering operations are applied to the finite volume formulation of the compressible Navier–Stokes equations (2.1). Three distinct filtering operations are performed on the equations to separate and remove the sub-grid terms for modeling purposes. The three filtering operations are grid, convolution, and Favre. Grid filtering is an implicit filtering step, as the information smaller than the grid is automatically removed. Figure 3.1 shows this. The orange vortices are smaller than the grid and therefore unable to be captured by the numerical method. Maroon vortices are much larger, and while not captured exactly, have most of their salient features

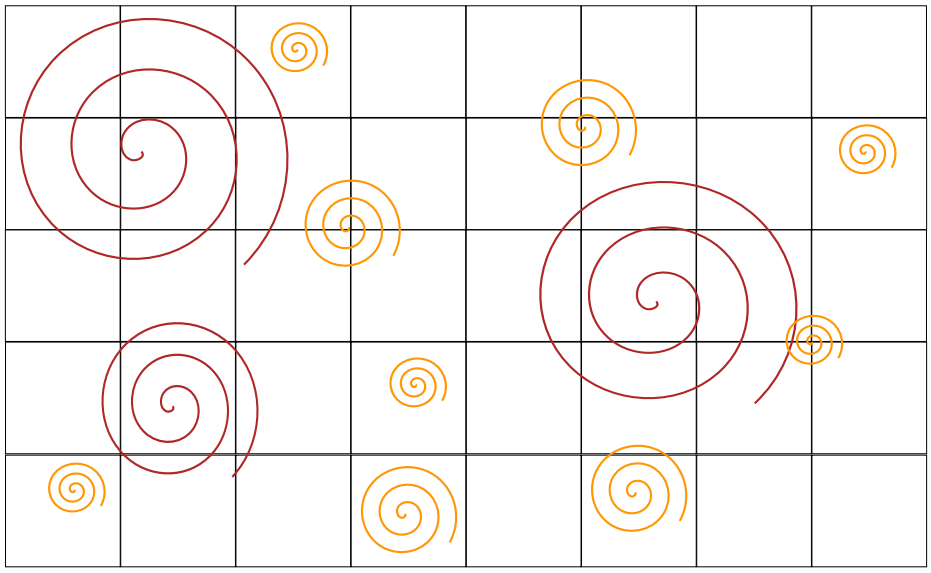


Figure 3.1: Grid Filtering of Small Scale Structures. Maroon spirals are captured by more than one grid point. Orange spirals are captured by at most one grid point.

represented by the mesh.

Convolution filtering is a high-pass filtering operation that removes the higher-frequency, smaller scale information from the model. The field is decomposed into large and small scales,

$$q(\mathbf{x},t) = \bar{q}(\mathbf{x},t) + q'(\mathbf{x},t), \tag{3.1}$$

where $\bar{()}$ are the large scales and $()'$ are the small scales. A convolution integral is then applied to (3.1) such that

$$\bar{q}(\mathbf{x},t) = G \star q \equiv \int_{\Omega} G(\mathbf{x},\boldsymbol{\xi}) q(\boldsymbol{\xi},t)\, d\boldsymbol{\xi} \tag{3.2}$$

where G is the filter function. There are a few options available for G, but in the context of finite volume schemes, G is automatically satisfied by the finite volume formulation, where G is explicitly a top-hat filter or volume averaging operation [131]:

$$G(\mathbf{x},\boldsymbol{\xi}) = \begin{cases} 1/\Delta^3 & \text{if } ||\mathbf{x}-\boldsymbol{\xi}|| \leq \Delta/2 \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

where Δ is the convolution filter length and the cell volume $\Omega = \Delta^3$. The convoluted variables are then expressed as:

$$\bar{q}(\mathbf{x}, t) = G \star q = \frac{1}{\Omega} \int_{\Omega} q(\boldsymbol{\xi}, t) d\Omega. \tag{3.4}$$

The governing equations now take the following form:

$$\frac{\partial}{\partial t} \int_{\Omega} \mathbf{Q} \, d\Omega + \oint_{\partial\Omega} (\mathbf{F}_{conv} - \mathbf{F}_{visc}) \cdot \mathbf{n} \, dS = \mathbf{0}, \tag{3.5}$$

where

$$\mathbf{Q} \equiv \begin{Bmatrix} \bar{\rho} \\ \bar{\rho}\bar{\mathbf{v}} \\ \bar{\rho}\bar{E} \end{Bmatrix}, \tag{3.6}$$

and $\bar{v}$ are the filtered velocity components, $\bar{\rho}$ is the filtered density, and $\bar{E}$ is the filtered total energy of a fluid element. It is important to point out that (3.5) is nearly identical to (2.1) (without the source term). The filtered convective and viscous flux vectors are [58]

$$\mathbf{F}_{conv} \equiv \begin{bmatrix} \bar{\rho}\bar{\mathbf{v}}^T \\ \bar{\rho}\bar{\mathbf{v}}\bar{\mathbf{v}}^T + \bar{p}\mathbf{I} \\ \bar{\rho}\bar{E}\bar{\mathbf{v}}^T + \bar{p}\bar{\mathbf{v}}^T \end{bmatrix}, \tag{3.7}$$

and

$$\mathbf{F}_{visc} \equiv \begin{bmatrix} SGS_D \\ \bar{\boldsymbol{\tau}} + SGS_M \\ (\bar{\boldsymbol{\tau}}\bar{\mathbf{v}} + \bar{\mathbf{q}} + SGS_E)^T \end{bmatrix}. \tag{3.8}$$

SGS_D, SGS_M, and SGS_E are the sub-grid terms for the continuity, momentum, and energy equa-

tions. They are given by [58]:

$$SGS_D \equiv \overline{\rho \mathbf{v}^T} - \bar{\rho}\bar{\mathbf{v}}^T \tag{3.9}$$

$$SGS_M \equiv (\bar{\rho}\overline{\mathbf{v}\mathbf{v}^T} - \bar{\rho}\bar{\mathbf{v}}\bar{\mathbf{v}}^T) + (\overline{\rho\mathbf{v}\mathbf{v}^T} - \bar{\rho}\overline{\mathbf{v}\mathbf{v}^T}) \tag{3.10}$$

$$SGS_E \equiv (\bar{\rho}\overline{E\mathbf{v}^T} - \bar{\rho}\bar{E}\bar{\mathbf{v}}^T) + (\overline{\rho E\mathbf{v}^T} - \bar{\rho}\overline{E\mathbf{v}^T}) + (\overline{p\mathbf{v}^T} - \bar{p}\bar{\mathbf{v}}^T) + (\overline{\boldsymbol{\tau}\mathbf{v}} - \bar{\boldsymbol{\tau}}\bar{\mathbf{v}}) \tag{3.11}$$

It is not desirable to have a sub-grid term in the continuity equation, so Favre filtering is utilized to remove the term. Favre filtering density-weighs the variables, such that

$$\tilde{q} = \frac{\overline{\rho q}}{\bar{\rho}}. \tag{3.12}$$

Terms such as $\overline{\rho\mathbf{v}\mathbf{v}^T}$ in the momentum sub-grid term are decomposed as:

$$\overline{\rho\mathbf{v}\mathbf{v}^T} = \bar{\rho}\widetilde{\mathbf{v}\mathbf{v}^T} = \bar{\rho}\tilde{\mathbf{v}}\tilde{\mathbf{v}}^T + [G\star, H](\rho, \mathbf{v}, \mathbf{v}^T) \tag{3.13}$$

where $[G\star, H]$ is the filter error term and

$$\boldsymbol{\sigma} = [G\star, H](\rho, \mathbf{v}, \mathbf{v}^T) = \bar{\rho}(\widetilde{\mathbf{v}\mathbf{v}^T} - \tilde{\mathbf{v}}\tilde{\mathbf{v}}^T) \tag{3.14}$$

is the sub-grid stress term. Favre filtering is not applied to density or pressure, since doing so would cancel out the density, but is applied to the temperature.

$$\bar{p} = \bar{\rho}R\tilde{T}. \tag{3.15}$$

After all of the filtering has been applied to the governing equations, the fully filtered LES equations are

$$\frac{\partial}{\partial t}\int_\Omega \mathbf{Q}\, d\Omega + \oint_{\partial\Omega}(\mathbf{F}_{conv} - \mathbf{F}_{visc}) \cdot \mathbf{n}\, dS = \mathbf{0},$$

where

$$\mathbf{Q} \equiv \begin{Bmatrix} \bar{\rho} \\ \bar{\rho}\tilde{\mathbf{v}} \\ \bar{\rho}\tilde{E} \end{Bmatrix}, \tag{3.16}$$

is the filtered state vector. The fluxes are given as [58]:

$$\mathbf{F}_{conv} \equiv \begin{bmatrix} \bar{\rho}\tilde{\mathbf{v}}^T \\ \bar{\rho}\tilde{\mathbf{v}}\tilde{\mathbf{v}}^T + \bar{p}\mathbf{I} \\ \bar{\rho}\tilde{E}\tilde{\mathbf{v}}^T + \bar{p}\tilde{\mathbf{v}}^T \end{bmatrix}, \tag{3.17}$$

and

$$\mathbf{F}_{visc} \equiv \begin{bmatrix} \mathbf{0}^T \\ \check{\boldsymbol{\tau}} + SGS_M \\ (\check{\boldsymbol{\tau}}\tilde{\mathbf{v}} + \check{\mathbf{q}} + SGS_E)^T \end{bmatrix}, \tag{3.18}$$

with the computable (not modeled) heat flux, total energy, and stress tensor defined as [58]:

$$\check{\mathbf{q}} \equiv \tilde{k}\nabla\tilde{T} \tag{3.19}$$

$$\bar{\rho}\tilde{E} \equiv \frac{\bar{p}}{\gamma - 1} + \frac{1}{2}\bar{\rho}\tilde{\mathbf{v}}^T\tilde{\mathbf{v}} + \text{diag}(\boldsymbol{\sigma}) \tag{3.20}$$

$$\check{\boldsymbol{\tau}} \equiv 2\tilde{\mu}\tilde{\mathbf{D}} - \frac{2}{3}\tilde{\mu}\left(\nabla \cdot \tilde{\mathbf{v}}\right)\mathbf{I} \tag{3.21}$$

where $\tilde{k}$ is the thermal conductivity of the fluid as a function of the Favre-average temperature $\tilde{T}$ and $\boldsymbol{\sigma}$ is the sub-grid stress [148]. The momentum and energy equations each have a sub-grid term remaining, which needs to be modeled using an appropriate sub-grid model. These terms are given by [58]:

$$SGS_M \equiv \boldsymbol{\sigma} + (\bar{\boldsymbol{\tau}} - \check{\boldsymbol{\tau}}) \tag{3.22}$$

$$SGS_E \equiv \frac{1}{2}\bar{\rho}(\widetilde{\mathbf{v}\mathbf{v}^T\mathbf{v}} - \tilde{\mathbf{v}}\widetilde{\mathbf{v}^T\mathbf{v}}) + c_p\bar{\rho}(\widetilde{\mathbf{v}^T T} - \tilde{\mathbf{v}}^T\tilde{T}) + (\overline{\boldsymbol{\tau}\mathbf{v}} - \bar{\boldsymbol{\tau}}\tilde{\mathbf{v}}) + (\bar{\mathbf{q}} - \check{\mathbf{q}}) \tag{3.23}$$

where c_p is the specific heat at constant pressure.

3.3 Reynolds-Averaged Navier–Stokes

The Reynolds-averaged Navier–Stokes model, like LES, averages the Navier–Stokes equations (2.1) to model the viscous stresses (2.6). Rather than averaging in space, the RANS model is a time average, or in some cases, phase average, of the Navier–Stokes equations. This averaging makes the underlying turbulence in the viscous stresses steady.

To derive the RANS equations, the finite volume formulation of the compressible Navier–Stokes equations (2.1) is first time average. The time average of a variable is given as:

$$\bar{q}(\mathbf{x}, t) = \frac{1}{T} \int_T q(\mathbf{x}, t) dt \tag{3.24}$$

The field is then decomposed using the Reynolds decomposition into the mean $\bar{()}$ and temporally fluctuating part $()'$.

$$q(\mathbf{x}, t) = \bar{q}(\mathbf{x}, t) + q'(\mathbf{x}, t) \tag{3.25}$$

For compressible flows, Favre averaging of the mean is introduced, similar to the operations when deriving the LES equations. The Favre density-weighted time average is

$$\tilde{q} = \frac{\overline{\rho q}}{\bar{\rho}}. \tag{3.26}$$

Note that (3.26) and (3.12) appear the same, but both work with different averages, LES with spatial averages and RANS with temporal averages. The Favre decomposition of the field is

$$q = \tilde{q} + q''. \tag{3.27}$$

These two decompositions, (3.25) and (3.27) are applied as follows: (1) the Reynolds decomposition is applied to the density and pressure, and the Favre decomposition is applied to the velocity,

total energy, and temperature.

$$\begin{aligned} \rho &= \bar{\rho} + \rho', \\ p &= \bar{p} + p', \end{aligned} \tag{3.28}$$

and (2) to the velocity, total energy, and temperature

$$\begin{aligned} \mathbf{v} &= \tilde{\mathbf{v}} + \mathbf{v}'', \\ E &= \tilde{E} + E'', \\ T &= \tilde{T} + T''. \end{aligned} \tag{3.29}$$

The fully-averaged RANS equations are then defined as:

$$\frac{\partial}{\partial t}\int_{\Omega}\mathbf{Q}\,d\Omega + \oint_{\partial\Omega}(\mathbf{F}_{conv} - \mathbf{F}_{visc})\cdot\mathbf{n}\,dS = \mathbf{0},$$

where

$$\mathbf{Q} \equiv \begin{Bmatrix} \bar{\rho} \\ \bar{\rho}\tilde{\mathbf{v}} \\ \bar{\rho}\tilde{E} \end{Bmatrix}, \tag{3.30}$$

is the averaged state vector. The fluxes are given as:

$$\mathbf{F}_{conv} \equiv \begin{bmatrix} \bar{\rho}\tilde{\mathbf{v}}^T \\ \bar{\rho}\tilde{\mathbf{v}}\tilde{\mathbf{v}}^T + \bar{p}\mathbf{I} \\ \bar{\rho}\tilde{E}\tilde{\mathbf{v}}^T + \bar{p}\tilde{\mathbf{v}}^T \end{bmatrix}, \tag{3.31}$$

and

$$\mathbf{F}_{visc} \equiv \begin{bmatrix} \mathbf{0}^T \\ \check{\boldsymbol{\tau}} + RANS_M \\ (\check{\boldsymbol{\tau}}\tilde{\mathbf{v}} + \check{\mathbf{q}} + RANS_E)^T \end{bmatrix}, \tag{3.32}$$

with the computable heat flux, total energy, and stress tensor defined as [12]:

$$\check{\mathbf{q}} \equiv k\nabla\tilde{T} \tag{3.33}$$

$$\bar{\rho}\tilde{E} \equiv \frac{\bar{p}}{\gamma-1} + \frac{1}{2}\bar{\rho}\tilde{\mathbf{v}}^T\tilde{\mathbf{v}} + \frac{1}{2}\bar{\rho}\widetilde{\mathbf{v}''^T\mathbf{v}''} \tag{3.34}$$

$$\check{\boldsymbol{\tau}} \equiv 2\mu\tilde{\mathbf{D}} - \frac{2}{3}\mu\left(\nabla\cdot\tilde{\mathbf{v}}\right)\mathbf{I}, \tag{3.35}$$

Similar to the LES equations, the RANS equations have Reynolds stress terms in the momentum and energy equations. They are given by [12, pg. 233-234]:

$$RANS_M \equiv -\widetilde{\rho\mathbf{v}''^T\mathbf{v}''} \tag{3.36}$$

$$RANS_E \equiv \frac{1}{2}(\tilde{\mathbf{v}}\widetilde{\rho\mathbf{v}''\cdot\mathbf{v}''}) + \tilde{\mathbf{v}}\widetilde{\rho\mathbf{v}''^T\mathbf{v}''} + \widetilde{\rho\mathbf{v}''^T T} - \widetilde{\boldsymbol{\tau}\mathbf{v}''} + \frac{1}{2}\widetilde{\rho\mathbf{v}''\mathbf{v}''^T\mathbf{v}''}. \tag{3.37}$$

The Reynolds stress terms of (3.36) and (3.37) must be further modeled. Blazek [12] and Pope [131] both are excellent starting places for Reynolds stress models. In this work, the Reynolds stresses are modeled using the Shear-Stress Transport (SST) model of Menter [115].

3.4 Similarities between RANS and LES

The similarities between the governing equations for the RANS and LES models are summarized in Table 3.1. The only difference between the models are the sub-grid and Reynolds stress terms. This is true for both incompressible and compressible flows, as proven by Germano [60] and Suman & Girimaji [151]. This means that any Navier–Stokes code can be both a RANS and an LES code, provided the viscous fluxes $\mathbf{F}_{visc}$ are appropriately modeled in both cases.

3.5 Sub-grid Models for Large-Eddy Simulation

Having shown in the previous section that an LES model can be implemented into an existing RANS model, the next difficulty arises in how the sub-grid stresses given by (3.22) and (3.23) are to be modeled. There are a plethora of sub-grid models available for the terms in (3.22) and (3.23) [141, 58]. The first sub-grid model proposed was by Smagorinsky [148] which mod-

Table 3.1: Similarities of LES and RANS Governing Equations.

	LES	RANS
Q	Identical	
$\mathbf{F}_{conv}$	Identical	
$\mathbf{F}_{visc}$	$\begin{bmatrix} \mathbf{0}^T \\ \check{\boldsymbol{\tau}} + SGS_M \\ (\check{\boldsymbol{\tau}}\tilde{\mathbf{v}} + \check{\mathbf{q}} + SGS_E)^T \end{bmatrix}$	$\begin{bmatrix} \mathbf{0}^T \\ \check{\boldsymbol{\tau}} + RANS_M \\ (\check{\boldsymbol{\tau}}\tilde{\mathbf{v}} + \check{\mathbf{q}} + RANS_E)^T \end{bmatrix}$

eled $\boldsymbol{\sigma}$ in (3.22). The various models presented in [141] and [58] work better for some flows than others, as is the case with RANS models, and should be selected with care.

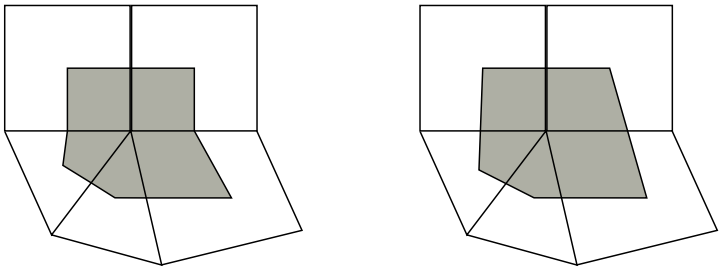
A consideration when selecting sub-grid models is the underlying accuracy of the numerical model. Typically, previous LES models implemented in the finite volume framework have been second-order accurate [129]. The seminal work of Ghosal [62] showed that the numerical error generated by the second-order finite volume method was significantly larger than the sub-grid modeled stresses. More recently, Dairay et al. [30] showed that even at high-orders, explicit sub-grid models, such as the Smagorinsky [148] model, show significant sensitivity to the numerical dissipation of the scheme. Therefore, to accurately apply LES in the finite volume framework, one must have a numerical accuracy of greater than second order regardless of the model. Boris [14] postulated even before Ghosal [62] that one could rely totally on the numerics in finite volume frameworks to generate the sub-grid model. Boris [14] called his method Monotonically Integrated LES or MILES. Now more popularly known as ILES, or Implicit LES, a higher-order finite volume method can generate the sub-grid stresses of (3.22) and (3.23) [72]. With this knowledge and the insight of Ghosal [62] and the recent work of Dairay et al. [30], the sub-grid terms herein are modeled implicitly with the numerical model. The numerical model used to achieve this will be discussed in the next two chapters.

4. NUMERICAL SCHEME AND DISCRETIZATION

This chapter will discuss how to implement a second-order scheme using the Finite Volume Method (FVM), which takes advantage of the integral form of the governing equations (2.1). The implementation of third- and higher-order scheme in the FVM context will be discussed in detail in the next chapter, but details discussed in this chapter are necessary for all orders of accuracy. The chapter presents the discretization of the computational domain, the spatial discretization of the governing equations, numerical implementation of the fluxes, including those on the boundaries, and the temporal discretization of the governing equations.

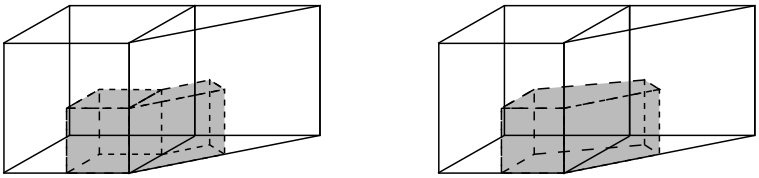
4.1 Spatial Discretization

The computational domain is constructed using a mesh composed of the combination of different polyhedral cells. This unstructured topology has the advantage of using fewer total nodes and better body-fitting capabilities. In this approach, the conserved state variables are stored and the governing equations are discretized at the vertices of the mesh. Control volumes constructed about the vertices of the mesh are made using the mesh duals. There are two approaches to constructed the mesh duals: median and centroid. In the median dual mesh paradigm, the control volume is made by connecting the lowest simplex centroids (cell, face, and edge). For the centroid duals, the highest simplex centroids are connected. Examples of these dual mesh paradigms are shown in Figure 4.1. The difference in strategies is more readily apparent in Figure 4.2, which shows the differing control volumes. For first- and second-order schemes, the median- and centroid-duals are effectively the same, since it is assumed that the flow variables are constant on dual volume faces. However, third- and higher-order schemes require the fluxes (2.4) and (2.5) be computed through each partial face of the dual volume. The median-dual approach guarantees that each face of the dual volume is a quadrilateral, which greatly simplifies the quadrature procedure [12, pg. 145]. The centroid-dual also generates quadrilateral faces, but their exact boundaries are more difficult to define. Additionally, control volumes generated by the median dual volume approach are typ-



(a) Median Dual (b) Centroid Dual

Figure 4.1: Examples of Cell-Vertex Dual Mesh Representations.



(a) Median Dual (b) Centroid Dual

Figure 4.2: Examples of Cell-Vertex Dual Control Volume Representations.

ically better, especially for skewed meshes. Therefore, the median dual volume approach is used herein, since most meshes used in typical CFD simulations have these skewed or stretched cells.

Fluxes are computed through each median dual volume face, or in an edge-based method, through each dual volume face associated with an edge. Figure 4.3 illustrates where fluxes are computed for first- and second-order. For first- and second-order schemes, it is sufficient to compute the fluxes at the midpoint of the edge. Single-point quadratures may not be sufficient for first and second-order schemes if the mesh is skewed. In those situations, the fluxes should be computed through each face centroid of the dual volume. The convective (2.4) and viscous (2.5) fluxes computed along an edge are:

$$\oint_{\partial\Omega} (\mathbf{F}_{conv} - \mathbf{F}_{visc}) \cdot \mathbf{n} dS = \sum_{j=k(i)} (\mathbf{F}_{conv} - \mathbf{F}_{visc}) \cdot |\mathbf{S}_{ij}| \tag{4.1}$$

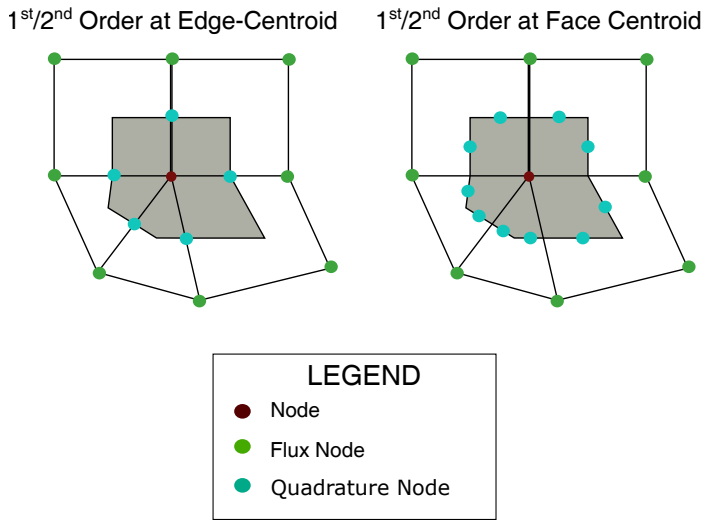


Figure 4.3: Quadrature Node Locations Using the Dual-Median Mesh Paradigm for 1^{st}- and 2^{nd}-Order Flux Computation.

where $|\mathbf{S}_{ij}|$ is the edge-projected area of the dual volumes' faces attached to an edge and $\mathbf{n}$ is the normal vector of the edge-projected area. The summation in (4.1) is over each dual face, based on the paradigm illustrated in Figure 4.3 for the specified order of the solver. The state vector and source terms are computed as cell averages:

$$\mathbf{Q}_i = \frac{\int_{\Omega} \mathbf{Q} d\Omega}{\Omega}, \tag{4.2}$$

$$\mathbf{G}_i = \frac{\int_{\Omega} \mathbf{G} d\Omega}{\Omega}. \tag{4.3}$$

The governing equation can be recast in semi-discrete form at each nodal location as:

$$\frac{\partial \mathbf{Q}_i \Omega}{\partial t} = -\sum_{j=k(i)} (\mathbf{F}_{conv} - \mathbf{F}_{visc}) \cdot |\mathbf{S}_{ij}| + \mathbf{G}\Omega. \tag{4.4}$$

If the mesh is stationary, the semi-discrete form of the governing equation at each nodal location

becomes:

$$\Omega \frac{\partial \mathbf{Q}_i}{\partial t} = - \sum_{j=k(i)} (\mathbf{F}_{conv} - \mathbf{F}_{visc}) \cdot |\mathbf{S}_{ij}| + \mathbf{G}\Omega. \tag{4.5}$$

4.2 Flux Computation

The real stumbling block in computing solutions to the Navier–Stokes lies in the computation of the convective and viscous fluxes of (4.4). The next two sections focus on the methods used to compute each.

4.2.1 Convective Fluxes

Convective fluxes (2.4) are computed using the Roe-Riemann flux-difference splitting method [138]. The convective fluxes at a dual face are approximated as [12]

$$\mathbf{F}_{conv} = \frac{1}{2} \left[\mathbf{F}_{conv} \mathbf{Q}_R + \mathbf{F}_{conv} \mathbf{Q}_L - |A_{\text{Roe}}| \cdot (\mathbf{Q}_R - \mathbf{Q}_L) \right] \tag{4.6}$$

where $|A_{\text{Roe}}|$ is the Roe matrix, and $\mathbf{Q}_R$ and $\mathbf{Q}_L$ are the states on the right and left sides of a dual face. The second term of (4.6) $|A_{\text{Roe}}| \cdot (\mathbf{Q}_R - \mathbf{Q}_L)$ is evaluated as:

$$|A_{\text{Roe}}| \cdot (\mathbf{Q}_R - \mathbf{Q}_L) = |\Delta \mathbf{F}_1| + |\Delta \mathbf{F}_{2,3,4}| + |\Delta \mathbf{F}_5| \tag{4.7}$$

where $|\Delta \mathbf{F}_1|$, $|\Delta \mathbf{F}_{2,3,4}|$, and $|\Delta \mathbf{F}_5|$ are

$$|\Delta \mathbf{F}_1| = |\tilde{V} - \tilde{c}| \frac{\Delta p - \tilde{\rho} \tilde{c} \Delta V}{2 \tilde{c}^2} \begin{bmatrix} 1 \\ \tilde{\mathbf{v}} - \tilde{c} \cdot \mathbf{n} \\ \tilde{H} - \tilde{c} \tilde{V} \end{bmatrix} \tag{4.8}$$

$$|\Delta \mathbf{F}_{2,3,4}| = |\tilde{V}| \left(\Delta \rho - \frac{\Delta p}{\tilde{c}^2} \right) \begin{bmatrix} 1 \\ \tilde{\mathbf{v}} \\ \tilde{q}^2/2 \end{bmatrix} + |\tilde{V}| \tilde{\rho} \begin{bmatrix} 0 \\ \Delta \mathbf{v} - \Delta V \cdot \mathbf{n} \\ \tilde{\mathbf{v}} \cdot \Delta \mathbf{v} - \tilde{V} \Delta V \end{bmatrix} \tag{4.9}$$

$$|\Delta \mathbf{F}_5| = |\tilde{V} + \tilde{c}| \frac{\Delta p + \tilde{\rho}\tilde{c}\Delta V}{2\tilde{c}^2} \begin{bmatrix} 1 \\ \tilde{\mathbf{v}} + \tilde{c} \cdot \mathbf{n} \\ \tilde{H} + \tilde{c}\tilde{V} \end{bmatrix}. \tag{4.10}$$

$\tilde{()}$ are the Roe-averages [138]

$$\tilde{q} = \frac{q_L\sqrt{\rho_L} + q_R\sqrt{\rho_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}} \tag{4.11}$$

$\Delta() = ()_R - ()_L$, $\tilde{H}$ is the Roe-averaged total enthalpy, $\tilde{c}$ is

$$\tilde{c} = \sqrt{(\gamma - 1)\left(\tilde{H} - \frac{\tilde{q}^2}{2}\right)}, \tag{4.12}$$

$\tilde{V}$ is

$$\tilde{V} = \tilde{\mathbf{v}} \cdot \mathbf{n}, \tag{4.13}$$

and

$$\tilde{q}^2 = \tilde{\mathbf{v}} \cdot \tilde{\mathbf{v}}. \tag{4.14}$$

It is important to note that the fluxes generated using (4.6) can have numerical issues. To explain this mathematically, $|A_{\text{Roe}}|$ in the Roe-Flux is determined as:

$$|A_{\text{Roe}}| = \mathbf{T}|\Lambda|_c\mathbf{T}^{-1} \tag{4.15}$$

where $\mathbf{T}$ and $\mathbf{T}^{-1}$ are the left and right eigenvectors, and $|\Lambda|_c$ is the diagonal matrix of the eigenvalues. The eigenvalue matrix is:

$$|\Lambda|_c = \begin{pmatrix} \tilde{V} - \tilde{c} & \tilde{V} & \tilde{V} & \tilde{V} & \tilde{V} + \tilde{c} \end{pmatrix} \mathbf{I} \tag{4.16}$$

$|\Lambda|_c$ becomes poorly conditioned or even singular if the flow approaches the speed of sound, such that $\tilde{V} - \tilde{c} \to 0$, or if $\tilde{V}$ is small. To avoid $|\Lambda|_c$ potentially becoming singular in these situations, a

perturbation, or correction, is applied to the eigenvalues:

$$|\Lambda|_{c,i} = \begin{cases} |\Lambda|_{c,i} & \text{if } |\Lambda|_{c,i} > \delta_f \\ \frac{|\Lambda|_{c,i}^2 + \delta_f^2}{2\delta_f} & \text{if } |\Lambda|_{c,i} < \delta_f \end{cases} \tag{4.17}$$

where δ_f is a small value, typically a fraction of the local speed of sound. This correction is commonly known as the Harten entropy fix [79]. The Harten entropy fix is also used to generate numerical dissipation in the scheme.

The fluxes in (4.6) require the left- and right-state $\mathbf{Q}_L$ and $\mathbf{Q}_R$ to be known. However, these states are not known directly, nor are their locations. Extrapolations from a node and quadratures on the dual faces must be determined to compute (4.6). Quadrature procedures for flux computation will be discussed in the next chapter. The extrapolation, or reconstruction, of a state at a node to the dual-face is necessary to determine either $\mathbf{Q}_L$ and $\mathbf{Q}_R$. Figure 4.4 shows a constant reconstruction of $\mathbf{Q}_L$ and $\mathbf{Q}_R$. Constant reconstructions are used only for first-order schemes. For second- and

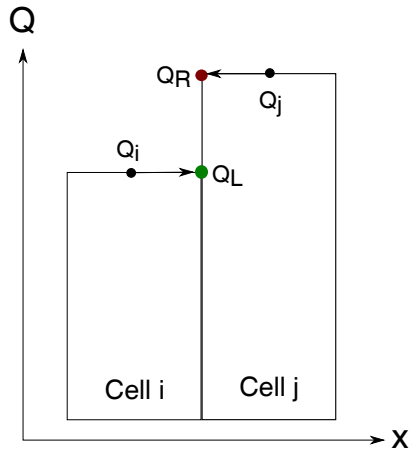


Figure 4.4: Constant Reconstruction of Left and Right States for Fluxes.

higher-order schemes, the reconstructions must use derivatives to determine $\mathbf{Q}_L$ and $\mathbf{Q}_R$. The left and right states are extrapolated to the quadrature points via Taylor expansion of the desired

order. Figure 4.5 shows a second-order linear reconstruction of $\mathbf{Q}_L$ and $\mathbf{Q}_R$. Mathematically, the

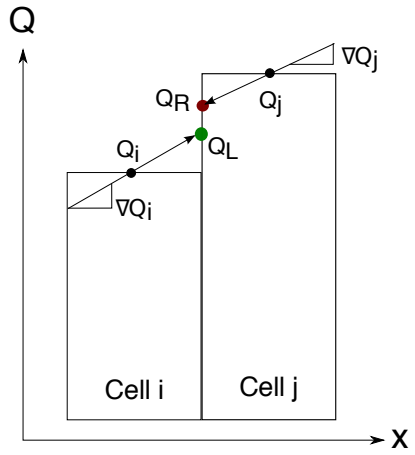


Figure 4.5: Linear Reconstruction of Left and Right States for Fluxes.

second-order linear reconstruction of the states, from the left and right, are:

$$\mathbf{Q}_L = \mathbf{Q}_i + \mathbf{r}^T \nabla \mathbf{Q}_i \tag{4.18}$$

$$\mathbf{Q}_R = \mathbf{Q}_j + \mathbf{r}^T \nabla \mathbf{Q}_j. \tag{4.19}$$

where $\mathbf{r}$ is the vector from node i or j to the edge mid-point. The left and right derivatives, $\nabla \mathbf{Q}_{i,j}$ are computed for the standard second-order method using least-squares [91].

To achieve a greater order of accuracy, a quadratic or greater piecewise reconstruction is required using a suitably accuracy gradient method to compute $\mathbf{Q}_L$ and $\mathbf{Q}_R$. Details on achieving this are given in the next chapter. A linear (or greater) piecewise reconstruction (4.19) does not guarantee monotonic solutions for linear schemes; therefore, limiters must be applied to avoid over- or undershoots of the state vector at the dual face [12, pgs. 168-169]. The general reconstruction

using limiters is now written as, from the left and right:

$$\mathbf{Q}_L = \mathbf{Q}_i + \Phi_i \cdot (\mathbf{r}^T \nabla \mathbf{Q}_i + \frac{1}{2}\mathbf{r}^T \nabla(\nabla \mathbf{Q}_i)\mathbf{r} + \ldots) \tag{4.20}$$

$$\mathbf{Q}_R = \mathbf{Q}_j + \Phi_j \cdot (\mathbf{r}^T \nabla \mathbf{Q}_j + \frac{1}{2}\mathbf{r}^T \nabla(\nabla \mathbf{Q}_j)\mathbf{r} + \ldots) \tag{4.21}$$

with $\Phi_{i,j}$ the limiter function. The limiter function can be computed by Venkatakrishnan's limiter [162] or other suitable limiter. The flux at a node is then computed as a sum of all the fluxes between a node and its neighbors:

$$\mathbf{F}_{\text{conv}} = \sum_{k \in \text{edges}} \mathbf{F}_{\text{conv, k}} |\mathbf{S}|_{i,k} \tag{4.22}$$

where $|S|$ is the edge-based area.

4.2.2 Viscous Fluxes

This section discusses how the viscous fluxes (2.5) are computed for (4.4). The viscous flux (2.5) requires both the velocity and temperature gradients as well as the state vector on the dual face. The state vector at the interface for first and second order is computed using an average of the left and right state

$$\mathbf{Q}_{\text{dual}} = \frac{1}{2}(\mathbf{Q}_R + \mathbf{Q}_L). \tag{4.23}$$

Gradients can be handled in a similar way, however, decoupling can occur when the average approach is applied directly. For second order, directional derivatives along the edge are computed as [12, pg. 176]:

$$\nabla \mathbf{Q}_{\text{dual}} = \overline{\nabla \mathbf{Q}_{\text{dual}}} - \left[\overline{\nabla \mathbf{Q}_{\text{dual}}} \cdot \frac{\Delta \mathbf{x}}{|\Delta \mathbf{x}|} - \frac{\mathbf{Q}_R - \mathbf{Q}_L}{|\Delta \mathbf{x}|} \right] \cdot \frac{\Delta \mathbf{x}}{|\Delta \mathbf{x}|} \tag{4.24}$$

where $|\Delta \mathbf{x}|$ is the edge length and $\overline{()}$ is the average. To achieve a higher-order viscous flux, direct computation of the states and gradients at the quadrature points are required. Details on how a higher-order accurate viscous flux is achieved are given in the next chapter.

4.2.3 Boundary Fluxes

On the boundaries, both the convective and viscous fluxes must be modified to accurately update the state on the boundaries. On non-wall boundaries, the convective fluxes are integrated using the states at the boundary node and state at the boundary face quadrature node. Figure 4.6 shows a typical boundary surface. From Equation (4.6), the left state is the state at the boundary

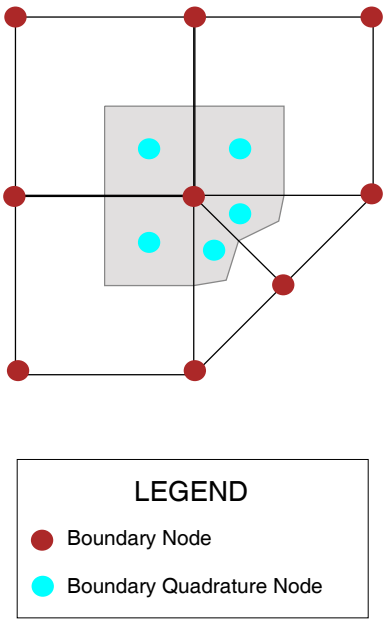


Figure 4.6: Typical Boundary Surface Showing Boundary Quadrature Locations.

node and the right state is the boundary face quadrature state. For second-order, only the left state is required to be reconstructed in (4.19), since the right state is directly known at the quadrature node. The state at the boundary quadrature node is determined from the boundary conditions using a method such as that presented in [12, Chapter 8].

For wall boundaries, the convective flux can be computed in two ways. The first approach applies the same methodology use for the other boundaries. The second approach recognizes that

on the wall boundary, the flow is subject to a non-slip condition:

$$\mathbf{v} \cdot \mathbf{n} = 0. \tag{4.25}$$

The convective flux on the wall boundary therefore reduces to:

$$\mathbf{F}_{conv} = \begin{pmatrix} 0 \\ P_w \mathbf{n} \\ 0 \end{pmatrix}, \tag{4.26}$$

where P_w is the wall pressure and set equal to the pressure at the boundary quadrature node

$$P_w = P_{qb} \tag{4.27}$$

and $\mathbf{n}$ is the normal vector on the wall boundary surface at the quadrature node. Equation (4.26) is true regardless of the numerical accuracy of the method.

The viscous fluxes described by Equation (2.5) are handled similarly to the convective fluxes on the boundaries; however the left state does not contribute to (4.23). The derivatives of the state vector in (4.24) are determined exclusively by the derivative at the boundary node in Figure 4.6. For wall boundaries, the viscous flux contribution is dependent on the thermal boundary condition. For adiabatic thermal wall boundaries, only the flux for the energy equation is computed:

$$\mathbf{F}_{visc} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ (\boldsymbol{\tau}\mathbf{v})^T \end{pmatrix} \tag{4.28}$$

For isothermal walls, $\mathbf{F}_{visc} = \mathbf{0}$ since the wall temperature is prescribed. Again, these fluxes are

true for all numerical orders of accuracy.

4.3 Temporal Discretization

The semi-discrete form of the Navier–Stokes equations (4.4) is written for a node i as:

$$\frac{\partial \mathbf{Q}_i}{\partial t}\Omega_i = \mathbf{R}_i \tag{4.29}$$

where the residuals $\mathbf{R}_i$ are:

$$\mathbf{R}_i = -\sum_{j=k(i)} (\mathbf{F}_{conv} - \mathbf{F}_{visc}) \cdot S_{ij} + \mathbf{G}_i \Omega_i - \frac{\partial \Omega_i}{\partial t}\mathbf{Q}_i \tag{4.30}$$

and $\frac{\partial \Omega_i}{\partial t}$ is the temporal evolution of the cell volume in the mesh deformation case. Herein, the mesh is fixed and $\frac{\partial \Omega_i}{\partial t} = 0$. For first- and second-order schemes, an explicit four-stage second-order Runge–Kutta scheme is used to integrate (4.29) forward in time:

$$\begin{aligned}
\mathbf{Q}_i^{(0)} &= \mathbf{Q}_i^n \\
\mathbf{Q}_i^{(1)} &= \mathbf{Q}_i^{(0)} + \alpha_1 \frac{\Delta t_i}{\Omega_i}\mathbf{R}\left(\mathbf{Q}_i^{(0)}\right) \\
\mathbf{Q}_i^{(2)} &= \mathbf{Q}_i^{(0)} + \alpha_2 \frac{\Delta t_i}{\Omega_i}\mathbf{R}\left(\mathbf{Q}_i^{(1)}\right) \\
\mathbf{Q}_i^{(3)} &= \mathbf{Q}_i^{(0)} + \alpha_3 \frac{\Delta t_i}{\Omega_i}\mathbf{R}\left(\mathbf{Q}_i^{(3)}\right) \\
\mathbf{Q}_i^{(4)} &= \mathbf{Q}_i^{(0)} + \alpha_4 \frac{\Delta t_i}{\Omega_i}\mathbf{R}\left(\mathbf{Q}_i^{(4)}\right) \\
\mathbf{Q}_i^{n+1} &= \mathbf{Q}_i^{(4)}
\end{aligned} \tag{4.31}$$

where δt_i is the time step at node i and the superscript $^{()}$ is the Runge–Kutta stage index. α are the Runge–Kutta stage coefficients. See [12, pgs. 184-185] for more details about the method. A four-stage method is used to increase the stability characteristics of the Runge–Kutta method [92, Chapter 8].

Equation (4.31) uses a global time step Δt at all of the nodes to ensure the second-order time

accuracy is met. The global time step is the minimum time step of all the dual cells:

$$\Delta t = \min_{i \in \Omega} \Delta t_i$$

where the local time step is based on the local stability conditions at the dual cell [12, pg. 189] using the Courant-Friedrichs-Lewy stability condition (CFL)

$$\Delta t_i = CFL \frac{\Omega_i}{\left(\Lambda_c^x + \Lambda_c^y + \Lambda_c^z\right)_i + 4\left(\Lambda_v^x + \Lambda_v^y + \Lambda_v^z\right)_i}, \tag{4.32}$$

where Λ_c are the convective spectral radii and Λ_v are the viscous spectral radii. The convective spectral radii are:

$$\begin{aligned} \Lambda_c^x &= (|u|+c)\Delta S_x \\ \Lambda_c^y &= (|v|+c)\Delta S_y \\ \Lambda_c^z &= (|w|+c)\Delta S_z \end{aligned} \tag{4.33}$$

where c is the local wave speed. ΔS is the projection of the control volume Ω_i onto the $yz-$, $xz-$, and $xy-$planes. They are given as:

$$\begin{aligned} \Delta S_x &= \frac{1}{2} \sum_{j=1}^{N_F \in \Omega_i} |S_x|_j \\ \Delta S_y &= \frac{1}{2} \sum_{j=1}^{N_F \in \Omega_i} |S_y|_j \\ \Delta S_z &= \frac{1}{2} \sum_{j=1}^{N_F \in \Omega_i} |S_z|_j \end{aligned} \tag{4.34}$$

where (S_x, S_y, S_z) are the components of the face vector $\mathbf{S} = \mathbf{n}\Delta S$. The viscous spectral radii

are [12, pg. 190]:

$$\begin{aligned}
\Lambda_v^x &= \frac{1}{\rho}\max\left(\frac{4}{3},\gamma\right)\left(\frac{\mu_L}{P_{rL}}+\frac{\mu_T}{P_{rT}}\right)\frac{(\Delta S_x)^2}{\Omega_i} \\
\Lambda_v^y &= \frac{1}{\rho}\max\left(\frac{4}{3},\gamma\right)\left(\frac{\mu_L}{P_{rL}}+\frac{\mu_T}{P_{rT}}\right)\frac{(\Delta S_y)^2}{\Omega_i} \\
\Lambda_v^z &= \frac{1}{\rho}\max\left(\frac{4}{3},\gamma\right)\left(\frac{\mu_L}{P_{rL}}+\frac{\mu_T}{P_{rT}}\right)\frac{(\Delta S_z)^2}{\Omega_i}
\end{aligned} \tag{4.35}$$

where P_r is the Prandtl number, μ is the dynamic viscosity, and the subscripts L and T designate laminar and turbulent values, respectively. For a purely laminar flow, $\frac{\mu_T}{P_{rT}}$ is ignored in the viscous spectral radii.

4.4 Implicit Residual Smoothing

While solving (4.29) using (4.31), the time step can become small and cause the convergence to stall. This is a particular problem for low-speed and sonic conditions. To alleviate the issues arising from small time steps, implicit residual smoothing is added to impart an implicit flavor to the explicit method of (4.31) [12, pg. 306]. The smoothing aids in stabilizing the solution, increasing convergence and CFL, and adding numerical dissipation.

The residual at a node is smoothed by taking a weighted average of the residuals of its neighbors, such that an iteration for the smoothed residual $\mathbf{R}^*$ at node i is:

$$\mathbf{R}_i^* + \epsilon\sum_{j=1}^{k}(\mathbf{R}_i^* - \mathbf{R}_j^*) = \mathbf{R}_i \tag{4.36}$$

or isolated for the smoothed residual at i,

$$\mathbf{R}_i^* = \frac{\mathbf{R}_i + \epsilon\sum_{j=1}^{k}\mathbf{R}_j^*}{1+\epsilon\sum_{j=1}^{k}1}, \tag{4.37}$$

where k is the number of neighbors of i and ϵ is the smoothing coefficient. Typically, $\epsilon \in [0.5, 0.8]$. (4.37) is solved using Jacobi iterations, converging in no more than 4 iterations [12, Chapter 10]. For viscous flows, after each iteration the residuals at walls must be prescribed to en-

sure the proper thermal condition is enforced. However, recent work has shown that using implicit residual smoothing on viscous flows can effectively destroy the solution in the boundary layer near a wall, and it is best avoided when solving (4.29) for viscous flows [18]. Additionally, any normal gradient boundary condition is essentially destroyed through the relaxation, which is undesirable. Once the residuals are smoothed, the modified residuals are used directly in (4.31) to solve the governing equations.

5. HIGHER-ORDER METHODS

This chapter describes the implementation of a higher-order (greater than second-order) scheme in the numerical model. Building upon the work in the previous chapter, this chapter presents and describes the areas from Chapter 4 that require additional work or terms to achieve higher-order accuracy. The specific areas that require additional attention to achieve higher-order accuracy are:

(1) Derivatives

(2) Quadratures

(3) Fluxes: Convective and Viscous

(4) Boundary Conditions

(5) Temporal Schemes

In the following sections, each method will be discussed in detail.

5.1 Higher-Order Derivatives

This section provides an overview of the methods for computing derivatives in CFD and describes and derives the Moving Least-Squares approach for computing derivatives with minimum restrictions.

5.1.1 Overview

When selecting ways to determine the derivatives needed for the reconstruction of the states on the left and right sides of the flux boundary, there would seem to be several viable options to choose from. Unfortunately, that is not the case. The traditional methods used to compute the derivatives in finite volume CFD codes are Green–Gauss, Least-Squares, and Weight Essentially Non-Oscillatory (WENO) schemes. Green–Gauss will only compute first derivatives. Least-Squares can be extended to higher-order derivatives but the systems become poorly conditioned and stiff with functional support over the entire domain, which can be very problematic [27].

WENO schemes can also be used to compute higher-order derivatives and have been used successfully in finite difference methods. However, the usage of WENO for finite volumes is limited to first-derivatives [179]. Additionally, WENO is undefined for triangular topologies above third order in any method [179, 39].

With the limitations of traditional methods prohibitive for higher-order schemes in finite volume methods, more recent developments for determining derivatives are explored. There are compact schemes, which are very popular for use with spectral schemes. In finite volume schemes, compact schemes are, at best, only third order when any curvature is introduced, and compact schemes are therefore only for structured topologies [149]. There are also Discontinuous Galerkin methods (DGM), which work well for both structured and unstructured meshes [147]. DGM are similar to finite elements where shape functions are generated internally to achieve a specified order [25]. Discontinuous Galerkin methods perform poorly when shocks or discontinuities are present [125], though much work is currently ongoing to rectify these deficiencies. Additionally, the computational cost of DGM is significantly higher than that of finite volume methods [106].

Meshless methods, however, do not suffer from the restrictions of any of the previous methods. Meshless methods can have arbitrary order, applied to any mesh topology, and have been used for flows with shocks and discontinuities. Stability (conditioning) issues do arise when using these methods; however, unlike standard least-squares methods, meshless methods have strategies to overcome these deficiencies. The meshless method chosen herein is Moving Least-Squares, first introduced by Lancaster & Salkauskas [95] as a tool for surface generation in computer graphics and to the CFD finite-volume community by Cueto-Felgueroso et al. [29].

5.1.2 Moving Least-Squares

This section presents the general theoretical framework and derivation of Moving Least-Squares (MLS) for computing gradients based partly on the work of Cueto-Felgueroso et al. [29]. MLS uses a weighted least-squares fit to approximate $\mathbf{q}(\mathbf{x})$ at a node in the mesh, $\mathbf{x}_I$, as $\hat{\mathbf{q}}(\mathbf{x})$ using the neighboring values of $\mathbf{q}$. Operating on each component of $\mathbf{q}(\mathbf{x})$ separately at $\mathbf{x}_I$, the MLS

approximation of q is using a set of basis vectors, $\boldsymbol{\phi}(\mathbf{x}_I)$, and the MLS shape functions, $\boldsymbol{\psi}(\mathbf{x}_I)$:

$$q(\mathbf{x}_I) \approx \hat{q}(\mathbf{x}_I) = \sum_{i=1}^{n} \phi_i(\mathbf{x}_I)\psi_i(\mathbf{x}_I) = \boldsymbol{\phi}^T(\mathbf{x}_I)\boldsymbol{\psi}(\mathbf{x}_I) \tag{5.1}$$

where n is the number of terms in the basis $\boldsymbol{\phi}(\mathbf{x}_I)$. The initial formulation given here assumes a monomial basis $\mathbf{p}(\mathbf{x})$ for $\boldsymbol{\phi}(\mathbf{x})$, though no restriction is placed on what the basis must be. Consequently, (5.1) becomes:

$$q(\mathbf{x}_I) = \sum_{i=1}^{n} p_i(\mathbf{x}_I)\psi_i(\mathbf{x}_I) = \mathbf{p}^T(\mathbf{x}_I)\boldsymbol{\psi}(\mathbf{x}_I)\ . \tag{5.2}$$

The size of the polynomial basis $\mathbf{p}(\mathbf{x}_I)$, n, is determined by the dimension, d, and order ℓ, of the polynomial basis. For example, a one-dimensional, third-order polynomial basis would have $n = 3$ terms:

$$\mathbf{p}(\mathbf{x}) = (1, x, x^2)^T.$$

A two-dimensional, third-order polynomial basis would have $n = 6$ terms:

$$\mathbf{p}(\mathbf{x}) = (1, x, y, x^2, xy, y^2)^T.$$

In three dimensions, an element of the basis $\mathbf{p}(\mathbf{x})$ is:

$$p_i(\mathbf{x}) = x^a y^b z^c, \qquad 0 \leq a + b + c \leq \ell - 1, \qquad a, b, c \in \mathbb{N} \tag{5.3}$$

Table 5.1 summarizes the size of $\mathbf{p}(\mathbf{x}_I)$, n, for one-, two-, and three-dimensional polynomial basis for first to fourth order. The full basis is written in the form:

$$\mathbf{p}(\mathbf{x}) = (p_1(\mathbf{x}), p_2(\mathbf{x}), p_3(\mathbf{x}), \cdots, p_n(\mathbf{x}))^T.$$

Table 5.1: Polynomial Basis Size Requirement for MLS.

		Order, ℓ			
		1	**2**	**3**	**4**
Dimension, d	**1**	1	2	3	4
	2	1	3	6	10
	3	1	4	10	20

To determine the MLS basis function $\boldsymbol{\psi}$ in (5.2), an error functional in a least-squares sense [66, pg. 234][92, pg. 434][44, pgs. 191-192] is written for (5.2)

$$J(\boldsymbol{\psi}\left(\mathbf{x}_I\right)) = \sum_{j \in \Omega_{\mathbf{x}_I}} W(\mathbf{x}_j - \mathbf{x}_I; r_{\text{support}}) \left[q(\mathbf{x}) - \mathbf{p}_j^T(\mathbf{x}) \boldsymbol{\psi}_j\left(\mathbf{x}_I\right)\right]^2. \tag{5.4}$$

W is a general weighting function, explicitly defined later, described over the support radius r_{support}

$$r_{\text{support}} = \frac{1}{k} r_{\max} = \frac{1}{k} \max_{j \in \Omega_{\mathbf{x}_I}} \|\mathbf{x}_j - \mathbf{x}_I\| = \frac{1}{k} \max_{j \in \Omega_{\mathbf{x}_I}} \|\Delta_{jI}\mathbf{x}\|$$

where $r_{\max}$ is the maximum radii between nodes j and I in the stencil, $\Delta_{jI} = ()_j - ()_I$, and k is a scaling parameter. Cueto-Felgueroso & Colominas [27] used $k \in [0.6, 0.7]$ in their work, but no basis was given for the choice in k. The effect of k on the support radius is shown in Figure 5.1. Extensive tests are performed herein to determine how the MLS reconstruction varies with varying scaling parameter k.

The weighting functions W in (5.4), and by extension the functional itself, are defined to have compact *local* support over the region $\Omega_{\mathbf{x}_I}$, the stencil for node $\mathbf{x}_I$. This is where MLS differs from weighted least-squares. In traditional *weighted* least-squares, the weighting function W has *global* support over the entire domain Ω [121]. In MLS, W is compact, and (5.4) is applied locally at each point in the domain [121]. Additionally, W is parameterized within the stencil by s, where s at a stencil node is defined as:

$$s_{jI} = \frac{\sqrt{(\Delta_{jI}x)^2 + (\Delta_{jI}y)^2 + (\Delta_{jI}z)^2}}{r_{\max}}. \tag{5.5}$$

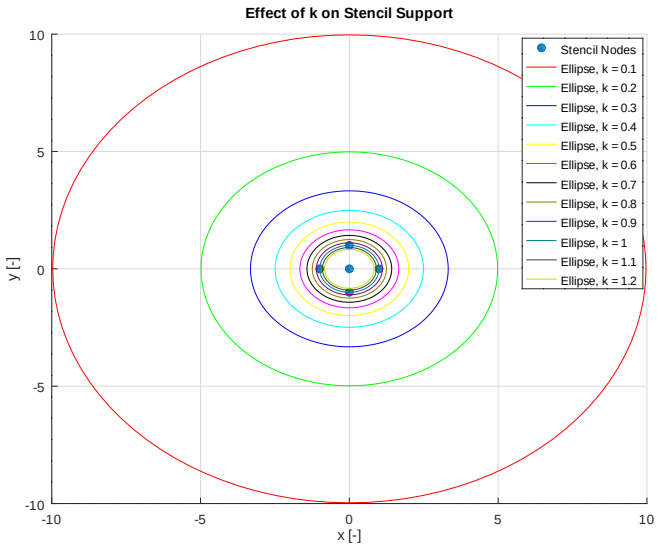


Figure 5.1: Effect of Varying the Scaling Parameter k on the Support Radius.

This function is sufficient if the points in $\Omega_{\mathbf{x}_I}$ are uniformly distributed, such as on a disk in 2D or a sphere in 3D. However, for meshes with anisotropic point distribution, such as those typically constructed for boundaries or viscous flux reconstructions, or just general triangular or tetrahedral topologies, the standard scaling function (5.5) is insufficient to accurately compute W without biasing one direction over another in $\Omega_{\mathbf{x}_I}$. To overcome this, a rectangular anisotropic weighting function may be used [124]. This method, however, is computationally expensive, since a weight for each direction is computed via a coordinate transformation. It is also difficult to determine the orientation of the points in $\Omega_{\mathbf{x}_I}$. To avoid the issues with rectangular anisotropic weights, a novel ellipsoidal anisotropic weighting function to describe $\Omega_{\mathbf{x}_I}$ is used. This approach is inspired by Principal Component Analysis, which tries to fit an n-dimensional ellipsoid to a set of data [172]. In particular, the ellipsoid generated should be a minimum volume enclosing ellipsoid (MVEE) [156]. Unlike rectangular anisotropic weights [124], the algorithm used to generate the MVEE automatically picks the correct direction of anisotropy. The MVEE of $\Omega_{\mathbf{x}_I}$ is a rather complex nonlinear programming problem, which is computationally expensive in its own right, though ways to limit convergence stalls have been implemented herein, as described in Appendix A. The

resultant of the MVEE are ellipsoidal coefficients, which are then used to define s as:

$$s_{jI} = \sqrt{\begin{array}{l} c_x(\Delta_{jI}x)^2 + c_y(\Delta_{jI}y)^2 + c_z(\Delta_{jI}z)^2 + \\ \quad 2(c_{yz}\Delta_{jI}y\Delta_{jI}z + c_{xz}\Delta_{jI}x\Delta_{jI}z + c_{xy}\Delta_{jI}x\Delta_{jI}y) \end{array}} \tag{5.6}$$

where $c_{()}$ are the ellipsoidal coefficients determined from the MVEE algorithm. It is important to state that when $\Omega_{\mathbf{x}_I}$ is nearly spherical, the MVEE algorithm tends to stall, and one should avoid using it in those cases. However, for truly anisotropic mesh topologies, the algorithm converges nicely. Further details on the MVEE algorithm can be found in [156] and Appendix A.

There are several choices for W; however, the appropriate W must be selected based on the following criteria [104]:

- $W > 0$ within $\Omega_{\mathbf{x}_I}$.
- $W = 0$ outside of $\Omega_{\mathbf{x}_I}$.
- W is a monotonically decreasing function, with a maximum at $\mathbf{x}_I$.
- W is sufficiently smooth in $\Omega_{\mathbf{x}_I}$.
- W is continuously differentiable up to the approximation order of $\hat{q}$ such that $\hat{q}$ is continuously differentiable up to the approximation order [101, 102].

It is important to note that the order of W does not contribute to the order of the MLS approximation, but only to the smoothness of the approximation [43]. The function used in this work that follows these criteria is the Wendland function [166, 146].The Wendland function is:

$$W(s) = \begin{cases} \int_s^1 t(1-t)^\mu \frac{(t^2-s^2)_+^{l-1}}{\Gamma(k)2^{l-1}} dt & s \leq 1 \\ \\ 0 & s > 1 \end{cases} \tag{5.7}$$

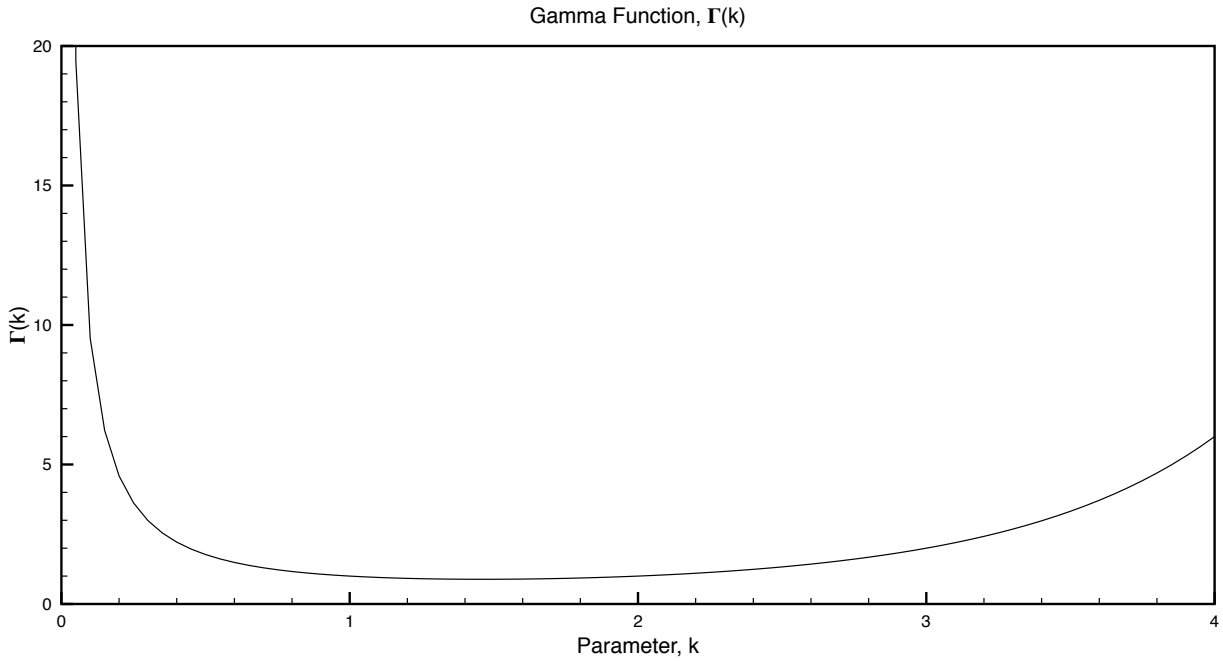


Figure 5.2: Gamma Function for a Range of Parameters k from $[0-4]$.

where $\mu = d/2 + l + 1$, Γ is the Gamma function:

$$\Gamma(k) = \int_0^\infty x^{k-1} e^{-x}\, dx \tag{5.8}$$

shown in Figure 5.2, d is the dimension of the problem, and l is the desired order. The family of Wendland functions used herein is presented in Figure 5.3.

With the weight function defined, the functional (5.4) is minimized to find $\boldsymbol{\psi}$ to such that:

$$\frac{\partial J(\boldsymbol{\psi})}{\partial \boldsymbol{\psi}} = \mathbf{0} = 2 \sum_{j \in \Omega_{\mathbf{x}_I}} W(s_j) \left[q_j(\mathbf{x}) - \mathbf{p}_j^T(\mathbf{x}) \boldsymbol{\psi}_j \left(\mathbf{x}_I \right) \right] (-\mathbf{p}_j). \tag{5.9}$$

Rearranging (5.9) and dropping $(\mathbf{x}_I)$ to simplify the notation, (5.9) becomes

$$\sum_{j \in \Omega_{\mathbf{x}_I}} W(s_j) q_j \mathbf{p}_j = \sum_{j \in \Omega_{\mathbf{x}_I}} W(s_j) \mathbf{p}_j^T \boldsymbol{\psi}_j \mathbf{p}_j \tag{5.10}$$

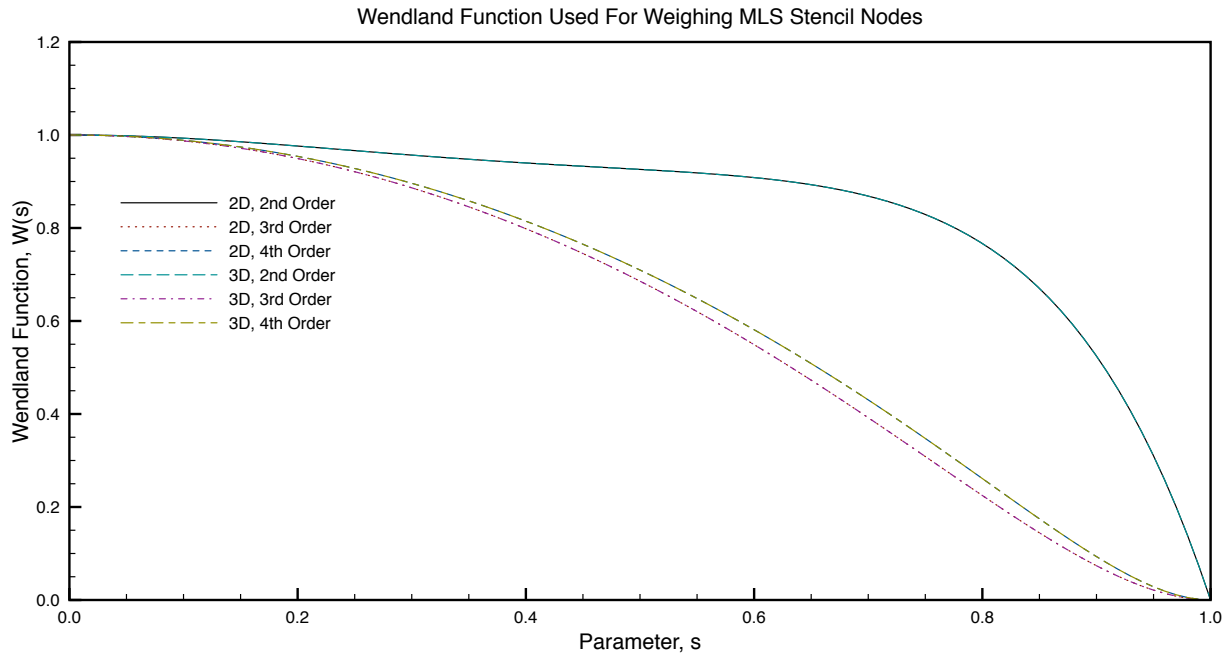


Figure 5.3: Wendland Functions for 2^{nd}-4^{th}-Order and Two- and Three-Dimensions.

With the vector identity,

$$\mathbf{a}^T\mathbf{b}\mathbf{a} = (\mathbf{a}\cdot\mathbf{b})\mathbf{a} = \mathbf{a}\mathbf{a}^T\mathbf{b},$$

where $\mathbf{a},\ \mathbf{b} \in \mathcal{R}^n$, (5.10) is rewritten:

$$\sum_{j\in\Omega_{\mathbf{x}_I}} W(s_j)q_j\mathbf{p}_j = \sum_{j\in\Omega_{\mathbf{x}_I}} W(s_j)\mathbf{p}_j\mathbf{p}_j^T\boldsymbol{\psi}_j. \tag{5.11}$$

Equation (5.11) is expressed in matrix-vector form as:

$$\mathbf{P}\mathbf{W}\mathbf{q}_{\Omega_{\mathbf{x}_I}} = \mathbf{P}\mathbf{W}\mathbf{P}^T\boldsymbol{\psi}, \tag{5.12}$$

where $\mathbf{W}$ is a diagonal $\mathcal{R}^{m,m}$ matrix of the weights W for m points in $\Omega_{\mathbf{x}_I}$, $\mathbf{P}$ is the $\mathcal{R}^{n,m}$ set of polynomial basis vectors of all the points in $\Omega_{\mathbf{x}_I}$

$$\mathbf{P} = \begin{bmatrix}\mathbf{p}(\mathbf{x}_1) & \mathbf{p}(\mathbf{x}_2) & \cdots & \mathbf{p}(\mathbf{x}_{\max_{\Omega_{\mathbf{x}_\mathbf{I}}}})\end{bmatrix}, \tag{5.13}$$

and $\mathbf{q}_{\Omega_{\mathbf{x}_I}}$ are the nodal values of the state vector in $\Omega_{\mathbf{x}_I}$. $\mathbf{P}$, using the monomial basis (5.3), forms the classic Vandermonde matrix [92, Chapter 6, Section 1], which could potentially cause problems from a stability standpoint as the order increases. Equation (5.12) is solved for the MLS shape functions $\boldsymbol{\psi}$

$$\boldsymbol{\psi} = \mathbf{M}^{-1}\mathbf{P}\mathbf{W}\mathbf{q}_{\Omega_{\mathbf{x}_I}} \tag{5.14}$$

where the moment matrix $\mathbf{M}$, a $\mathcal{R}^{m,m}$ Gram, Symmetric Positive Definite (SPD) matrix [43], is defined as:

$$\mathbf{M} = \mathbf{P}\mathbf{W}\mathbf{P}^T. \tag{5.15}$$

Substituting (5.14) into (5.2), the MLS approximation is:

$$\hat{q}(\mathbf{x}_I) = \mathbf{p}^T\mathbf{M}^{-1}\mathbf{P}\mathbf{W}\mathbf{q}_{\Omega_{\mathbf{x}_I}} = \boldsymbol{\Psi}^T\mathbf{q}_{\Omega_{\mathbf{x}}} = \sum_{j\in\Omega_{\mathbf{x}_I}} \Psi_j(\mathbf{x}_I)q_j(\mathbf{x}_j) \tag{5.16}$$

where $\boldsymbol{\Psi} = \mathbf{p}^T\mathbf{M}^{-1}\mathbf{P}\mathbf{W}$ is the $\mathcal{R}^m$ MLS basis vector. $\boldsymbol{\Psi}$ is a function of $\mathbf{x}_I$ and $\mathbf{x}_{j\in\Omega_{\mathbf{x}_I}}$ only.

5.1.3 Stability of the Moving Least-Squares

The accuracy of the MLS reconstruction of $\mathbf{q}$ is strongly tied to the order of the basis $\mathbf{p}$. An ℓ^{th} order MLS reconstruction (using an ℓ^{th}-order polynomial basis) has an approximation order of $O(\ell+1)$ [43]. Numerical tests for an ℓ^{th}-order MLS reconstruction and a general irregularly-spaced set of neighbor points, the s^{th}-order gradient accuracy is approximately $(\ell - s + 1)$ [69].

For higher-order problems, the Gram Matrix $\mathbf{M}$ becomes ill-conditioned, even for 'nice' stencils $\Omega_{\mathbf{x}_I}$, and the condition of the matrix must be considered when attempting to solve systems [92, pg. 403]. The condition of the matrix can be accessed by computing the condition number κ:

$$\kappa(\mathbf{A}) = ||\mathbf{A}||\,||\mathbf{A}^{-1}|| \tag{5.17}$$

where $\mathbf{A} \in \mathcal{R}^{n,n}$ matrix and $||\,||$ is a general matrix norm [92, pg. 190-191]. Note that there is no exact condition number that determines a 'well-conditioned' system, but in general a larger

condition number κ implies a poorly conditioned system [92, pg. 193]. Even though there are specialized methods to compute the inverse of $\mathbf{M}$ (see any of the SPD inverse methods available from LAPACK [3]), these methods may not be sufficient to overcome the conditioning issues of $\mathbf{M}$ to produce an accurate inverse. There are a few ways to modify the condition number of $\mathbf{M}$ to improve the stability of the method.

5.1.3.1 Scaled Polynomial Basis

The first method used to improve the stability of the system is to scale the polynomial basis $\mathbf{p}(\mathbf{x})$. $\mathbf{p}(\mathbf{x})$ is shifted and scaled such that when evaluated at the point $\mathbf{x}_I$, the basis takes the form $\mathbf{p}(\frac{\mathbf{x}_I - \mathbf{x}}{r_{\max}})$ [27]. An element of the scaled polynomial in three dimensions is:

$$p_i(\mathbf{x}_I - \mathbf{x}; r_{\max}) = \left(\frac{x_I - x}{r_{\max}}\right)^a \left(\frac{y_I - y}{r_{\max}}\right)^b \left(\frac{z_I - z}{r_{\max}}\right)^c, \qquad 0 \leq a + b + c \leq l - 1 \tag{5.18}$$

The weights of (5.7) are implicitly scaled through s (5.5) or (5.6). With this change to the polynomial, the MLS shape functions now become:

$$\mathbf{\Psi}^T = \mathbf{p}^T(\mathbf{0})\mathbf{C} = \mathbf{p}^T(\mathbf{0})\mathbf{M}^{-1}\mathbf{P}\mathbf{W} \,. \tag{5.19}$$

where $\mathbf{C}$ is defined as:

$$\mathbf{C} = \mathbf{M}^{-1}\mathbf{P}\mathbf{W} \,. \tag{5.20}$$

However, this is often not sufficient to significantly reduce the condition number of $\mathbf{M}$.

5.1.3.2 QR Decomposition

The next choice to improve the stability of the method is to use a matrix decomposition method to invert the Gram Matrix $\mathbf{M}$. Without modifying the neighborhood of $\mathbf{x}_I$, two matrix decomposition methods are available. One is singular value decomposition (SVD); the other method is QR decomposition. Both QR and singular value decomposition (SVD) have been used frequently in the framework of MLS. The QR method for MLS was developed herein and by Mirzaei et al.

[117]. For SVD versions of MLS, consult [23] and [69], among others. While either approach is acceptable, SVD was shown to perform slower than QR by Mirzaei et al. [117], and both cases will perform slower than the standard approach. The QR decomposition method is therefore selected to solve the inversion issue.

To derive QR decomposition for MLS, we define the decomposition

$$\mathbf{QR} = \sqrt{\mathbf{W}}\mathbf{P}^T, \tag{5.21}$$

where $\mathbf{Q}$ is an orthogonal $m \times n$ matrix and $\mathbf{R}$ is a square $n \times n$ upper-triangular matrix. With this definition of $\mathbf{QR}$, the moment matrix $\mathbf{M}$ then becomes:

$$\mathbf{M} = \mathbf{PWP}^T = (\mathbf{QR})^T\mathbf{QR} = \mathbf{R}^T\mathbf{Q}^T\mathbf{QR} = \mathbf{R}^T\mathbf{R} \tag{5.22}$$

where $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$. Furthermore, $\mathbf{C}$ becomes:

$$\begin{aligned}
\mathbf{C} &= \mathbf{M}^{-1}\mathbf{PW} = (\mathbf{R}^T\mathbf{R})^{-1}\mathbf{R}^T\mathbf{Q}^T\sqrt{\mathbf{W}} \\
\mathbf{RC} &= \mathbf{R}(\mathbf{R}^T\mathbf{R})^{-1}\mathbf{R}^T\mathbf{Q}^T\sqrt{\mathbf{W}} \\
\mathbf{RC} &= \mathbf{Q}^T\sqrt{\mathbf{W}} \\
\mathbf{C} &= \mathbf{R}^{-1}\mathbf{Q}^T\sqrt{\mathbf{W}}
\end{aligned} \tag{5.23}$$

where the identity

$$\mathbf{R}(\mathbf{R}^T\mathbf{R})^{-1}\mathbf{R}^T = \mathbf{I} \tag{5.24}$$

is used, where $\mathbf{R} \in \mathcal{R}^{nxn}$. The QR decomposed shape functions are then defined as:

$$\mathbf{\Psi}^T = \mathbf{p}^T(\mathbf{0})\mathbf{R}^{-1}\mathbf{Q}^T\sqrt{\mathbf{W}} \tag{5.25}$$

QR decomposed version of MLS only requires the inversion of $\mathbf{R}$ to determine the shape functions computed by (5.25), since the inversion for the diagonal matrix $\mathbf{W}$ is simple.

To improve the numerical stability and accuracy of the QR decomposition, pivoting can be introduced [32, 165]. Pivoting during the decomposition delays the deteriorating effects of rounding errors and has been shown to improve the decomposition of rank-deficient least-square problems [32], as in the current situation. The QR decomposition, with column pivoting can be written as:

$$\mathbf{QRP}_v^T = \sqrt{\mathbf{W}}\mathbf{P}^T, \tag{5.26}$$

where $\mathbf{P}_v$ is the pivot coefficient matrix. With this definition, the moment matrix $\mathbf{M}$ is expressed as:

$$\mathbf{M} = \mathbf{PWP}^T = (\mathbf{QRP}_v^T)^T\mathbf{QRP}_v^T = \mathbf{P}_v\mathbf{R}^T\mathbf{Q}^T\mathbf{QRP}_v^T = \mathbf{P}_v\mathbf{R}^T\mathbf{RP}_v^T = \mathbf{R}^T\mathbf{R} \tag{5.27}$$

Note that $\mathbf{M}$ is the same in both decompositions since pivoting is an orthogonal transformation. The definition of $\mathbf{C}$, however, is changed by the pivoting. Let $\mathbf{B} = \mathbf{RP}_v^T$, then

$$\begin{aligned}
\mathbf{C} &= \mathbf{M}^{-1}\mathbf{PW} = (\mathbf{P}_v\mathbf{R}^T\mathbf{RP}_v^T)^{-1}\mathbf{P}_v\mathbf{R}^T\mathbf{Q}^T\sqrt{\mathbf{W}} \\
\mathbf{C} &= (\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T\mathbf{Q}^T\sqrt{\mathbf{W}} \\
\mathbf{BC} &= \mathbf{B}(\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T\mathbf{Q}^T\sqrt{\mathbf{W}} \\
\mathbf{BC} &= \mathbf{Q}^T\sqrt{\mathbf{W}} \\
\mathbf{C} &= \mathbf{B}^{-1}\mathbf{Q}^T\sqrt{\mathbf{W}} \\
\mathbf{C} &= (\mathbf{RP}_v^T)^{-1}\mathbf{Q}^T\sqrt{\mathbf{W}} \\
\mathbf{C} &= \mathbf{P}_v\mathbf{R}^{-1}\mathbf{Q}^T\sqrt{\mathbf{W}}
\end{aligned} \tag{5.28}$$

The shape function using QR decomposition with pivoting is then:

$$\mathbf{\Psi}^T = \mathbf{p}^T(\mathbf{0})\mathbf{P}_v\mathbf{R}^{-1}\mathbf{Q}^T\sqrt{\mathbf{W}}. \tag{5.29}$$

5.1.3.3 Orthogonal Polynomial Basis

The third method used to improve the numerical stability of MLS is the usage of an orthogonal basis instead of a monomial basis (5.3) when constructing the Gram Matrix $\mathbf{M}$ [17]. The advantage of using orthogonal polynomials as the basis is that $\mathbf{P}$ will no longer be linearly (or near linearly) dependent, since orthogonal polynomials are designed as such over a scaled interval in one dimension or more dimensions. There are a few options on how to incorporate or build an orthogonal basis. First, one might consider taking $\mathbf{P}$, which with a monomial basis is a classic Vandermonde matrix [92, Chapter 6, Section 1], and perform a Gram–Schmidt process [92, Chapter 5, Section 3] operation such that $\mathbf{P}$ becomes a linearly independent set. This approach works, but is expensive, and it is difficult to extract the coefficients of the orthogonal polynomial basis created by this procedure for a general $\Omega_{\mathbf{x}_I}$. Another approach finds an optimal basis for a given stencil using an SVD decomposition of a positive-definite system $\mathbf{G} = \mathbf{X}\mathbf{X}^T$, where $\mathbf{X}$ is a square system of full rank of monomial powers spanning the stencil [17]. This approach also works, but the SVD method becomes increasingly expensive with larger stencil sizes in addition to forming $\mathbf{G}$ for each location [17]. Another issue is that the orthogonal set generated by the procedure does not include a complete set of the monomials for a given order, and 'extra' stencil points, which are naturally added to the stencil and would normally be required to fully span the monomial basis, instead create an incomplete higher-order basis [17].

Both methods may be circumvented by predefining the orthogonal basis. Noting that the stencil $\Omega_{\mathbf{x}_I}$ is easily scaled on the interval$[-1, 1]^2$ or $[-1, 1]^3$, any orthogonal polynomial for any set of nodes may be selected, such as Legendre, Hermite, or Chebyshev (first or second kind). While these orthogonal polynomials are defined in one-dimension over the interval $[-1, 1]$, a two- or three-dimensional orthogonal polynomial set may be constructed [40]:

$$P_o(x, y, z) = P_i(x)P_j(y)P_k(z) \quad 0 < i + j + k < n \tag{5.30}$$

where P_o is the combined orthogonal basis, (i, j, k) are permutation indexes, and n is the specified

order of the polynomial. Another advantage to this method is that the coefficients of the polynomial are known in advance, so $\mathbf{\Psi}^T$ may be computed correctly:

$$\mathbf{\Psi}^T = \hat{\mathbf{p}}^{\,T}(\mathbf{0})\mathbf{C}\,, \tag{5.31}$$

where $\hat{\mathbf{p}}^{\,T}(\mathbf{0})$ is the vector resulting from evaluating each polynomial in the span at $\mathbf{0}$. This happens in the monomial case, but since the combined monomial basis $(1, x, y, z...)$ evaluated at $\mathbf{0}$ is the same as evaluating $(1(\mathbf{0}), x(\mathbf{0}), y(\mathbf{0}), z(\mathbf{0}), ...)$, the change in notation is warranted for clarification. A similar procedure must be carried out when computing derivatives.

The orthogonal polynomials mentioned above are the obvious candidates; however, there are special orthogonal polynomials that are better suited for the natural distribution of the nodes in $\Omega_{\mathbf{x}_I}$. $\Omega_{\mathbf{x}_I}$ forms a radial basis, so orthogonal polynomials defined over the unit sphere might be more appropriate than those defined over essentially the unit box. There are two orthogonal polynomials that are used herein: the Zernike [111] and the Gegenbauer [153, 109]. Zernike are a class of orthogonal polynomials used on the unit sphere in the optics community [111]. The Zernike polynomials are based on a radial function, but have expressions in Cartesian coordinates that are used herein [111]. The Gegenbauer polynomials are ultra-spherical polynomials as determined by their generating weight function [153]. The exact form of the Gegenbauer polynomial is [153]:

$$\frac{1}{(1-2xt+t^2)^\alpha} = \sum_{n=0}^{\infty} C_n^\alpha(x)t^n \tag{5.32}$$

where C_n^α is the Gegenbauer polynomial. If $\alpha = 1/2$, the Gegenbauer polynomials are equivalent to the Legendre polynomials [1]. If $\alpha = 1$, the Gegenbauer polynomials are equivalent to the Chebyshev polynomials of the second kind [1]. In this work, $\alpha = 2$, such that the weighting function used to define the orthogonalization ($n \neq m$):

$$\int_{-1}^{1} C_n^\alpha(x)C_m^\alpha(x)w(x)dx = 0 \tag{5.33}$$

is:

$$w(z) = (1 - z^2)^{\alpha - \frac{1}{2}} = (1 - z^2)^{1.5}. \tag{5.34}$$

The first five Gegenbauer polynomials with $\alpha = 2$ are plotted in Figure 5.4.

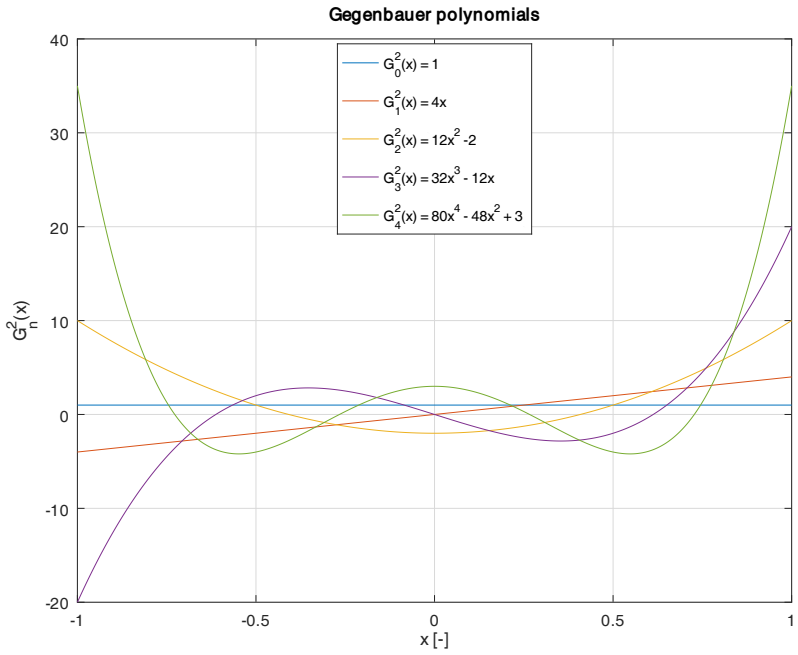


Figure 5.4: First Five Gegenbauer Polynomials with $\alpha = 2$.

Both the Zernike and Gegenbauer polynomials may be used directly when using (5.5) to define W. However, when using (5.6) to determine W, a transformation is required to map $\Omega_{\mathbf{x}_I}$ to a unit disk or sphere before either radial orthogonal polynomials are valid.

5.1.3.4 Affine Moving Least-Squares

The last method used to improve the stability is by using an affine map on the stencil $\Omega_{\mathbf{x}_I}$. The nodes in $\Omega_{\mathbf{x}_I}$ are 'quasi' spherical. While the s (5.5) assumes the points are distributed in a spherical manner about $\mathbf{x}_I$, as mentioned earlier in 5.1.2 and in A, the better ellipsoidal fitting using (5.6) is explicitly not spherical. Writing the ellipsoidal coefficients in the quadratic form $\mathbf{H}$ (see Appendix A, (A.50)), a mapping between the ellipsoid and the unit sphere can be generated [132].

These results are derived from John's ellipsoid theorem, for any ellipsoid there exists a unit ball (or sphere) under an affine transformation [73]. There are two choices to map to the unit sphere: SVD or Cholesky [132]. The SVD method is essentially a principle component analysis [172] of the ellipsoid $\mathbf{H}$:

$$\mathbf{H} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* \tag{5.35}$$

where $\mathbf{U}$ is an $m \times m$ orthogonal matrix composed of the eigenvectors of $\mathbf{H}^T\mathbf{H}$, $\mathbf{V}$ is an $n \times n$ orthogonal matrix of the eigenvectors of $\mathbf{H}^T\mathbf{H}$, and $\mathbf{\Sigma}$ is a diagonal matrix of the eigenvalues of $\mathbf{H}^T\mathbf{H}$. The mapping to the unit sphere is [132]

$$\mathbf{S} = \mathbf{H}\sqrt{\mathbf{\Sigma}}\mathbf{V}^* \tag{5.36}$$

which, mapped to the ellipsoid is

$$\mathbf{H} = \mathbf{U}\sqrt{\mathbf{\Sigma}}\mathbf{S}, \tag{5.37}$$

where $\mathbf{S}$ is the quadratic form of the unit sphere. While mathematically the SVD mapping is affine (*invertible*), numerically the transformation is in general non-affine, that is when applying the 'inverse' transformation to the unit sphere, the original ellipsoid is not restored. This problem is due to the nature of the method, that while stable, the SVD decomposition involves operations that have known round-off and truncation issues. The Cholesky decomposition is much stabler and always affine. Additionally, the Cholesky and SVD method are equivalent when performed on $\mathbf{H}$, since $\mathbf{H}$ is symmetric positive-definite (see Appendix G for details). The Cholesky mapping of $\mathbf{H}$ is

$$\mathbf{H} = \mathbf{L}\mathbf{L}^* \tag{5.38}$$

where $\mathbf{L}$ is a lower triangular matrix and $\mathbf{L}^*$ is the conjugate transpose. The forward (to unit sphere) transformation is

$$\mathbf{S} = \mathbf{H}\mathbf{L}^* \tag{5.39}$$

and the inverse transformation is

$$\mathbf{H} = \mathbf{LS} \tag{5.40}$$

Graphically, the transformation of the MLS basis is shown in Figure 5.5. An additional benefit to

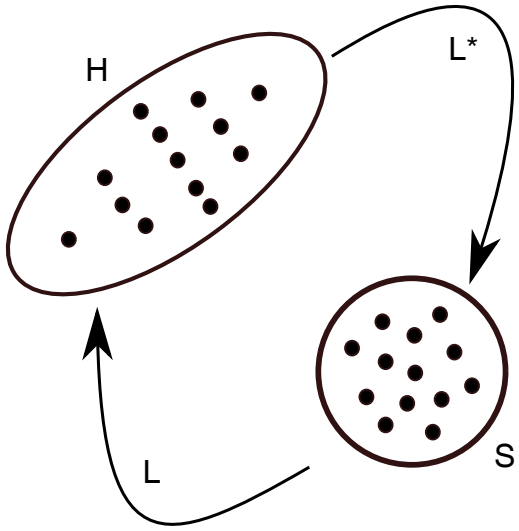


Figure 5.5: Graphical Representation of the Forward and Backward Affine Transformation of $\Omega_{\mathbf{x}_I}$ from a General Ellipsoid $\mathbf{H}$ to a Unit Sphere $\mathbf{S}$.

the Cholesky Decomposition is that it is substantially faster ($O(mn^2)$) than the SVD decomposition ($O(mn^2 + n^3)$). The transformed coordinates of $\Omega_{\mathbf{x}_I}$ are [132]:

$$\Delta_{jI}\mathbf{x_S} = \mathbf{L}^*\Delta_{jI}\mathbf{x_H} \tag{5.41}$$

where $()_{\text{us}}$ are the coordinates of the unit sphere and $()_{\text{ell}}$ are the coordinates of the ellipsoid. The maximum radius is now $r_{\text{max}} = 1$. Since the unit sphere is a dual of the ellipsoid, from John's Theorem [73], the MLS basis $\mathbf{\Phi}$ of the unit sphere is the same basis as on the ellipsoid. The derivatives of $\mathbf{\Phi}$, however, have to be mapped back to the ellipsoid. This is discussed in the derivative section. The procedure given above to compute the MLS basis $\mathbf{\Phi}$ using an affine map is as follows:

Algorithm 5.1 Computing MLS Basis Functions of the Affine Map.

Require: Quadratic form of the ellipsoid determined via the Todd Algorithm A.6 described in Appendix A

1: Perform the Cholesky Decomposition of $\mathbf{H}$ via (5.38).

2: Transform the coordinates of the stencil $\Omega_{\mathbf{x}_I}$ to the unit sphere using (5.41).

3: Compute MLS basis functions as described in Section 5.1.2

The effect of the algorithm, coupled with an orthogonal polynomial basis (5.30), is to greatly reduce the condition number of the Gram Matrix $\mathbf{M}$. For example, on a flat plate-type grid, the condition number using the general MLS formulation is on average 10^{21}. When using the affine MLS formulation and the Gegenbauer basis, the average condition number of the Gram Matrix is only 40, which greatly improves the overall stability and accuracy of the method. Algorithm 5.1 can also be used with (5.5) general spherical weight scaling function, since a sphere is a special case of an ellipsoid.

5.1.4 Stencil Selection for MLS

With the overall method for computing MLS described, it is now time to discuss how $\Omega_{\mathbf{x}_I}$ is determined. Beginning with a node $\mathbf{x}_I$, neighboring points are collected into a stencil. In a meshed framework, the nodal connectivities are known *a priori*, and this information is used to build the stencil for $\mathbf{x}_I$. Groups of nodes, or layers in the stencil, are collected from the nodal and edge connectivities based on the desired spatial order. This is shown in Figure 5.6 for second- and third-order stencils in two dimensions. An additional layer of nodes would generally be needed for a fourth-order reconstruction. The polynomial basis given by (5.3) and (5.18), and by extension the methods given to compute the shape functions and their derivatives, require a minimum number of nodes (and layers) to exist within the stencil. The concept of a minimum set comes from interpolation theory [92, Chapter 6, Section 1]. To avoid (5.15) becoming singular, the minimum number of nodes required for a given polynomial order is equal to the number of terms in the

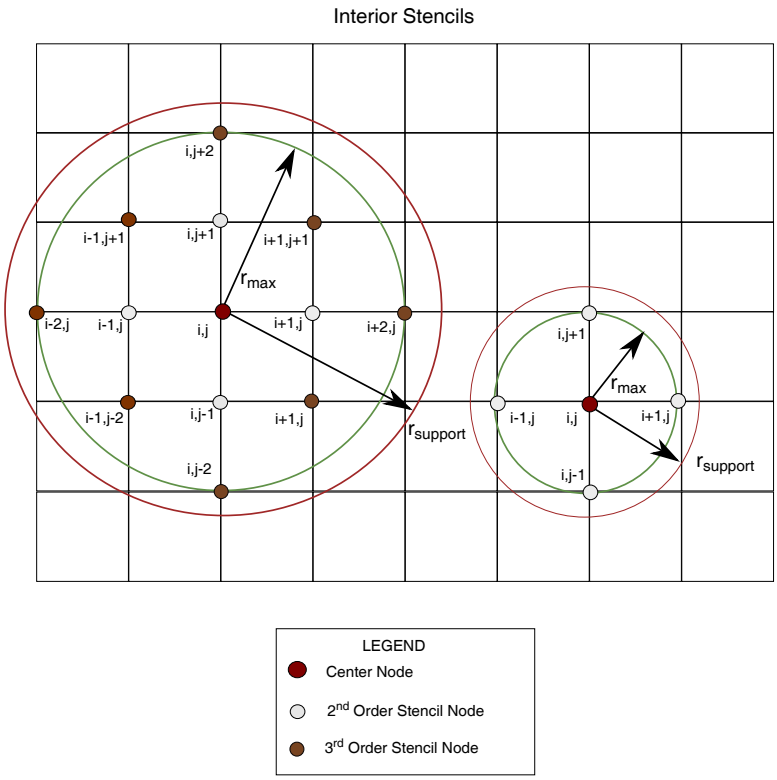


Figure 5.6: 2^{nd} and 3^{rd}-Order $\Omega_{\mathbf{x}_I}$ for MLS Showing Support Radius.

polynomial basis. For example, (5.3) requires at minimum of 20 nodes to be contained within the stencil. Often, this minimum requirement is not sufficient to fully resolve the shape functions, or in the worst-case scenario, even compute them. Therefore, it is common practice to include 20-50% more nodes than are minimally required in the stencil to improve the condition of M from (5.15) [69]. Table 5.2 summarizes this information for first through sixth orders in two dimensions. Gossler [69] has also done extensive work in showing the effects of the size of the stencil on the accuracy and computability of the shape functions and reconstructed derivatives, and the interested reader is referred to that work for more information.

5.1.4.1 Periodic Boundary Stencils

Stencils for periodic boundary nodes take the same general form as interior nodes. Figure 5.7 shows an example translational periodic boundary stencil. Nodes from the slave stencil are translated over to the master stencil for completeness. Slave stencil nodes that are on the slave periodic

Table 5.2: Minimum Size of the $\Omega_{\mathbf{x}_I}$.

	Size of Neighborhood $\Omega_{\mathbf{x}_I}$	
Order, ℓ	Minimum	In-Practice Minimum
1	1	3
2	4	6
3	10	15
4	20	30
5	35	53
6	56	84

boundary are not translated, since they have matching counterparts on the master boundary and would be redundant. The relative location of a slave stencil node to the master is:

$$d\mathbf{x}_{\text{sst}\to m} = \mathbf{x}_{\text{sst}} - \mathbf{x}_s \tag{5.42}$$

where $()_{\text{sst}}$ is the slave stencil node, $()_{\text{s}}$ is the slave node, and $()_{\text{m}}$ is the master node. Since in the relative frame the slave and master are the same node, the coordinate difference between a slave stencil node and the slave is the same after it is shifted. When multiple periodic boundaries are used (two or three dimensional periodic boundaries), special care should be taken to ensure that the proper relative location is maintained.

For rotational periodic boundaries, the process to generate stencils is the same as for translational periodic boundaries, except when computing the relative distance. Instead of computing a simple difference, the difference is rotated

$$d\mathbf{x}_{\text{sst}\to m} = \mathbf{R}_\theta(\mathbf{x}_{\text{sst}} - \mathbf{x}_s) \tag{5.43}$$

with the rotation matrix $\mathbf{R}_\theta$ to properly locate the slave stencil node within the master stencil.

An important result from the generation of periodic stencils is that the explicit requirement of geometric quantities on either side of the master or slave periodic boundary faces must be periodic with their counterparts. Before, when dealing with first- and second-order methods, one generally

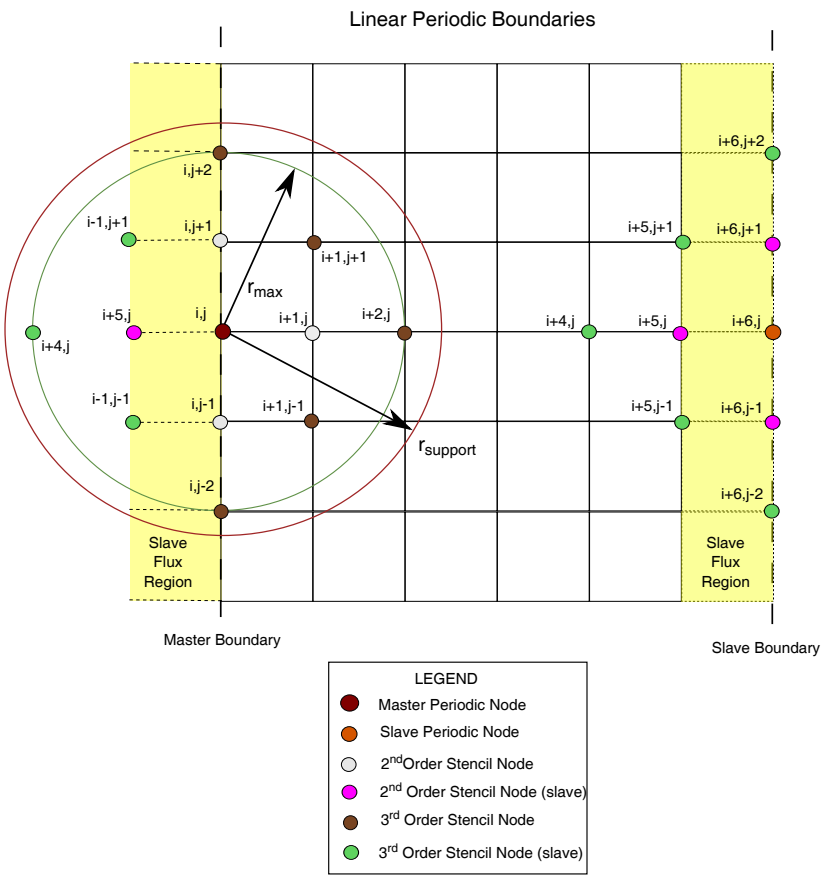


Figure 5.7: 2^{nd} and 3^{rd}-Order Translational Periodic Boundary $\Omega_{\mathbf{x}_I}$ for MLS Showing Support Radius.

assumes that only the information closest (only the first cells) to the master/slave boundaries is periodic with respect to the boundaries. However, since the stencils for third- and fourth-order span at least four cells away from the boundary all of the related geometric information is also now subject to the periodicity requirements. By extension, this also includes any information related to the state vector, as is the case with lower-order methods. This effectively means that any fluxes or other computations carried out in the slave periodic buffer zone do not need to be computed. However, in practice, the logic involved to not compute information over this large of an area is rather difficult and involved. Therefore, as with the standard lower-order approach, calculations are avoided only on the slave boundary itself.

5.1.4.2 *Non-Periodic Boundary Stencils*

Non-periodic boundary node stencils require special care. In general, stencils for boundary nodes will never automatically satisfy the minimum size requirement, due to their lack of connectivity. Figure 5.8 shows two approaches to reach the least-squares minimum threshold: augmented stencils and ghost nodes.

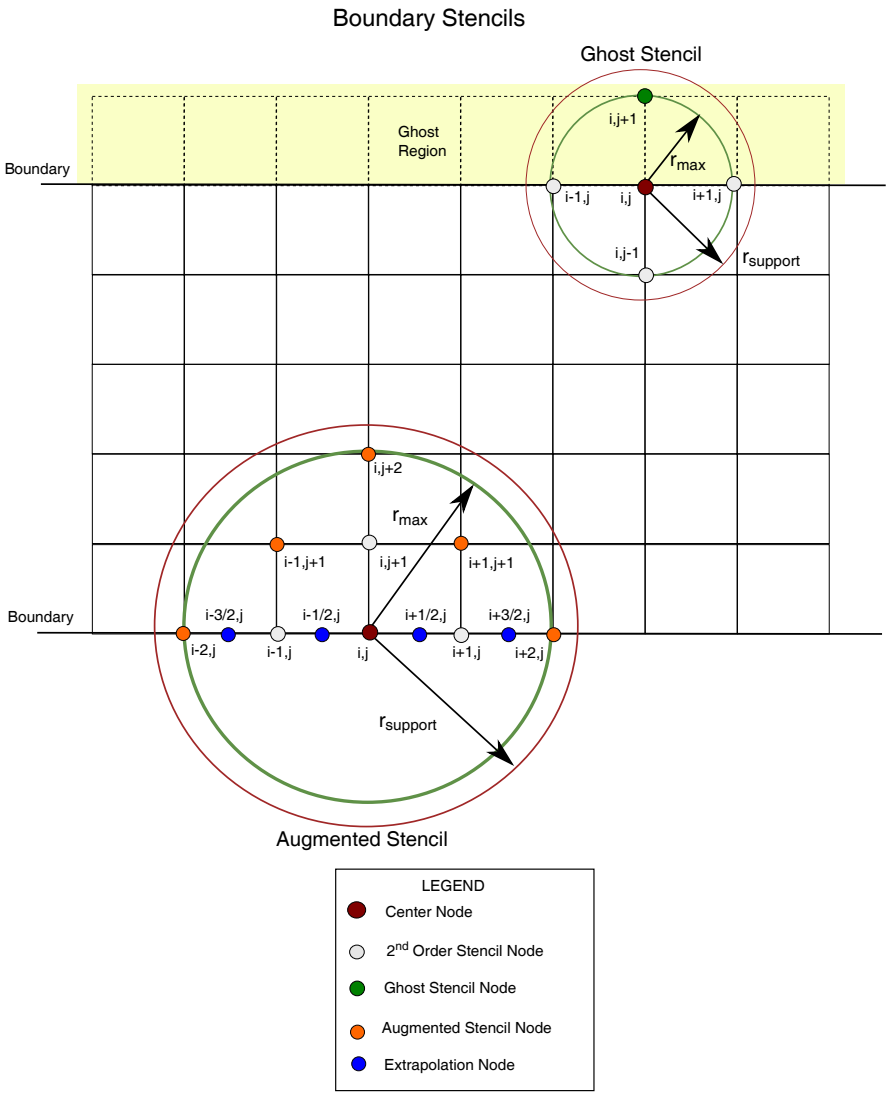


Figure 5.8: Augemented and Ghost Boundary $\Omega_{\mathbf{x}_I}$ for 2^{nd}-Order MLS Showing Support Radius.

5.1.4.2.1 Augmented Stencils The augmented stencil is the first approach for non-periodic boundary stencils. This approach is shown at the bottom of Figure 5.8. Augmented stencils are inspired from finite differences. In finite differences, an additional node or nodes is added to the extremity of the finite-difference stencil to maintain the desired order of the difference. This approach is commonly referred to as forward or backward differencing, depending on how the boundary and node are oriented. In the framework of this dissertation, typically a single layer is enough to increase the population of the stencil for node (i, j) to least-square levels and improve the condition number of $\mathbf{M}$ from (5.15) [23]. Referring to Figure 5.8, the nodes necessary to maintain a second-order MLS reconstruction at node (i, j) would be $(i-2, j)$, $(i-1, j+1)$, $(i, j+2)$, $(i+1, j+1)$, and $(i+2, j)$. No extrapolation is needed to define these extra nodes, as they exist in the interior of the mesh. Additional boundary nodes, known as extrapolation or zero-area nodes [28, 23], can also be added to the augmented stencil. Zero-area nodes are places at the midpoints of the boundary faces. From Figure 5.8, these nodes are $(i-3/2, j)$, $(i-1/2, j)$, $(i+1/2, j)$, and $(i+3/2, j)$. Zero-area nodes by themselves do not add enough nodes to the overall stencil of (i, j) to satisfy the least-squares problem. These nodes are normally just used to enrich the data set for the reconstruction of the derivatives. However, since the extrapolation nodes are usually a function of neighboring values, the data added by them is minimal.

5.1.4.2.2 Ghost Nodes The other approach for defining a boundary stencil uses true ghost nodes, as shown at the top of Figure 5.8. Ghost nodes are points outside of the regular domain of interior points. A ghost node is generated during the mesh generation process. This is the easiest and safest approach, as the interior mesh can generally be exactly extruded into the exterior of the domain without too much difficulty. Figure 5.9 shows an example structured mesh and its ghost mesh generated using the extruded approach.

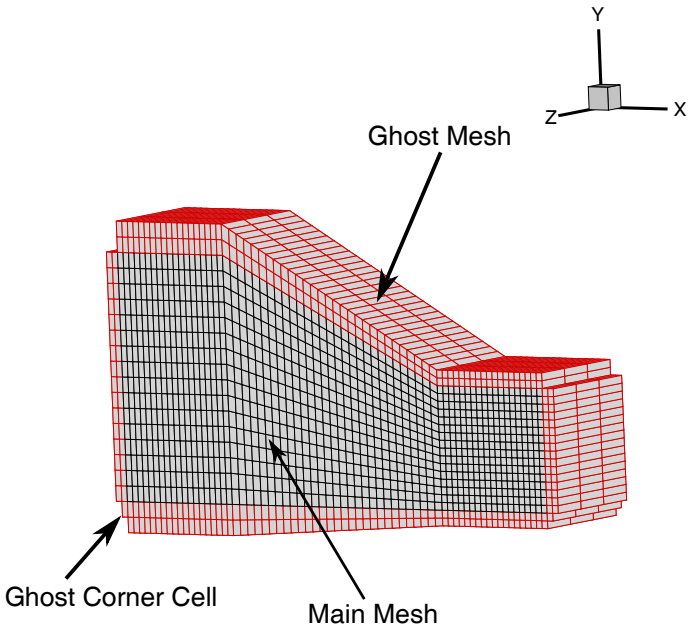


Figure 5.9: Example Ghost Node Mesh Topology Built from Extrusion of Main Mesh.

To properly enforce the boundary conditions at boundary nodes, the ghost nodes are aligned normal to the boundary after the initial extrusion. Effectively, the ghost nodes are projected onto the boundary normals through a vector projection:

$$proj_{gn} = \frac{\mathbf{g} \cdot \mathbf{n}}{\mathbf{n} \cdot \mathbf{n}} \mathbf{n} \tag{5.44}$$

where $\mathbf{n}$ is the boundary normal and $\mathbf{g} = \mathbf{x}_g - \mathbf{x}_m$. The subscripts g and m represent the ghost and main mesh nodes. Figure 5.10 shows the resultant projection of the ghost mesh of Figure 5.9.

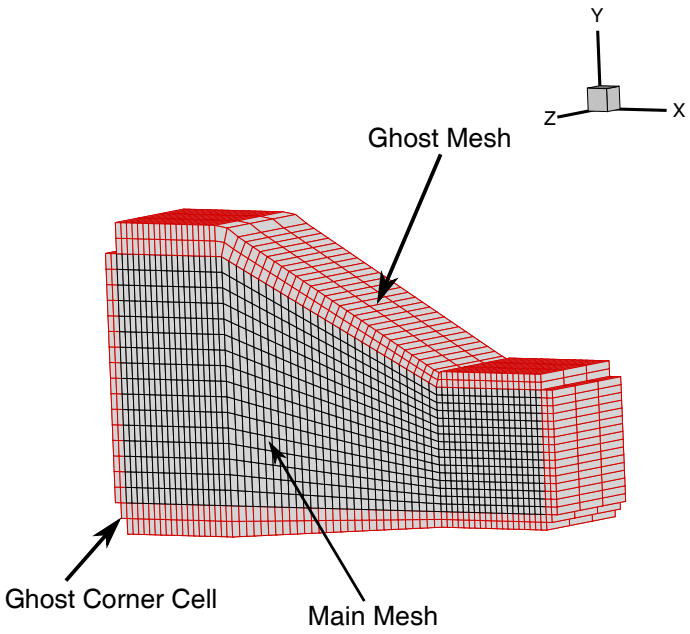
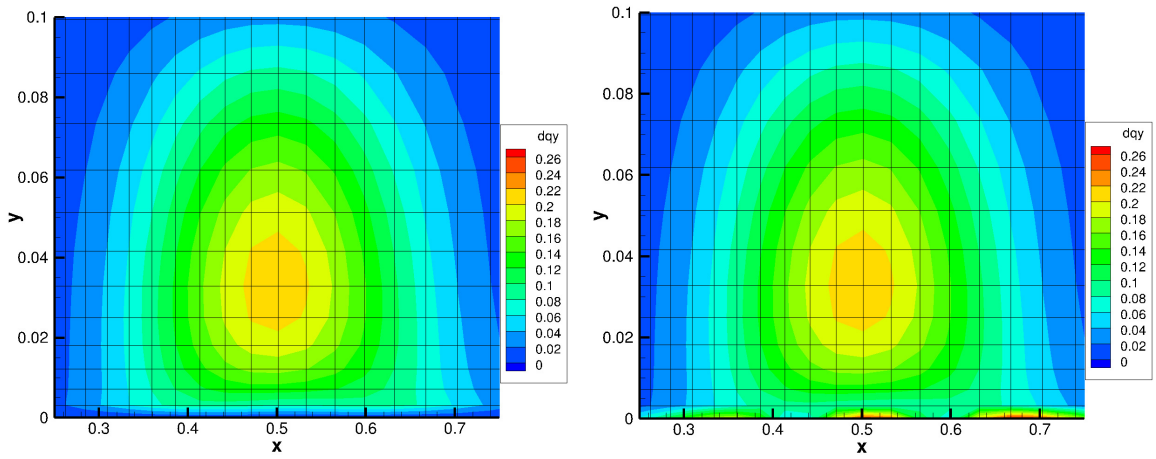


Figure 5.10: Example Ghost Node Mesh Topology Built from Projection onto the Boundary Normals of the Main Mesh.

Corners are added for the ghost mesh where non-wall boundaries meet. This ensures the geometric quantities, such as the cell volume and edge-based area $|\mathbf{S}|$, are correctly computed at these locations. Ghost nodes in corner regions are not used to update the solution, since the flow field is ambiguous in these regions [12, Chapter 8, pg. 200]. Additionally, ghost nodes from corner regions are not used in stencils. The procedure to define the flow variables at a ghost node is described later in the chapter when boundary conditions are discussed.

5.1.4.2.3 Comparison of Stencils Having introduced both the augmented stencil and the concept of ghost nodes, it's imperative to determine which is better in the context of the MLS reconstruction. For that purpose, a simple demonstration is made using a Gaussian:

$$f(x,y) = e^{-\frac{(x-x_o)^2+(y-y_o)^2}{\delta^2}} \tag{5.45}$$

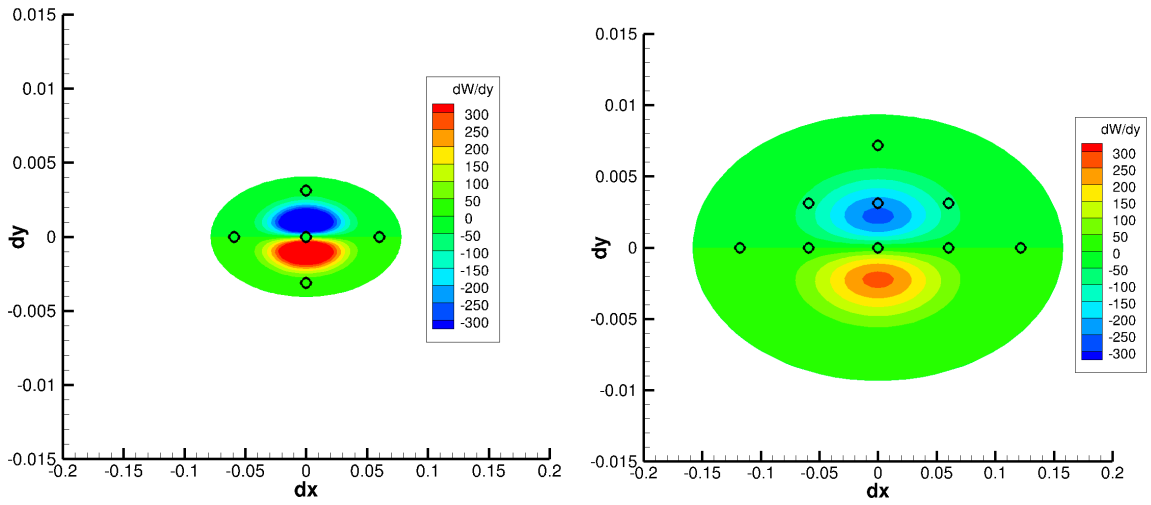


(a) Ghost Nodes

(b) Augmented Stencil

Figure 5.11: Error for $\frac{\partial G}{\partial y}$ Near Boundary for 81^2 Stretched Mesh using Augmented and Ghost Node $\Omega_{\mathbf{x}_I}$ Paradigm.

with $\delta^2 = 0.02$ centered at $(x_0, y_0) = (0.5, 0.0)$, or on the boundary. The mesh used here has 81 nodes with an initial spacing of 0.031625 in each direction, and a dimensional growth rate of 1.3. The growth rate is defined as the ratio between the height (or width) of a lower cell and an adjacent cell. The mesh is meant to mimic one that would typically be used for a flat plate. The Wendland weight function (5.7) was used when constructing the MLS approximation. Since this case is used only to show the differences between the two boundary stencil methods, only first derivatives (second-order) are computed. Each method was solved using the Pivoting QR decomposition technique (5.29). For the ghost node results, the values at the ghost are directly determined from (5.45). Results are shown for the wall normal derivative on the boundary in Figure 5.11. For the interior field in both cases, the results are the same as expected, since the stencils are the same there. When looking at the boundary, however, it is clear the answers are different. For the augmented stencil, over and undershoots of the derivative are manifested, and the largest error of the field occurs on the boundary. The ghost node results, on the other hand, look excellent at the wall. In fact, some of the lowest error occurs at the wall, most likely due to the small spacing in the wall-normal direction there. Why does the augmented stencil not work here? For finite differences, one-sided differences are typically used on boundaries and can even be easily constructed



(a) Ghost Nodes

(b) Augmented Stencil

Figure 5.12: Weights for $\frac{\partial G}{\partial y}$ for a Boundary Node $\Omega_{\mathbf{x}_I}$ Using an Augmented or Ghost Node $\Omega_{\mathbf{x}_I}$ Paradigm.

to have the same order of accuracy of centered differences, even for nonuniform meshes [49]. However, for radial functions (see (5.7)), a one-sided difference is not sufficient to compute radial functions or their derivatives [51]. Additionally, errors for radial function reconstructions are typically the highest near the boundary, as shown by Wang & Liu [164] and others. In Figure 5.12, the derivatives of the weight function over the entire stencil are shown for the ghost node and augmented stencil case. For one, the stencil using ghost nodes is much more compact than the augmented stencil, and the aim should be to have as compact stencils as possible. Disregarding the magnitude of the weight function derivative, which will be different between the two since their footprint is different, notice that the augmented stencil fails to capture any of the lower half of the radius of influence. Without any through-center information of the radial function, one cannot hope to accurately compute the derivatives in that direction. This assessment is true whether using isotropic (5.5) or anisotropic (5.6) definition of the weights, since the MVEE algorithm will still choose the 'larger' wall normal dimension for the minor axis and have similar weights as to those shown in Figure 5.12. Therefore, it is beneficial to eliminate issues near boundaries as best as possible. With ghost nodes, the boundaries are effectively eliminated, and their derivatives can

reasonably and accurately be computed.

5.1.5 Derivatives: Full, Semi-Diffuse, and Fully-Diffuse

This section describes the methodology for computing derivatives for both the standard and QR versions of MLS described herein. The derivatives of the shape functions are used to compute the reconstructions of the derivative fields, where the reconstruction is:

$$\frac{\partial^{a+b+c} q(\mathbf{x}_I)}{\partial x^a \partial y^b \partial z^c} \approx \sum_{i \in \Omega_{\mathbf{x}_I}} \frac{\partial^{a+b+c} \boldsymbol{\Psi}(\mathbf{x})}{\partial x^a \partial y^b \partial z^c} q_i, \qquad 0 \leq a+b+c \leq l \tag{5.46}$$

where $\frac{\partial^{a+b+c} \boldsymbol{\Psi}(\mathbf{x})}{\partial x^a \partial y^b \partial z^c}$ are the derivatives of the MLS shape functions given by (5.19) and (5.25). There are three methods used to compute the MLS derivatives: full, semi-diffuse, and full-diffuse.

5.1.5.1 Full Derivatives

The full derivatives of the MLS shape functions are derived by differentiating (5.19) with respect to the *local* coordinates centered at x_I to yield:

$$\frac{\partial \boldsymbol{\Psi}}{\partial x_i} = \frac{\partial \mathbf{p}^T(\mathbf{0})}{\partial x_i} \mathbf{C} + \mathbf{p}^T(\mathbf{0}) \frac{\partial \mathbf{C}}{\partial x_i}. \tag{5.47}$$

Higher-order derivatives for the shape function follow directly from (5.47). The second and third derivatives for the shape function are then:

$$\frac{\partial^2 \boldsymbol{\Psi}}{\partial x_j \partial x_i} = \frac{\partial^2 \mathbf{p}^T(\mathbf{0})}{\partial x_j \partial x_i} \mathbf{C} + \frac{\partial \mathbf{p}^T(\mathbf{0})}{\partial x_i} \frac{\partial \mathbf{C}}{\partial x_j} + \frac{\partial \mathbf{p}^T(\mathbf{0})}{\partial x_j} \frac{\partial \mathbf{C}}{\partial x_i} + \mathbf{p}^T(\mathbf{0}) \frac{\partial^2 \mathbf{C}}{\partial x_j \partial x_i} \tag{5.48}$$

$$\begin{aligned} \frac{\partial^3 \boldsymbol{\Psi}}{\partial x_k \partial x_j \partial x_i} &= \frac{\partial^3 \mathbf{p}^T(\mathbf{0})}{\partial x_k \partial x_j \partial x_i} \mathbf{C} + \frac{\partial^2 \mathbf{p}^T(\mathbf{0})}{\partial x_j \partial x_i} \frac{\partial \mathbf{C}}{\partial x_k} + \frac{\partial^2 \mathbf{p}^T(\mathbf{0})}{\partial x_k \partial x_i} \frac{\partial \mathbf{C}}{\partial x_j} \\ &+ \frac{\partial \mathbf{p}^T(\mathbf{0})}{\partial x_i} \frac{\partial^2 \mathbf{C}}{\partial x_k \partial x_j} + \frac{\partial^2 \mathbf{p}^T(\mathbf{0})}{\partial x_k \partial x_j} \frac{\partial \mathbf{C}}{\partial x_i} + \frac{\partial \mathbf{p}^T(\mathbf{0})}{\partial x_j} \frac{\partial^2 \mathbf{C}}{\partial x_k \partial x_i} \\ &+ \frac{\partial \mathbf{p}^T(\mathbf{0})}{\partial x_k} \frac{\partial^2 \mathbf{C}}{\partial x_j \partial x_i} + \mathbf{p}^T(\mathbf{0}) \frac{\partial^3 \mathbf{C}}{\partial x_k \partial x_j \partial x_i}. \end{aligned} \tag{5.49}$$

Examining (5.47), (5.48), and (5.49), the main computational effort and difficulty will not come from any of the derivatives of $\mathbf{p}^T(0)$, but from the first, second, and third derivatives of $\mathbf{C}$. The derivatives of a monomial $\mathbf{p}^T(0)$ are given component-wise as:

$$\frac{\partial^{a+b+c} p(\mathbf{0})}{\partial x^a \partial y^b \partial z^c} = \frac{a * b * c}{r^{a+b+c}}, \qquad 0 \leq a+b+c \leq l-1 \tag{5.50}$$

where if a, b, or c is equal to zero, they are instead equal to 1 in the numerator. Using the definition of $\mathbf{C}$ given by (5.20), and the definition of the derivative of an inverse matrix [180]:

$$\frac{\partial \mathbf{A}^{-1}(\mathbf{x})}{\partial x_i} = -\mathbf{A}^{-1} \frac{\partial \mathbf{A}(\mathbf{x})}{\partial x_i} \mathbf{A}^{-1}, \tag{5.51}$$

the first derivatives of $\mathbf{C}$ are:

$$\begin{aligned}
\frac{\partial \mathbf{C}}{\partial x_i} &= \frac{\partial}{\partial x_i}(\mathbf{M}^{-1}\mathbf{P}\mathbf{W}) \\
&= \frac{\partial}{\partial x_i}(\mathbf{M}^{-1})\mathbf{P}\mathbf{W} + \mathbf{M}^{-1}\mathbf{P}\frac{\partial}{\partial x_i}(\mathbf{W}) \\
&= -\mathbf{M}^{-1}\frac{\partial \mathbf{M}}{\partial x_i}\mathbf{M}^{-1}\mathbf{P}\mathbf{W} + \mathbf{M}^{-1}\mathbf{P}\frac{\partial \mathbf{W}}{\partial x_i} \\
&= -\mathbf{M}^{-1}\mathbf{P}\frac{\partial \mathbf{W}}{\partial x_i}\mathbf{P}^T\mathbf{M}^{-1}\mathbf{P}\mathbf{W} + \mathbf{M}^{-1}\mathbf{P}\frac{\partial \mathbf{W}}{\partial x_i} \\
&= \mathbf{M}^{-1}\mathbf{P}\frac{\partial \mathbf{W}}{\partial x_i}(\mathbf{I} - \mathbf{P}^T\mathbf{C}) \\
\frac{\partial \mathbf{C}}{\partial x_i} &= \mathbf{C}\mathbf{W}^{-1}\frac{\partial \mathbf{W}}{\partial x_i}(\mathbf{I} - \mathbf{P}^T\mathbf{C}),
\end{aligned} \tag{5.52}$$

which is similar to the expression given by [21]. $\frac{\partial \mathbf{W}}{\partial x_i}$ are the derivatives of the weighting function, where the explict form is given in Appendix C. The QR version of the derivative of $\mathbf{C}$ is:

$$\frac{\partial \mathbf{C}}{\partial x_i} = \mathbf{R}^{-1}\mathbf{Q}^T\sqrt{\mathbf{W}^{-1}}\frac{\partial \mathbf{W}}{\partial x_i}\left(\mathbf{I} - \sqrt{\mathbf{W}^{-1}}\mathbf{Q}\mathbf{Q}^T\sqrt{\mathbf{W}}\right). \tag{5.53}$$

With pivoting, the QR derivative is:

$$\frac{\partial \mathbf{C}}{\partial x_i} = \mathbf{P}_v \mathbf{R}^{-1} \mathbf{Q}^T \sqrt{\mathbf{W}^{-1}} \frac{\partial \mathbf{W}}{\partial x_i} \left(\mathbf{I} - \sqrt{\mathbf{W}^{-1}} \mathbf{Q} \mathbf{Q}^T \sqrt{\mathbf{W}} \right). \tag{5.54}$$

A few observations can be made from (5.52),(5.53), and (5.54). Since the differentiation is local, the polynomial basis remains constant through the differentiation due to the global definition of the basis. Therefore, only the weighting kernels are differentiable, and $\mathbf{M}^{-1}$ is only differentiable insofar as it contains $\mathbf{W}$. Computing successive derivatives of $\mathbf{C}$ is both complex and very expensive, due to the additional matrix/matrix and matrix/vector multiplications. The second derivatives of $\mathbf{C}$ are:

$$\begin{aligned} \frac{\partial^2 \mathbf{C}}{\partial x_j \partial x_i} &= \left(\mathbf{C} \mathbf{W}^{-1} \right) \left(\frac{\partial \mathbf{W}}{\partial x_j} \left(\mathbf{I} - \mathbf{P}^T \mathbf{C} \right) \mathbf{W}^{-1} \frac{\partial \mathbf{W}}{\partial x_i} - \frac{\partial \mathbf{W}}{\partial x_j} \mathbf{W}^{-1} \frac{\partial \mathbf{W}}{\partial x_i} \right. \\ &+ \left. \frac{\partial^2 \mathbf{W}}{\partial x_j \partial x_i} - \frac{\partial \mathbf{W}}{\partial x_i} \mathbf{P}^T \mathbf{C} \mathbf{W}^{-1} \frac{\partial \mathbf{W}}{\partial x_j} \right) \left(\mathbf{I} - \mathbf{P}^T \mathbf{C} \right). \end{aligned} \tag{5.55}$$

The QR form for the second derivative, using the definition of $\frac{\partial \mathbf{C}}{\partial x_i}$ given by (5.53) is

$$\begin{aligned} \frac{\partial^2 \mathbf{C}}{\partial x_j \partial x_i} &= \left(\mathbf{R}^{-1} \mathbf{Q}^T \sqrt{\mathbf{W}^{-1}} \right) \left(\frac{\partial \mathbf{W}}{\partial x_j} \left(\mathbf{I} - \sqrt{\mathbf{W}^{-1}} \mathbf{Q} \mathbf{Q}^T \sqrt{\mathbf{W}} \right) \mathbf{W}^{-1} \frac{\partial \mathbf{W}}{\partial x_i} \right. \\ &- \left. \frac{\partial \mathbf{W}}{\partial x_j} \mathbf{W}^{-1} \frac{\partial \mathbf{W}}{\partial x_i} + \frac{\partial^2 \mathbf{W}}{\partial x_j \partial x_i} - \frac{\partial \mathbf{W}}{\partial x_i} \sqrt{\mathbf{W}^{-1}} \mathbf{Q} \mathbf{Q}^T \sqrt{\mathbf{W}^{-1}} \frac{\partial \mathbf{W}}{\partial x_j} \right) \\ &\left(\mathbf{I} - \sqrt{\mathbf{W}^{-1}} \mathbf{Q} \mathbf{Q}^T \sqrt{\mathbf{W}} \right). \end{aligned} \tag{5.56}$$

The third derivatives of $\mathbf{C}$ are:

$$\begin{aligned}
\frac{\partial^3 \mathbf{C}}{\partial x_k \partial x_j \partial x_i} &= \mathbf{A}\left(\frac{\partial^2 \mathbf{B}}{\partial x_k \partial x_j}\mathbf{D}\mathbf{W}^{-1}\frac{\partial \mathbf{W}}{\partial x_i} - \frac{\partial \mathbf{W}}{\partial x_j}\mathbf{D}\mathbf{W}^{-1}\frac{\partial \mathbf{W}}{\partial x_k}\mathbf{W}^{-1}\frac{\partial \mathbf{W}}{\partial x_i}\right. \\
&+ \frac{\partial \mathbf{W}}{\partial x_j}\mathbf{D}\mathbf{W}^{-1}\frac{\partial^2 \mathbf{W}}{\partial x_k \partial x_i} - \frac{\partial \mathbf{W}}{\partial x_j}\mathbf{D}\mathbf{W}^{-1}\frac{\partial \mathbf{W}}{\partial x_i}\mathbf{E}\frac{\partial \mathbf{W}}{\partial x_k} \\
&- \frac{\partial \mathbf{W}}{\partial x_k}\mathbf{D}\mathbf{W}^{-1}\frac{\partial \mathbf{W}}{\partial x_j}\mathbf{W}^{-1}\frac{\partial \mathbf{W}}{\partial x_i} + \frac{\partial \mathbf{W}}{\partial x_k}\mathbf{W}^{-1}\frac{\partial \mathbf{W}}{\partial x_j}\mathbf{W}^{-1}\frac{\partial \mathbf{W}}{\partial x_i} \\
&- \frac{\partial^2 \mathbf{W}}{\partial x_k \partial x_j}\mathbf{W}^{-1}\frac{\partial \mathbf{W}}{\partial x_i} + \frac{\partial \mathbf{W}}{\partial x_j}\mathbf{W}^{-1}\frac{\partial \mathbf{W}}{\partial x_k}\mathbf{W}^{-1}\frac{\partial \mathbf{W}}{\partial x_i} \\
&- \frac{\partial \mathbf{W}}{\partial x_j}\mathbf{W}^{-1}\frac{\partial^2 \mathbf{W}}{\partial x_k \partial x_i} + \frac{\partial \mathbf{W}}{\partial x_j}\mathbf{W}^{-1}\frac{\partial \mathbf{W}}{\partial x_i}\mathbf{E}\frac{\partial \mathbf{W}}{\partial x_k} \\
&+ \frac{\partial \mathbf{W}}{\partial x_k}\mathbf{D}\mathbf{W}^{-1}\frac{\partial^2 \mathbf{W}}{\partial x_j \partial x_i} - \frac{\partial \mathbf{W}}{\partial x_k}\mathbf{W}^{-1}\frac{\partial^2 \mathbf{W}}{\partial x_j \partial x_i} \\
&+ \frac{\partial^3 \mathbf{W}}{\partial x_k \partial x_j \partial x_i} - \frac{\partial^2 \mathbf{W}}{\partial x_j \partial x_i}\mathbf{E}\frac{\partial \mathbf{W}}{\partial x_k} \\
&- \frac{\partial \mathbf{W}}{\partial x_k}\mathbf{D}\mathbf{W}^{-1}\frac{\partial \mathbf{W}}{\partial x_i}\mathbf{E}\frac{\partial \mathbf{W}}{\partial x_j} \\
&+ \frac{\partial \mathbf{W}}{\partial x_k}\mathbf{W}^{-1}\frac{\partial \mathbf{W}}{\partial x_i}\mathbf{E}\frac{\partial \mathbf{W}}{\partial x_j} \\
&- \left.\frac{\partial^2 \mathbf{W}}{\partial x_k \partial x_i}\mathbf{E}\frac{\partial \mathbf{W}}{\partial x_j} + \frac{\partial \mathbf{W}}{\partial x_i}\mathbf{E}\frac{\partial^2 \mathbf{B}}{\partial x_k \partial x_j}\right)\mathbf{D} \qquad (5.57)
\end{aligned}$$

where

$$\mathbf{A} = \mathbf{C}\mathbf{W}^{-1}, \qquad (5.58)$$

$$\mathbf{D} = \mathbf{I} - \mathbf{P}^T\mathbf{C}, \qquad (5.59)$$

$$\mathbf{E} = \mathbf{P}^T\mathbf{A}, \qquad (5.60)$$

$$(5.61)$$

and

$$\frac{\partial^2 \mathbf{B}}{\partial x_j \partial x_i} = \left(\frac{\partial \mathbf{W}}{\partial x_j}\mathbf{D}\mathbf{W}^{-1}\frac{\partial \mathbf{W}}{\partial x_i} - \frac{\partial \mathbf{W}}{\partial x_j}\mathbf{W}^{-1}\frac{\partial \mathbf{W}}{\partial x_i} + \frac{\partial^2 \mathbf{W}}{\partial x_j \partial x_i} - \frac{\partial \mathbf{W}}{\partial x_i}\mathbf{E}\frac{\partial \mathbf{W}}{\partial x_j}\right). \qquad (5.62)$$

The QR version of (5.57) is identical to the standard version, but $\mathbf{A}$, $\mathbf{D}$, and $\mathbf{E}$ are defined as:

$$\mathbf{A} = \mathbf{R}^{-1}\mathbf{Q}^T\sqrt{\mathbf{W}^{-1}}, \tag{5.63}$$

$$\mathbf{D} = \mathbf{I} - \sqrt{\mathbf{W}^{-1}}\mathbf{Q}\mathbf{Q}^T\sqrt{\mathbf{W}}, \tag{5.64}$$

$$\mathbf{E} = \sqrt{\mathbf{W}^{-1}}\mathbf{Q}\mathbf{Q}^T\sqrt{\mathbf{W}^{-1}}. \tag{5.65}$$

A simpler form exists for the second and third derivatives of $\mathbf{C}$ than (5.55) and (5.57). The successive derivatives of $\mathbf{C}$ can instead be built recursively, such that the second and third derivatives of $\mathbf{C}$ are:

$$\frac{\partial^2\mathbf{C}}{\partial x_j \partial x_i} = \mathbf{C}\mathbf{W}^{-1}\frac{\partial^2\mathbf{W}}{\partial x_j \partial x_i}\left(\mathbf{I} - \mathbf{P}^T\mathbf{C}\right) - \mathbf{C}\mathbf{W}^{-1}\left(\frac{\partial\mathbf{W}}{\partial x_j}\mathbf{P}^T\frac{\partial\mathbf{C}}{\partial x_i} + \frac{\partial\mathbf{W}}{\partial x_i}\mathbf{P}^T\frac{\partial\mathbf{C}}{\partial x_j}\right) \tag{5.66}$$

and

$$\begin{aligned}\frac{\partial^3\mathbf{C}}{\partial x_k \partial x_j \partial x_i} &= \mathbf{C}\mathbf{W}^{-1}\frac{\partial^3\mathbf{W}}{\partial x_k \partial x_j \partial x_i}\left(\mathbf{I} - \mathbf{P}^T\mathbf{C}\right) \\ &- \mathbf{C}\mathbf{W}^{-1}\left(\frac{\partial\mathbf{W}}{\partial x_k}\mathbf{P}^T\frac{\partial^2\mathbf{C}}{\partial x_j \partial x_i} + \frac{\partial\mathbf{W}}{\partial x_j}\mathbf{P}^T\frac{\partial^2\mathbf{C}}{\partial x_k \partial x_i} + \frac{\partial\mathbf{W}}{\partial x_i}\mathbf{P}^T\frac{\partial^2\mathbf{C}}{\partial x_k \partial x_j}\right) \\ &- \mathbf{C}\mathbf{W}^{-1}\left(\frac{\partial^2\mathbf{W}}{\partial x_j \partial x_i}\mathbf{P}^T\frac{\partial\mathbf{C}}{\partial x_k} + \frac{\partial^2\mathbf{W}}{\partial x_k \partial x_i}\mathbf{P}^T\frac{\partial\mathbf{C}}{\partial x_j} + \frac{\partial^2\mathbf{W}}{\partial x_k \partial x_j}\mathbf{P}^T\frac{\partial\mathbf{C}}{\partial x_i}\right)\end{aligned} \tag{5.67}$$

This form of the derivatives, while more efficient than (5.55) and (5.57), is still expensive to compute. Two methods for reducing the costs of computing derivatives are discussed in the following two sections.

5.1.5.2 Semi-Diffuse Derivatives

Rather than compute the full form for the second and third derivatives of the shape functions given by (5.48) and (5.49), and by extension the second and third derivatives of $\mathbf{C}$ given by (5.55), (5.56), and (5.57) and (5.66) and (5.67), Chassaing et al. [21] introduces the concept of *semi*-diffuse derivatives of the shape functions. The first derivatives of $\mathbf{C}$ (5.52) and (5.53) are still computed for the semi-diffuse higher-order derivatives of the shape functions, but the higher-order

derivatives of $\mathbf{C}$ are dropped. Chassaing et al. [21] showed that the spatial accuracy does not suffer any appreciable drop when this is done. The second derivatives for the shape functions then become:

$$\frac{\partial^2 \mathbf{\Psi}}{\partial x_j \partial x_i} = \frac{\partial^2 \mathbf{p}^T(\mathbf{0})}{\partial x_j \partial x_i}\mathbf{C} + \frac{\partial \mathbf{p}^T(\mathbf{0})}{\partial x_i}\frac{\partial \mathbf{C}}{\partial x_j} + \frac{\partial \mathbf{p}^T(\mathbf{0})}{\partial x_j}\frac{\partial \mathbf{C}}{\partial x_i}, \tag{5.68}$$

and the third derivatives, which were not included in [21], are:

$$\frac{\partial^3 \mathbf{\Psi}}{\partial x_k \partial x_j \partial x_i} = \frac{\partial^3 \mathbf{p}^T(\mathbf{0})}{\partial x_k \partial x_j \partial x_i}\mathbf{C} + \frac{\partial^2 \mathbf{p}^T(\mathbf{0})}{\partial x_j \partial x_i}\frac{\partial \mathbf{C}}{\partial x_k} + \frac{\partial^2 \mathbf{p}^T(\mathbf{0})}{\partial x_k \partial x_i}\frac{\partial \mathbf{C}}{\partial x_j} + \frac{\partial^2 \mathbf{p}^T(\mathbf{0})}{\partial x_k \partial x_j}\frac{\partial \mathbf{C}}{\partial x_i}\,. \tag{5.69}$$

Another form of the semi-diffuse derivatives, if the cost associated with computing the second derivatives (5.55) or (5.56) is not prohibitive, would be to compute the full form of (5.48) and (5.55) or (5.56) and only drop the third derivatives (5.57) from (5.49).

5.1.5.3 Fully Diffuse Derivatives

If cost of computing any of the derivatives of $\mathbf{C}$ is prohibitive, then the derivatives of the shape function can be approximated using only the derivatives of $\mathbf{p}^T(0)$. The derivatives for the shape functions of first, second, and third order are then computed as:

$$\frac{\partial \mathbf{\Psi}}{\partial x_i} = \frac{\partial \mathbf{p}^T(0)}{\partial x_i}\mathbf{C} \tag{5.70}$$

$$\frac{\partial^2 \mathbf{\Psi}}{\partial x_j \partial x_i} = \frac{\partial^2 \mathbf{p}^T(0)}{\partial x_j \partial x_i}\mathbf{C} \tag{5.71}$$

$$\frac{\partial^3 \mathbf{\Psi}}{\partial x_k \partial x_j \partial x_i} = \frac{\partial^3 \mathbf{p}^T(0)}{\partial x_k \partial x_j \partial x_i}\mathbf{C}. \tag{5.72}$$

Chassaing et al. [21], along with several others, have shown that the diffuse derivatives are comparable and converge to the full derivatives as the mesh is refined. Optionally, one could compute the full first derivatives of the shape function and semi- or fully diffuse derivatives of the higher-order derivatives of the shape functions.

5.1.5.4 *Derivatives of the Affine MLS*

As mentioned in 5.1.3, while the MLS basis functions are not affected by the affine mapping to the unit sphere, the derivatives are. This comes from the derivatives of the weighting function, which are explicitly a function of the coordinate system, *i.e.* (x, y, z). To finish the computation of the derivatives in Cartesian space, the first derivatives are transformed back to Cartesian from the affine unit-sphere space as:

$$\frac{\partial \mathbf{\Psi}}{\partial x_i}_{\text{Cart}} = \mathbf{L}\frac{\partial \mathbf{\Psi}}{\partial x_i}_{\mathbf{S}} \tag{5.73}$$

where $()_{\mathbf{S}}$ denotes the derivatives on the unit sphere. The second derivatives mapped to Cartesian derivatives are:

$$\mathbf{\Psi}_{d,\text{Cart}} = \mathbf{L}\mathbf{\Psi}_{d,\mathbf{S}}\mathbf{L}^{*} \tag{5.74}$$

where $\mathbf{\Psi}_d$ is the Hessian:

$$\mathbf{\Psi}_d = \begin{pmatrix} \frac{\partial^2 \mathbf{\Psi}}{\partial x^2} & \frac{\partial^2 \mathbf{\Psi}}{\partial xy} & \frac{\partial^2 \mathbf{\Psi}}{\partial xz} \\ \frac{\partial^2 \mathbf{\Psi}}{\partial xy} & \frac{\partial^2 \mathbf{\Psi}}{\partial y^2} & \frac{\partial^2 \mathbf{\Psi}}{\partial yz} \\ \frac{\partial^2 \mathbf{\Psi}}{\partial xz} & \frac{\partial^2 \mathbf{\Psi}}{\partial yz} & \frac{\partial^2 \mathbf{\Psi}}{\partial z^2} \end{pmatrix} \tag{5.75}$$

The third derivative mapping does not have a tensor form, but a third derivative in Cartesian coordinates is:

$$\frac{\partial^3 \mathbf{\Psi}}{\partial x_k \partial x_j \partial x_i}_{\text{Cart}} = \mathbf{L}_{i,m}\mathbf{L}_{j,n}\mathbf{L}_{k,o}\frac{\partial^3 \mathbf{\Psi}}{\partial x_o \partial x_n \partial x_m}_{\mathbf{S}} \tag{5.76}$$

The mappings of (5.73), (5.74), and (5.76) are valid for the full (5.47), semi-diffuse (5.69), and fully-diffuse (5.72) derivatives.

5.2 Higher-Order Quadrature

This section describes the quadrature procedure for higher-order methods. As with lower-order methods, the fluxes are computed by cycling over the edges. However, unlike lower-order methods, the quadrature does not take place at the edge midpoint. For higher-order methods, each component dual face of a dual cell must be integrated to correctly compute the fluxes. Since the dual volume faces are quadrilaterals, exact weights and positions of the quadrature are known [136].

The weights and locations for the quadratures on a quadrilateral are given in Appendix E. Since the quadratures are given for a perfect quadrilateral, the dual face must be mapped to isoparametric space such that the quadrature can correctly defined. An example of this is shown in Figure 5.13. If the mesh is highly skewed, the quadrilaterals can be subdivided into triangles, and a similar

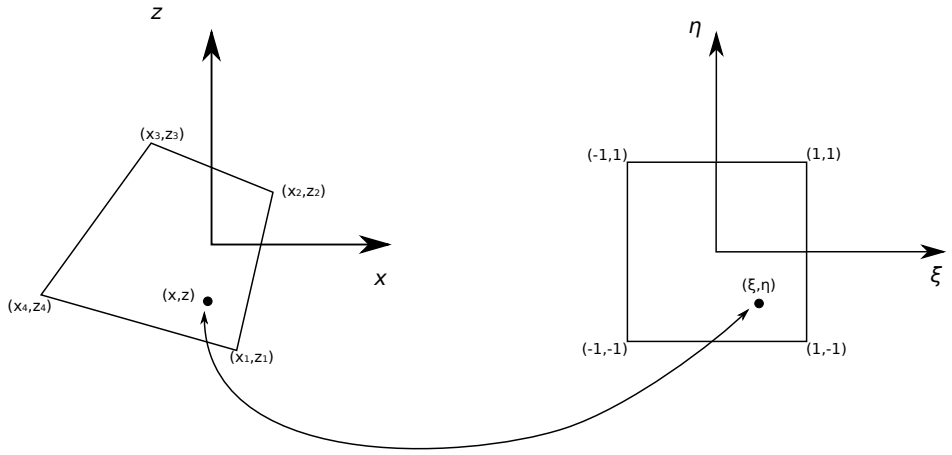


Figure 5.13: Isoparametric Mapping of a Quadrature Point in Parametric Space to a Flux Face [47].

mapping procedure can be utilized to determine the quadrature locations and weights. A more thorough discussion on isoparametric mapping can be found in [107] or [47].

5.3 Higher-Order Fluxes

This section presents higher-order convective and viscous fluxes. To generate higher-order fluxes, reconstructions and quadratures are increased in order. Higher-order quadratures were discussed previously, but reconstructions are discussed here. Fluxes at the boundaries will be discussed in Section 5.4.3 on higher-order boundary condition.

5.3.1 Convective Fluxes

The first step to generate a higher-order convective flux, the reconstructions at a quadrature node on a dual face are determined using higher-order reconstructions. In Section 4.2.1, the linear reconstruction was discussed. For third- and fourth-order accuracy, a quadratic or cubic recon-

struction at the quadrature location is required. Figure 5.14 shows a third-order quadratic reconstruction of $\mathbf{Q}_L$ and $\mathbf{Q}_R$ required for the convective flux. Mathematically, the third-order quadratic

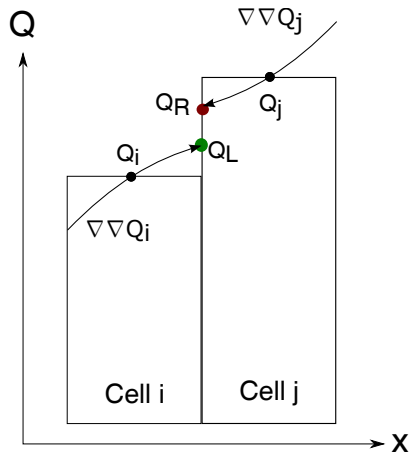


Figure 5.14: Quadratic Reconstruction of Left and Right States for Fluxes.

reconstruction of the state vector at the quadrature location is:

$$\mathbf{Q}_{L,R} = \mathbf{Q}_{i,j} + \mathbf{r}^T_{(i,j)\to q}\nabla\mathbf{Q}_{i,j} + \frac{1}{2}\mathbf{r}^T_{(i,j)\to q}\nabla(\nabla\mathbf{Q}_{i,j})\mathbf{r}_{(i,j)\to q} \tag{5.77}$$

where $\mathbf{r}_{(i,j)\to q}$ represents the distance from either node i or j to the quadrature node. Figure 5.15 shows a fourth-order cubic reconstruction of $\mathbf{Q}_L$ and $\mathbf{Q}_R$. Mathematically, the fourth-order cubic reconstruction of the state vector at the quadrature location is:

$$\mathbf{Q}_{L,R} = \mathbf{Q}_{i,j} + \mathbf{r}^T_{(i,j)\to q}\nabla\mathbf{Q}_{i,j} + \frac{1}{2}\mathbf{r}^T_{(i,j)\to q}\nabla(\nabla\mathbf{Q}_{i,j})\mathbf{r} + \frac{1}{6}(\mathbf{r}^2)^T_{(i,j)\to q}\nabla(\nabla(\nabla\mathbf{Q}_{i,j}))\mathbf{r}_{(i,j)\to q} \tag{5.78}$$

The state vectors computed using (5.77), and (5.78), just as for (4.19), do not guarantee monotonic solutions. As with the second-order case (4.21), limiters are applied to enforce TVD conditions. There is some debate on how limiters should be applied to a higher order reconstruction, and higher-order limiters is still an active area of research. The first method applies separate limiters

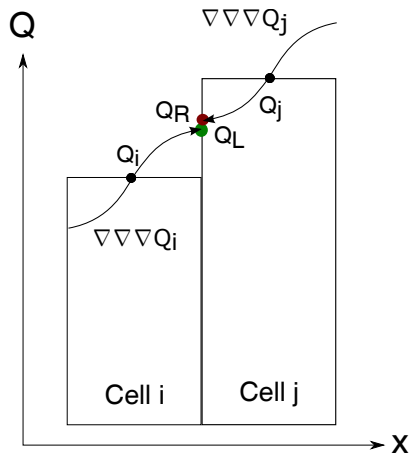


Figure 5.15: Cubic Reconstruction of Left and Right States for Fluxes.

to the first derivatives and higher-order derivatives [116, 122]. This has the general form:

$$\mathbf{Q}_{L,R} = \mathbf{Q}_{i,j} + \Phi_{i,j} \cdot (\mathbf{r}^T_{(i,j)\to q} \nabla \mathbf{Q}_{i,j}) \\ + \Psi_{i,j} \cdot (\frac{1}{2} \mathbf{r}^T_{(i,j)\to q} \nabla(\nabla \mathbf{Q}_{i,j}) \mathbf{r}_{(i,j)\to q} + \frac{1}{6} (\mathbf{r}^2)^T_{(i,j)\to q} \nabla(\nabla(\nabla \mathbf{Q}_{i,j})) \mathbf{r}_{(i,j)\to q} + \ldots) \quad (5.79)$$

where $\Phi_{i,j}$ is the limiter for the first derivatives and $\Psi_{i,j}$ is the limiter, or sometimes referred to as the discontinuity detector, for the higher-order derivatives. This approach can be problematic, as $\Psi_{i,j}$ can over-limit and violate monotonicity. In fact, the higher-order derivatives may contribute to reducing overshooting of the reconstruction, and if limited with $\Psi_{i,j}$, cause $\Phi_{i,j}$ to be insufficient to enforce monotonicity [116].

The second approach, and the one used herein, applies a single limiter to all the derivatives [88]. This looks similar to the second-order limited reconstruction (4.21):

$$\mathbf{Q}_{L,R} = \mathbf{Q}_{i,j} + \Phi_{i,j} \cdot (\mathbf{r}^T_{(i,j)\to q} \nabla \mathbf{Q}_{i,j} + \frac{1}{2} \mathbf{r}^T_{(i,j)\to q} \nabla(\nabla \mathbf{Q}_{i,j}) \mathbf{r}_{(i,j)\to q} \\ + \frac{1}{6} (\mathbf{r}^2)^T_{(i,j)\to q} \nabla(\nabla(\nabla \mathbf{Q}_{i,j})) \mathbf{r}_{(i,j)\to q} + \ldots) \quad (5.80)$$

where $\Phi_{i,j}$ is computed in a analog manner to a more traditional second-order limiter. In this work, the Michalak-Gooch [116] and Venkatakrishan [162] limiters are used.. For (5.77), (5.78),

and (5.80), the limiters are based on the unlimited reconstruction of the state vector of nodes i and j at the quadrature locations.

The second step to compute a higher-order convective flux is to integrate the flux through each dual face attached on an edge. Figure 5.16 shows an edge with its duals and quadrature locations. The total flux computed about an edge (i, j) is

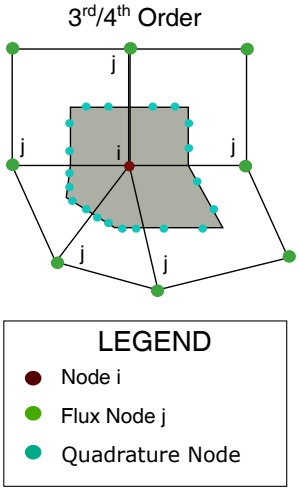


Figure 5.16: Quadrature Node Locations Using the Dual-Median Mesh Paradigm for 3^{rd}- and 4^{th}-Order Flux Computation.

$$\mathbf{F}_{\text{conv},(i,j)} = \sum_{k \in \text{dual}} \left(\sum_{q \in \text{nquad}_{dual}} w_q \mathbf{F}_{\text{conv, q}} \right) |\mathbf{S}|_{\text{dual}} \tag{5.81}$$

where w_q is the weight at the quadrature node for the dual face, as defined explicitly in Appendix E, and $|\mathbf{S}|_{\text{dual}}$ is the dual face area. The edge flux described by (5.87) can also be used in areas of high curvature, where the composite $|\mathbf{S}|_{\text{dual}}$ may not be accurate. Additionally, this method is valid for the convective fluxes in RANS turbulence models.

5.3.2 Viscous Flux

Viscous fluxes, unlike convective fluxes, require the velocities, temperature, and their derivatives at the quadrature nodes q in 5.16. To compute the states and derivatives for higher-order methods, there are three ways. The first method simply averages the state vector and derivative at the quadrature location using (4.23) as discussed previously in Section 4.2.2. This method is the least accurate but potentially the most stable since the fields are not reconstructed via unlimited reconstructions.

The second computes the unlimited reconstruction at the quadrature node from the left and right state for both the state and the derivative. For third-order, the states are reconstructed using (5.77), and the derivatives are reconstructed using

$$\nabla \mathbf{Q}_{L,R} = \nabla \mathbf{Q}_{i,j} + \mathbf{r}^T \nabla (\nabla \mathbf{Q}_{i,j}) \tag{5.82}$$

For fourth-order, the state is reconstructed using (5.78), and the derivatives are reconstructed using:

$$\nabla \mathbf{Q}_{L,R} = \nabla \mathbf{Q}_{i,j} + \mathbf{r}^T \nabla (\nabla \mathbf{Q}_{i,j}) + \frac{1}{2} \mathbf{r}^T \nabla (\nabla (\nabla \mathbf{Q}_{i,j})) \mathbf{r}. \tag{5.83}$$

As discussed in Section 4.2.2, the states and derivatives are still averaged using (4.23). This method could have issues if the unlimited reconstructions over- or under-shoot the bounds in the adjoining cells. It can be shown that even if one side of the average over- or under-shoots while the other does not, the method could over- or under-predict $\mathbf{Q}_{L,R}$ or $\nabla \mathbf{Q}_{L,R}$.

The last method directly determines the states and derivatives at the quadrature nodes q in Figure 5.16. For each quadrature node q, a stencil is built for an MLS reconstruction about the quadrature node q, with q not included explicitly in the stencil. The viscous flux stencil at q is build as a union of the stencils at node i and j such that:

$$\Omega_{\mathbf{x}_q} = \Omega_{\mathbf{x}_i} \cup \Omega_{\mathbf{x}_j} \tag{5.84}$$

An example viscous flux stencil is shown in Figure 5.17. The MLS reconstruction at the quadrature

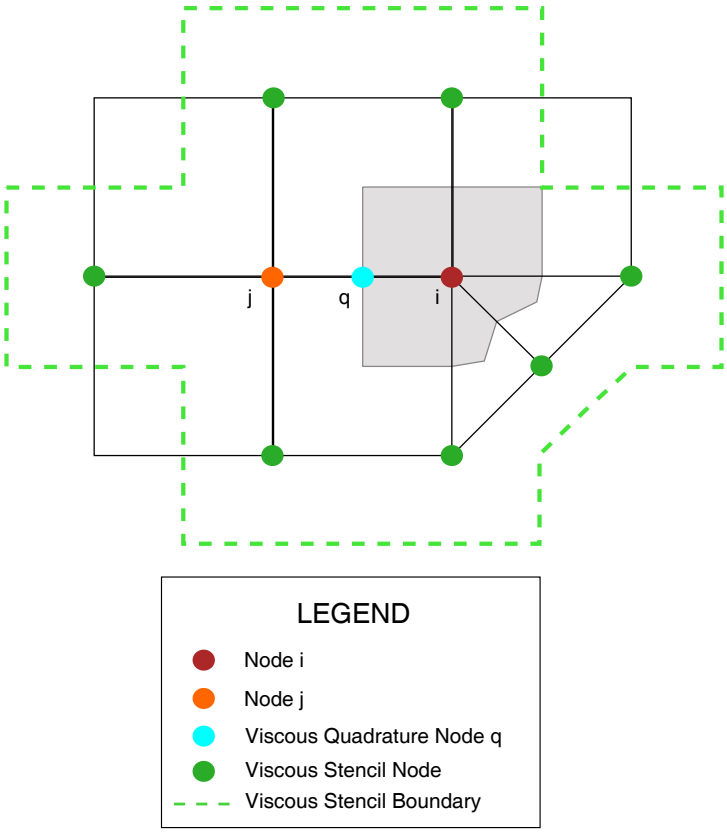


Figure 5.17: Example 2D Viscous Flux Stencil for 2^{nd}-Order.

node of the velocity and temperature using the viscous flux stencil is:

$$\mathbf{v}_q = \sum_{k \in \Omega_{\mathbf{x}_q}} \Psi_k \mathbf{v}_k$$
$$T_q = \sum_{k \in \Omega_{\mathbf{x}_q}} \Psi_k T_k. \tag{5.85}$$

The MLS derivative reconstruction at the quadrature node for the velocity and temperature using

the viscous flux stencil is:

$$\left(\frac{\partial \mathbf{v}}{\partial \mathbf{x}}\right)_q = \sum_{k \in \Omega_{\mathbf{x}_q}} \left(\frac{\partial \Psi}{\partial \mathbf{x}}\right)_k \mathbf{v}_k$$
$$\left(\frac{\partial T}{\partial \mathbf{x}}\right)_q = \sum_{k \in \Omega_{\mathbf{x}_q}} \left(\frac{\partial \Psi}{\partial \mathbf{x}}\right)_k T_k. \tag{5.86}$$

Both (5.85) and (5.86) are then used in (2.5) to compute the viscous flux at the quadrature node. While this method is the most accurate, it is also the most expensive of the three viscous flux methods. This is due to the MLS reconstruction of the states and derivatives at each quadrature location. Additionally, as the order of MLS reconstruction increases, the size of the viscous flux stencil also increases, typically twice the size of a nodal stencil. Limits could be placed on the size of the viscous stencil, but this could potentially reduce the accuracy of (5.85) and (5.86). The total viscous flux for an edge (i, j) is then:

$$\mathbf{F}_{\text{visc},(i,j)} = \sum_{k \in \text{dual}} \left(\sum_{q \in \text{nquad}_{dual}} w_q \mathbf{F}_{\text{visc, q}} \right) \tag{5.87}$$

As with the higher-order convective flux, the higher-order viscous flux can be used with the viscous fluxes in RANS turbulence models.

5.3.3 Leading-Edge Flux Modification

When solving flows with wall boundaries interfaced with symmetry-type boundaries, a 'pressure' rise will occur at the leading-edge of the wall. For viscous flows, this would be the expected situation. For inviscid flows, this pressure rise should not happen, since the boundaries are both effectively slip-walls. However, ghost node-based boundary conditions will cause a pressure rise. Figure 5.18 shows the pressure rise when using ghost nodes.

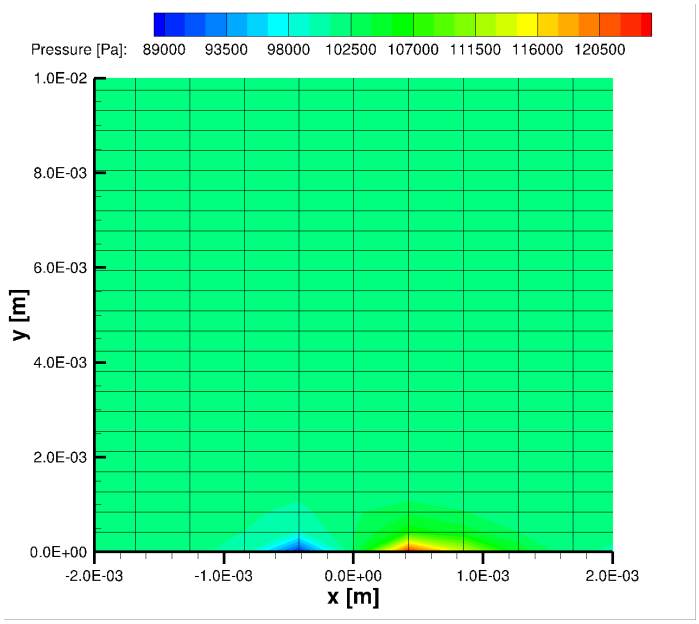


Figure 5.18: Pressure Rise at a Leading-Edge ($x = 0$) Due to the Flow 'Seeing' a Corner when Using Ghost Nodes.

This is due to the control volumes at the leading edge. The symmetry boundary node will have a full control volume, while the wall boundary will only have the interior control volume, which means that the flux at the control volume face will be twice as large as it should be. A modification to the flux, both convective and diffusive, must be made for the nodes along the leading-edge. The leading-edge flux should be removed below the wall/symmetry plane for the wall node. Instead of recomputing the flux and removing it directly, the normal convective flux is computed. However, when summing the contributions for the left and right state, the quadrature weight is set to zero for the wall contribution. Effectively, the flux face has left and right weights that can be turned on or off depending on the situation. The convective flux now reads:

$$\mathbf{F}_{\text{conv},(i,j)} = \sum_{k \in \text{dual}} \left(\sum_{q \in \text{nquad}_{dual}} w_{q,(i,j)} \mathbf{F}_{\text{conv, q}} \right) |\mathbf{S}|_{\text{dual}} \tag{5.88}$$

where $w_{q,(i,j)}$ is the quadrature weight from the left or right. Figure 5.19 illustrates this.

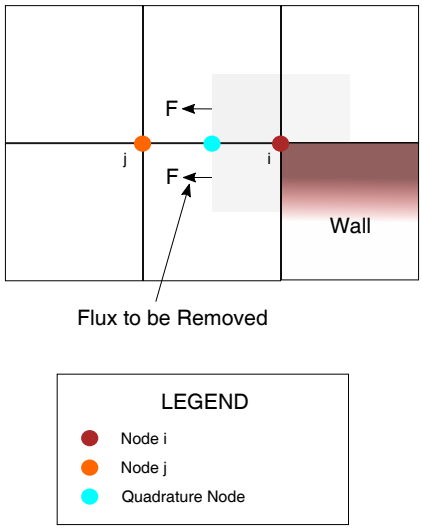


Figure 5.19: Illustration of Removing Flux from Ghost Region at a Leading-Edge.

The effect of the leading-edge flux modification is shown in Figure 5.20.

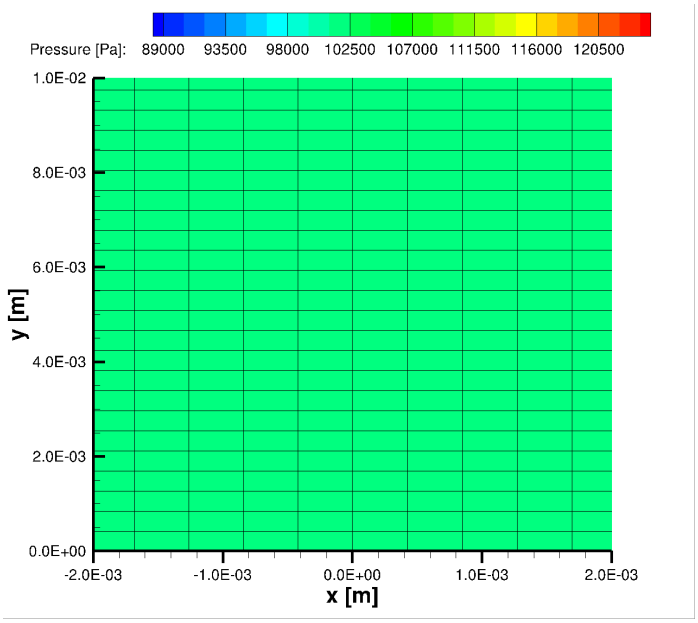


Figure 5.20: Effect of Correction of the Leading-Edge ($x = 0$) Fluxes.

It should be noted that this type of boundary (symmetry-wall) causes instabilities in the flow, so the usefulness of this correction is probably limited. Meshes with this boundary interface should best be avoided to avoid these issues when possible.

5.4 Higher-Order Boundary Conditions

This section presents the higher-order boundary conditions required for implementing a higher-order flow solver. A general overview of higher-order boundary conditions will be presented, followed by specific implementation details. Additionally, boundary fluxes will be discussed at the end of the section.

5.4.1 Periodic Boundary Conditions

Periodic boundary conditions are used to reduce the size of the computational domain. To facilitate the reduction, information must be transferred from one side (or sides) of the reduced domain to the other. This information transfer is handled through the use of boundary faces or directly with the nodes. With periodic nodes, two- and three-dimensional periodic boundaries are possible, which is necessary for many DNS cases and for LES benchmarking. Additionally, periodic nodes allow for a smoother implementation of continuous derivatives across the boundaries. Mathematically, the conditions at periodic nodes are:

$$\begin{aligned} \mathbf{q}_{\mathrm{s}} &= \mathbf{q}_{\mathrm{m}} \\ \frac{\partial^{a+b+c}\mathbf{q}_{\mathrm{s}}}{\partial x^a y^b z^c} &= \frac{\partial^{a+b+c}\mathbf{q}_{\mathrm{m}}}{\partial x^a y^b z^c} \qquad 0 \leq a+b+c \leq l \end{aligned} \tag{5.89}$$

where ℓ is the maximum order of the derivatives. Figure 5.21 shows a second-order periodic boundary. Cell volumes, Ω, and edge-based areas, $\mathbf{S}$, for a periodic node are computed at the master and slave nodes normally. Figure 5.22 shows how a slave cell volume is added to a master volume. Since volumes are direction-invariant, the partial dual-volumes obtained in the slave flux region are directly added to the master dual-volumes to obtain the full dual-volume on the master periodic boundary. The slave dual-volumes are then set equal to the master dual-volumes. For a

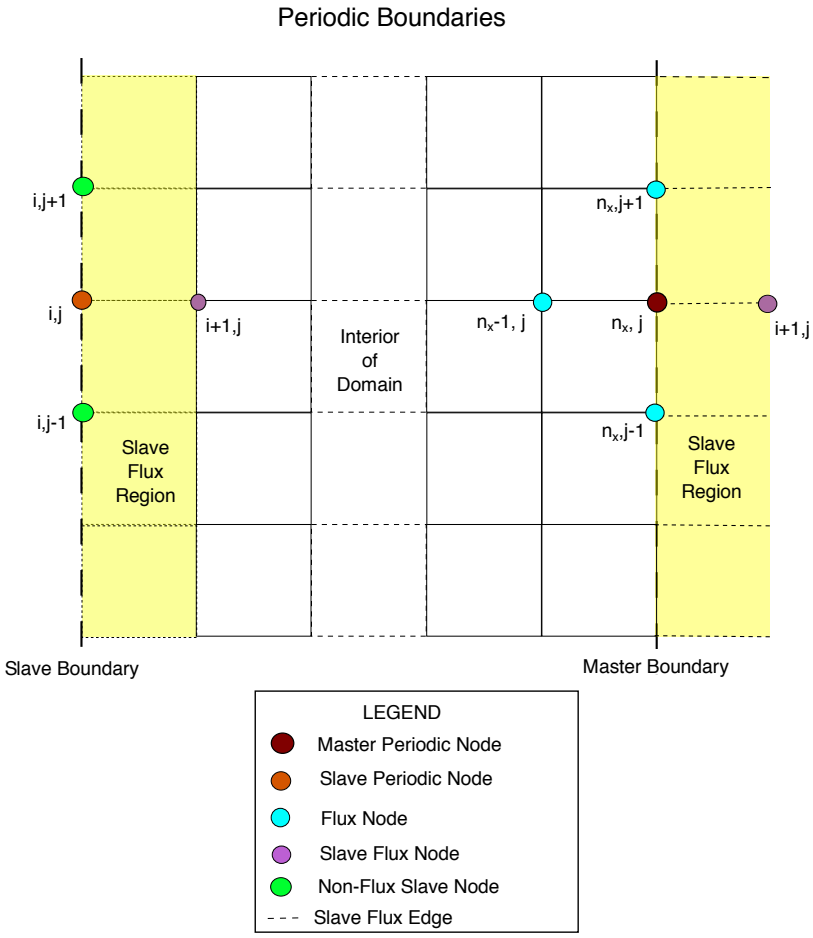


Figure 5.21: Illustration of a Linear 2^{nd}-Order Periodic Boundary.

discussion on dual-volumes, refer to Section 4.1.

Edge-based areas, unlike dual-volumes, are not direction-invariant, and therefore care must be taken to compute the periodic edge-based areas needed for the fluxes. In general, edges on the master and slave boundaries will not be oriented in the same direction. Figure 5.23 shows how the edge-based areas are added. If the dot product of the edge-based areas of the master and slave edges is less than zero, the slave edge-based areas are rotated 180°.

$$\mathbf{R}_{s\to m} = \begin{cases} \mathbf{R}(180^\circ) & \text{if} \quad \mathbf{s}_\text{m} \cdot \mathbf{s}_\text{s} < 0 \\ \\ \mathbf{R}(0^\circ) & \text{if} \quad \mathbf{s}_\text{m} \cdot \mathbf{s}_\text{s} > 0 \end{cases} \tag{5.90}$$

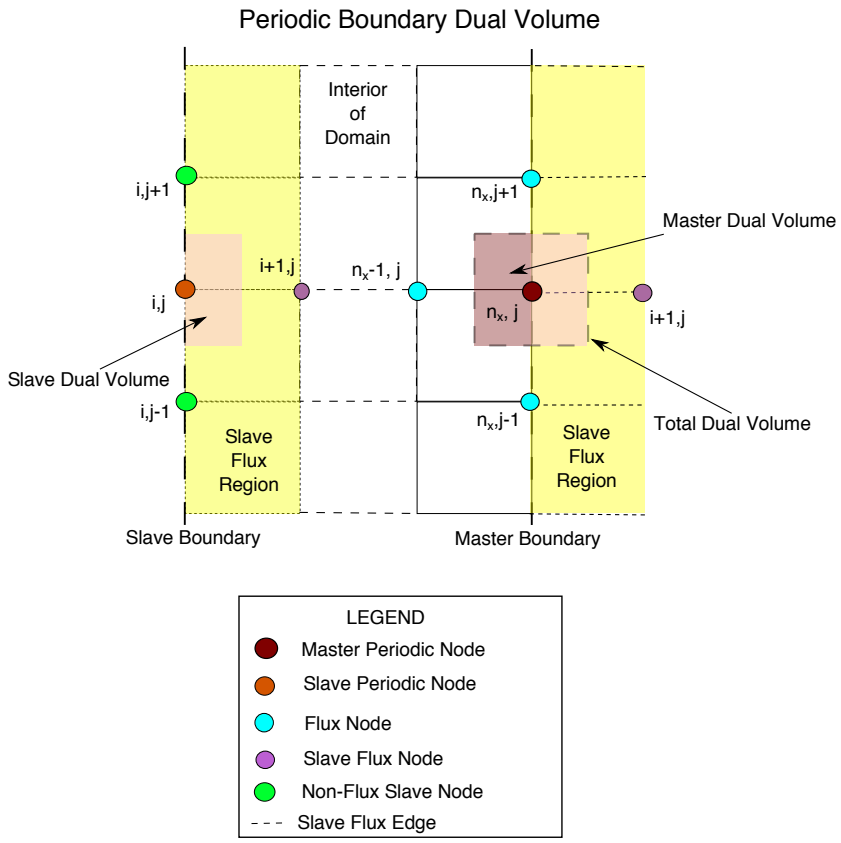


Figure 5.22: Illustration of Corrected Control Volumes for Periodic Boundaries.

Adding the edge-based areas together is then:

$$\mathbf{s}_{\mathrm{m}} = \mathbf{s}_{\mathrm{m}} + \mathbf{R}_{s \to m}\mathbf{s}_{\mathrm{s}} \tag{5.91}$$

$$\mathbf{s}_{\mathrm{s}} = \mathbf{s}_{\mathrm{m}}$$

To compute derivatives at the master mode (n_x, j) in Figure 5.21, the stencil used in the MLS approximation 5.16 would also include node $(i+1, j)$. Note that nodes $(i, j+1)$ and $(i, j-1)$ would not be included in the stencil, since they are slave nodes of nodes already in the master node stencil. The distance from node $(i+1, j)$ to the master is computed as the distance from $(i+1, j)$ to the slave node, since the relative location is the same. Once the stencil is determined for a periodic boundary node, the steps outlined for determining a derivative about a non-periodic

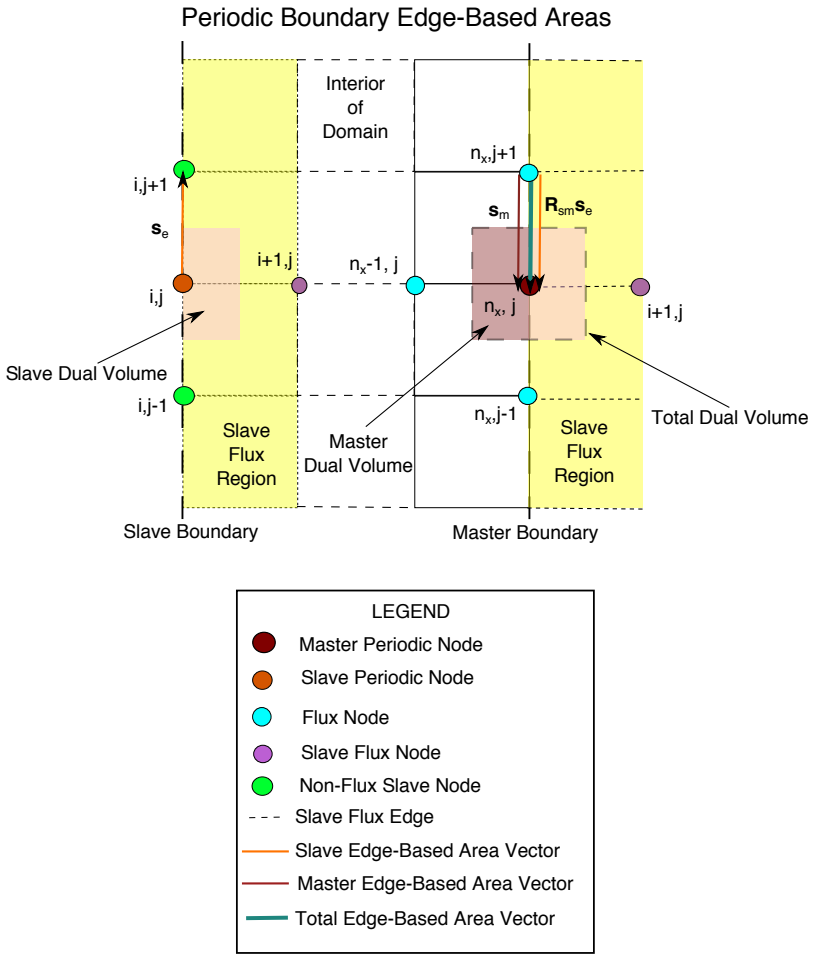


Figure 5.23: Illustration of Corrected Edge-Based Areas for Periodic Boundaries.

node in Section 5.1.5 are used. Fluxes are computed along active edges according to (4.6) and (2.5), where an active edge is an edge with either no slave nodes or only one slave node attached to it. In Figure 5.21, the non-active edges would be the slave edges; that means that no flux is computed between the slave (i, j) and nodes $(i, j + 1)$ and $(i, j - 1)$. For periodic boundaries, the flux between the slave (i, j) and node $(i - 1, j)$ is added to the sum of the fluxes (4.22) to finish updating the master node's residual.

$$\mathbf{R}_{\mathrm{m}} = \mathbf{R}_{\mathrm{m}} + \mathbf{R}_{\mathrm{s}} \tag{5.92}$$

At the end of the time step or sub-iteration, the state vector determined at the master node is

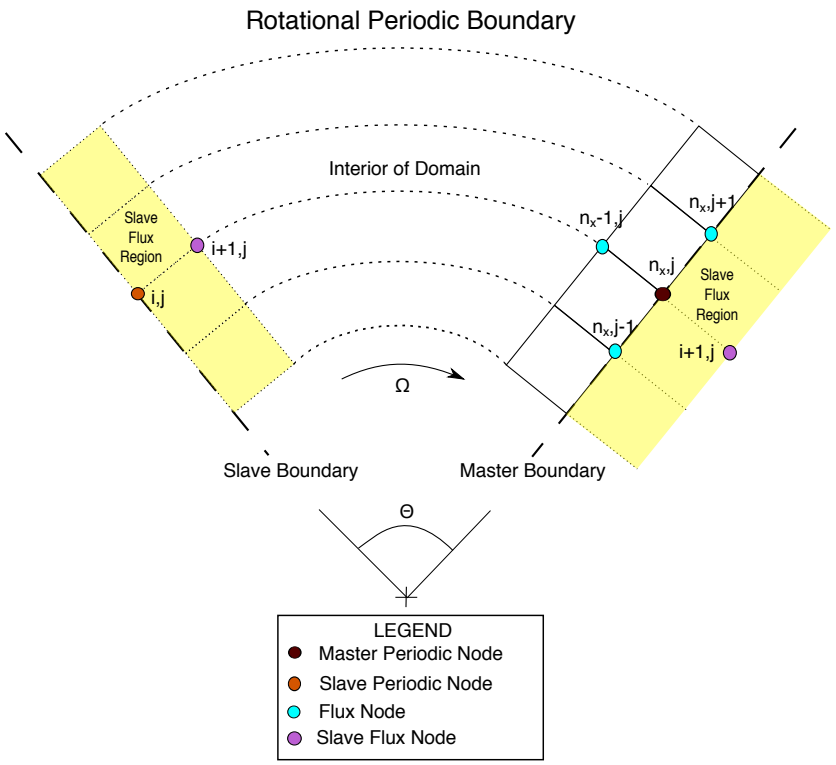


Figure 5.24: Illustration of a 2^{nd}-Order Rotational Periodic Boundary.

transferred to the slave node.

Rotational periodic boundaries are a special case of periodic boundaries. Figure 5.24 shows an example of a rotational periodic boundary. The conditions of (5.89) are modified to include the rotation matrix for velocities on the master and slave boundaries:

$$\mathbf{v}_{\text{s}} = \mathbf{R}_{\theta}\mathbf{v}_{\text{m}} \tag{5.93}$$

where $\mathbf{R}_{\theta}$ is the rotation matrix defined by the angle θ between the master and slave periodic boundaries. As for periodic boundaries, the orientation of edge-based areas of the master and slave edges must be made compatible. The rotational matrix $\mathbf{R}_{\theta}$ is used to properly orient the slave

edges with the master edges, such that $\mathbf{R}_{s\to m}$ in (5.91) becomes:

$$\mathbf{R}_{s\to m} = \begin{cases} \mathbf{R}_\theta \mathbf{R}(180^\circ) & \text{if} \quad \mathbf{s}_\text{m} \cdot \mathbf{s}_\text{s} < 0 \\ \\ \mathbf{R}_\theta \mathbf{R}(0^\circ) & \text{if} \quad \mathbf{s}_\text{m} \cdot \mathbf{s}_\text{s} > 0 \end{cases} \tag{5.94}$$

For derivatives, the relative distance between node $(i+1, j)$ and the slave node in Figure 5.24 is rotated using $\mathbf{R}_\theta$ to give the correct distance between $(i+1, j)$ and the master node. Derivatives are rotated to ensure continuity across the boundary:

$$\frac{\partial^{a+b+c}\mathbf{q}_\text{s}}{\partial x^a y^b z^c} = \mathbf{R}_\theta \frac{\partial^{a+b+c}\mathbf{q}_\text{m}}{\partial x^a y^b z^c} \qquad 0 \leq a+b+c \leq l \tag{5.95}$$

The rotation is performed on the coordinates specifically, not the thermodynamic variables. Derivatives of velocities require a double rotation, a result of (5.93).

$$\frac{\partial^{a+b+c}\mathbf{v}_\text{s}}{\partial x^a y^b z^c} = \mathbf{R}_\theta \mathbf{R}_\theta \frac{\partial^{a+b+c}\mathbf{v}_\text{m}}{\partial x^a y^b z^c} \qquad 0 \leq a+b+c \leq l \tag{5.96}$$

Finally, after the fluxes are computed along each active edge, the residuals of the slave node are rotated and added to the master.

$$\mathbf{R}_\text{m} = \mathbf{R}_\text{m} + \mathbf{R}_\theta \mathbf{R}_\text{s} \tag{5.97}$$

Updated state quantities at the master are transferred using $\mathbf{R}_\theta$.

An important issue is raised when using periodic boundaries for higher-order methods. Typically, the master and slave are assumed to be co-located if the requisite translation is performed. Referring to Figure 5.7, it is not just sufficient for nodes (i, j) and $(i+6, j)$ to be periodic, but also $(i+1, j)$ with $(i+5, j)$, $(i+2, j)$ with $(i+4, j)$, and all the remaining points to the right of (i, j) with the remaining right points of $(i+6, j)$. In effect, all the geometric and flow quantities $n-1$-orders of magnitude away from the master periodic boundaries *must* be periodic with $n-1$-orders of magnitude. This condition could also allow for less computation since more points are effectively fixed within the domain. However, practically speaking, only the outer-most slave flow

vector is not directly computed since the logic involved to limit computation would be significant. It is sufficient, though, to require only the geometry to be explicitly periodic for the higher-order slaves, and allow the flow solver to take care implicitly to ensure the periodicity of the higher-order slave flow variables.

It should be noted that results using the improved periodic boundaries have been published in Freno et al. [53], Freno et al. [54] and Krath et al. [93].

5.4.2 Non-Periodic Boundaries

Non-periodic boundaries present substantial difficulty in implementation compared to periodic boundaries for higher-order methods. Whereas for periodic boundaries the requirement is only to properly attach the slave information to the master, non-periodic boundaries require determination of the stencils and each corresponding state vector. From Section 5.1.4, stencils near boundaries are either augmented or utilize ghost nodes. Therefore, the boundary conditions may be enforced two ways: with augmented stencils or ghost nodes. Using augmented stencils, boundary conditions can only be enforced in a manner similar to the method used by Kim [91] and Gargoloff [57]. To increase the order, the quadrature on the boundary flux would require higher accuracy in addition to a higher-order interpolation/extrapolation to determine the state vector at the quadrature nodes, which would in turn require higher-order derivatives. Using ghost nodes, the stencils are similar to their interior counterparts, but the ghost nodes only need to be defined.

Ghost nodes are used to enforce the boundary conditions. The ghost nodes are defined such that the flux integration at the boundary nodes using (4.6) and (2.5) enforces the required conditions for each boundary. Fluxes are computed between nodes and ghost nodes along ghost flux edges. Ghost flux edges are edges containing a boundary node and a ghost node. In general, both (4.6) and (2.5) are computed without adding extra conditions.

The difficulty in using ghost nodes comes from defining the ghost node state. One could define the flow variables at a ghost node similar to the method given in [12, Chapter 8, pg. 272-273]. However, this assumes that the ghost and interior nodes are normal to the boundary [12, Chapter 8]. This may not always be true. When the ghost grid is not orthogonal to the boundary, the grid

deficiencies must be taken into account in order to accurately reconstruct the flow variables at the ghost nodes. Finite element-type approaches would be the logical choice to define ghost nodes, where shape functions constructed at the boundary could be leveraged, with the necessary conditions applied, to determine what the ghost node states should be. However, this is also problematic since interpolation error could occur and degrade the solution. Additionally, this method may not account for the boundary, as in the case of walls. To circumvent both issues discussed above, image and intercept nodes are introduced to aid in setting the boundary conditions. The ghost nodes are required to be orthogonal to the boundary, though the interior may not always be.

5.4.2.1 Generation of Image and Intercept Nodes

The concept of image and intercept nodes is taken from immersed boundary methods [118]. In immersed boundary methods, the flow variables at the ghost nodes are set using these nodes, where the ghost nodes are defined as any node lying within the immersed boundary surface. What is presented within is a similar situation; however, the current case is simpler as the boundary is explicitly known as well as the ghost nodes.

To generate the image and intercept nodes, Algorithm 5.2 is used, with reference to Figure 5.25.

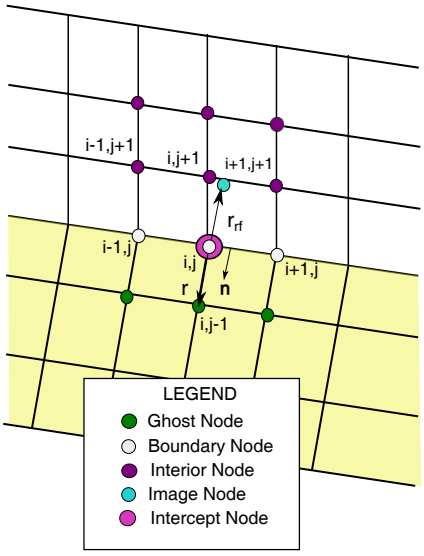


Figure 5.25: Generation of Image and Intercept Nodes for Ghost Node $(i, j-1)$.

Algorithm 5.2 Image and Intercept Node Generation

1: Take the ghost node, its corresponding main mesh boundary node, and the boundary surface normal at the boundary node. These nodes are nodes $(i, j-1)$ and (i, j) in Figure 5.25.

2: Using the *inverse* of the affine transformation outlined in 5.1.4.2.2, the ghost node is reflected into the interior of the domain, such that

$$\begin{Bmatrix} x_{im} - x_{i,j} \\ y_{im} - y_{i,j} \\ z_{im} - z_{i,j} \\ 1 \end{Bmatrix} = \mathbf{A} \begin{Bmatrix} x_g - x_{i,j} \\ y_g - y_{i,j} \\ z_g - z_{i,j} \\ 1 \end{Bmatrix}. \tag{5.98}$$

where $\mathbf{A}$ is the affine matrix, and (x_{im}, y_{im}, z_{im}) are the coordinates of the image node. In Figure 5.25,

$$\mathbf{r}_{rf} = \begin{Bmatrix} x_{im} - x_{i,j} \\ y_{im} - y_{i,j} \\ z_{im} - z_{i,j} \end{Bmatrix} \tag{5.99}$$

and

$$\mathbf{r} = \begin{Bmatrix} x_g - x_{i,j} \\ y_g - y_{i,j} \\ z_g - z_{i,j} \end{Bmatrix} \tag{5.100}$$

The stencils for the image and intercept nodes are presented in Figure 5.26. The stencils built for these nodes are based on (b) of State 5 in Algorithm 5.2. For the image node stencils, the stencils do not include any ghost nodes to avoid coupling. At boundaries where a Neumann condition is required, the intercept stencil can be trimmed to only include nodes lying along the boundary nodal normal. These normal intercept stencils are shown in Figure 5.27.

Algorithm 5.2 Image and Intercept Node Generation, Continued

3: Find the image node within the main mesh. This is handled with an octree partitioning [145] of the main mesh. This search must performed since the location of the image node is not explicitly known. It is possible to have the image node located:

   (**a**) Co-located with a main mesh node.

   (**b**) Located on a face in the main mesh.

   (**c**) Located within a cell of the main mesh.

   (**d**) Any of the above three options, but within the ghost mesh.

   The octree is discussed in Appendix F.

4: Once the location of the image node is known, the intercept node is taken as the average of the image and ghost nodes. This is generally sufficient, but additional refinement of the intercept location could be made for a curved surface, if desired.

5: Stencils for both the image and intercept nodes are generated. For the image node, the stencil is generated by:

   (a) The matching main mesh node. This effectively sets the image node quantities exactly equal to those of the interior mesh node. This only occurs if the image node is co-located with a main mesh node.

   (b) The stencil of the closest node on the face or within the cell of the main mesh. The stencil taken from the closest node is adjusted such that it is centered on the image node.

   (c) A union of the stencils on the face or within the cell that the image is located on/in. Duplicate nodes are removed, and the stencil is adjusted to be centered on the image node.

   For the intercept node, the stencil is generated in the same way as the image node, except the stencil is directly copied from the matching main mesh node. This is because the derivative is required at the intercept location. Additionally, if the ghost node and image node do not appear in the copied list, they are added to the intercept node stencil. If the image node is not explicitly a node in the main mesh, then the image node stencil nodes are added to the intercept node stencil.

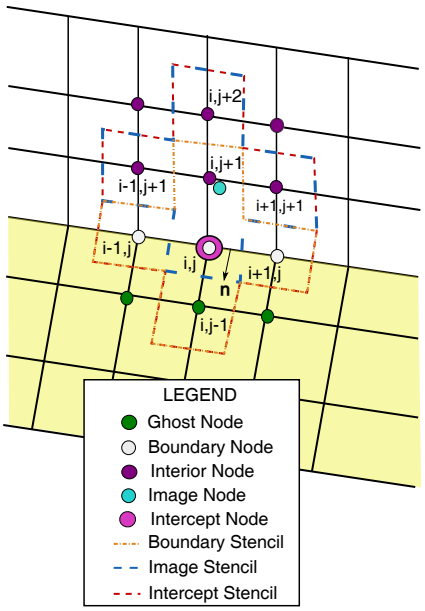


Figure 5.26: Illustration of Image and Intercept Nodes and Stencils Related to Ghost Node $(i, j-1)$.

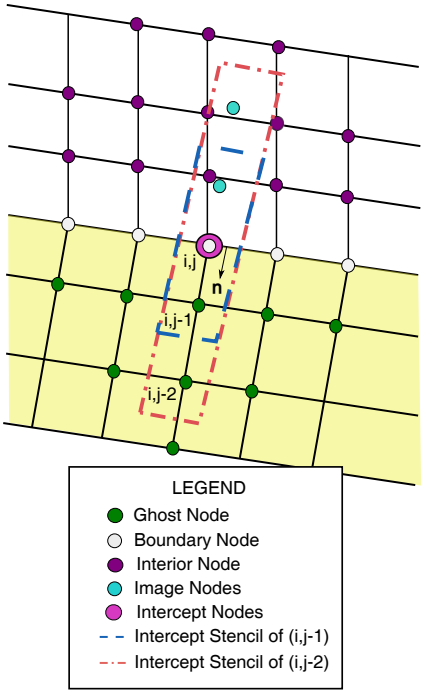


Figure 5.27: Illustration of Normal Intercept Stencils Related to Ghost Nodes $(i, j-1)$ and $(i, j-2)$.

5.4.2.2 *Building the Boundary System with Ghost, Image, & Intercept Nodes*

Once the image, intercept, and ghost node stencils are built, the boundary system used to solve for the boundary conditions can be built. In this work, the Moving-Least Square basis functions (5.14) are used to determine the values of the flow variables at the image and intercept nodes. The flow variables at the ghost nodes are then determined using a combination of finite-differences and the MLS basis functions (5.14). An advantage of using (5.14) is that the ghost nodes are explicitly included in the stencil. This is important for building a system of equations that can be used to determine ghost node states. The linear system of equations for the ghost nodes takes the form:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{5.101}$$

where $\mathbf{A}$ is a $N_g \times N_g \times N_{eq}$ sparse matrix, $\mathbf{b}$ is a $N_g \times N_{eq}$ vector, and $\mathbf{x}$ is a $N_g \times N_{eq}$ vector containing the ghost nodes. N_g is the number of ghost nodes and N_{eq} is the number of equations. For the purposes of the discussion, boundary conditions and derivation of values for the ghost nodes will be based on the node indexes in Figure 5.28.

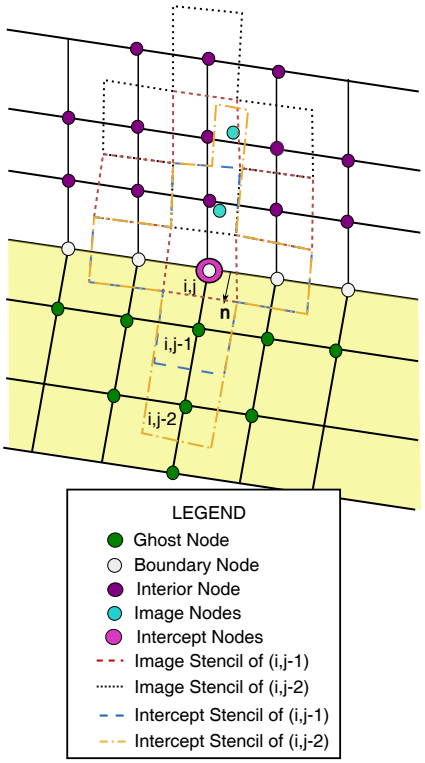


Figure 5.28: Illustration of a General Boundary with Ghost, Image, and Intercept Nodes Used to Derive Boundary Conditions. Stencils are Outlined to Highlight Relationships Between Each Stencil.

The system is built using equations relating the image and intercept nodes to the ghost nodes. Four options are available to relate the ghost nodes to corresponding image and intercept nodes: Dirichlet, Neumann, Robin, or reflection conditions. Extrapolation is also possible, but not used herein.

The Dirichlet condition are set by equating the value at the ghost node to a desired value:

$$q_b = \hat{q}_b \tag{5.102}$$

where $\hat{q}$ is the desired value for the boundary node. Using (5.16) as a guide, the Dirichlet boundary condition is defined as:

$$\hat{q}_b = \sum_{k \in \Omega_{\mathbf{x}_I}} \Psi_k q_k. \tag{5.103}$$

Equation (5.103) is manipulated to isolate the ghost nodes:

$$\sum_{\substack{k\in\Omega_{\mathbf{x}_I}\\ k=\text{ghost}}} \Psi_k q_k = \hat{q}_b - \sum_{\substack{k\in\Omega_{\mathbf{x}_I}\\ k\neq\text{ghost}}} \Psi_k q_k \tag{5.104}$$

Where applicable, this system is a boundary node and put into the general system (5.101).

Neumann boundary conditions enforce a gradient condition on the boundary. This is expressed as:

$$\nabla q_{\text{int}} \cdot \mathbf{n}_{i,j} = \widehat{\left(\frac{\partial q}{\partial \mathbf{n}}\right)}_{\text{int}}$$

where $\widehat{\left(\frac{\partial q}{\partial \mathbf{n}}\right)}_{\text{int}}$ is the desired value for the normal gradient on the boundary at the intercept node, and $\mathbf{n}$ is the boundary normal at the boundary node. An example of a Neumann boundary is the pressure gradient on a viscous wall, where the normal pressure gradient is zero. In the MLS framework, the Neumann boundary condition is set using (5.46):

$$\left(\frac{\partial \hat{q}_{\text{int}}}{\partial x}, \frac{\partial \hat{q}_{\text{int}}}{\partial y}, \frac{\partial \hat{q}_{\text{int}}}{\partial z}\right) \cdot \mathbf{n}_{i,j} = \widehat{\left(\frac{\partial q}{\partial \mathbf{n}}\right)}_{\text{int}} = \sum_{k\in\Omega,\text{int}} \left(\frac{\partial \Psi_k}{\partial x}, \frac{\partial \Psi_k}{\partial y}, \frac{\partial \Psi_k}{\partial z}\right) \cdot \mathbf{n}_{i,j}\, q_k \tag{5.105}$$

where $\mathbf{n}_{i,j}$ is the normal of the boundary at node (i, j). Isolating for ghost nodes, Equation (5.105) is written as:

$$\sum_{\substack{k\in\Omega_{\mathbf{x}_I}\\ k=\text{ghost}}} \left(\frac{\partial \Psi}{\partial \mathbf{n}}\right)_k q_k = \widehat{\left(\frac{\partial q}{\partial \mathbf{n}}\right)}_{i,j} - \sum_{\substack{k\in\Omega_{\mathbf{x}_I}\\ k\neq\text{ghost}}} \left(\frac{\partial \Psi}{\partial \mathbf{n}}\right)_k q_k \tag{5.106}$$

If the target normal $\widehat{\left(\frac{\partial q}{\partial \mathbf{n}}\right)}_{i,j}$ is non-zero, then the Neumann condition is effectively an extrapolation condition. The Neumann-extrapolation condition assumes that the flow is continuous at the boundary of interest. Where applicable, (5.106) is written for each node and used to populate (5.101).

The previous description of the Neumann condition (5.106) can have deficiencies leading to unacceptable errors. The intercept stencil includes non-normal nodes which can be overly weighted. Also, from numerical experiments, the anisotropic weighted-MLS, while producing

accurate derivatives, will not highly weight unnecessary points for the specified order. For example, using Figure 5.29 as a guide, a second-order MLS reconstruction using anisotropic weights will only weight $(i, j+1)$ and $(i, j-1)$ for $\frac{\partial \mathbf{\Psi}}{\partial y}$. Nodes $(i, j+2)$ and $(i, j-2)$ have very small weights for $\frac{\partial \mathbf{\Psi}}{\partial y}$, causing numerical issues in (5.101).

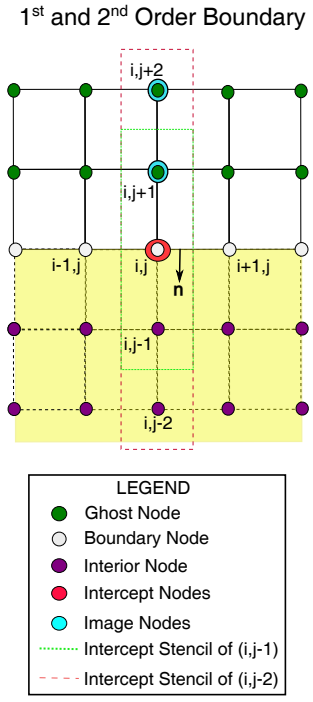


Figure 5.29: Illustration of Normal Intercept Stencils Related to Ghost Nodes $(i, j-1)$ and $(i, j-2)$.

To overcome this, the boundary normal nodes in the intercept stencil are isolated to build a stencil purely normal to the boundary, as shown in Figure 5.28. Once the normal intercept stencil is built, which is a straightforward process, the stencil is used to build finite differences. The choice in computing the weights for the finite difference is the Fornberg algorithm [48, 50], which allows for unequal spacing in the stencil. However, the intercept stencil is purposefully build to have equal spacing to avoid issues with large unphysical weights for the center node. Once the weights are

determined, the Neumann system can be built:

$$\left(\widehat{\frac{\partial q}{\partial \mathbf{n}}}\right)_{\text{int}} = \sum_{k\in\Omega,\text{int, Fornberg}} \frac{\partial \psi_k}{\partial \mathbf{n}}, q_k \tag{5.107}$$

where ψ is the Fornberg finite difference weight.

The next step is to determine how the boundary stencils are to be coupled for the Neumann condition, or how are the ghost node values are prescribed using sets of finite differences. To simplify the analysis, the points are assumed to be equally spaced, since it eliminates the spacing h from the equations. For a first- or second-order problem, the stencils are oriented as shown in Figure 5.29. The possible finite difference equations, with h eliminated, are:

$$\begin{aligned}
q_{i,j+1} - q_{i,j-1} &= 0 \\
q_{i,j+2} - q_{i,j-2} &= 0 \\
q_{i,j+1} - q_{i,j-2} &= 0 \\
q_{i,j+2} - q_{i,j-1} &= 0 \\
-q_{i,j+2} + 8q_{i,j+1} - 8q_{i,j-1} + q_{i,j-2} &= 0 \\
2q_{i,j+1} + 3q_{i,j} - 6q_{i,j-1} + q_{i,j-2} &= 0 \\
-q_{i,j+2} + 6q_{i,j+1} - 3q_{i,j} - 2q_{i,j-1} &= 0 \\
q_{i,j+2} + 12q_{i,j} - 16q_{i,j-1} + 3q_{i,j-2} &= 0 \\
-3q_{i,j+2} + 16q_{i,j+1} - 12q_{i,j} - q_{i,j-2} &= 0.
\end{aligned}$$

A least-squares approach could be used with these stencils, such that the system would be:

$$\mathbf{A}\mathbf{q}_g = \mathbf{B}\mathbf{q}_m \tag{5.108}$$

where

$$\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 8 & -1 \\ 6 & -1 \\ 2 & 0 \\ 16 & -3 \\ 0 & 1 \end{pmatrix} \quad , \mathbf{B} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 8 & -1 \\ 3 & 2 & 0 \\ -3 & 6 & -1 \\ 0 & 12 & 1 \\ -12 & 16 & -3 \end{pmatrix} . \tag{5.109}$$

The vectors $\mathbf{q}_g$ and $\mathbf{q}_m$ are the state vectors for the ghost and main mesh nodes, respectively. If

$$\mathbf{q}_m = \begin{pmatrix} q_{i,j} \\ q_{i,j+1} \\ q_{i,j+2} \end{pmatrix} = \begin{pmatrix} 0.714 \\ 0.714 \\ 0.714 \end{pmatrix} \tag{5.110}$$

or, a uniform non-dimensional pressure field for sea-level pressure, and (5.108) is solved in a least-squares sense for the pressure at the ghost nodes, one will correctly determine the ghost node pressure to be:

$$\mathbf{q}_g = \begin{pmatrix} q_{i,j-1} \\ q_{i,j-2} \end{pmatrix} = \begin{pmatrix} 0.714 \\ 0.714 \end{pmatrix} . \tag{5.111}$$

If a similar procedure for the third-order ghost nodes using a reduced set of finite differences, $\mathbf{A}$

and $\mathbf{B}$ would be:

$$\mathbf{A} = \begin{pmatrix} -8 & 1 & 0 \\ -45 & 9 & -1 \\ -27 & 0 & 1 \\ 0 & -9 & 2 \\ -45 & 0 & 1 \end{pmatrix} \quad , \mathbf{B} = \begin{pmatrix} 0 & 8 & -1 & 0 \\ 0 & -45 & 9 & -1 \\ 0 & -27 & 0 & 1 \\ 60 & -90 & 27 & -4 \\ 60 & -135 & 36 & 5 \end{pmatrix}. \tag{5.112}$$

Using an initial uniform pressure field

$$\mathbf{q}_m = \begin{pmatrix} q_{i,j} \\ q_{i,j+1} \\ q_{i,j+2} \\ q_{i,j+3} \end{pmatrix} = \begin{pmatrix} 0.714 \\ 0.714 \\ 0.714 \\ 0.714 \end{pmatrix} \tag{5.113}$$

, the ghost node pressures would be:

$$\mathbf{q}_g = \begin{pmatrix} q_{i,j-1} \\ q_{i,j-2} \\ q_{i,j-3} \end{pmatrix} = \begin{pmatrix} 0.4887 \\ -0.6269 \\ -0.0430 \end{pmatrix} \tag{5.114}$$

which is unphysical. Even if all of the second-order stencils are included from (5.108) into (5.112) to increase the size of the third-order set, the ghost node values would be:

$$\mathbf{q}_g = \begin{pmatrix} q_{i,j-1} \\ q_{i,j-2} \\ q_{i,j-3} \end{pmatrix} = \begin{pmatrix} -0.7966 \\ -2.8271 \\ -10.020 \end{pmatrix} \tag{5.115}$$

which is worse. This error is alluded to when the MLS stencils are coupled at the boundary to generate Neumann conditions using (5.106), namely that coupling these sets will lead to unphysical

results. Both of these methods are therefore unacceptable for use herein.

This still leaves open the question on how ghost nodes are defined for Neumann conditions. One suggestion is using a Lagrange interpolating polynomial for the ghost nodes is suggested by Berthelsen & Faltinsen [9]; however, Lagrange polynomials have numerical issues as the order of the polynomial is increased (think ill-conditioned system) and the Runge phenomenon [92, Chapter 6]. The Runge phenomenon shows that for equally spaced nodes, the error can become unbounded, especially at extrema [92, Chapter 6, pg. 319]. This is less than ideal, since the ghost nodes are located at the extrema of the stencils. Some authors suggest shifting the extrapolation polynomial, effectively using finite differences where the boundary is not the 'central' node, though this reduces the accuracy of the approximation for the ghost nodes [64, 55]. Effectively, these methods use non-symmetric stencils at the boundaries, which can produce unphysical values at the ghost nodes. Another approach is to solve a so-called extrapolation partial differential equation of varying-order to a steady state using interior information and a level-set function (defined as the signed distance to a boundary) [5]. This method has been used exclusively for Cartesian overset grids [5] and would be difficult to implement for finite-volume discretizations. Therefore, to avoid the issues discussed above, the ghost nodes for Neumann boundary conditions are solved by using a lower-order extrapolation, as used by Motheau et al. [120]. The ghost nodes are prescribed using a set of finite differences (where h is the step size):

$$\begin{aligned}
\frac{\partial q}{\partial n} &= \frac{q_{i,j+1} - q_{i,j-1}}{2h} \\
\frac{\partial q}{\partial n} &= \frac{-q_{i,j+2} + 8q_{i,j+1} - 8q_{i,j-1} + q_{i,j-2}}{12h} \\
\frac{\partial q}{\partial n} &= \frac{q_{i,j+3} - 9q_{i,j+2} + 45q_{i,j+1} - 45q_{i,j-1} + 9q_{i,j-2} - q_{i,j-3}}{60h} \\
\frac{\partial q}{\partial n} &= \frac{-3q_{i,j+4} + 32q_{i,j+3} - 168q_{i,j+2} + 672q_{i,j+1} - 672q_{i,j-1} + 168q_{i,j-2} - 32q_{i,j-3} + 3q_{i,j-4}}{840h}
\end{aligned} \tag{5.116}$$

or the equivalent set generated by the Fornberg algorithm, based on the order desired. While it may not seem to be the most accurate, it is the most stable method. Based on construction of the stencils, the image nodes in the stencil are either known directly from the interior or approximated

with MLS, the definition of the ghost nodes is considered higher-order [157, 8]. Writing the stencil set (5.116) in terms of (5.107) and isolating for the ghost nodes, the boundary condition is written as

$$\left(\frac{\partial\psi}{\partial\mathbf{n}}q\right)_{i,j-2} + \left(\frac{\partial\psi}{\partial\mathbf{n}}q\right)_{i,j-1} = \left(\widehat{\frac{\partial q}{\partial\mathbf{n}}}\right)_{\text{int, Fornberg}} - \left\{\left(\frac{\partial\psi}{\partial\mathbf{n}}q\right)_{i,j} + \left(\frac{\partial\psi}{\partial\mathbf{n}}q\right)_{im:(i,j-1)} + \left(\frac{\partial\psi}{\partial\mathbf{n}}q\right)_{im:(i,j-2)} + \right\}. \tag{5.117}$$

In compact form, (5.117) is:

$$\sum_{\substack{k\in\Omega_{\mathbf{x}_I}\\ k=\text{ghost}}} \left(\frac{\partial\psi}{\partial\mathbf{n}}\right)_k q_k = \left(\widehat{\frac{\partial q}{\partial\mathbf{n}}}\right)_{i,j} - \sum_{\substack{k\in\Omega_{\mathbf{x}_I}\\ k\neq\text{ghost}}} \left(\frac{\partial\Psi}{\partial\mathbf{n}}\right)_k q_k - \sum_{\substack{k\in\Omega_{\mathbf{x}_I}\\ k=\text{image}}} \left(\frac{\partial\Psi}{\partial\mathbf{n}}\right)_k q_k \tag{5.118}$$

Where applicable, (5.118) is written for each node and used to populate (5.101). If the image nodes are not main mesh nodes, then the stencil for the image, $\Omega_{\mathbf{x}_{\text{image}}}$, is used to interpolate the value at the image when solving (5.118). Also, if the intercept node is not explicitly in the main mesh nodes, the intercept node is interpolated using $\Omega_{\mathbf{x}_{\text{int}}}$ with the ghost nodes excluded.

Robin, or mixed, boundary conditions apply both the Dirichlet and Neumann boundary conditions. Robin conditions are common in heat transfer on a surface. Without repeating the previous two sections, the compact form of the Robin condition using (5.118) and (5.104) is:

$$\sum_{\substack{k\in\Omega_{\mathbf{x}_I}\\ k=\text{ghost}}} \left(\Psi + \left(\frac{\partial\Psi}{\partial\mathbf{n}}\right)\right)_k q_k = \left(\widehat{\frac{\partial q}{\partial\mathbf{n}}}\right)_{i,j} + \hat{q}_b - \sum_{\substack{k\in\Omega_{\mathbf{x}_I}\\ k\neq\text{ghost}}} \left(\Psi + \left(\frac{\partial\Psi}{\partial\mathbf{n}}\right)\right)_k q_k \tag{5.119}$$

$\frac{\partial\Psi}{\partial\mathbf{n}}$ in 5.119 can be either the 1D derivative coefficients or the full MLS derivative basis for the intercept node. Where applicable, (5.119) is written for each node and used to populate (5.101).

The last condition that can be enforced on the ghost nodes is the reflection condition. This boundary condition is most notably used on a symmetry boundary to mirror all of the information

to the ghost nodes. For scalar quantities, this is straightforward. The ghost node is set directly by the image node

$$q_{(i,j-1)} = \sum_{k \in \Omega_{\mathbf{x}_{\text{image}}}} \Psi_k q_k \tag{5.120}$$

with the compact form taking the same appearance as the Dirichlet compact form. A vector quantity is handled differently. First, an affine reflection matrix is built using the nodal normal at the boundary node following the same procedure outlined in 5.1.4.2.2. The rotation matrix takes an identical form to (D.16). The translation matrix is the identity matrix since the velocities are not being shifted. Effectively, the affine transformation is $\mathbf{A} = \mathbf{R}_{rot}$. The velocity at the ghost nodes can then be determined as:

$$\begin{bmatrix} \mathbf{v}_{(i,j-1)} \\ 1 \end{bmatrix} = \sum_{k \in \Omega_{\mathbf{x}_{\text{image}}}} \Psi_k \mathbf{A} \begin{bmatrix} \mathbf{v}_k \\ 1 \end{bmatrix}, \tag{5.121}$$

where the velocity is augmented with 1 for consistency. In compact form, the reflection for the velocity is written as:

$$-\sum_{\substack{k \in \Omega_{\mathbf{x}_I} \\ k=\text{ghost} \\ k \neq i,j-1}} \mathbf{A}_t(\Psi_k \mathbf{1}) + (\mathbf{1} - \mathbf{A}_t(\Psi_{i,j-1}\mathbf{1})) = \sum_{\substack{k \in \Omega_{\mathbf{x}_I} \\ k \neq \text{ghost}}} \mathbf{A}_t \Psi_k \mathbf{v}_k \tag{5.122}$$

where $\mathbf{A}_t$ is the truncated affine transformation, where the last row and column are removed.

Equations (5.104), (5.118), and (5.122) are used to populate $\mathbf{A}$ and $\mathbf{b}$ in (5.101). Since $\mathbf{A}$ is only composed of the MLS basis functions and therefore dependent only on the mesh, $\mathbf{A}$ is precomputed and stored, and only needs updating if there is mesh movement. In this work, the sparse matrix package UMFpack in the SuiteSparse [31] matrix suite was used to solve (5.101). UMFpack can symbolically factorize $\mathbf{A}$ and store it; $\mathbf{A}$ stays the same if its structure remains nearly constant. This is important for two reasons: 1) $\mathbf{A}$ can change structurally only if nodes are added or removed from the mesh and 2) the structure of $\mathbf{A}$ is essentially the same for all the of variables in the state vector. In effect, $\mathbf{A}$ only needs to be factored once symbolically and numerically and

can be stored for the entire simulation. Even though this can be taxing from a memory stand-point, it will cut down significantly on the computational cost to solve (5.101). UMFpack can also solve the system exactly or iteratively.

It is important to note that (5.101) will only define the ghost nodes. In general, specifying the ghost nodes will be sufficient for setting the boundary conditions. In special cases, such as a viscous wall, the boundary nodes will also require prescribing a value as well. Thus, the boundary main mesh nodes are discussed as well in the following sections.

5.4.2.3 Inlet

Inlet boundary conditions are used to enforce conditions present at $-\infty$. The ghost nodes 'bridge' the $-\infty$ conditions and the interior. Figure 5.30 illustrates the general construction of the inlet boundary using ghost nodes. For sonic or supersonic flows, the ghost nodes are directly

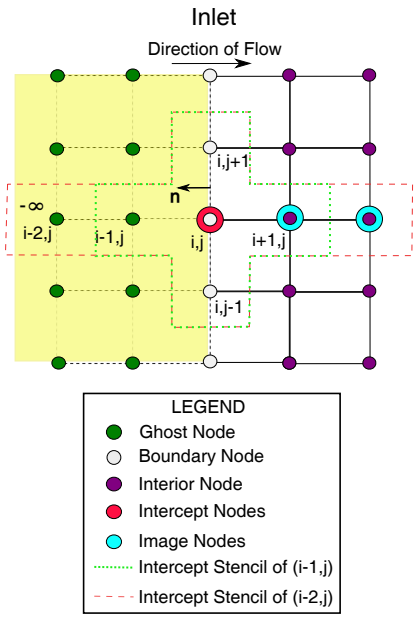


Figure 5.30: Illustration of an Inlet Bop;lundary with ghost nodes.

determined from the $-\infty$:

$$\mathbf{q}_g = \hat{\mathbf{q}} = \begin{pmatrix} \rho_{-\infty} \\ \mathbf{v}_{-\infty} \\ p_{-\infty} \end{pmatrix} \tag{5.123}$$

If the flow is subsonic, the ghost nodes are more difficult to define. A Dirichlet condition based on the image node/stencil is one way to define the ghost nodes, but this is only effective for the ghost closest to the inlet. Ghost nodes further away from the inlet are defined using the freestream conditions, but this approach is too restrictive in not allowing information to travel outward through the inlet. The inlet ghost nodes must be defined with consideration given to allowing information pass through the boundary. It is important to illustrate then what physically happens at the inlet. Figure 5.31 shows an example inlet. At the inlet, the flow is described by five waves: two acoustic

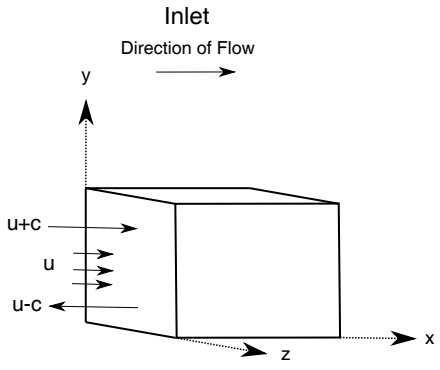


Figure 5.31: Illustration of an Inlet Boundary with Characteristic Waves.

wavs, two vortical waves, and one entropic wave. Four waves enter the domain at speed u (vortical and entropic) and $u+c$ (acoustic), while the other acoustic wave traveling at speed $u-c$ is allowed to exit the domain. c is the local speed of sound. The inlet boundary condition should allow for the correct propagation of these waves.

There are three common methods to implement a non-reflecting boundary condition. The most accurate approach is to numerically extract all of the waves that would pass through the boundary

with a full-eigenvector decomposition at the boundaries [24]. This approach is computationally expensive. On the other end of the spectrum, the least expensive approach is to use Riemann Invariants [12, Chapter 8]. This method has been successfully used in [91, 57, 18], but does not readily allow for the ghost states beyond the closet ghost node to the inlet to be defined. The final approach is to use Navier–Stokes Characteristic Boundary Conditions (NSCBC) or the Locally One-Dimensional Inviscid (LODI) boundary conditions [130, 120]. The NSCBC decompose the flow into a set of one-dimensional characteristic waves. This allows for numerical control of information into and out of the domain. These waves, as shown in Figure 5.31, are the vortical and entropic waves. Formally, the NSCBC are given as [130]:

$$\begin{aligned}
\frac{\partial \rho}{\partial t} + \frac{1}{c^2}\left[\mathcal{L}_2 + \frac{1}{2}(\mathcal{L}_5 + \mathcal{L}_1)\right] &= 0 \\
\frac{\partial u}{\partial t} + \frac{1}{2\rho c}(\mathcal{L}_5 - \mathcal{L}_1) &= 0 \\
\frac{\partial v}{\partial t} + \mathcal{L}_3 &= 0 \\
\frac{\partial w}{\partial t} + \mathcal{L}_4 &= 0 \\
\frac{\partial p}{\partial t} + \frac{1}{2}(\mathcal{L}_5 + \mathcal{L}_1) &= 0
\end{aligned} \tag{5.124}$$

$\mathcal{L}$ are LODI relations in (5.124):

$$\begin{aligned}
\mathcal{L}_1 &= (u - c)\frac{1}{2}\left(\frac{\partial p}{\partial x} - \rho c \frac{\partial u}{\partial x}\right) \\
\mathcal{L}_2 &= u\left(c^2 \frac{\partial \rho}{\partial x} - \frac{\partial p}{\partial x}\right) \\
\mathcal{L}_3 &= u\frac{\partial v}{\partial x} \\
\mathcal{L}_4 &= u\frac{\partial w}{\partial x} \\
\mathcal{L}_5 &= (u + c)\frac{1}{2}\left(\frac{\partial p}{\partial x} + \rho c \frac{\partial u}{\partial x}\right)
\end{aligned} \tag{5.125}$$

The gradients in (5.125) are determined via one-sided differences. Additional terms to account for viscous [152] and transverse [178] effects may be added to (5.124) to better model the flow at the

nonreflecting boundary. Accounting for the transverse terms, the NSCBC in characteristic form is [178]:

$$\begin{aligned}
\frac{1}{2}\left(\frac{\partial p}{\partial t}-\rho c\frac{\partial u}{\partial t}\right)+\mathcal{L}_1-\mathcal{T}_1^1=0\\
\frac{\partial \rho}{\partial t}-\frac{1}{c^2}\frac{\partial p}{\partial t}+\mathcal{L}_2-\mathcal{T}_2^1=0\\
\frac{\partial v}{\partial t}+\mathcal{L}_3-\mathcal{T}_3^1=0\\
\frac{\partial w}{\partial t}+\mathcal{L}_4-\mathcal{T}_4^1=0\\
\frac{1}{2}\left(\frac{\partial p}{\partial t}+\rho c\frac{\partial u}{\partial t}\right)+\mathcal{L}_5-\mathcal{T}_5^1=0
\end{aligned} \tag{5.126}$$

where $\mathcal{T}^1$ are the characteristic transverse terms:

$$\begin{aligned}
\mathcal{T}_1^1&=\mathfrak{T}_5^1-\rho c\mathfrak{T}_2^1\\
\mathcal{T}_2^1&=c^2\mathfrak{T}_1^1-\mathfrak{T}_5^1\\
\mathcal{T}_3^1&=\mathfrak{T}_3^1\\
\mathcal{T}_4^1&=\mathfrak{T}_4^1\\
\mathcal{T}_5^1&=\mathfrak{T}_5^1+\rho c\mathfrak{T}_2^1
\end{aligned} \tag{5.127}$$

and the transverse terms $\mathfrak{T}^1$ are:

$$\begin{aligned}
\mathfrak{T}_1&=-\frac{\partial \rho v}{\partial y}\\
\mathfrak{T}_2&=-v\frac{\partial u}{\partial y}\\
\mathfrak{T}_3&=-v\frac{\partial v}{\partial y}-\frac{1}{\rho}\frac{\partial p}{\partial y}\\
\mathfrak{T}_4&=-v\frac{\partial w}{\partial y}\\
\mathfrak{T}_5&=-v\frac{\partial p}{\partial y}-\gamma p\frac{\partial v}{\partial y}
\end{aligned} \tag{5.128}$$

At the inlet, four variables are prescribed corresponding to the four incoming waves: the velocity $\mathbf{v}$ and temperature $T_{-\infty}$. The LODI relations are modified to prescribe these variables:

$$\begin{aligned}
\mathcal{L}_2 &= \beta_2(T - T_{-\infty}) + \mathcal{T}_2^1 \\
\mathcal{L}_3 &= \beta_3(v - v_{-\infty}) + \mathcal{T}_3^1 \\
\mathcal{L}_4 &= \beta_4(w - w_{-\infty}) + \mathcal{T}_4^1 \\
\mathcal{L}_5 &= \beta_5(u - u_{-\infty}) + \mathcal{T}_5^1
\end{aligned} \tag{5.129}$$

where β are relaxation parameters, $(T, u, v, w)_{-\infty}$ are the desired target values at the boundary, and (T, u, v, w) are the current values of the state vector on the boundary. $\mathcal{T}_5^1$ is modified to [105]:

$$\mathcal{T}_5^1 = \mathfrak{T}_5^1 - \rho c \mathfrak{T}_2^1 \tag{5.130}$$

if the inlet is at $x_1 = 0$. The relaxation parameters are [178, 120]:

$$\begin{aligned}
\beta_2 &= \eta_2 \frac{\rho c R_{\text{gas}}}{l_x} \\
\beta_3 &= \eta_3 \frac{c}{l_x} \\
\beta_4 &= \eta_4 \frac{c}{l_x} \\
\beta_5 &= \eta_5 \frac{\rho c^2 (1 - M_\infty^2)}{l_x}
\end{aligned} \tag{5.131}$$

where l_x is the characteristic length of the domain, and η are further constants used to tune the flow. Typically, $\eta_3 = \eta_4 = 0$ and $\eta_2 = -\eta_5 = -0.278$ [178]. Additionally, the transverse terms may not be included since the majority of the flow variables are prescribed at the inlet [178]. However, if the waves propagating from the interior of the domain are non-planar, the transverse waves should not be neglected. The NSCBC conditions are written in terms of the flow gradients [154, 130, 120]:

$$\frac{\partial q}{\partial x}_{\text{NSCBC}} = \frac{\partial q}{\partial x} = \mathbf{S}_x \mathbf{\Lambda}^{-1} \mathcal{L} \tag{5.132}$$

where $\mathbf{S}_x$ is the eigenvector matrix:

$$\mathbf{S}_x = \begin{pmatrix} \frac{1}{2c^2} & \frac{1}{c^2} & 0 & 0 & \frac{1}{2c^2} \\ \frac{-1}{2\rho c} & 0 & 0 & 0 & \frac{1}{2\rho c} \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} \end{pmatrix} \tag{5.133}$$

and $\mathbf{\Lambda}$ is the eigenvalue matrix $\boldsymbol{\lambda}\mathbf{I}$.

The final issue with the inlet boundary conditions is where the condition is applied. For traditional finite-volume cell-vertex and finite difference methods, the boundary condition is applied on the boundaries, and fluxes are computed on the boundary face. For finite-volume cell-center schemes, the boundary condition is also applied on the boundary face, but since the state vector is stored at the cell center, gradient conditions are applied directly in the flux. Since the current implementation is a 'mix' of the cell-center and cell-vertex, the application of the inlet boundary condition is applied at the flux face of the augmented inlet boundary cell, as shown in Figure 5.32. As the flux location lies outside of the domain, for second-order or higher, the state vector and gradients are extrapolated to the flux face from the boundary/intercept location to determine the nonreflecting gradients (5.132). The inlet boundary condition could be set at the interior, but that would cause the boundary to become decoupled from the rest of the field. Once the LODI conditions are determined, the ghost node system for inlet nodes is built using the Neumann condition (5.118) for the ghost node system (5.101).

For the SST-Menter RANS turbulence model [115], the ghost node state vector is set to constant

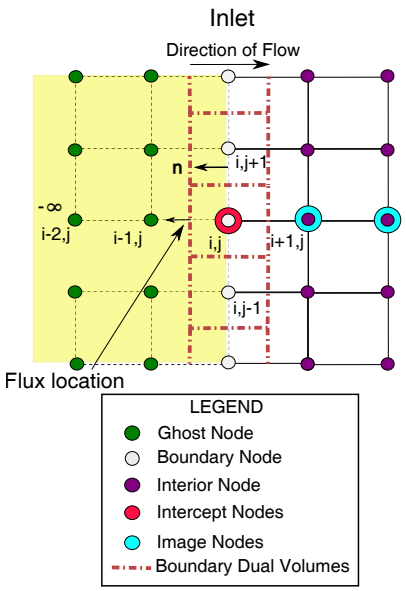


Figure 5.32: Flux Location of Inlet Boundary Cells.

values prescribed by [13, Chapter 7, pg 249]:

$$\begin{aligned} \omega_\infty &= C_1 \frac{||\mathbf{v}_\infty||}{L} \\ (\mu_T)_\infty &= (\mu_L)_\infty 10^{-C_2} \\ K_\infty &= \frac{(\mu_T)_\infty}{\rho_\infty} \omega_\infty \end{aligned} \tag{5.134}$$

where $\infty = g$, $1 \leq C_1 \leq 10$, $2 \leq C_2 \leq 5$, and L is the length of the domain.

5.4.2.4 Outlet

Outlet boundary conditions enforce the conditions at ∞. Figure 5.33 illustrates the conditions enforced at the outlet boundary. There are three options to set the outlet boundary condition. The first is the zero-normal gradient condition:

$$\widehat{\left(\frac{\partial \mathbf{q}}{\partial \mathbf{n}}\right)} = 0. \tag{5.135}$$

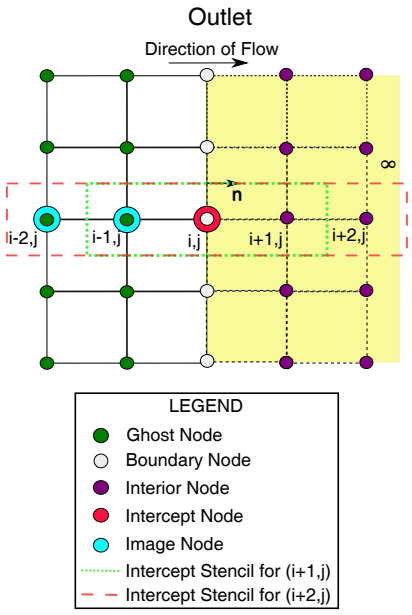


Figure 5.33: Illustration of an Outlet Boundary with Ghost Nodes.

This boundary condition does not explicitly set the back pressure but relies on the upstream values to maintain the desired flow. The second option is the convective boundary condition. This takes the form:

$$\widehat{\left(\frac{\partial \mathbf{q}}{\partial \mathbf{n}}\right)} = \left(\frac{\partial \mathbf{q}}{\partial \mathbf{n}}\right)_{\text{interior}} \tag{5.136}$$

where $\left(\frac{\partial \mathbf{q}}{\partial \mathbf{n}}\right)_{\text{interior}}$ is the derivative normal to the boundary determined by a one-sided backward finite difference at the intercept node. This condition is only applied to the pressure and density, as the zero-normal gradient condition is applied to the velocity fields. Applying the convective boundary to the velocities will cause the flow to become unstable at the outlet. $\left(\frac{\partial \mathbf{q}}{\partial \mathbf{n}}\right)_{\text{interior}}$ is computed either with a one-sided finite difference or a one-sided MLS derivative reconstruction (5.46) at the intercept node. The MLS derivative reconstruction (5.46) is used when the intercept node is not explicitly a node in the main mesh. A convective boundary is preferred when the flow does not exit normal to the boundary or for supersonic flows. It is not recommended for subsonic flows since the back pressure is difficult to maintain.

The final approach is using the NSCBC/LODI conditions as at the inlet (5.132). For the outlet

boundary, only the acoustic wave is allowed to enter the domain, as shown in Figure 5.34. The

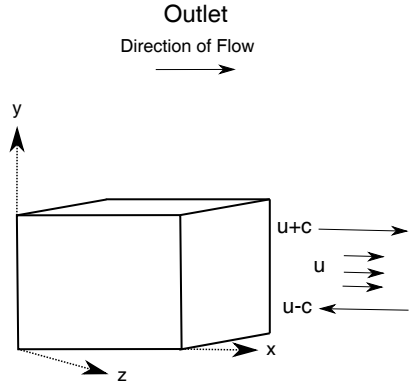


Figure 5.34: Illustration of an Outlet Boundary with Characteristic Waves.

LODI relations at the outlet are modified to prescribe the pressure [177]:

$$\mathcal{L}_1 = \alpha_x(p - p_\infty) + (1 - \beta_1)\mathcal{T}_1^1 \tag{5.137}$$

where α_x is the pressure relaxation term

$$\alpha_x = \frac{\sigma c(1 - M^2)}{l_x} \tag{5.138}$$

$\beta_1 = M$ is the transverse relaxation term, and $\mathcal{T}_1$ is given by (5.128). σ is the pressure relaxation coefficient, typically set to $\sigma = 0.28$ [139, 140]. The pressure relaxation term keeps the pressure from drifting [139, 140, 130]. For inviscid flows, the remaining transverse terms given by (5.127) are included. However, for viscous flows, the remaining transverse terms are neglected, since large wall-normal gradients near the wall, especially during transients, can cause the solution to diverge quickly. The nonreflecting gradients (5.132) using (5.137) are determined for each ghost node and are used to build the Neumann condition (5.118) for the ghost node system (5.101).

For the SST-Menter RANS turbulence model [115], the ghost node turbulent state vector is

defined via the zero-normal derivative

$$\widehat{\left(\frac{\partial \mathbf{q_T}}{\partial \mathbf{n}}\right)} = \mathbf{0}. \tag{5.139}$$

using (5.118).

5.4.2.5 Far-Field

Far-field boundary conditions enforce the far field conditions at ∞. Figure 5.35 shows the layout for a typical far-field boundary. The far-field boundary is considered an 'open', or free

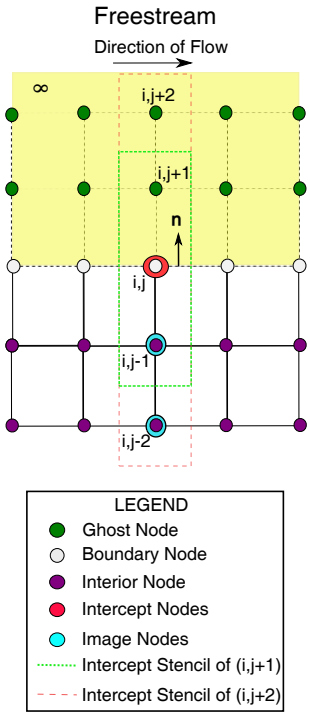


Figure 5.35: Illustration of a Far-Field Boundary with Ghost Nodes.

boundary, allowing the boundary to effectively switch between an inlet or an outlet. There are a few methods available to define the state at the ghost nodes. The first is to simply define the ghost

nodes values as the freestream conditions:

$$\mathbf{q}_g = \hat{\mathbf{q}} = \begin{pmatrix} \rho_\infty \\ \mathbf{v}_\infty \\ p_\infty \end{pmatrix} \tag{5.140}$$

While this method does lead to overall good convergence and solutions, it does not explicitly allow information to pass in or out of the domain, since the ghost nodes state values are rigidly defined. The second approach is consider the boundary as streamline, which effectively makes the far-field boundary a symmetry boundary. This approach is discussed in more detail the next section. The symmetry-far-field boundary is a pure reflective boundary, though, and can cause stalling (or no) convergence of the solution. The final approach is to use the LODI relations, which should allow information to pass freely through the boundary. Figure 5.36 illustrates the waves entering and exiting the far-field boundary. The eigenvalues are $\xi = (v - c, v, v, v, v + c)$. It is important to

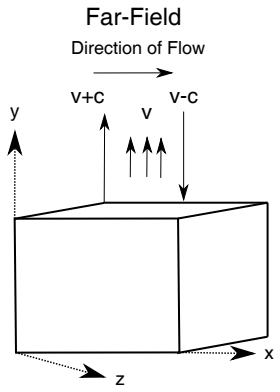


Figure 5.36: Illustration of a Far-Field Boundary with Characteristic Waves.

note that the vortical and entropic waves (those traveling at speed v) are not guaranteed to exit the

domain as shown in Figure 5.36. The LODI relations in the y-direction, with transverse terms, are:

$$\begin{aligned}
\frac{1}{2}\left(\frac{\partial p}{\partial t} - \rho c \frac{\partial v}{\partial t}\right) + \mathcal{M}_1 - \mathcal{T}_1^2 &= 0 \\
\frac{\partial u}{\partial t} + \mathcal{M}_2 - \mathcal{T}_2^2 &= 0 \\
\frac{\partial \rho}{\partial t} - \frac{1}{c^2}\frac{\partial p}{\partial t} + \mathcal{M}_3 - \mathcal{T}_3^2 &= 0 \\
\frac{\partial w}{\partial t} + \mathcal{M}_4 - \mathcal{T}_4^2 &= 0 \\
\frac{1}{2}\left(\frac{\partial p}{\partial t} + \rho c \frac{\partial v}{\partial t}\right) + \mathcal{M}_5 - \mathcal{T}_5^2 &= 0
\end{aligned} \tag{5.141}$$

where $\mathcal{M}$ are the y-direction LODI relations:

$$\begin{aligned}
\mathcal{M}_1 &= (v-c)\frac{1}{2}\left(\frac{\partial p}{\partial y} - \rho c \frac{\partial v}{\partial y}\right) \\
\mathcal{M}_2 &= v\frac{\partial u}{\partial y} \\
\mathcal{M}_3 &= v\left(c^2\frac{\partial \rho}{\partial y} - \frac{\partial p}{\partial y}\right) \\
\mathcal{M}_4 &= v\frac{\partial w}{\partial y} \\
\mathcal{M}_5 &= (v+c)\frac{1}{2}\left(\frac{\partial p}{\partial y} + \rho c \frac{\partial v}{\partial y}\right),
\end{aligned} \tag{5.142}$$

$\mathcal{T}^2$ are the characteristic transverse terms:

$$\begin{aligned}
\mathcal{T}_1^2 &= \mathfrak{T}_5^2 - \rho c \mathfrak{T}_3^2 \\
\mathcal{T}_2^2 &= \mathfrak{T}_2^2 \\
\mathcal{T}_3^2 &= c^2\mathfrak{T}_1^2 - \mathfrak{T}_5^2 \\
\mathcal{T}_4^2 &= \mathfrak{T}_4^2 \\
\mathcal{T}_5^2 &= \mathfrak{T}_5^2 + \rho c \mathfrak{T}_3^2,
\end{aligned} \tag{5.143}$$

and the transverse terms $\mathfrak{T}^2$ are:

$$\begin{aligned}
\mathfrak{T}_1^2 &= -\frac{\partial \rho u}{\partial x} \\
\mathfrak{T}_2^2 &= -u\frac{\partial u}{\partial x} \\
\mathfrak{T}_3^2 &= -u\frac{\partial u}{\partial x} - \frac{1}{\rho}\frac{\partial p}{\partial x} \\
\mathfrak{T}_4^2 &= -u\frac{\partial w}{\partial x} \\
\mathfrak{T}_5^2 &= -u\frac{\partial p}{\partial x} - \gamma p\frac{\partial u}{\partial x}.
\end{aligned} \tag{5.144}$$

If the characteristics are traveling outward, the LODI relations are determined directly from the interior one-sided differences. If the characteristics are traveling into the domain, then the LODI $\mathcal{M} = 0$ [84]. The pressure relation, $\mathcal{M}_1$, however, is determined using a relationship similar to the outlet:

$$\mathcal{M}_1 = \alpha_y(p - p_\infty) + (1 - \beta_1)\mathcal{T}_1^2 \tag{5.145}$$

where α_y is the pressure relaxation term

$$\alpha_y = \frac{\sigma c(1 - M^2)}{l_y} \tag{5.146}$$

and l_y is the characteristic height of the domain [35]. At the far-field boundary,

$$\frac{\partial q}{\partial y}_{\text{LODI}} = \frac{\partial q}{\partial y} = \mathbf{S}_y \boldsymbol{\Xi}^{-1} \mathcal{M} \tag{5.147}$$

where $\mathbf{S}_y$ is the eigenvector matrix:

$$\mathbf{S}_y = \begin{pmatrix} \frac{1}{2c^2} & 0 & \frac{1}{c^2} & 0 & \frac{1}{2c^2} \\ 0 & 1 & 0 & 0 & 0 \\ \frac{-1}{2\rho c} & 0 & 0 & 0 & \frac{1}{2\rho c} \\ 0 & 0 & 0 & 1 & 0 \\ \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} \end{pmatrix} \tag{5.148}$$

and $\mathbf{\Xi}^{-1}$ is only applied to non-zero waves. $\mathcal{M}$ are not multiplied by ξ to avoid numerical issues when $\xi \leftarrow 0$. The characteristic approach is the preferred method herein, and it is used to build the system for each far-field ghost node and added to the system (5.101).

5.4.2.6 Non-Reflecting Compatibility Boundaries

Before continuing with the remaining boundary conditions, the special cases of adjacent non-reflecting boundary conditions must be discussed. At these boundaries, the waves in two directions become coupled, and special care must be taken to ensure robustness and stability of the boundary condition. The NSCBC written in two-dimensions are:

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \frac{1}{c^2}\left[\mathcal{L}_2 + \frac{1}{2}(\mathcal{L}_5 + \mathcal{L}_1)\right] + \frac{1}{c^2}\left[\mathcal{M}_3 + \frac{1}{2}(\mathcal{M}_5 + \mathcal{M}_1)\right] &= 0 \\ \frac{\partial u}{\partial t} + \frac{1}{2\rho c}(\mathcal{L}_5 - \mathcal{L}_1) + \mathcal{M}_2 &= 0 \\ \frac{\partial v}{\partial t} + \mathcal{L}_3 + \frac{1}{2\rho c}(\mathcal{M}_5 - \mathcal{M}_1) &= 0 \\ \frac{\partial w}{\partial t} + \mathcal{L}_4 + \mathcal{M}_4 &= 0 \\ \frac{\partial p}{\partial t} + \frac{1}{2}(\mathcal{L}_5 + \mathcal{L}_1) + \frac{1}{2}(\mathcal{M}_5 + \mathcal{M}_1) &= 0 \end{aligned} \tag{5.149}$$

In characteristic form, the NSCBC equations are written separately. In the x-direction, the NSCBC equations are:

$$\begin{aligned}
\frac{1}{2}\left(\frac{\partial p}{\partial t} - \rho c \frac{\partial u}{\partial t}\right) + \mathcal{L}_1 - \mathcal{T}_1^1 &= 0 \\
\frac{\partial \rho}{\partial t} - \frac{1}{c^2}\frac{\partial p}{\partial t} + \mathcal{L}_2 - \mathcal{T}_2^1 &= 0 \\
\frac{\partial v}{\partial t} + \mathcal{L}_3 - \mathcal{T}_3^1 &= 0 \\
\frac{\partial w}{\partial t} + \mathcal{L}_4 - \mathcal{T}_4^1 &= 0 \\
\frac{1}{2}\left(\frac{\partial p}{\partial t} + \rho c \frac{\partial u}{\partial t}\right) + \mathcal{L}_5 - \mathcal{T}_5^1 &= 0
\end{aligned} \tag{5.150}$$

and in the $y-$direction:

$$\begin{aligned}
\frac{1}{2}\left(\frac{\partial p}{\partial t} - \rho c \frac{\partial v}{\partial t}\right) + \mathcal{M}_1 - \mathcal{T}_1^2 &= 0 \\
\frac{\partial u}{\partial t} + \mathcal{M}_2 - \mathcal{T}_2^2 &= 0 \\
\frac{\partial \rho}{\partial t} - \frac{1}{c^2}\frac{\partial p}{\partial t} + \mathcal{M}_3 - \mathcal{T}_3^2 &= 0 \\
\frac{\partial w}{\partial t} + \mathcal{M}_4 - \mathcal{T}_4^2 &= 0 \\
\frac{1}{2}\left(\frac{\partial p}{\partial t} + \rho c \frac{\partial v}{\partial t}\right) + \mathcal{M}_5 - \mathcal{T}_5^2 &= 0
\end{aligned} \tag{5.151}$$

The formal difference between these equations, (5.150) and (5.151), and the original LODI equations, (5.126) and (5.141), lies in the transverse terms. The transverse terms are now written in

terms of the other characteristics. The characteristic transverse terms in the $x-$direction are:

$$\begin{aligned}
\mathcal{T}_1^1 &= -\frac{1}{2}(\mathcal{M}_5 + \mathcal{M}_1) + \rho c \mathcal{M}_2 \\
\mathcal{T}_2^1 &= -\mathcal{M}_3 \\
\mathcal{T}_3^1 &= -\frac{1}{2\rho c}(\mathcal{M}_5 - \mathcal{M}_1) \\
\mathcal{T}_4^1 &= -\mathcal{M}_4 \\
\mathcal{T}_5^1 &= -\frac{1}{2}(\mathcal{M}_5 + \mathcal{M}_1) - \rho c \mathcal{M}_2
\end{aligned} \tag{5.152}$$

and in the $y-$direction

$$\begin{aligned}
\mathcal{T}_1^2 &= -\frac{1}{2}(\mathcal{L}_5 + \mathcal{L}_1) + \rho c \mathcal{L}_3 \\
\mathcal{T}_2^2 &= -\frac{1}{2\rho c}(\mathcal{L}_5 - \mathcal{L}_1) \\
\mathcal{T}_3^2 &= -\mathcal{L}_2 \\
\mathcal{T}_4^2 &= \mathcal{L}_4 \\
\mathcal{T}_5^2 &= -\frac{1}{2}(\mathcal{L}_5 + \mathcal{L}_1) - \rho c \mathcal{L}_3,
\end{aligned} \tag{5.153}$$

Both (5.150) and (5.151) are modified appropriately to related the conditions in the $x-$ and $y-$ direction. The derivatives for the ghost nodes at these locations are defined by:

$$\frac{\partial q}{\partial x}_{\text{LODI}} = \frac{\partial q}{\partial x} = \mathbf{S}_x \mathbf{\Lambda}^{-1} \mathcal{L} \tag{5.154}$$

and

$$\frac{\partial q}{\partial y}_{\text{LODI}} = \frac{\partial q}{\partial y} = \mathbf{S}_y \mathbf{\Xi}^{-1} \mathcal{M} \tag{5.155}$$

The three situations discussed herein on how (5.150) and (5.151):

- Outlet/Far-Field
- Inlet/Far-Field

- Wall/Symmetry & Inlet/Outlet

5.4.2.6.1 Outlet/Far-field At the boundary where an outlet and far-field boundary meet, two waves, one from each direction normal to the boundary, enter the domain, while the remaining waves are allowed to exit the domain. Figure 5.37 illustrates this. Both incoming waves are

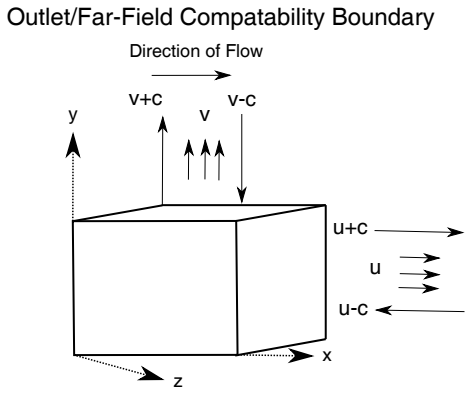


Figure 5.37: Illustration of an Outlet/Far-Field Boundary with Characteristic Waves.

acoustic waves, and defined by:

$$\begin{aligned} \frac{1}{2}\left(\frac{\partial p}{\partial t} - \rho c \frac{\partial u}{\partial t}\right) + \mathcal{L}_1 - \mathcal{T}_1^1 = 0 \\ \frac{1}{2}\left(\frac{\partial p}{\partial t} - \rho c \frac{\partial v}{\partial t}\right) + \mathcal{M}_1 - \mathcal{T}_1^2 = 0 \end{aligned} \tag{5.156}$$

with the transverse terms defined by (5.152) and (5.153) [105]. The pressure, as before with the outlet and far-field, is relaxed:

$$\begin{aligned} \frac{1}{2}\left(\frac{\partial p}{\partial t} - \rho c \frac{\partial u}{\partial t}\right) + \alpha_x(p - p_\infty) + (1-\beta)\mathcal{T}_1^1 = 0 \\ \frac{1}{2}\left(\frac{\partial p}{\partial t} - \rho c \frac{\partial v}{\partial t}\right) + \alpha_y(p - p_\infty) + (1-\beta)\mathcal{T}_1^2 = 0 \end{aligned} \tag{5.157}$$

Using (5.156) and (5.157), the compatibility relations for the acoustic waves are expressed as [105]:

$$\begin{aligned}\mathcal{L}_1 + \frac{1-\beta}{2}\mathcal{M}_1 &= \alpha_x(p - p_\infty) + (1-\beta)\mathcal{T}_1^1 \\ \frac{1-\beta}{2}\mathcal{L}_1 + \mathcal{M}_1 &= \alpha_y(p - p_\infty) + (1-\beta)\mathcal{T}_1^2\end{aligned} \tag{5.158}$$

with the characteristic transverse terms modified to:

$$\begin{aligned}\mathcal{T}_1^1 &= -\frac{1}{2}\mathcal{M}_5 - \rho c\mathcal{M}_2 \\ \mathcal{T}_1^2 &= -\frac{1}{2}\mathcal{L}_5 - \rho c\mathcal{L}_3\end{aligned} \tag{5.159}$$

The system (5.158) is solved and used in (5.154) and (5.155) to define the ghost nodal values.

5.4.2.6.2 Inlet/Far-Field At the boundary where an inlet and far-field boundary meet, five waves, four from the inlet and one from the far-field, enter the domain. Figure 5.38 illustrates this. The

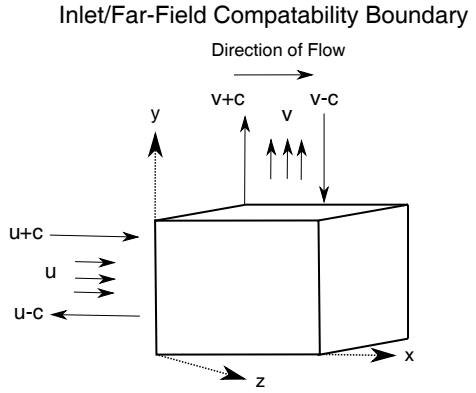


Figure 5.38: Illustration of an Inlet/Far-Field Boundary with Characteristic Waves.

compatibility relationship could be defined as for the outlet/far-field boundary, but this situation has known stability issues [105]. Instead, the pressure relation at the far-field, $\mathcal{M}_1$ is set to zero, allowing the pressure to adjust to the inlet parameters. The LODI relations for the inlet are defined

as [105]:

$$\begin{aligned}
\mathcal{L}_2 &= \beta_2(T - T_{-\infty}) - \mathcal{T}_2^1 \\
\mathcal{L}_3 &= \beta_3(v - v_{-\infty}) - \mathcal{T}_3^1 \\
\mathcal{L}_4 &= \beta_4(w - w_{-\infty}) - \mathcal{T}_4^1 \\
\mathcal{L}_5 &= \beta_5(u - u_{-\infty}) - \mathcal{T}_5^1
\end{aligned} \tag{5.160}$$

where the characteristic transverse terms $\mathcal{T}^1$ are defined as:

$$\begin{aligned}
\mathcal{T}_1^1 &= -\frac{1}{2}\mathcal{M}_5 + \rho c \mathcal{M}_2 \\
\mathcal{T}_2^1 &= -\mathcal{M}_3 \\
\mathcal{T}_3^1 &= -\frac{1}{2\rho c}\mathcal{M}_5 \\
\mathcal{T}_4^1 &= -\mathcal{M}_4 \\
\mathcal{T}_5^1 &= -\frac{1}{2}\mathcal{M}_5 - \rho c \mathcal{M}_2
\end{aligned} \tag{5.161}$$

For the far-field, the LODI relations are defined as:

$$\begin{aligned}
\mathcal{M}_1 &= -\mathcal{T}_1^2 \\
\mathcal{M}_2 &= \mathcal{M}_2 \\
\mathcal{M}_3 &= \mathcal{M}_3 \\
\mathcal{M}_4 &= \mathcal{M}_4 \\
\mathcal{M}_5 &= \mathcal{M}_5 - \mathcal{T}_5^2
\end{aligned} \tag{5.162}$$

where the characteristic transverse terms $\mathcal{T}^2$ are defined as in (5.153). The ghost nodal values are then defined using (5.154) and (5.155).

5.4.2.7 *Inflow/Wall or Symmetry and Outflow/Wall or Symmetry*

The last grouping of combined boundary conditions is where an inlet or outflow meet a symmetry or wall boundary. In all cases, the wall and symmetry are reflective boundaries. At the outflow/wall boundary, $\mathcal{L}_{2,3,4} = 0$ and the acoustic waves are equal $\mathcal{L}_5 = \mathcal{L}_1$. When the outflow interfaces with a symmetry boundary, $\mathcal{L}_2$ is not zero, but the remaining LODI relations are the same as for a outflow/wall interface. At the inflow boundary, the same relations hold as at the outflow/wall and outflow/symmetry interfaces, but instead $L_1 = L_5$.

5.4.2.8 *Symmetry and Inviscid Walls*

The symmetry and inviscid wall boundaries require zero-normal derivatives for all flow variables as well as zero-normal velocity:

$$\begin{aligned} \frac{\partial q}{\partial \mathbf{n}} &= \mathbf{0} \\ \mathbf{v} \cdot \mathbf{n} &= 0 \end{aligned} \tag{5.163}$$

Figure 5.39 illustrates a typical symmetry boundary. For the orthogonal grid case, the pressure and density for the ghost node can be directly specified from the image node:

$$\begin{aligned} \rho_g &= \rho_{im} \\ p_g &= p_{im} \end{aligned} \tag{5.164}$$

For the non-orthogonal case, the pressure and density for the ghost node are determined by the Neumann condition (5.118) with a zero-normal gradient

$$\begin{aligned} \widehat{\left(\frac{\partial \rho}{\partial \mathbf{n}}\right)} &= \mathbf{0} \\ \widehat{\left(\frac{\partial p}{\partial \mathbf{n}}\right)} &= \mathbf{0}. \end{aligned} \tag{5.165}$$

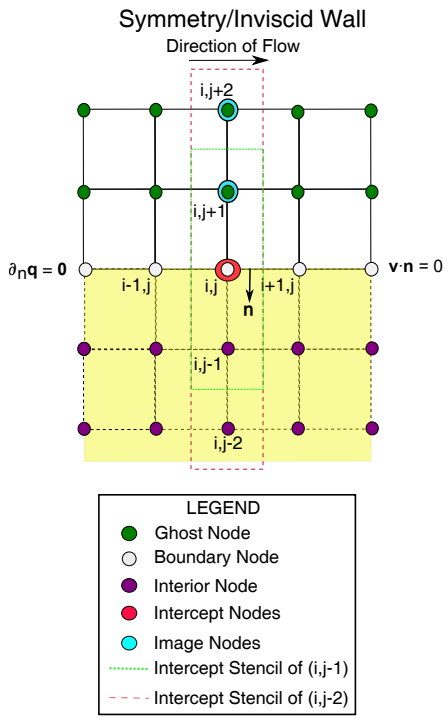


Figure 5.39: Illustration of a Symmetry Boundary with Ghost Nodes.

The velocities are determined from the reflected velocity at the image node, as given by (5.122) and as outlined by Algorithm D.2 in Appendix D. The relationships for the density, pressure, and velocities for inviscid wall and symmetry ghost nodes are built and added into the system (5.101). Additionally, the velocity for inviscid wall and symmetry boundary nodes may be set to:

$$\mathbf{v}_i \cdot \mathbf{n}_i = 0 \tag{5.166}$$

where $\mathbf{n}_i$ is the boundary normal at the node, typically taken as an average normal of the related boundary faces. This strongly enforces the boundary conditions, but enforcing the boundary conditions in this way can cause the solution convergence to stall.

For curved inviscid walls, the zero-gradient pressure condition is not correct. Instead, the

pressure gradient at the wall is a function of the curvature [134]:

$$\widehat{\left(\frac{\partial p}{\partial \mathbf{n}}\right)} = \frac{\rho \mathbf{v}_t}{R} \tag{5.167}$$

where $\mathbf{v}_t$ is the tangential velocity and R is the average radius of curvature of surface, determined by:

$$R_j = \sum_i \frac{(\mathbf{n}_i - \mathbf{n}_j) \cdot (\mathbf{x}_i - \mathbf{x}_j)}{||\mathbf{x}_i - \mathbf{x}_j||} \tag{5.168}$$

where i is the connected surface nodes, and $\mathbf{n}$ are the averaged nodal normals. The normal density gradient is, using the ideal gas law and $T =$ constant in an inviscid flow:

$$\begin{aligned}
P &= \rho R_g T \\
\frac{\partial P}{\partial \mathbf{n}} &= R_g \frac{\partial (\rho T)}{\partial \mathbf{n}} \\
\frac{\partial P}{\partial \mathbf{n}} &= R_g T_w \frac{\partial \rho}{\partial \mathbf{n}} + R_g \rho_w \frac{\partial T}{\partial \mathbf{n}} \\
\frac{\partial P}{\partial \mathbf{n}} &= R_g T_w \frac{\partial \rho}{\partial \mathbf{n}} \\
\frac{\partial \rho}{\partial \mathbf{n}} &= \frac{1}{R_g T_w} \frac{\partial P}{\partial \mathbf{n}} \\
\frac{\partial \rho}{\partial \mathbf{n}} &= \frac{1}{R_g T_w} \frac{\rho_w \mathbf{v}_t}{R}
\end{aligned} \tag{5.169}$$

where R_g is the gas constant and $()_w$ is a quantity at the wall. At the symmetry boundary for viscous flows, the SST-Menter RANS turbulence model [115], the ghost node turbulent state vector is defined via the zero-normal derivative

$$\widehat{\left(\frac{\partial \mathbf{q_T}}{\partial \mathbf{n}}\right)} = \mathbf{0}. \tag{5.170}$$

using (5.118). The Neumann condition is built for each ghost inviscid wall or symmetry node and added into the system (5.101).

5.4.2.9 *Viscous Walls*

Viscous, no-slip, walls require zero velocity

$$\mathbf{v} = \mathbf{0} \tag{5.171}$$

and zero normal pressure gradient

$$\frac{\partial p}{\partial \mathbf{n}} = 0 \tag{5.172}$$

at the wall. Constraints on the temperature require either zero-normal gradient for adiabatic walls

$$\frac{\partial T}{\partial \mathbf{n}} = 0 \tag{5.173}$$

or constant temperature on the wall for isothermal walls

$$T_w = T_{\text{wall}}. \tag{5.174}$$

The boundary conditions applied at a viscous wall are shown in Figure 5.40.

The velocities for the ghost nodes of both adiabatic and isothermal walls are defined using a zero-second derivative [12, Chapter 8, pg 281]:

$$\frac{\partial^2 \mathbf{v}}{\partial \mathbf{n}^2} = \mathbf{0} \tag{5.175}$$

For an adiabatic wall, both the pressure and density are enforced using (5.118) with a zero-normal gradient condition. For an isothermal wall, only the pressure is enforced using the zero-normal gradient. The temperature for the wall ghost nodes is the isothermal wall temperature, T_w. The density for the ghost nodes is determined directly from T_w and the pressure gradient. One could be tempted to differentiate the ideal gas law to derive an expression for the density:

$$\frac{\partial \rho}{\partial \mathbf{n}} = -\frac{\rho_w}{T_w}\frac{\partial T}{\partial \mathbf{n}} \tag{5.176}$$

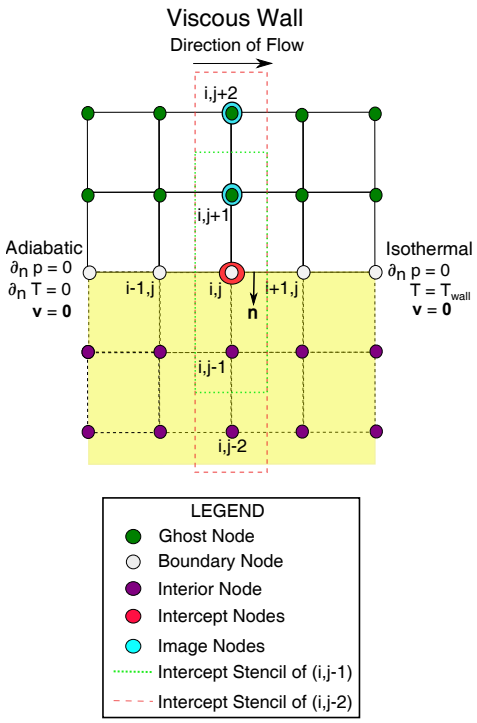


Figure 5.40: Illustration of a Viscous Wall with Ghost Nodes.

but this is unstable and will set unphysical (low) temperatures and (high) densities in the wall. What is instead done herein is to compute the ghost density after ghost system is solved using the normal pressure and isothermal wall temperature. After the ghost states are determined, the wall pressure at the boundary is determined using a one-sided finite difference.

For an adiabatic wall, the states at the ghost are defined using (5.118) and (5.175), and for an isothermal wall the states at the ghost are defined using (5.118), (5.175), and (5.176). These relationships are built and added into the system (5.101).

For the SST-Menter RANS turbulence model [115], the ghost node turbulent state vector is defined via the zero-normal derivative

$$\widehat{\left(\frac{\partial \mathbf{q_T}}{\partial \mathbf{n}}\right)} = \mathbf{0}. \tag{5.177}$$

using (5.118). K and ω in the SST-Menter RANS model is also directly applied to the wall nodes:

$$K = 0$$
$$\omega = 10\frac{6\mu_L}{\rho\beta_1 d_1^2} \tag{5.178}$$

where d_1 is the distance to the first wall node. The Neumann condition is built for each ghost wall node and added into the system (5.101).

5.4.2.10 Rotating Walls

Rotating walls are handled the same as their stationary viscous or inviscid counterparts for the flow variables. Figure 5.41 illustrates a rotating wall boundary.

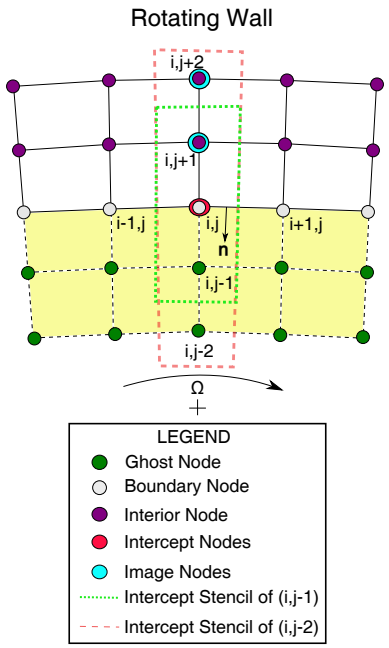


Figure 5.41: Illustration of Ghost Node Definitions for a Rotating Wall.

The modification made for rotational wall boundaries is the velocity on the wall is now non-

zero. The velocity on the wall is with the rotation about the $x-$axis is:

$$\begin{aligned} u &= 0 \\ v &= \Omega z_b \\ w &= -\Omega y_b \end{aligned} \tag{5.179}$$

where Ω is the rotational speed of the wheel, and y_b and z_b are the coordinates of the boundary node. To define the wall ghost nodes, (5.118) and (5.175) for adiabatic walls and (5.118), (5.175), and (5.176) for isothermal walls are used. These relationships are built and added into the system (5.101).

For the SST-Menter RANS turbulence model [115], the ghost node turbulent state vector is defined via the zero-normal derivative

$$\widehat{\left(\frac{\partial \mathbf{q_T}}{\partial \mathbf{n}}\right)} = \mathbf{0}. \tag{5.180}$$

using (5.118). The Neumann condition is built for each ghost rotating wall node and added into the system (5.101).

5.4.3 Boundary Fluxes

As discussed in Sections 4.2 and 5.3, special care is required when integrating both the convective and diffusive fluxes at the boundaries. When ghost nodes are used, the non-wall boundary fluxes are effectively computed using the same methodology applied to an internal edge. The flux is computed between a boundary node and a ghost node, which comprise a ghost flux edge. Figure 5.42 illustrates a ghost-flux edge at one of the boundaries. For the convective flux, the state vector and derivative at the ghost node takes the place of either the left or right state in (4.19), (5.77), or (5.78) for the reconstruction and (4.6) for the flux. The reconstruction and quadrature takes place on the shared dual faces of the boundary dual cell with the ghost dual cell. When determining the flux limiter for the reconstruction in (4.19), (5.77), or (5.78), the ghost nodes are included.

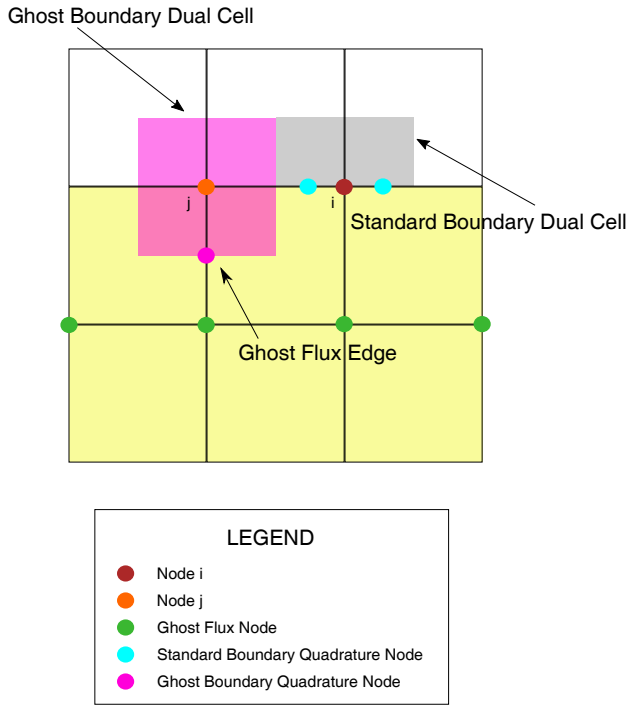


Figure 5.42: Ghost-Flux Edge for Integration of the Fluxes along Boundaries.

Once the convective fluxes along the ghost flux edge is determined, the boundary node residual is updated since the ghost node is determined from the ghost node system (5.101).

For the diffusive flux, the state vector and derivative at the ghost node again takes the place of either the left or right state in (5.82) or (5.83) for the averaged reconstructed viscous flux. In the case of the MLS reconstructed viscous flux, the viscous flux stencil is built at the quadrature nodes of the dual faces such that (5.85) and (5.86) can be determined. Once the reconstructions are determined using either approach, (2.5) is determined. The boundary nodes residuals are updated as with the convective fluxes.

Wall boundaries using ghost require a slight modification when computing the fluxes. Instead of computing the flux on the interface dual faces, the flux is computed at the quadrature nodes on the wall, since the wall is a physical boundary. These are the standard quadrature nodes shown in Figure 5.42. For viscous walls, wall boundary fluxes may not be computed. For adiabatic wall boundary conditions, only the flux for the energy equation is computed. For isothermal walls, the

flux is not computed, since the state at the wall is fully defined.

5.5 Higher-Order Temporal Schemes

This section discusses the final step to creating a higher-order method: the temporal scheme. Unlike first- and second-order accurate methods, where a single register is sufficient to integrate accurately, third- and higher-order accurate methods require more than one register to achieve the desired numerical accuracy. A register is an array of data. As an example, in (4.31) the register is the residual $\mathbf{R}\left(\mathbf{Q}_i^{()}\right)$. Third- and fourth-order accuracy is achieved by utilizing an additional register. Two-register Runge–Kutta schemes are widely used in the DNS community, as implicit schemes smooth information that is critical to solving these flows [89].

There are two approaches to implementing two-register Runge–Kutta schemes. Both are implemented in this work. The first approach is attributed to Williamson [170]. Williamson's approach, or 2N schemes, is written for one stage:

$$\begin{aligned}\Delta\mathbf{Q}_i^{(j)} &= \alpha_j \Delta\mathbf{Q}_i^{(j-1)} + \frac{\Delta t_i}{\Omega_i}\mathbf{R}\left(\mathbf{Q}_i^{(j-1)}\right) \\ \mathbf{Q}_i^{(j)} &= \mathbf{Q}_i^{(j-1)} + \beta_j \Delta\mathbf{Q}_i^{(j)}\end{aligned} \tag{5.181}$$

where $\Delta\mathbf{Q}_i^{(j)}$ is the intermediate change in $\mathbf{Q}_i$ which now also needs to be stored. α and β are related to the Butcher coefficients. The second approach is Wray's van der Houwen scheme [89]. In Wray's method, or 2R schemes, only two registers are used to achieve the desired accuracy. Over two stages, a 2R scheme is written as:

$$\begin{aligned}\mathbf{Q}_i^{(j+1)} &= \Delta\mathbf{Q}_i^{(j)} + \alpha_{j+1}\frac{\Delta t_i}{\Omega_i}\mathbf{R}\left(\mathbf{Q}_i^{(j)}\right) \\ \Delta\mathbf{Q}_i^{(j+1)} &= \mathbf{Q}_i^{(j+1)} + (\beta_j - \alpha_{j+1})\frac{\Delta t_i}{\Omega_i}\mathbf{R}\left(\mathbf{Q}_i^{(j)}\right) \\ \mathbf{Q}_i^{(j+2)} &= \Delta\mathbf{Q}_i^{(j+1)} + \alpha_{j+2}\frac{\Delta t_i}{\Omega_i}\mathbf{R}\left(\mathbf{Q}_i^{(j+1)}\right) \\ \Delta\mathbf{Q}_i^{(j+2)} &= \mathbf{Q}_i^{(j+2)} + (\beta_{j+1} - \alpha_{j+2})\frac{\Delta t_i}{\Omega_i}\mathbf{R}\left(\mathbf{Q}_i^{(j+1)}\right)\end{aligned} \tag{5.182}$$

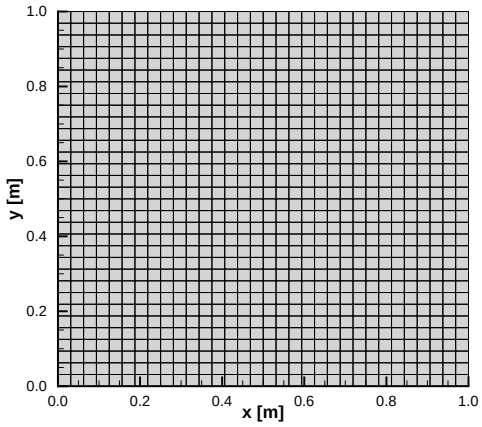
In the 2R schemes, the previous solution $\mathbf{Q}_i^{()}$ is written over, and $\Delta\mathbf{Q}_i^{()}$ is used to transfer information from one stage to another. The overwrite does not have to take place, and in some cases is not desired, but the resultant scheme has different coefficients than (5.181). α and β in (5.182) are also related to the Butcher coefficients. Specifics on both 2N and 2R schemes, as well as coefficients α and β, are presented in [19, 89].

6. MOVING LEAST-SQUARES 2D RESULTS

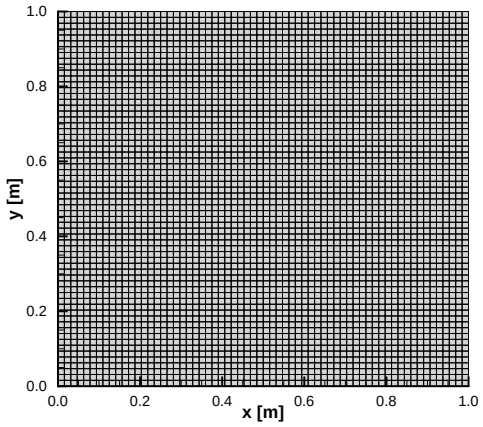
6.1 Introduction

This chapter presents results of the implementation of Moving Least-Squares for computing derivatives. First, a brief study on the methodology used to compute the inverse of the moment matrix $\mathbf{M}$ needed for the MLS basis vector $\mathbf{\Psi}$ in the MLS reconstruction (5.16) is presented. A full study of the effectiveness and accuracy of Moving Least-Squares is performed using three difference mesh topologies. The structured, unstructured, and stretched grids are shown in Figures 6.1, 6.2, and 6.3. All of the meshes are two-dimensional square $[0-1]^2$ domains with node counts of 32^2 to 256^2. The unstructured grids are padded with structured cells to handle boundary terms. The stretched grid spacing in the corners is initially $dx = dy = 0.0625$ and halved each refinement. The spacing is grown at a ratio of 1.3 from the corner. On each set of grids, results for the studies are generated for each of the weighting strategies, unless otherwise noted:

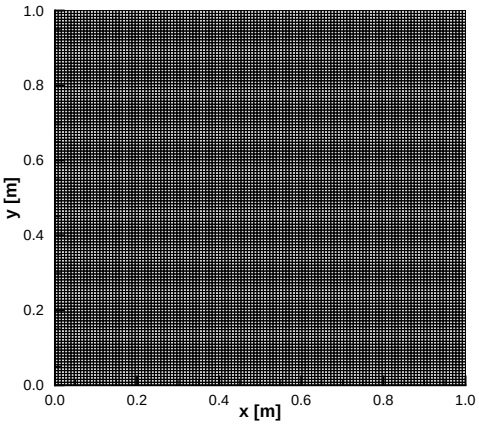
- Isotropic Weights
- Anisotropic Weights
- Affine MLS with Isotropic Weights, also referred to as Affine (1)
- Affine MLS with Anisotropic Weights, also referred to as Affine (2)



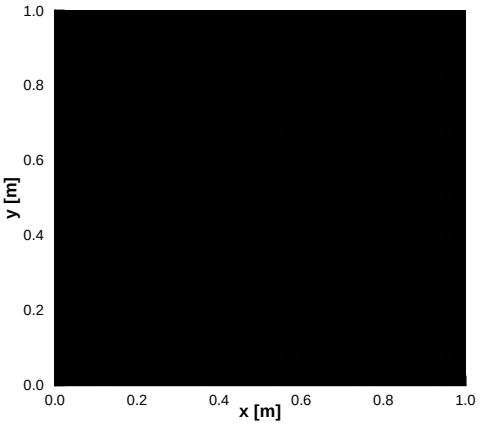
(a) 32x32



(b) 64x64

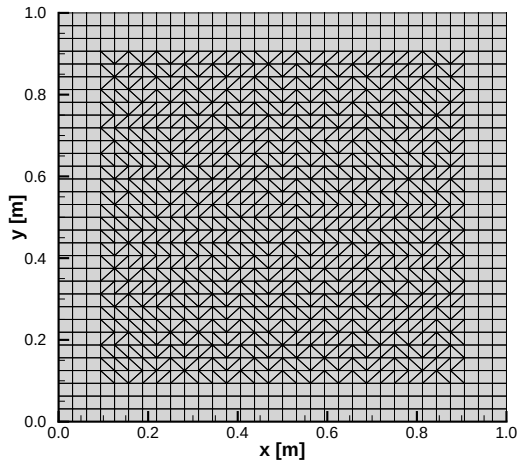


(c) 128x128

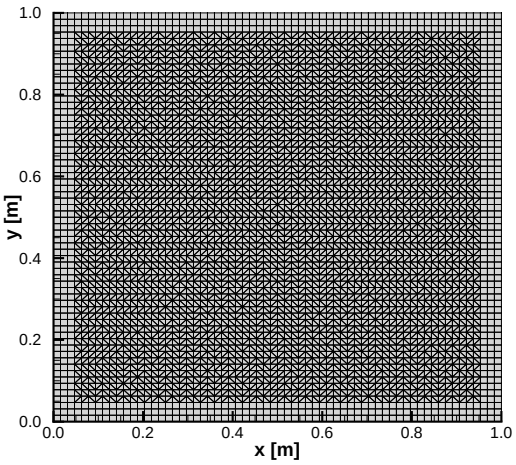


(d) 256x256

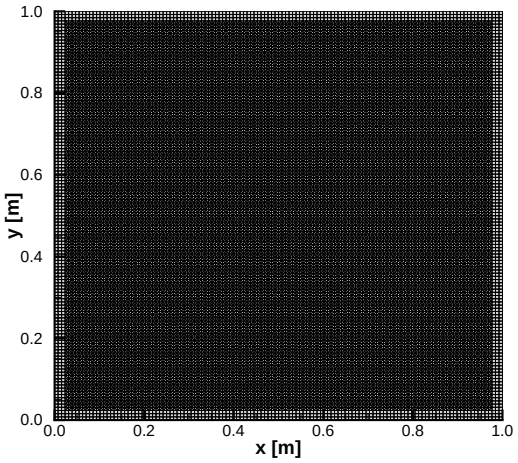
Figure 6.1: Structured Meshes Used for Analysis of MLS.



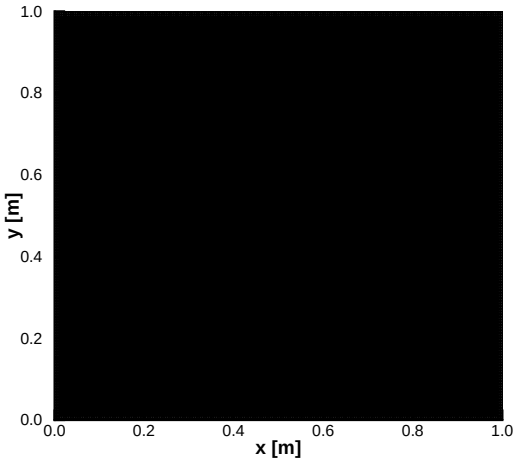
(a) 32x32



(b) 64x64

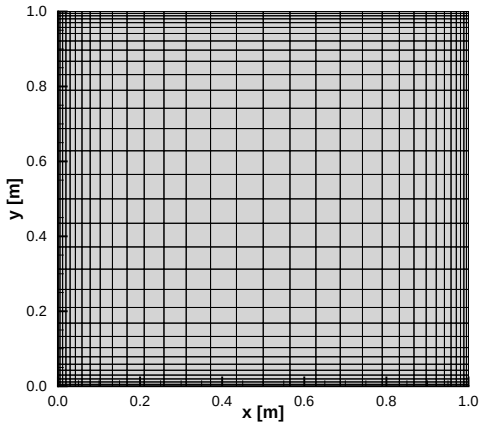


(c) 128x128

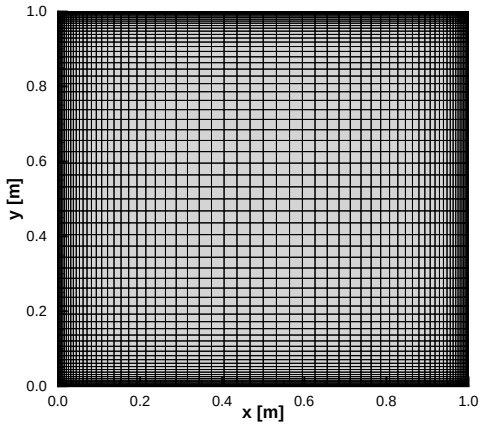


(d) 256x256

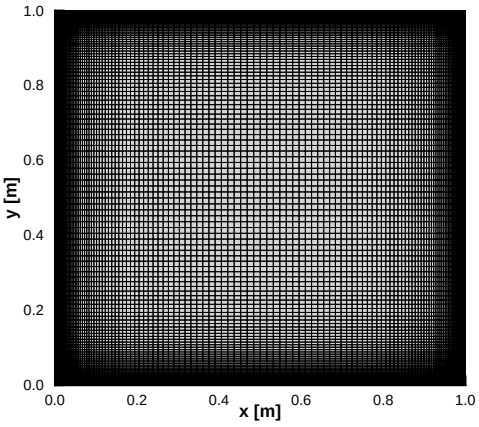
Figure 6.2: Unstructured Meshes Used for Analysis of MLS.



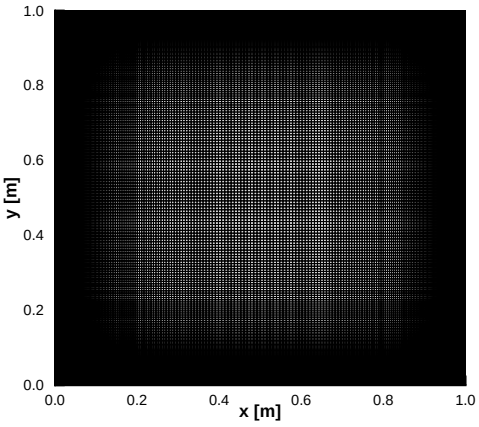
(a) 32x32



(b) 64x64



(c) 128x128



(d) 256x256

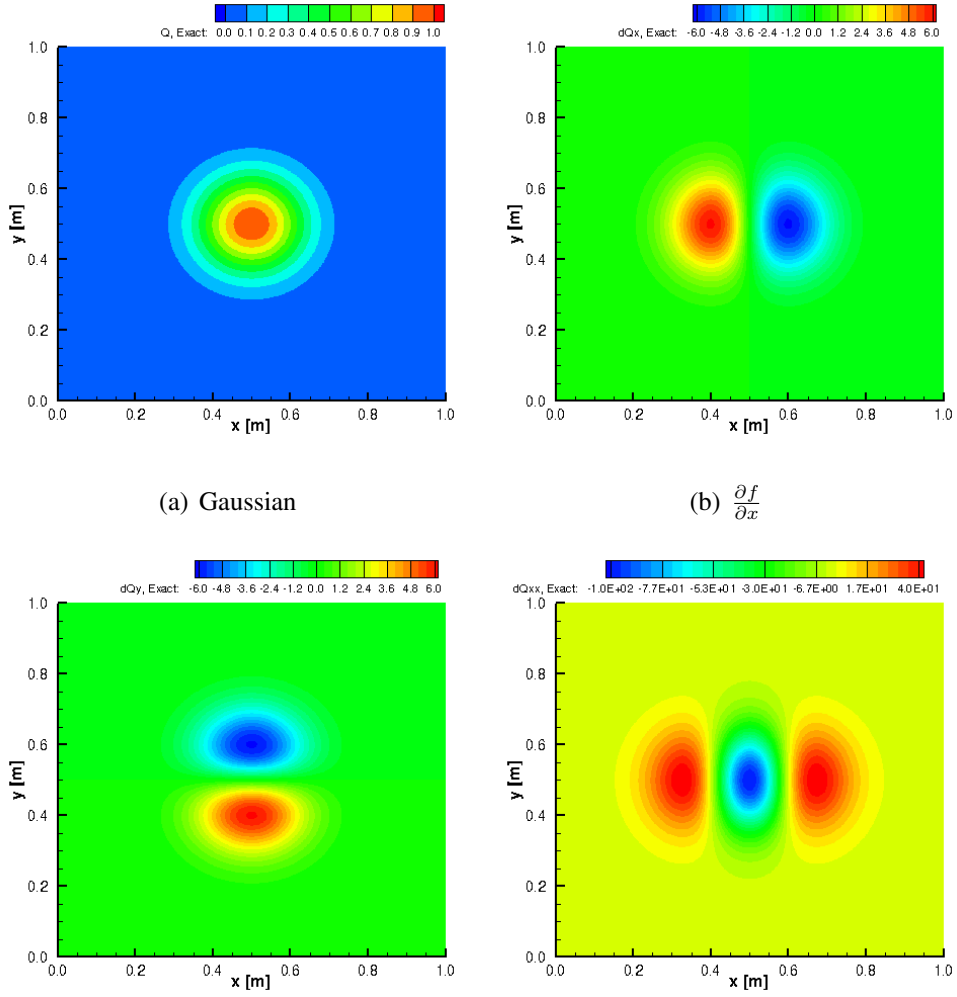
Figure 6.3: Stretched Meshes Used for Analysis of MLS.

Each weighting strategy is tested with four different analytic functions:

- Gaussian, shown with its derivatives in Figures 6.4 and 6.5 :

$$f(x,y) = e^{-\frac{(x-x_o)^2+(y-y_o)^2}{\delta^2}} \tag{6.1}$$

with $\delta^2 = 0.02$ centered at $(x_0, y_0) = (0.5, 0.5)$.



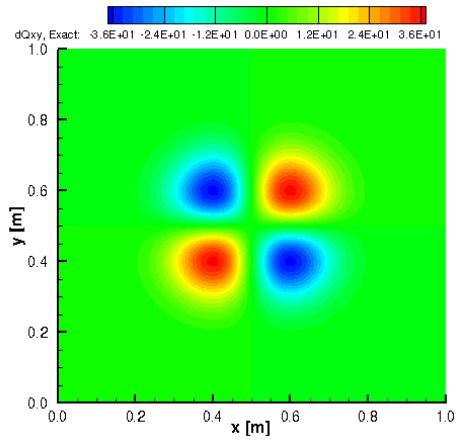
(a) Gaussian

(b) $\frac{\partial f}{\partial x}$

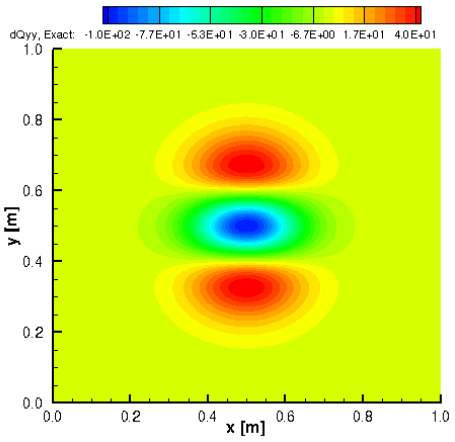
(c) $\frac{\partial f}{\partial y}$

(d) $\frac{\partial^2 f}{\partial x^2}$

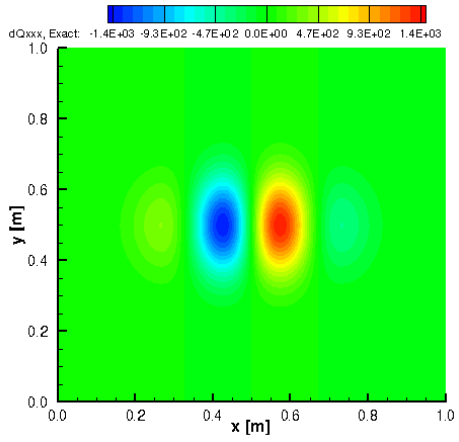
Figure 6.4: Gaussian Function and Some of Its Derivatives.



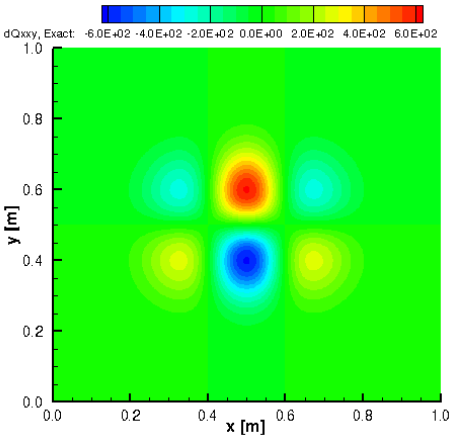
(a) $\frac{\partial^2 f}{\partial xy}$



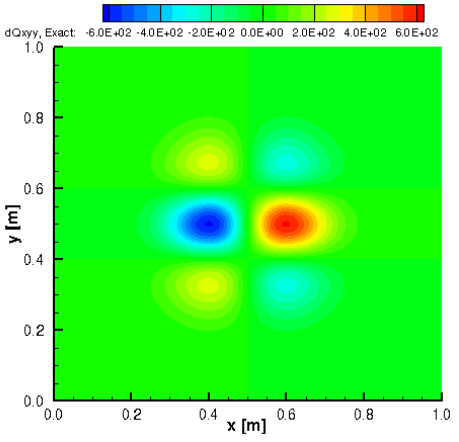
(b) $\frac{\partial^2 f}{\partial y^2}$



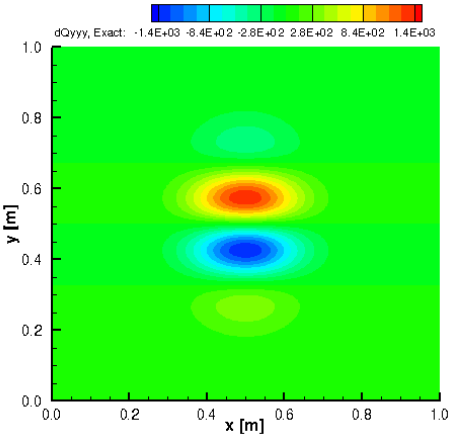
(c) $\frac{\partial^3 f}{\partial x^3}$



(d) $\frac{\partial^3 f}{\partial x^2 y}$



(e) $\frac{\partial^3 f}{\partial xy^2}$

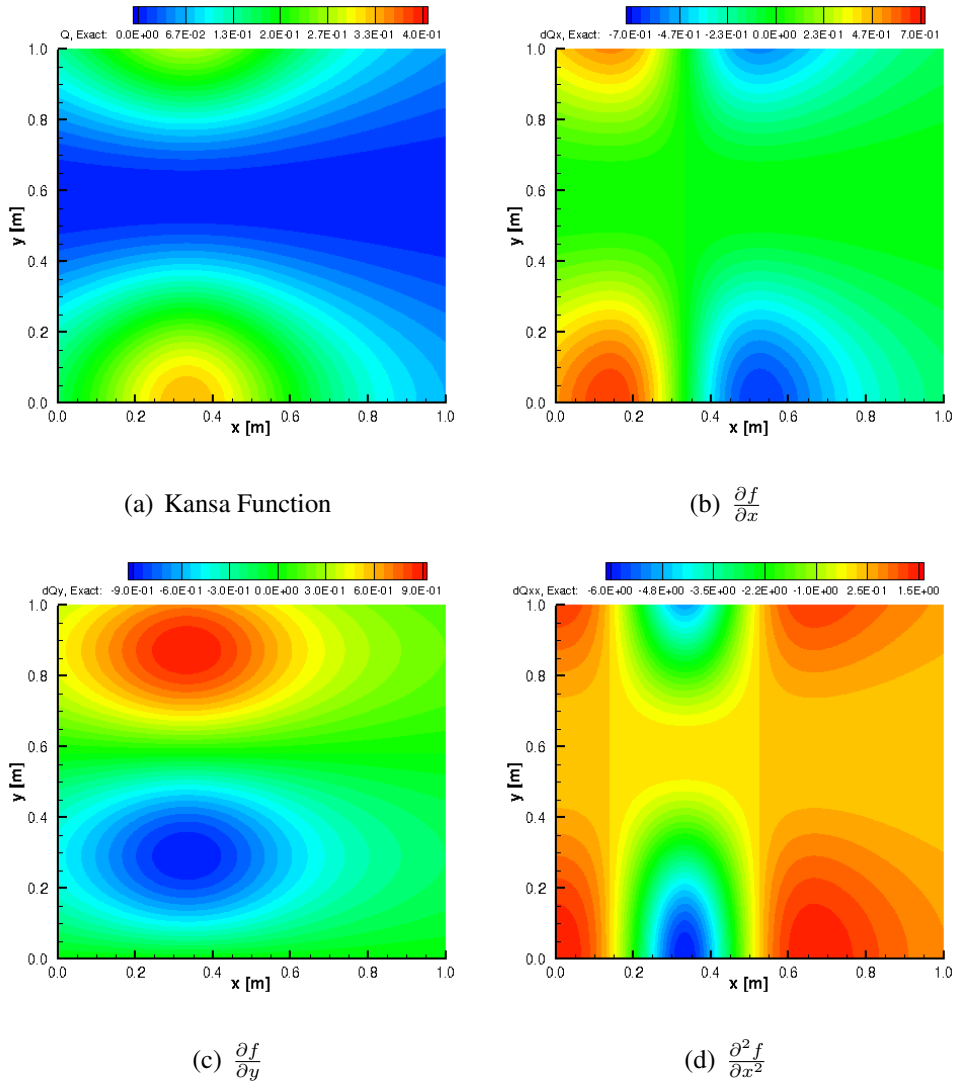


(f) $\frac{\partial^3 f}{\partial y^3}$

Figure 6.5: Additional Derivatives of the Gaussian Function.

- An equation used by [85], shown with its derivatives Figures 6.6 and 6.7:

$$f(x,y) = \frac{\frac{5}{4} + \cos(5.4y)}{6 + 6(3x-1)^2}. \tag{6.2}$$



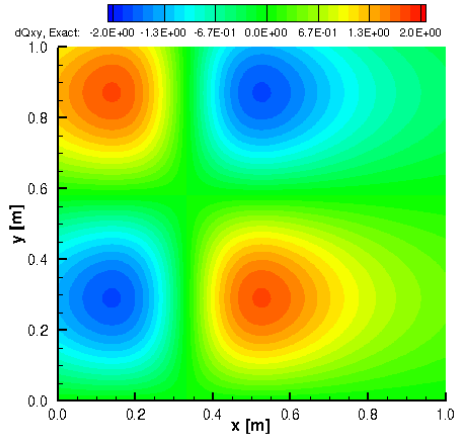
(a) Kansa Function

(b) $\frac{\partial f}{\partial x}$

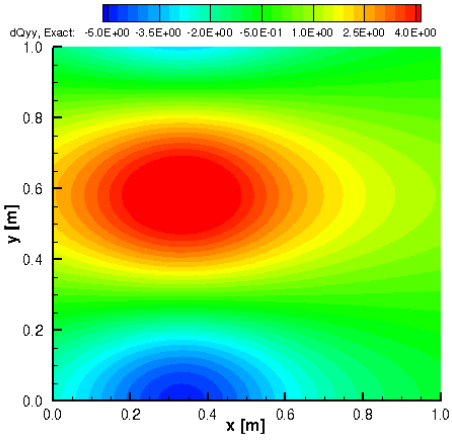
(c) $\frac{\partial f}{\partial y}$

(d) $\frac{\partial^2 f}{\partial x^2}$

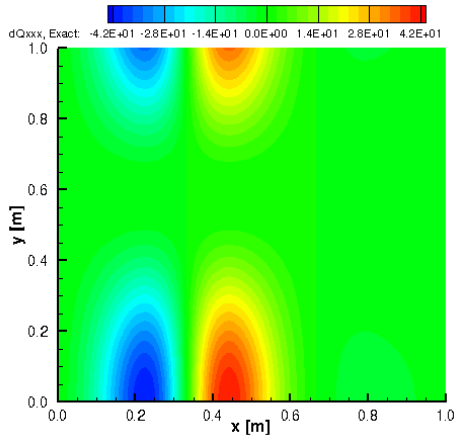
Figure 6.6: Kansa Function and Some of Its Derivatives.



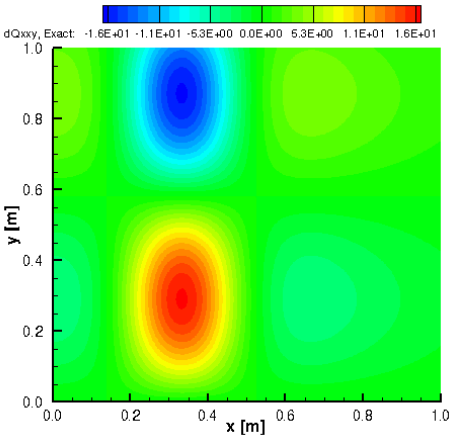
(a) $\frac{\partial^2 f}{\partial xy}$



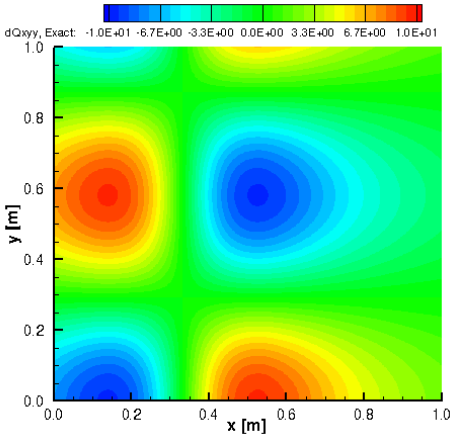
(b) $\frac{\partial^2 f}{\partial y^2}$



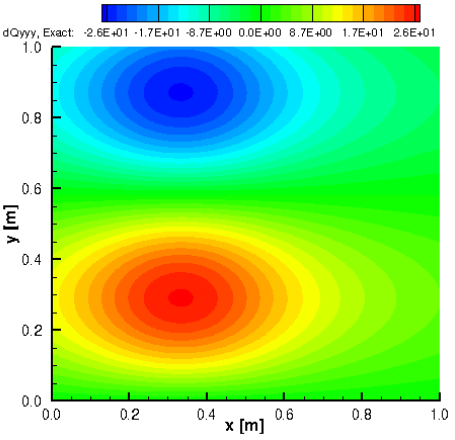
(c) $\frac{\partial^3 f}{\partial x^3}$



(d) $\frac{\partial^3 f}{\partial x^2 y}$



(e) $\frac{\partial^3 f}{\partial xy^2}$

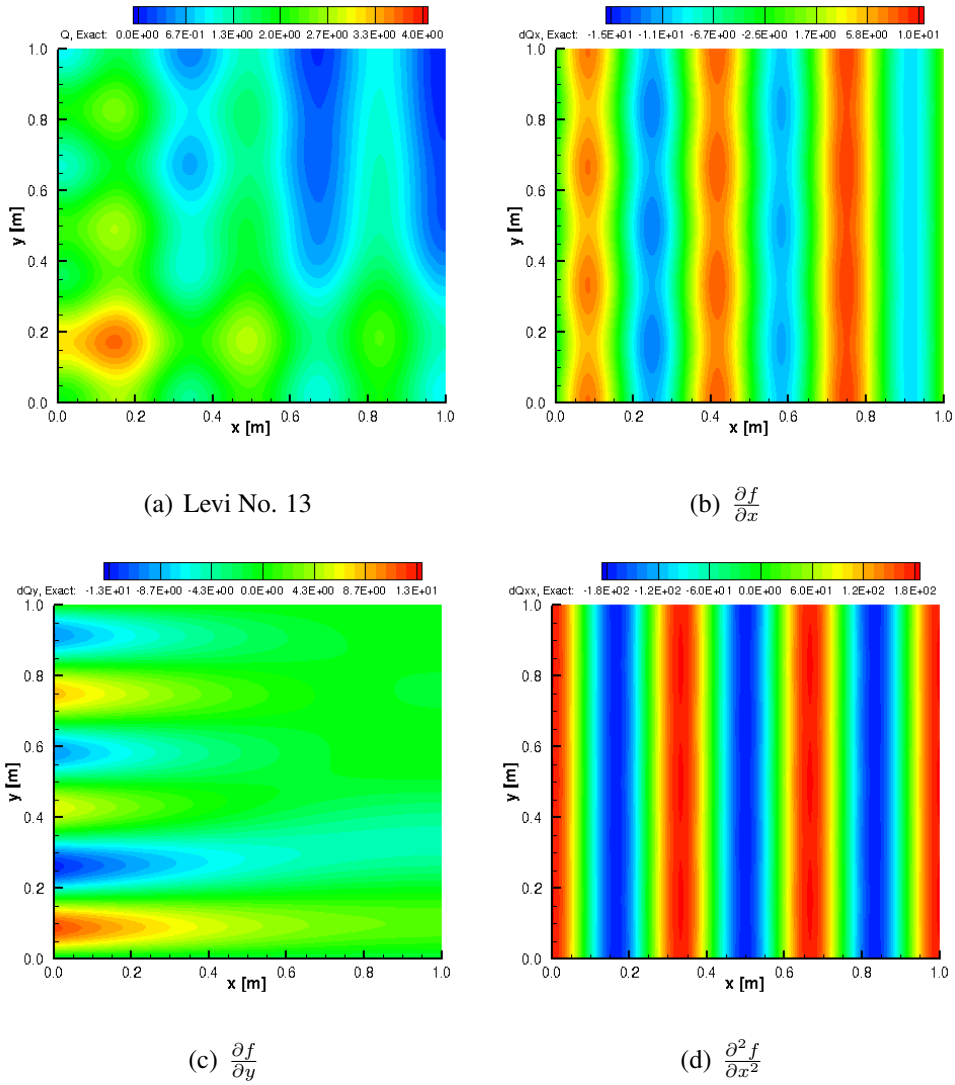


(f) $\frac{\partial^3 f}{\partial y^3}$

Figure 6.7: Additional Derivatives of the Kansa Function.

- Levi No. 13 Function, also referred herein as to Levi, shown with its derivatives in Figures 6.8 and 6.9:

$$f(x,y) = \sin^2(3\pi x) + (x-1)^2(1+\sin^2(3\pi y)) + (y-1)^2(1+\sin^2(2\pi y)) \tag{6.3}$$



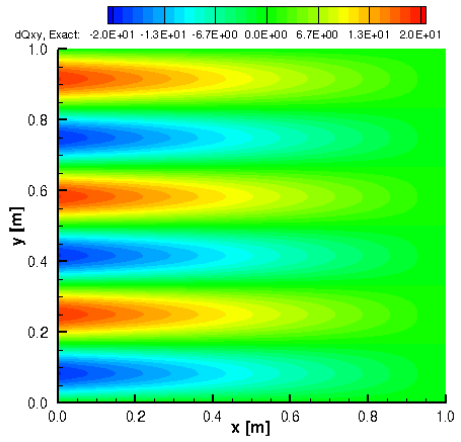
(a) Levi No. 13

(b) $\frac{\partial f}{\partial x}$

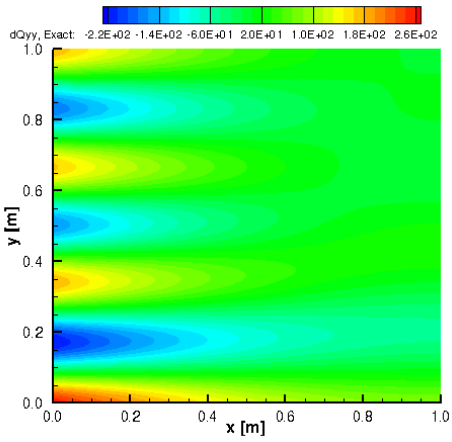
(c) $\frac{\partial f}{\partial y}$

(d) $\frac{\partial^2 f}{\partial x^2}$

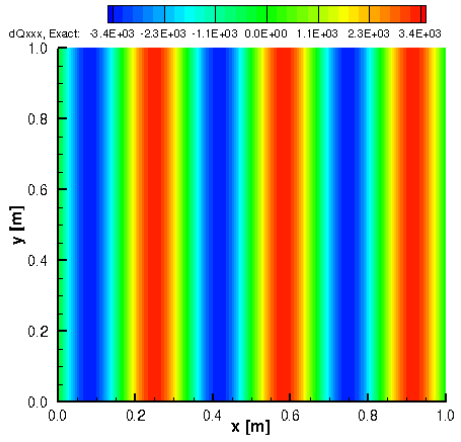
Figure 6.8: Levi No. 13 Function and Some of Its Derivatives.



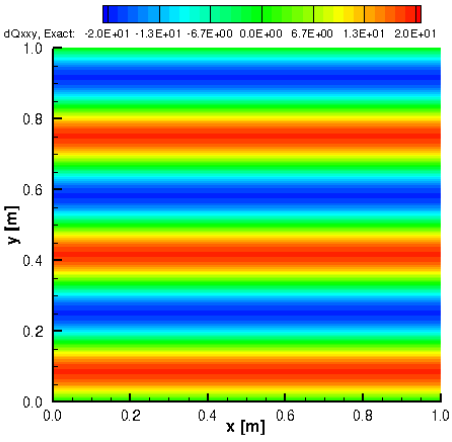
(a) $\frac{\partial^2 f}{\partial xy}$



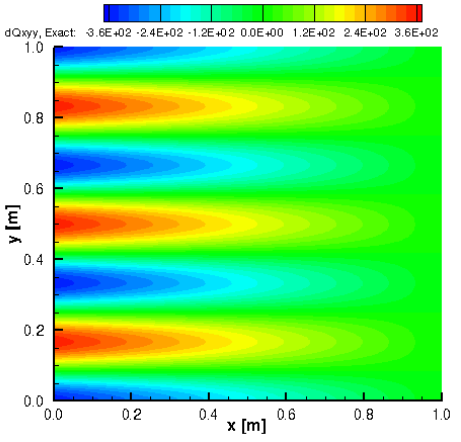
(b) $\frac{\partial^2 f}{\partial y^2}$



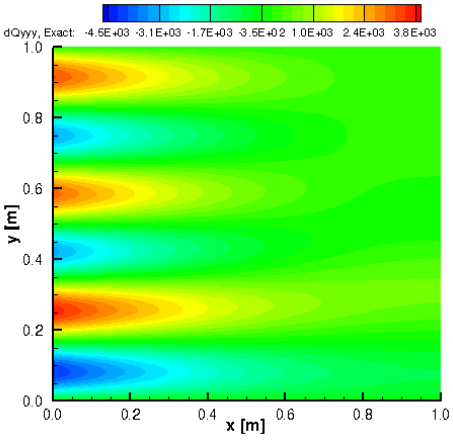
(c) $\frac{\partial^3 f}{\partial x^3}$



(d) $\frac{\partial^3 f}{\partial x^2 y}$



(e) $\frac{\partial^3 f}{\partial xy^2}$

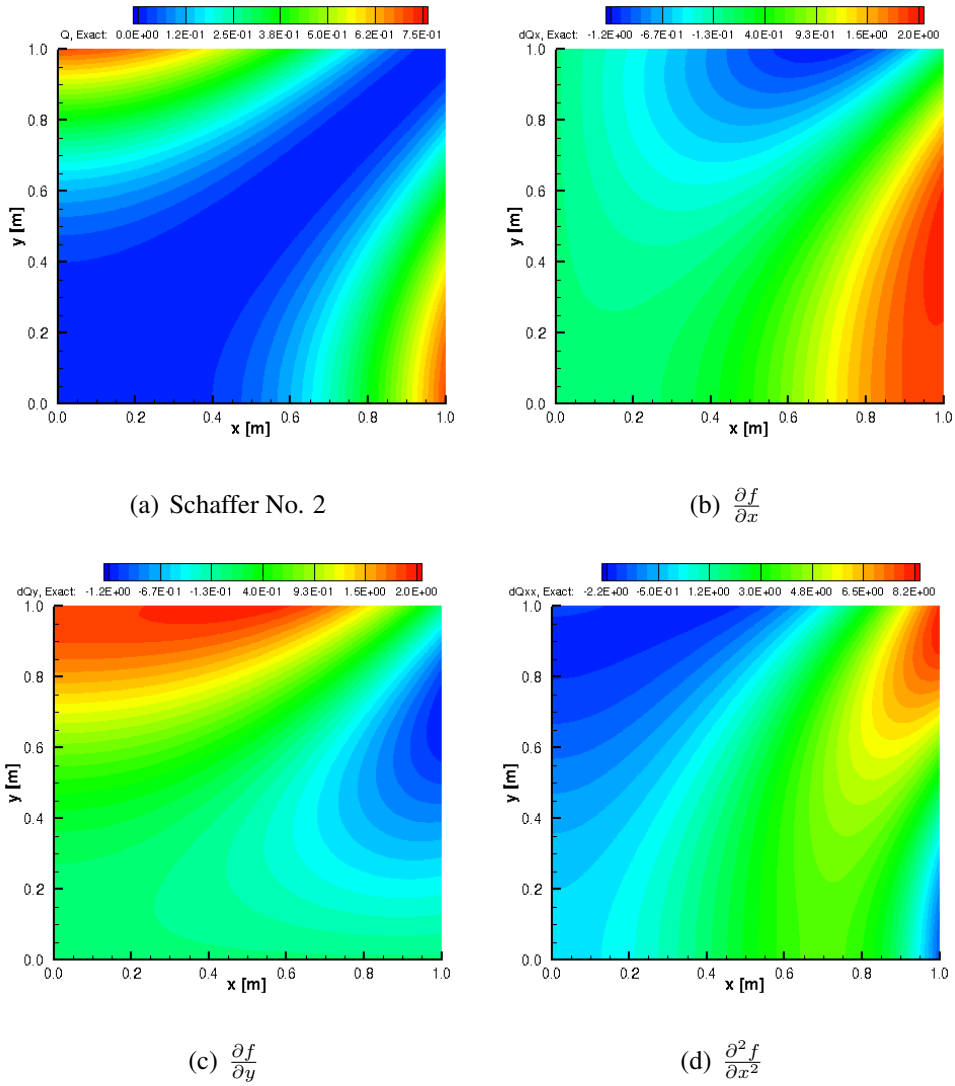


(f) $\frac{\partial^3 f}{\partial y^3}$

Figure 6.9: Additional Derivatives of the Levi No. 13 Function.

- Schaffer No. 2 Function, also herein referred to as Schaffer, shown with its derivatives in Figures 6.10 and 6.11:

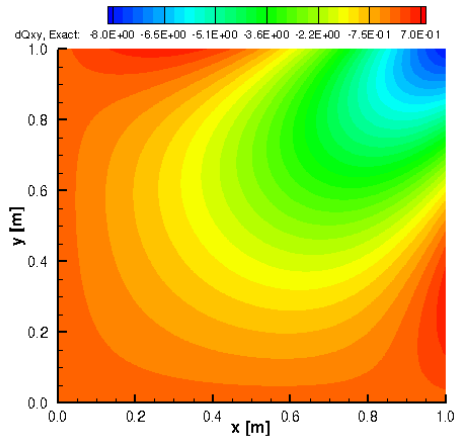
$$f(x,y) = \frac{1}{2} + \frac{\sin^2(x^2 - y^2)}{(1 + 0.001(x^2 + y^2))^2} \tag{6.4}$$



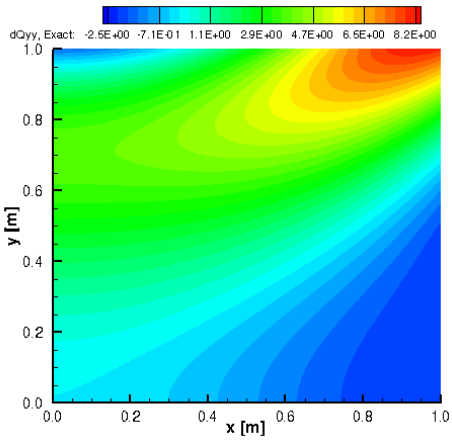
(a) Schaffer No. 2 (b) $\frac{\partial f}{\partial x}$

(c) $\frac{\partial f}{\partial y}$ (d) $\frac{\partial^2 f}{\partial x^2}$

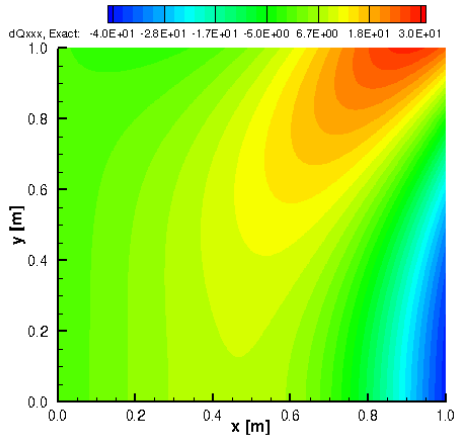
Figure 6.10: Schaffer No. 2 Function and Some of Its Derivatives.



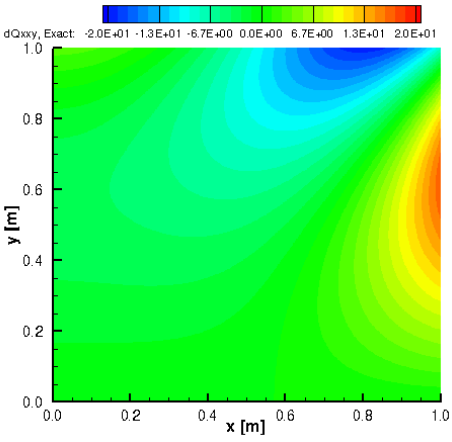
(a) $\frac{\partial^2 f}{\partial xy}$



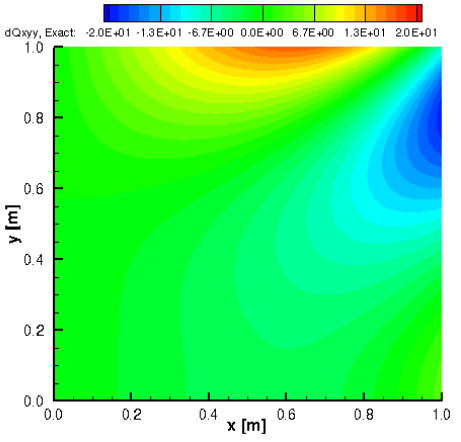
(b) $\frac{\partial^2 f}{\partial y^2}$



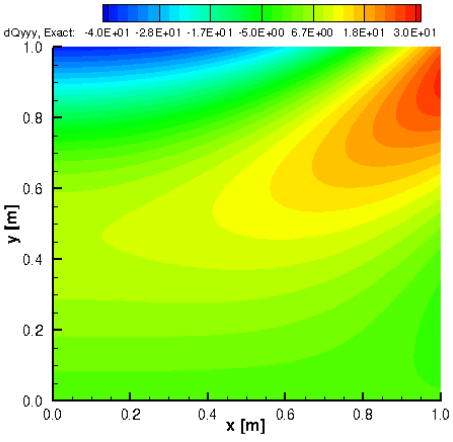
(c) $\frac{\partial^3 f}{\partial x^3}$



(d) $\frac{\partial^3 f}{\partial x^2 y}$



(e) $\frac{\partial^3 f}{\partial xy^2}$



(f) $\frac{\partial^3 f}{\partial y^3}$

Figure 6.11: Additional Derivatives of the Schaffer No. 2 Function.

Both the Levi No. 13 and Schaffer No. 2 functions are commonly used functions to test optimization problems [4]. For each study, the error for the state q and derivatives is computed using the L_2 norm

$$L_2(j) = \sqrt{\frac{1}{n \cdot \max_n \partial_j q} \sum_{i=1}^{n} (\partial_j q - \partial_j q_{analytic})^2} \tag{6.5}$$

where j is the current derivative compared. Next, a study on the accuracy of the MLS diffusive derivatives is performed. A full study on the effect of the scaling parameter k on the MLS reconstruction is done. Finally, the effect of changing the polynomial basis is studied.

6.2 Inverse of Moment Matrix $\mathbf{M}$ for 2D Meshes

This section presents a study on the usage of different inversion methods for the determination of the inverse of the moment matrix $\mathbf{M}$ when computing the MLS basis vector $\mathbf{\Psi}$ needed for the MLS reconstruction (5.16). In Section 5.1.3.2, three methods for computing the inverse of $\mathbf{M}$ were presented:

- General SPD Inverse: $inv(M) = \mathbf{M}^{-1}$
- QR Decomposition: $inv(M) = (\mathbf{R}^T\mathbf{R})^{-1}$
- Pivoting QR Decomposition: $inv(M) = (\mathbf{P}_v\mathbf{R}^T\mathbf{R}\mathbf{P}_v^T)^{-1}$

The general SPD Inverse algorithm used herein is the LAPACK algorithm **dpotri** in conjunction with **dpotrf**, which computes the inverse of a symmetric positive definite matrix via a LU inverse of a Cholesky Decomposed matrix [3]. The SVD method for matrix decomposition was not included, since it was shown in previous studies to be slower than QR decomposition [117] and numerically non-affine (see Section 5.1.3.4 for more details). Rather than looking at the numerical accuracy of the overall MLS method, this study will look at the average condition number of $\mathbf{M}$ on the meshes to be used in later studies herein. The condition number κ is determined using the LAPACK algorithm **dpocon** in conjunction with the $\mathcal{L}_\infty$ matrix norm

$$||A||_\infty = \max_i \sum_{j=1}^{n} |a_{i,j}| \tag{6.6}$$

for the general SPD Inverse and **dtrcon** for the QR methods [3]. The meshes, structured, unstructured, and stretched, were shown in Figures 6.1, 6.2, and 6.3. For each set of grids, the MLS reconstruction from second- to fourth-order is performed. The background equation is not important for $\mathbf{M}$, since it is purely a function of the nodal locations in the stencil $\Omega_{\mathbf{x}_I}$. Only isotropic weights are used in this study, since the other weighting strategies are examined later. The monomial polynomial is scaled as shown in (5.18). Ultimately, this section aims to show the improvement in the condition of $\mathbf{M}$ when using the Pivoting QR method.

6.2.1 Structured Meshes

The average condition number of the second-order reconstruction is presented in Table 6.1 for each of the matrix inversion methods. For the two-dimensional case, the overall condition

Table 6.1: Average Condition Number of $\mathbf{M}$ on Structured Grids: 2^{nd}-Order.

Method	Mesh Size			
	32	64	128	256
General SPD	4.17	4.15	4.15	4.14
QR	2.04	2.04	2.04	2.04
Pivoting QR	2.04	2.04	2.04	2.04

number is low, so no matter the method the inverse of $\mathbf{M}$ should be accurate. As expected, the general SPD method has the largest average condition number. The QR and Pivoting QR method condition numbers are also two times lower than the general SPD method. As the order of the reconstruction is increased, this difference should get larger. It is also interesting to note that the Pivoting QR and QR method have the same condition number, which would imply that the stencils $\Omega_{\mathbf{x}_I}$ do not require any pivoting. Table 6.2 shows the average condition number for the third-order reconstruction. As with the second-order reconstruction, the average condition number for the moment matrix $\mathbf{M}$ is largest for the general SPD method. The condition number for the general SPD method is almost eleven times larger than both the QR and Pivoting QR method. Additionally, the Pivoting QR method again has the same (or nearly) condition number as the general

Table 6.2: Average Condition Number of $\mathbf{M}$ on Structured Grids: 3^{rd}-Order.

Method	Mesh Size			
	32	64	128	256
General SPD	153.6	152.7	152.4	152.4
QR	13.82	13.79	13.79	13.79
Pivoting QR	13.82	13.79	13.79	13.79

QR method, so as with the second-order case, pivoting in not required for these stencils. For third-order, one should start considering using one of the advanced methods for matrix inversion, though the general SPD methods still produce relatively accurate inverses for the condition numbers seen here. Table 6.3 shows the average condition number for the fourth-order reconstruction. For the

Table 6.3: Average Condition Number of $\mathbf{M}$ on Structured Grids: 4^{th}-Order.

Method	Mesh Size			
	32	64	128	256
General SPD	3185	3142	3131	3128
QR	74.07	73.80	73.73	73.71
Pivoting QR	54.17	53.95	53.89	53.88

fourth-order reconstruction, the inverse methods now all have different condition numbers for $\mathbf{M}$. The general method has average condition numbers nearly sixty times larger than the Pivoting QR method, and for these stencils one should use advanced decomposition methods to ensure some level of accuracy. The condition number for the general QR method is about 1.3 times larger than the Pivoting QR method. While the difference is not large, one should use Pivoting QR since it would produce a more accurate inverse. Overall, the results presented in Tables 6.1, 6.2, and 6.3 confirm that the Pivoting QR method should be used since it always produces the lowest average condition number. However, for some systems, the general QR method will produce the same average condition number, though it is difficult to know *a priori* if the stencils used will behave in the manner shown here.

6.2.2 Unstructured Meshes

The average condition number of the second-order reconstruction is presented in Table 6.4 for each of the matrix inversion methods. For the two-dimensional case, the overall condition number

Table 6.4: Average Condition Number of $\mathbf{M}$ on Unstructured Grids: 2^{nd}-Order.

Method	Mesh Size			
	32	64	128	256
General SPD	6.96	7.18	7.43	7.46
QR	2.72	2.85	2.96	2.97
Pivoting QR	2.72	2.85	2.96	2.97

is, as with the structured mesh, low. This again means that any of the decomposition methods will produce accurate inverses of $\mathbf{M}$. As with the structured mesh results of Table 6.1, the general SPD method has the largest average condition number. In this case, the general method has an average condition number 2.5 times larger than the Pivoting QR method. Also, unlike the structured mesh results of Table 6.1, the QR and Pivoting QR method condition numbers are slightly different, with the Pivoting QR method producing a better, though only slightly, average condition number for these meshes. These results should hold for the third-order reconstruction. Table 6.5 shows the average condition number for the third-order reconstruction. As with the second-order

Table 6.5: Average Condition Number of $\mathbf{M}$ on Unstructured Grids: 3^{rd}-Order.

Method	Mesh Size			
	32	64	128	256
General SPD	195.3	196.4	192.6	186.1
QR	15.21	15.09	15.09	14.72
Pivoting QR	14.30	14.20	14.05	13.81

reconstruction, the average condition number for the moment matrix $\mathbf{M}$ is largest for the general

SPD method. The condition number for the general SPD method is approximately fourteen times larger than the Pivoting QR method. Additionally, the Pivoting QR method has a slightly better condition number compared to the general QR method. As with the structured mesh results, one should start considering using one of the advanced methods for matrix inversion, though the general SPD methods still produce relatively accurate inverses for the condition numbers seen here. Table 6.3 shows the average condition number for the fourth-order reconstruction. For the fourth-

Table 6.6: Average Condition Number of $\mathbf{M}$ on Unstructured Grids: 4^{th}-Order.

Method	Mesh Size			
	32	64	128	256
General SPD	3538	3433	3264	3076
QR	70.11	67.67	65.51	63.35
Pivoting QR	54.11	52.37	51.65	49.96

order reconstruction, the general method average condition numbers are approximately sixty-five times larger than the Pivoting QR method, which is similar to the structured mesh results shown in Table 6.3. Again, the results highlight the requirement to use an advanced decomposition method for solving any higher-order least squares problem. The condition number for the general QR method is about 1.3 times larger than the Pivoting QR method. While the difference is not large, one should use Pivoting QR since it would produce a more accurate inverse. Overall, the results presented in Tables 6.4, 6.5, and 6.6 confirm that the Pivoting QR method should be used since it always produces the lowest average condition number.

6.2.3 Stretched Meshes

The average condition number of the second-order reconstruction is presented in Table 6.7 for each of the matrix inversion methods. For the two-dimensional case, the overall condition number is relatively low, but unlike the structured and unstructured mesh results shown in Tables 6.1 and 6.4 the SPD method condition number is on average ten times larger than the QR methods.

Table 6.7: Average Condition Number. of $\mathbf{M}$ on Stretched Grids: 2^{nd}-Order.

Method	Mesh Size			
	32	64	128	256
General SPD	80.43	67.29	60.65	57.33
QR	7.11	6.51	6.20	6.05
Pivoting QR	7.11	6.51	6.20	6.05

The QR and Pivoting QR method condition numbers are the same for these meshes. These results should hold for the third-order reconstruction. Table 6.8 shows the average condition number for the third-order reconstruction. As with the second-order reconstruction, the average condition

Table 6.8: Average Condition Number of $\mathbf{M}$ on Stretched Grids: 3^{rd}-Order.

Method	Mesh Size			
	32	64	128	256
General SPD	36851	32714	28391	25620
QR	135.8	122.3	111.9	105.7
Pivoting QR	118.0	116.2	97.26	92.00

number for the moment matrix $\mathbf{M}$ is largest for the general SPD method. For the stretched grids, the condition number for the general SPD method is over 300 times larger than the Pivoting QR method. Additionally, the Pivoting QR method has a slightly better condition number compared to the general QR method. Unlike the structured and unstructured meshes, for third-order, one should never use a general matrix method for inversion due to the conditioning of $\mathbf{M}$. Table 6.3 shows the average condition number for the fourth-order reconstruction. For the fourth-order reconstruction, the general method average condition numbers are nearly 8000 times larger than the Pivoting QR method. Any results with the general SPD method are worthless at these high condition numbers. The user should only use the advanced matrix decomposition methods for fourth-order reconstructions. For the advanced methods, the condition number for the general QR method is about 1.2 times larger than the Pivoting QR method. While the difference is not large,

Table 6.9: Average Condition Number of $\mathbf{M}$ on Stretched Grids: 4^{th}-Order.

Method	Mesh Size			
	32	64	128	256
General SPD	$1.61E7$	$2.15E7$	$2.10E7$	$1.92E7$
QR	2450	2502	2325	2166
Pivoting QR	2028	2082	1938	1806

one should use Pivoting QR since it would produce a more accurate inverse. Overall, the results presented in Tables 6.7, 6.8, and 6.9 confirm that the Pivoting QR method should be used since it always produces the lowest average condition number.

6.3 Order of Accuracy Study

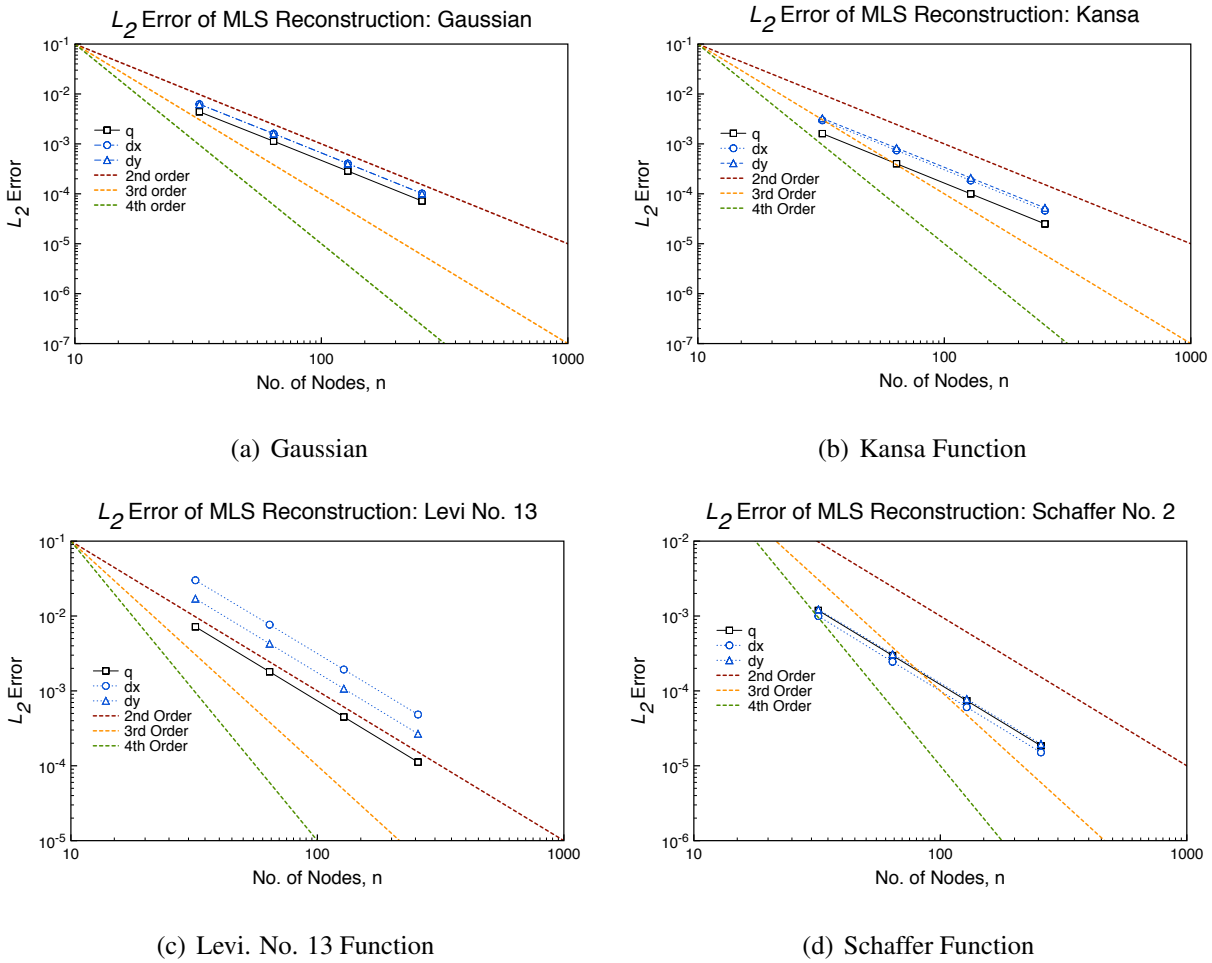
This section performs a full assessment of the order of accuracy of the MLS reconstruction method on the structured, unstructured, and stretched grids shown in Figures 6.1- 6.3. The Gaussian (6.1), Kansa (6.2), Levi No. 13 (6.3), and Schaffer (6.4) functions are analyzed on these grids. All of the weighting strategies are examined and compared to determine how changing the weighting strategy affects the accuracy and overall error of the MLS reconstruction. The variation of the weighting strategies is shown at the end of each mesh section. The scaling parameter in this section is $k = 0.8$, and the polynomial basis used is the standard monomial basis. The inversion method of choice for this study is the Pivoting QR method, since it always produced the minimum condition number for $\mathbf{M}$ for each of the grids, as shown in Section 6.2.

6.3.1 Structured Grids

This section presents the results of order of accuracy of the MLS derivatives on structured grids. The field is reconstructed using the isotropic, anisotropic weighting strategies as well as the Affine MLS with isotropic weights. This section concludes with a comparison study for each function and each weighting strategy.

6.3.1.1 *Isotropic Weights*

This subsection presents the results for MLS reconstructions on structured grids using an isotropic weighting scheme. The second-order results using isotropic weights are shown in Figure 6.12.



(a) Gaussian

(b) Kansa Function

(c) Levi. No. 13 Function

(d) Schaffer Function

Figure 6.12: Error Norms on Structured Grids: 2^{nd}-Order MLS with Isotropic Weights.

Figure 6.12 shows that the function and first derivative achieve approximately second-order accuracy. Table 6.10 summarizes the computed order of accuracy from the data. There also appears to be no significant change in the order of accuracy based on which function is used.

Table 6.10: Accuracy on Structured Grids: 2^{nd}-Order MLS with Isotropic Weights.

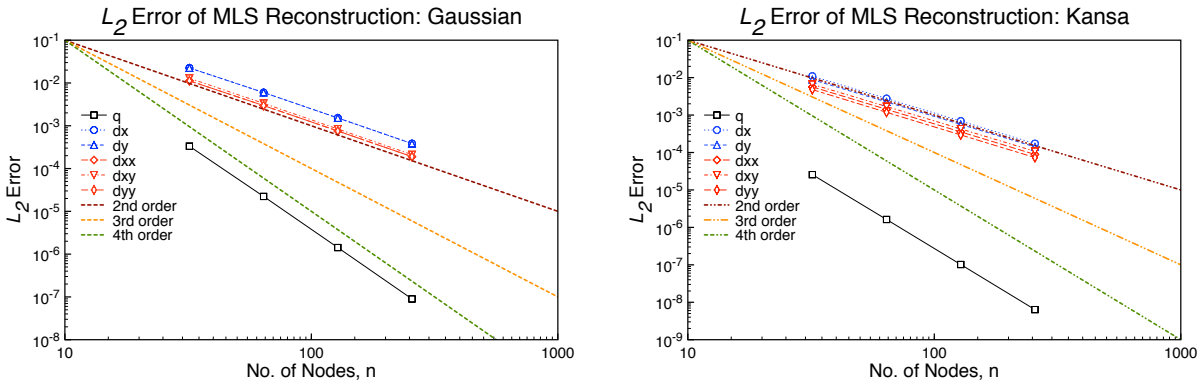
Equation	Function		
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
Gaussian	1.98	1.98	1.98
Kansa	2.00	2.00	1.99
Levi	2.00	1.99	2.00
Schaffer	2.00	2.02	1.99

Third-order results are shown in Figure 6.13. Figure 6.13 shows that the function achieves approximately fourth-order accuracy and the first- and second-derivative achieve approximately second-order accuracy. Table 6.11 summarizes the computed order of accuracy for each equation on structured grids using isotropic weights. The best accuracy is achieved when reconstructing the Schaffer function and worse for the Gaussian.

Table 6.11: Accuracy on Structured Grids: 3^{rd}-Order MLS with Isotropic Weights.

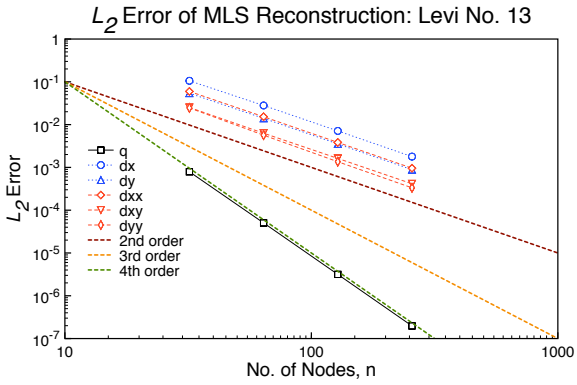
Equation	Function					
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
Gaussian	3.95	1.96	1.96	1.96	1.98	1.96
Kansa	3.99	1.99	1.99	2.00	1.98	2.01
Levi	3.99	1.96	1.98	1.99	1.97	2.07
Schaffer	3.98	2.02	2.01	2.00	2.00	1.97

Fourth-order results are shown in Figure 6.14. Figure 6.14 shows that the function and first-derivatives achieve approximately fourth-order accuracy, while the second- and third-derivatives

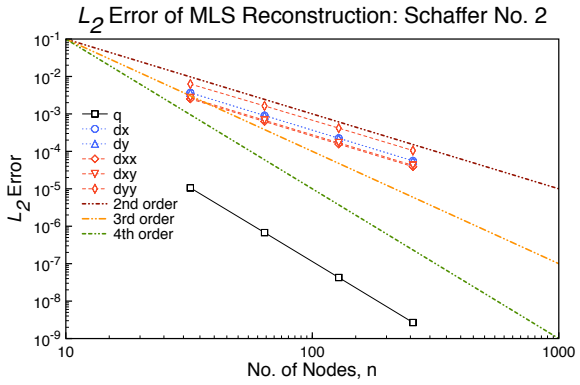


(a) Gaussian

(b) Kansa Function



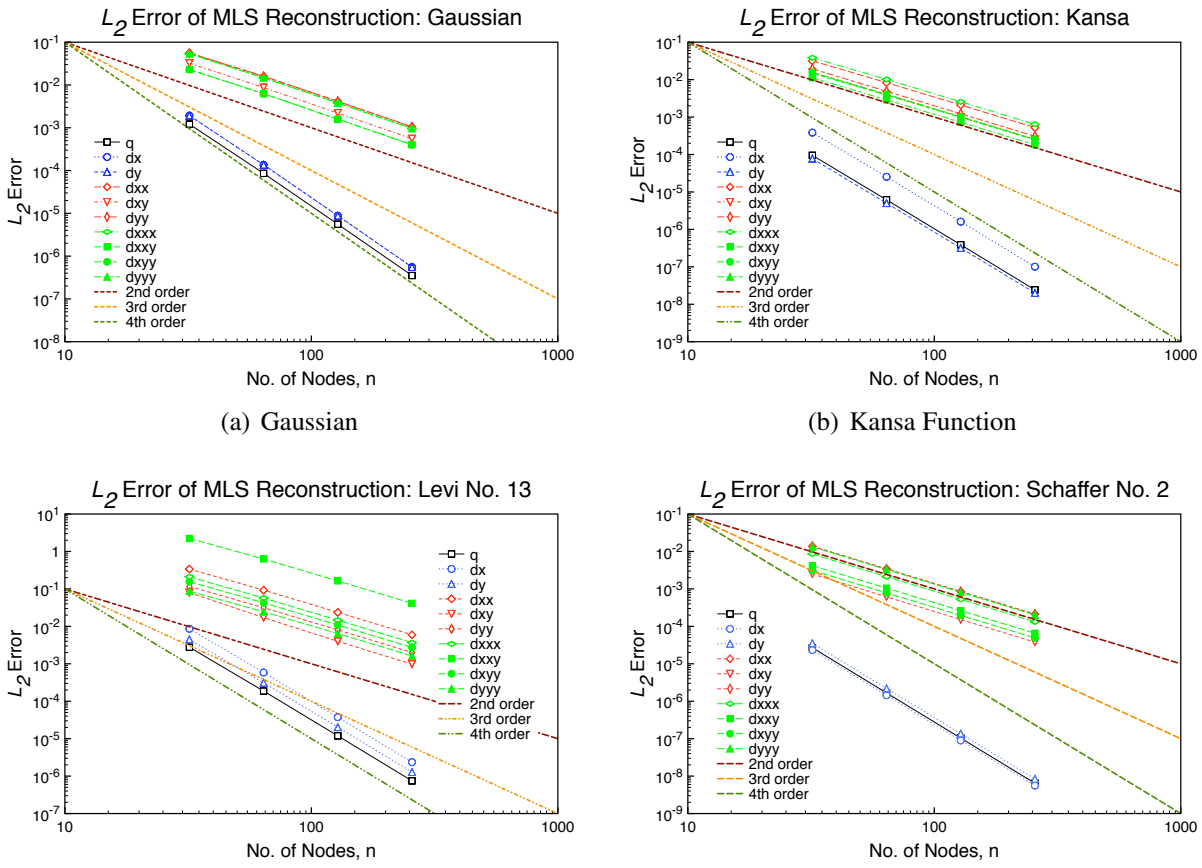
(c) Levi Function



(d) Schaffer Function

Figure 6.13: Error Norms on Structured Grids: 3^{rd}-Order MLS with Isotropic Weights.

are approximately second-order accurate. Table 6.12 summarizes the computed order of accuracy for each equation on structured grids using isotropic weights. As with the third-order solution, the reconstruction performs best with the Schaffer function and worse with the Gaussian.



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

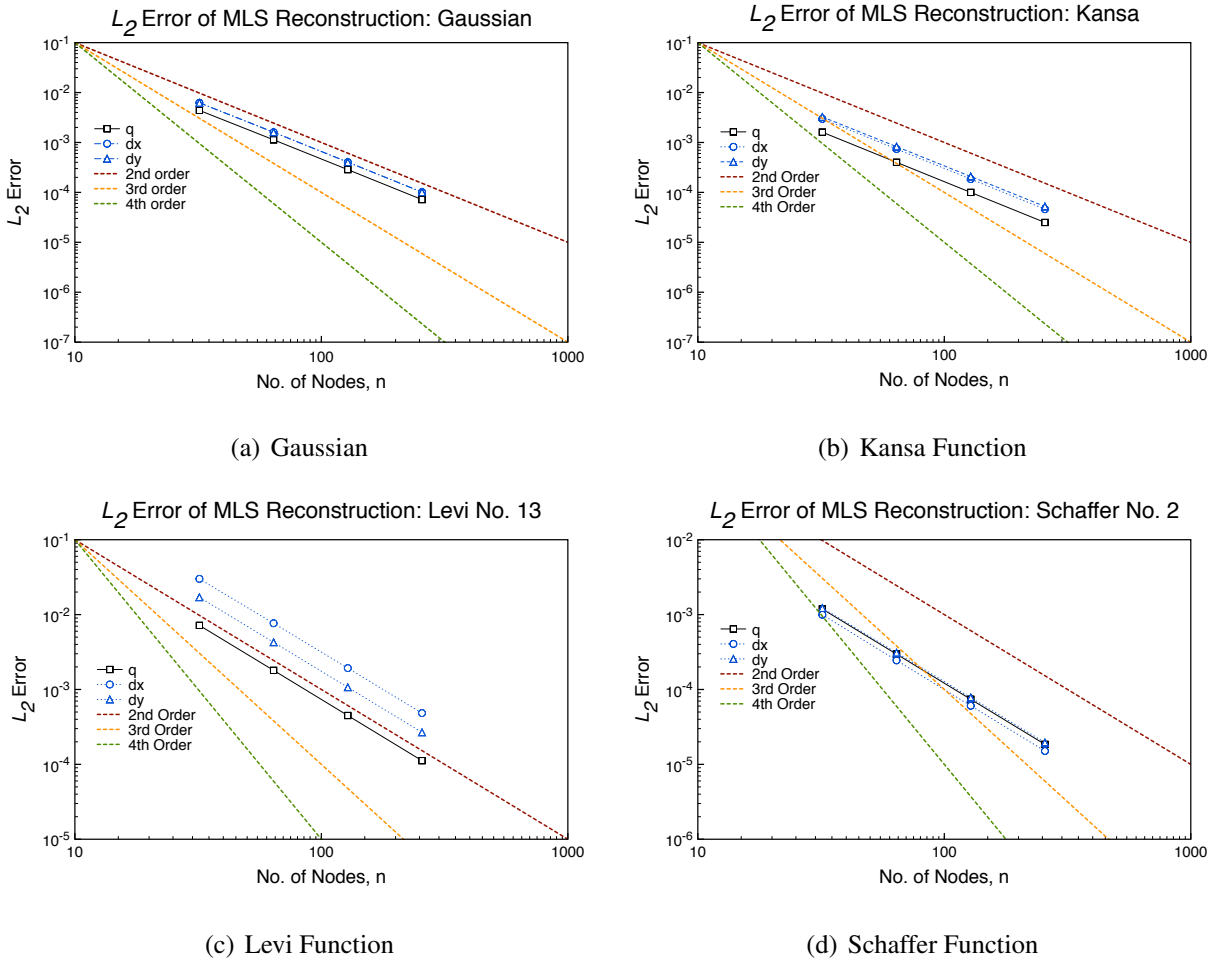
Figure 6.14: Error Norms on Structured Grids: 4^{th}-Order MLS with Isotropic Weights.

Table 6.12: Accuracy on Structured Grids: 4^{th}-Order MLS with Isotropic Weights.

Equation	Function									
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial xy^2}$	$\frac{\partial^3 q}{\partial y^3}$
Gaussian	3.92	3.92	3.92	1.91	1.95	1.91	1.93	1.95	1.95	1.93
Kansa	3.98	3.97	3.97	1.97	1.98	1.99	1.97	1.98	1.98	1.97
Levi	3.95	3.95	3.92	1.94	2.12	1.95	1.95	1.92	1.95	1.92
Schaffer	4.01	4.01	4.01	2.01	2.02	2.01	2.01	2.00	1.99	2.01

6.3.1.2 *Anisotropic Weights*

This subsection presents the results for MLS reconstructions on structured grids using an anisotropic weighting scheme. The second-order results using anisotropic weights are shown in Figure 6.15.



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.15: Error Norms on Structured Grids: 2^{nd}-Order MLS with Anisotropic Weights.

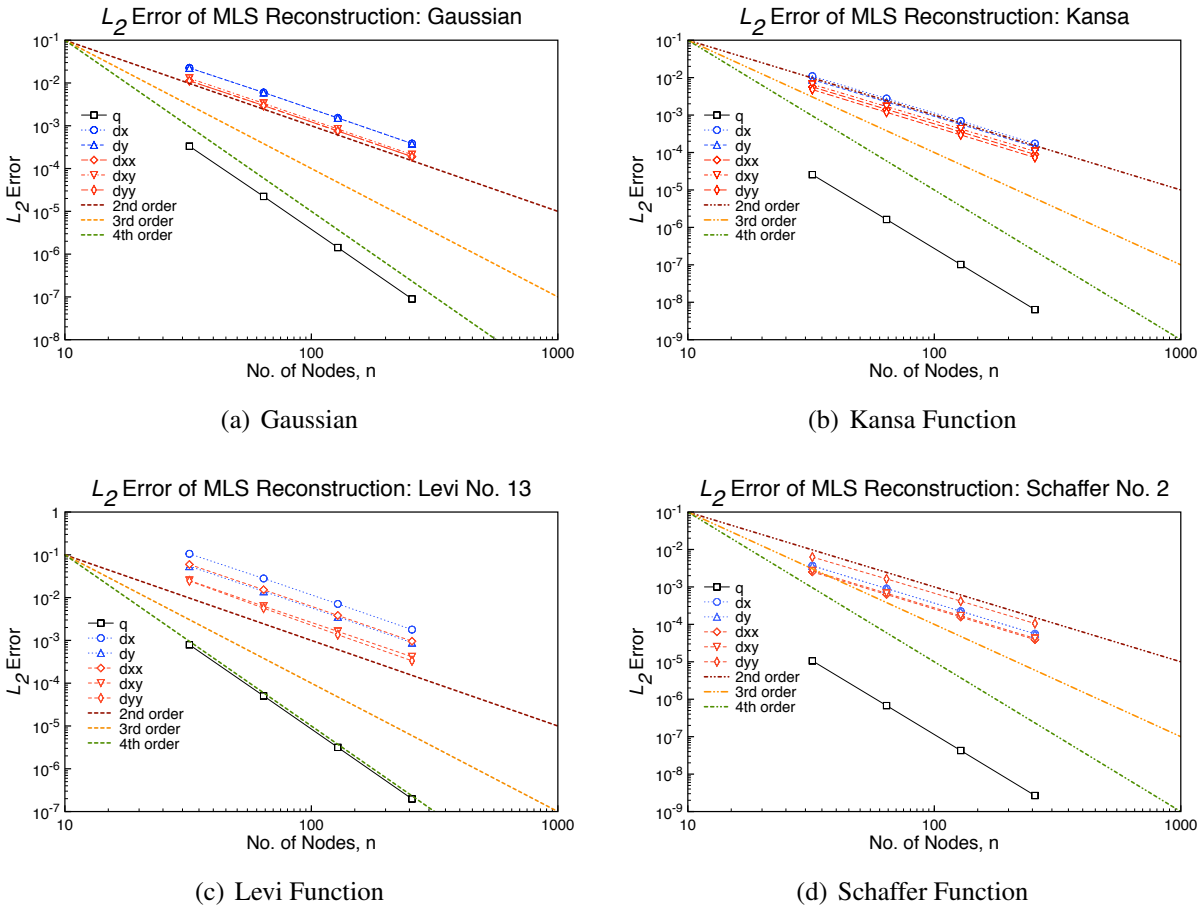
Figure 6.15 shows that the function and first derivative achieve approximately second-order accuracy. Table 6.13 summarizes the computed order of accuracy for each equation on structured grids using anisotropic weights. The results are same as those produced by the isotropic weights

presented in Table 6.10, as one would expect since the anisotropic weight would degenerate to the isotropic case for these grids. As with the isotropic case, the reconstruction performs similar for all the equations.

Table 6.13: Accuracy on Structured Grids: 2^{nd}-Order MLS with Anisotropic Weights.

	Function		
Equation	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
Gaussian	1.98	1.98	1.98
Kansa	2.00	2.00	1.99
Levi	2.00	1.99	2.00
Schaffer	2.00	2.02	1.99

Third-order results are shown in Figure 6.16. Figure 6.16 shows that the function achieves approximately fourth-order accuracy and the first- and second-derivative achieve approximately second-order accuracy. Table 6.14 summarizes the computed order of accuracy for each equation on structured grids using anisotropic weights. As with the second-order reconstruction, the third-order reconstruction with anisotropic weights produces the same level of accuracy as the isotropic weighted MLS shown in Table 6.11. The reconstruction performs the best for the Schaffer function, and the reconstruction is worse for the Gaussian.



(a) Gaussian

(b) Kansa Function

(c) Levi Function

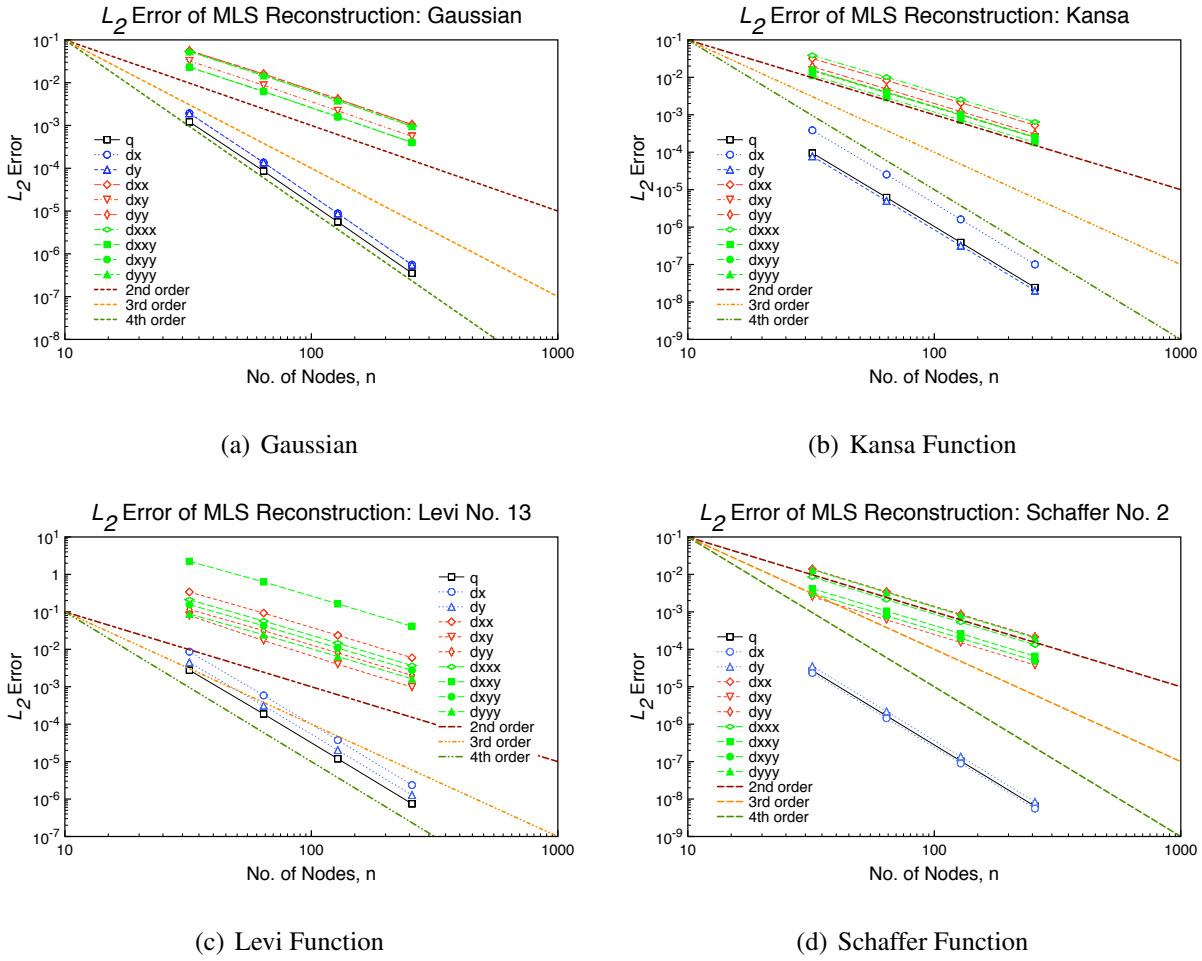
(d) Schaffer Function

Figure 6.16: Error Norms on Structured Grids: 3^{rd}-Order MLS with Anisotropic Weights.

Table 6.14: Accuracy on Structured Grids: 3^{rd}-Order MLS with Anisotropic Weights.

	Function					
Equation	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial x y}$	$\frac{\partial^2 q}{\partial y^2}$
Gaussian	3.95	1.96	1.96	1.96	1.98	1.96
Kansa	3.99	1.99	1.99	2.00	1.98	2.01
Levi	3.99	1.96	1.98	1.99	1.97	2.07
Schaffer	3.98	2.02	2.01	2.00	2.00	1.97

Fourth-order results are shown in Figure 6.17.



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.17: Error Norms on Structured Grids: 4^{th}-Order MLS with Anisotropic Weights.

Figure 6.17 shows that the function and first-derivatives achieve approximately fourth-order accuracy, while the second- and third-derivatives are approximately second-order accurate. Table 6.15 summarizes the computed order of accuracy for each equation on structured grids using isotropic weights. As with the second- and third-order reconstruction, the fourth-order reconstruction with anisotropic weights produces the same level of accuracy as the isotropic weighted MLS shown in Table 6.12. The reconstruction is best with the Schaffer No. 2 function, and it performs

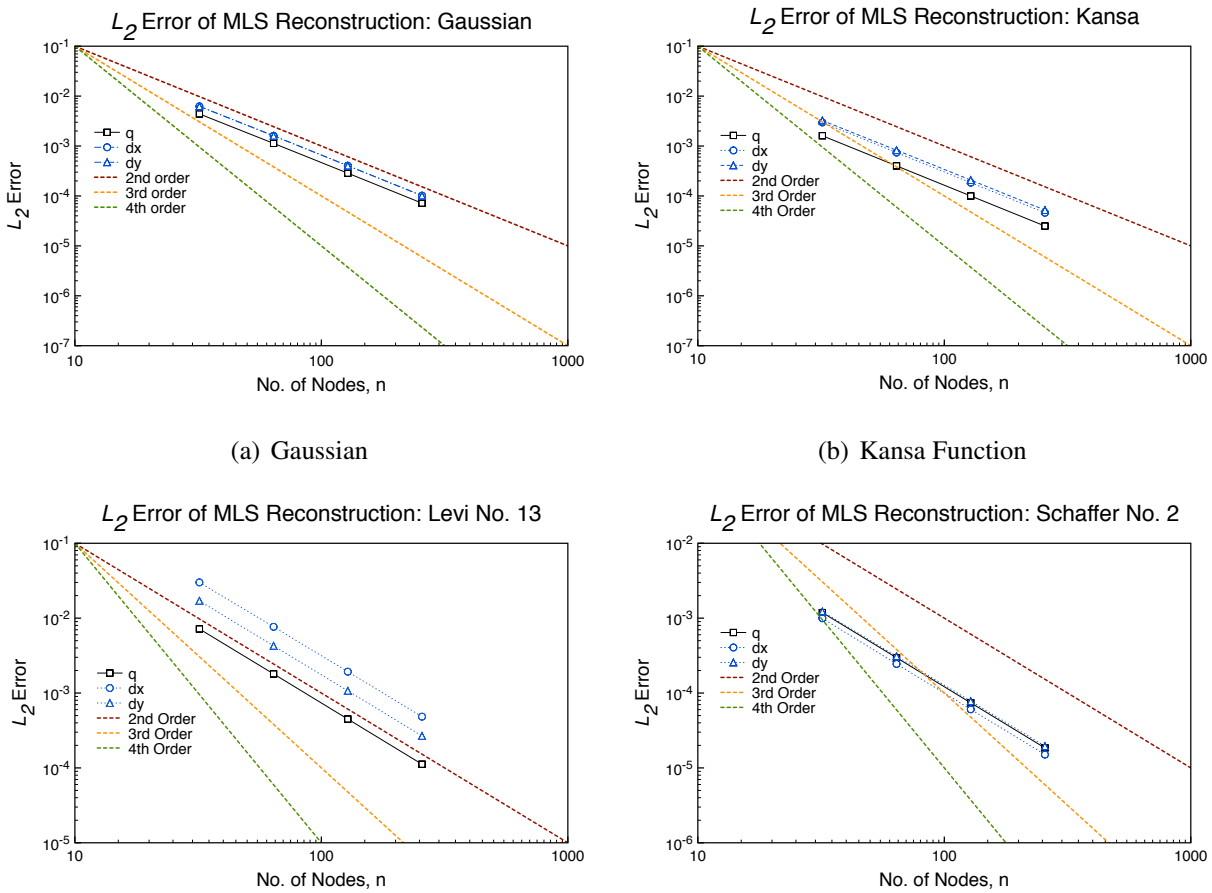
worse for the Gaussian.

Table 6.15: Accuracy on Structured Grids: 4^{th}-Order MLS with Anisotropic Weights.

Equation	Function									
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial xy^2}$	$\frac{\partial^3 q}{\partial y^3}$
Gaussian	3.92	3.92	3.92	1.91	1.95	1.91	1.93	1.95	1.95	1.93
Kansa	3.98	3.97	3.97	1.97	1.98	1.99	1.97	1.98	1.98	1.97
Levi	3.95	3.95	3.92	1.94	2.12	1.95	1.95	1.92	1.95	1.92
Schaffer	4.01	4.01	4.01	2.01	2.02	2.01	2.01	2.00	1.99	2.01

6.3.1.3 Affine MLS

This subsection presents the results for Affine MLS reconstructions on structured grids using an isotropic weighting scheme. The second-order results using Affine MLS with isotropic weights are shown in Figure 6.18. Note that anisotropic weights are not used for the structured mesh test, since they degenerate to the isotropic case as shown in the previous section.



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.18: Error Norms on Structured Grids: 2^{nd}-Order Affine MLS with Isotropic Weights

Figure 6.18 shows that the reconstruction and first-derivatives achieve approximately second-order accuracy. Table 6.16 summarizes the computed order of accuracy for each equation on structured grids using Affine MLS with isotropic weights. As with the previous two methods, the Affine MLS produces second-order results for both the function and first derivatives. Affine MLS also produces the same results as general method using isotropic and anisotropic method as presented in Tables 6.10 and 6.13. As with the previous two sections, the reconstruction performs similarly for all equations.

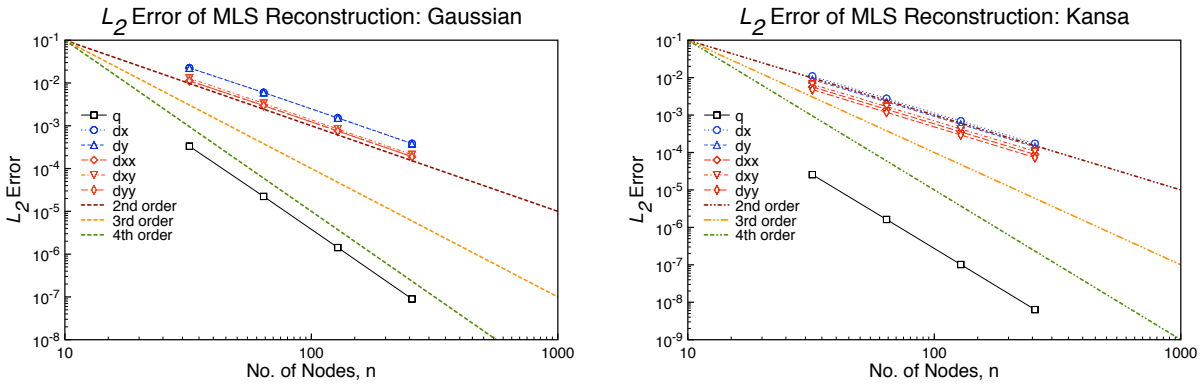
Table 6.16: Accuracy on Structured Grids: 2^{nd}-Order Affine MLS with Isotropic Weights

Equation	Function		
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
Gaussian	1.98	1.98	1.98
Kansa	2.00	2.00	1.99
Levi	2.00	1.99	2.00
Schaffer	2.00	2.02	1.99

Third-order results are shown in Figure 6.19. Figure 6.19 shows that the function is approximately fourth-order accurate, while the first- and second-order derivatives are approximately second-order accurate. Table 6.17 summarizes the computed order of accuracy for each equation using Affine MLS on structured grids with isotropic weights. As with the second-order reconstruction, Affine MLS reproduces the same level of accuracy as the isotropic and anisotropic weighted MLS as shown in Tables 6.11 and 6.14. The reconstruction performs best for the Schaffer function, and it performs worse for the Gaussian.

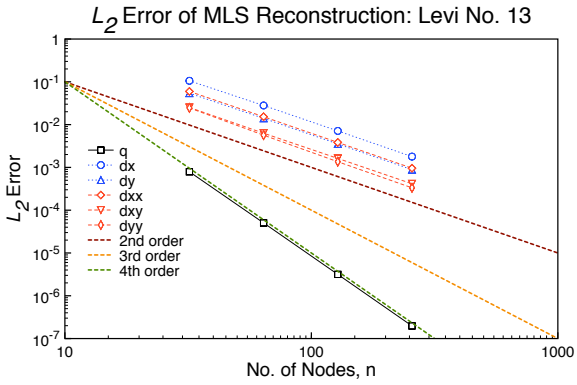
Table 6.17: Accuracy on Structured Grids: 3^{rd}-Order Affine MLS with Isotropic Weights.

Equation	Function					
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
Gaussian	3.95	1.96	1.96	1.96	1.98	1.96
Kansa	3.99	1.99	1.99	2.00	1.98	2.01
Levi	3.99	1.96	1.98	1.99	1.97	2.07
Schaffer	3.98	2.02	2.01	2.00	2.00	1.97

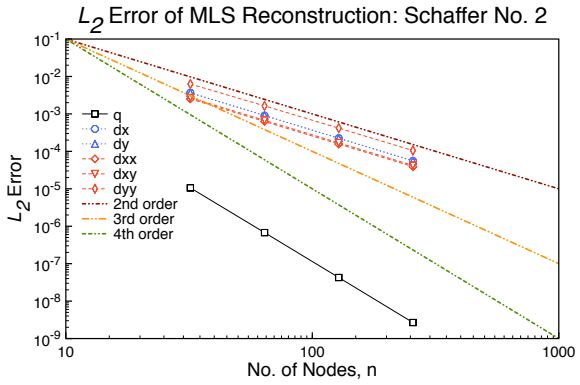


(a) Gaussian

(b) Kansa Function



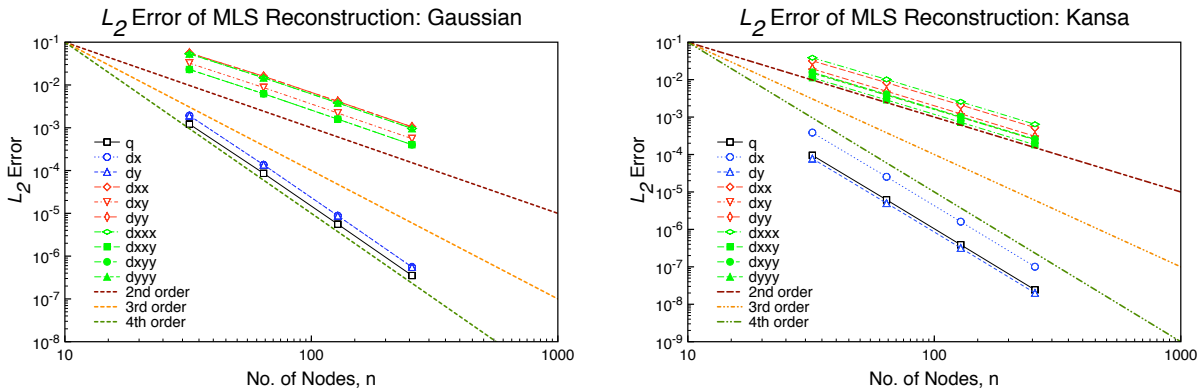
(c) Levi Function



(d) Schaffer Function

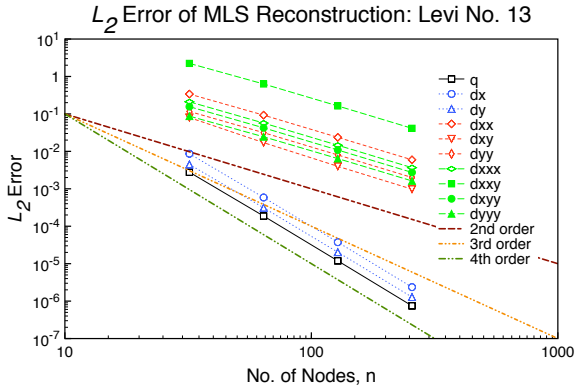
Figure 6.19: Error Norms on Structured Grids: 3^{rd}-Order Affine MLS with Isotropic Weights.

Fourth-order results are shown in Figure 6.20. Figure 6.20 shows that the reconstruction of the function and first-derivatives achieves approximately fourth-order accuracy, while the second- and third-derivatives are approximately second-order accurate. Table 6.18 summarizes the computed order of accuracy for each equation using Affine MLS on structured grids with isotropic weights. Affine MLS reproduces the same level of accuracy as the isotropic and anisotropic weighted MLS as shown in Tables 6.12 and 6.15. As with the previous sections, the reconstruction performs best for the Schaffer function, and it performs worse for the Gaussian.

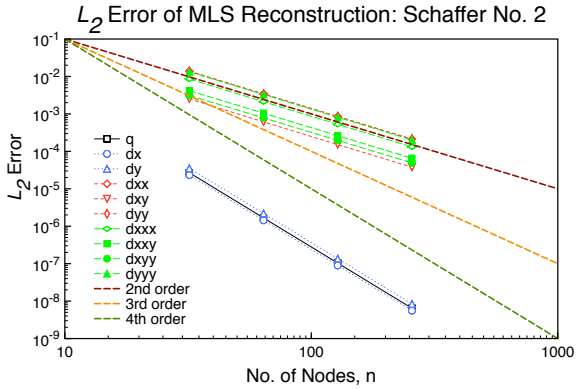


(a) Gaussian

(b) Kansa Function



(c) Levi Function



(d) Schaffer Function

Figure 6.20: Error Norms on Structured Grids: 4^{th}-Order Affine MLS with Isotropic Weights.

Table 6.18: Accuracy on Structured Grids: 4^{th}-Order Affine MLS with Isotropic Weights.

Equation	Function									
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial xy^2}$	$\frac{\partial^3 q}{\partial y^3}$
Gaussian	3.92	3.92	3.92	1.91	1.95	1.91	1.93	1.95	1.95	1.93
Kansa	3.98	3.97	3.97	1.97	1.98	1.99	1.97	1.98	1.98	1.97
Levi	3.95	3.95	3.92	1.94	2.12	1.95	1.95	1.92	1.95	1.92
Schaffer	4.01	4.01	4.01	2.01	2.02	2.01	2.01	2.00	1.99	2.01

6.3.1.4 Summary

After computing all of the reconstructions using the various weighting strategies, it is worth comparing the accuracy and error computed from each strategy for each function. Table 6.19 summarizes the order of accuracy for each of the functions using a second-order MLS reconstruction.

Table 6.19: Accuracy on Structured Grids: 2^{nd}-Order MLS Weight Comparison.

Equation	Weight	Function		
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
Gaussian	Iso.	1.98	1.98	1.98
	Aniso.	1.98	1.98	1.98
	Affine (1)	1.98	1.98	1.98
Kansa	Iso.	2.00	2.00	1.99
	Aniso.	2.00	2.00	1.99
	Affine (1)	2.00	2.00	1.99
Levi	Iso.	2.00	1.99	2.00
	Aniso.	2.00	1.99	2.00
	Affine (1)	2.00	1.99	2.00
Schaffer	Iso.	2.00	2.02	1.99
	Aniso.	2.00	2.02	1.99
	Affine (1)	2.00	2.02	1.99

Table 6.19 shows that the weighting strategy does not change the order of accuracy for any of the functions. This is the expected and desired effect, since on a structured mesh each of the strategies degenerate to the isotropic weighting case. Table 6.20 summarizes the order of accuracy for each of the functions using a third-order MLS reconstruction.

Table 6.20: Accuracy on Structured Grids: 3^{rd}-Order MLS Weight Comparison.

Equation	Weight	Function					
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
Gaussian	Iso.	3.95	1.96	1.96	1.96	1.98	1.96
	Aniso.	3.95	1.96	1.96	1.96	1.98	1.96
	Affine (1)	3.95	1.96	1.96	1.96	1.98	1.96
Kansa	Iso.	3.99	1.99	1.99	2.00	1.98	2.01
	Aniso.	3.99	1.99	1.99	2.00	1.98	2.01
	Affine (1)	3.99	1.99	1.99	2.00	1.98	2.01
Levi	Iso.	3.99	1.96	1.98	1.99	1.97	2.07
	Aniso.	3.99	1.96	1.98	1.99	1.97	2.07
	Affine (1)	3.99	1.96	1.98	1.99	1.97	2.07
Schaffer	Iso.	3.98	2.02	2.01	2.00	2.00	1.97
	Aniso.	3.98	2.02	2.01	2.00	2.00	1.97
	Affine (1)	3.98	2.02	2.01	2.00	2.00	1.97

Table 6.20 shows that the weighting strategy does not change the order of accuracy for any of the functions. This is the expected and desired effect, since on a structured mesh each of the strategies degenerate to the isotropic weighting case. Table 6.21 summarizes the order of accuracy for each of the functions using a third-order MLS reconstruction.

Table 6.21: Accuracy on Structured Grids: 4^{th}-Order MLS Weight Comparison.

Equation	Weight	Function									
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial xy^2}$	$\frac{\partial^3 q}{\partial y^3}$
Gaussian	Iso.	3.92	3.92	3.92	1.91	1.95	1.91	1.93	1.95	1.95	1.93
	Aniso.	3.92	3.92	3.92	1.91	1.95	1.91	1.93	1.95	1.95	1.93
	Affine (1)	3.92	3.92	3.92	1.91	1.95	1.91	1.93	1.95	1.95	1.93
Kansa	Iso.	3.98	3.97	3.97	1.97	1.98	1.99	1.97	1.98	1.98	1.97
	Aniso.	3.98	3.97	3.97	1.97	1.98	1.99	1.97	1.98	1.98	1.97
	Affine (1)	3.98	3.97	3.97	1.97	1.98	1.99	1.97	1.98	1.98	1.97
Levi	Iso.	3.95	3.95	3.92	1.94	2.12	1.95	1.95	1.92	1.95	1.92
	Aniso.	3.95	3.95	3.92	1.94	2.12	1.95	1.95	1.92	1.95	1.92
	Affine (1)	3.95	3.95	3.92	1.94	2.12	1.95	1.95	1.92	1.95	1.92
Schaffer	Iso.	4.01	4.01	4.01	2.01	2.02	2.01	2.01	2.00	1.99	2.01
	Aniso.	4.01	4.01	4.01	2.01	2.02	2.01	2.01	2.00	1.99	2.01
	Affine (1)	4.01	4.01	4.01	2.01	2.02	2.01	2.01	2.00	1.99	2.01

Table 6.21 shows that the weighting strategy does not change the order of accuracy for any of the functions, as with the second- and third-order case. By inspection, one would want to choose the isotropic weighting strategy, since it produces the same order of accuracy for the lowest computational cost across any of the orders of reconstruction. However, this does not tell the full story, as the condition number should also be assessed when making this decision. A condition number comparison is given for each order in Section 6.6.

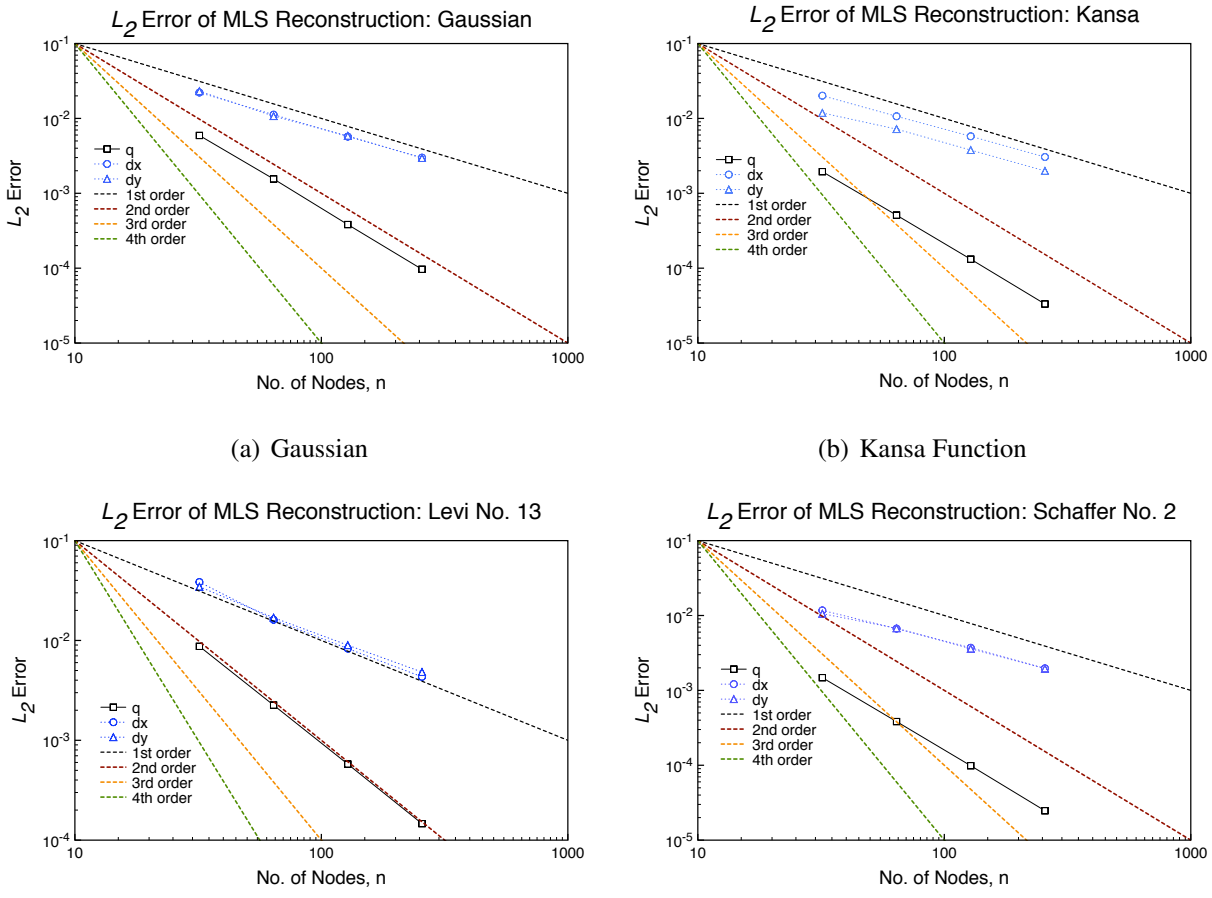
6.3.2 Unstructured Grids

This section presents the results of order of accuracy of the MLS derivatives on unstructured grids. First, the MLS derivatives using isotropic weights is presented. Next, the MLS derivatives

using anisotropic weights are shown. The section concludes with the Affine MLS derivatives using anisotropic weights.

6.3.2.1 Isotropic Weights

This subsection presents the results for MLS reconstructions on unstructured grids using an isotropic weighting scheme. The second-order results using isotropic weights are shown in Figure 6.21.



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

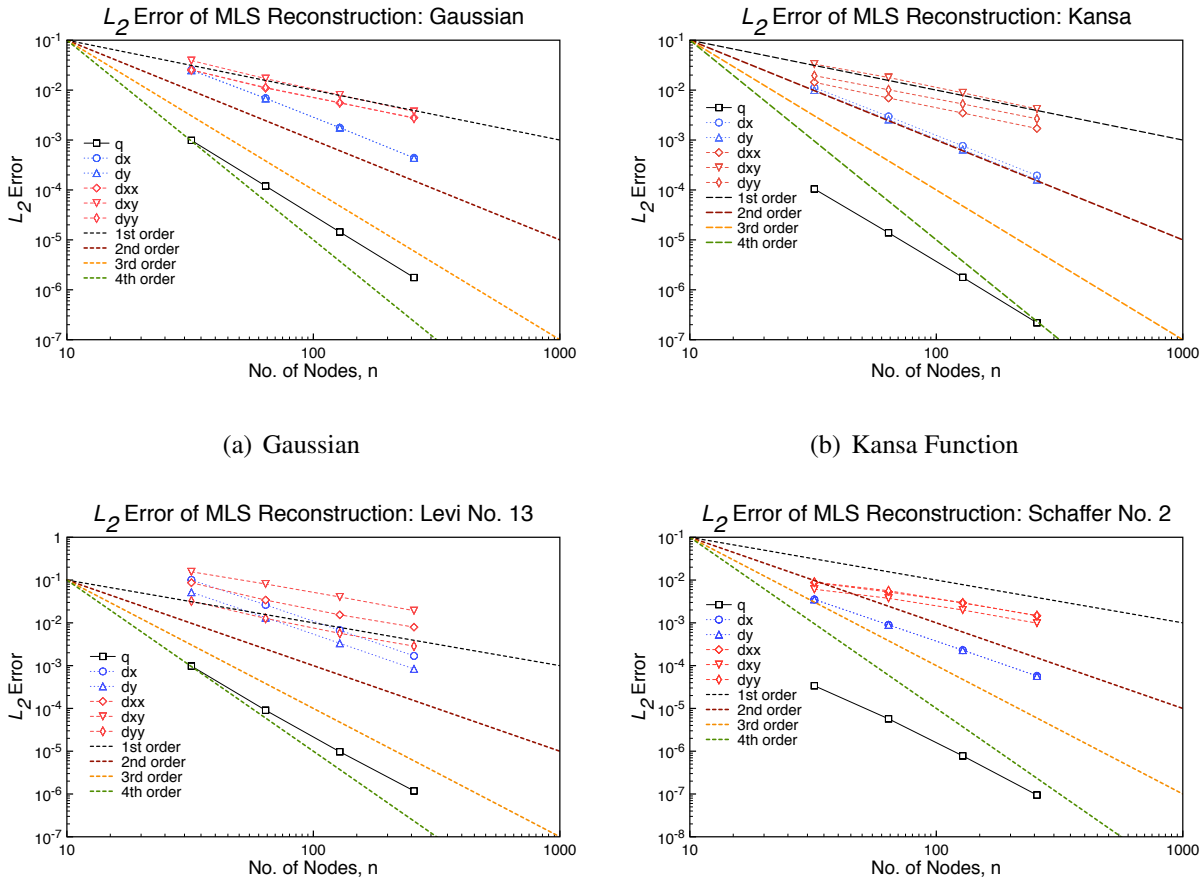
Figure 6.21: Error Norms on Unstructured Grids: 2^{nd}-Order MLS with Isotropic Weights.

From Figure 6.21, the function is reconstructed approximately second-order, and the first

derivatives are reconstructed approximately first-order. Table 6.22 summarizes the computed order of accuracy for each equation on unstructured grids using isotropic weights. The accuracy on the unstructured grids is reduced as expected compared to structured grids; however, the first-derivatives are as accurate as the theory would predict (first-order for a second-order approximation). The reconstruction of the first-derivatives is best for the Gaussian and worse for the Schaffer function. Third-order results are shown in Figure 6.22.

Table 6.22: Accuracy on Unstructured Grids: 2^{nd}-Order MLS with Isotropic Weights.

Equation	Function		
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
Gaussian	2.00	0.95	0.93
Kansa	1.97	0.90	0.93
Levi	1.97	0.95	0.90
Schaffer	1.98	0.88	0.89



(a) Gaussian

(b) Kansa Function

(c) Levi Function

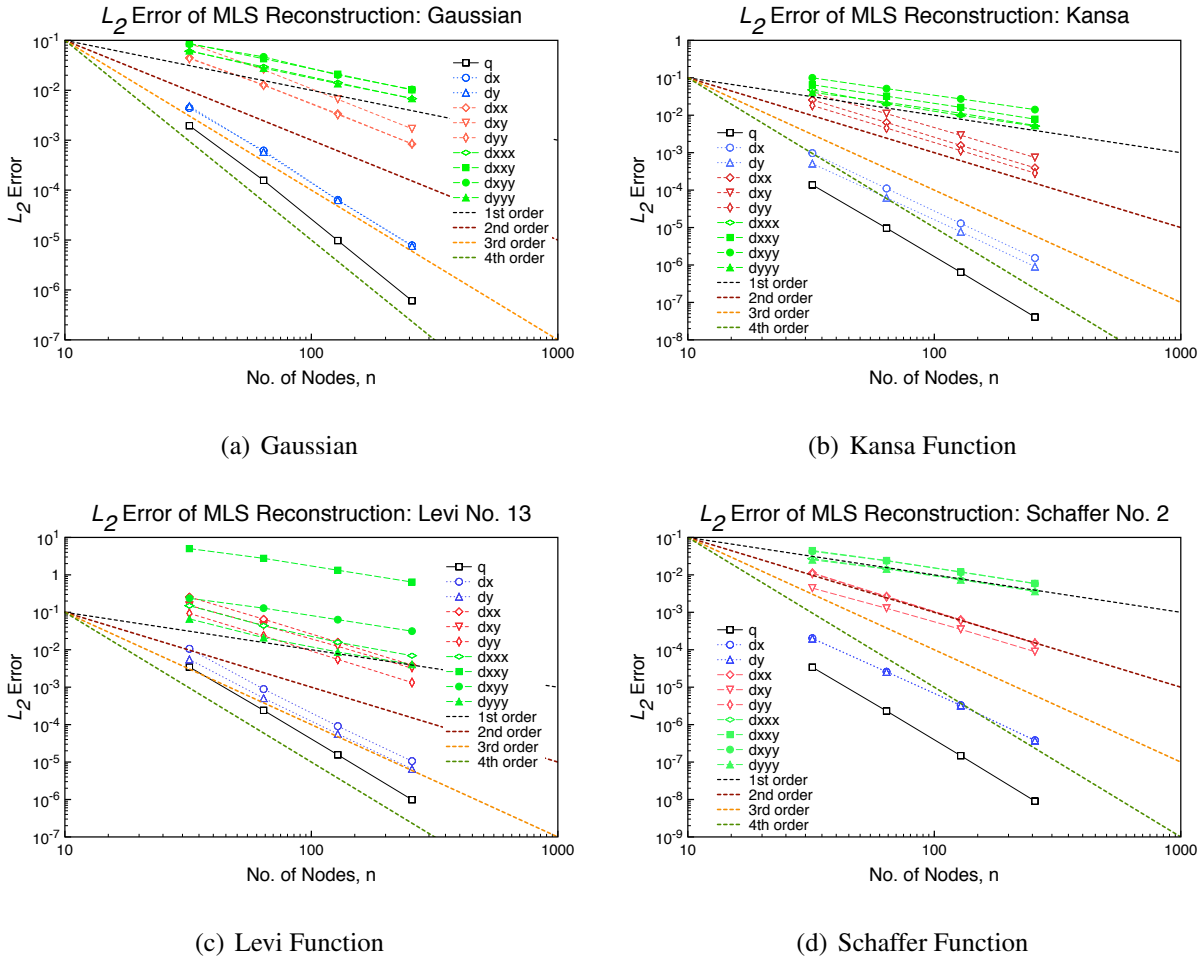
(d) Schaffer Function

Figure 6.22: Error Norms on Unstructured Grids: 3rd-Order MLS with Isotropic Weights.

Figure 6.22 shows that the accuracy for the function is approximately third-order accurate, and the first-derivatives second-order accurate, and second-derivatives first-order accurate. Table 6.23 summarizes the computed order of accuracy for each equation on unstructured grids using isotropic weights. The accuracy is similar to the second-order results on unstructured grids; the accuracy achieved is more based on the level of derivation than the structured case. The reconstruction is best for the Gaussian function overall, but the reconstruction of the Levi is nearly as accurate. Fourth-order results are shown in Figure 6.23.

Table 6.23: Accuracy on Unstructured Grids: 3^{rd}-Order MLS with Isotropic Weights.

Equation	Function					
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
Gaussian	3.05	1.99	1.98	0.99	1.10	1.02
Kansa	2.99	1.97	2.00	1.01	1.05	0.96
Levi	3.13	1.98	1.99	1.05	1.04	1.08
Schaffer	2.96	1.98	1.99	0.92	0.97	0.96



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.23: Error Norms on Unstructured Grids: 4^{th}-Order MLS with Isotropic Weights.

Figure 6.23 shows that the function is approximately fourth-order accurate, first-derivatives ap-

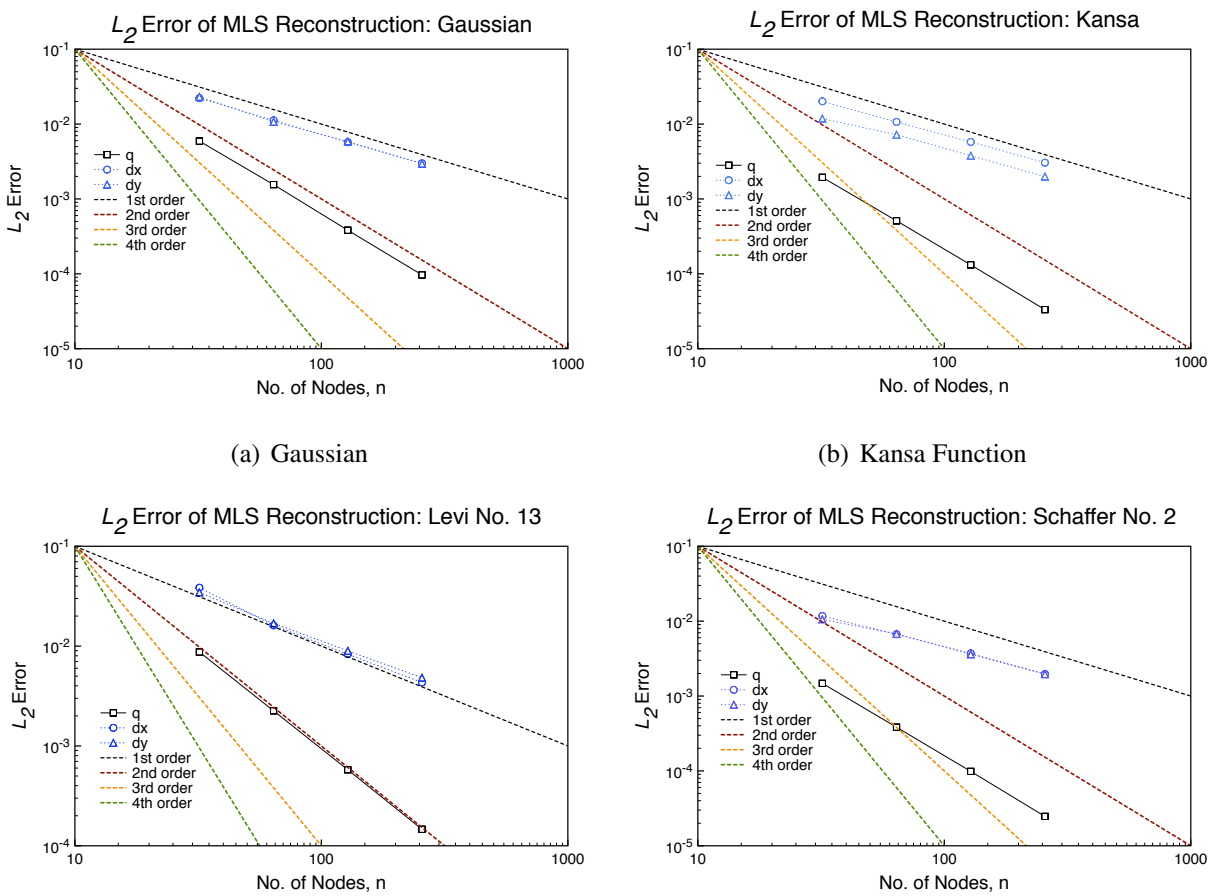
proximately third-order accurate, the second-derivatives second-order accurate, and third-derivatives first-order accurate. Table 6.24 summarizes the computed order of accuracy for each equation on unstructured grids using isotropic weights. The best overall reconstruction is with the Gaussian function.

Table 6.24: Accuracy on Unstructured Grids: 4^{th}-Order MLS with Isotropic Weights.

Equation	Function									
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial xy^2}$	$\frac{\partial^3 q}{\partial y^3}$
Gaussian	4.01	3.15	3.13	1.97	1.96	1.95	1.06	1.04	1.09	1.01
Kansa	3.94	3.08	3.06	2.01	1.95	2.00	0.99	1.02	0.93	1.03
Levi	3.96	3.20	3.14	2.04	1.90	2.07	1.33	1.05	1.02	1.19
Schaffer	4.00	3.04	3.05	2.07	1.93	2.06	1.00	1.02	1.01	0.99

6.3.2.2 Anisotropic Weights

This subsection presents the results for MLS reconstructions on unstructured grids using an anisotropic weighting scheme. The second-order results using anisotropic weights are shown in Figure 6.24.



(a) Gaussian

(b) Kansa Function

(c) Levi Function

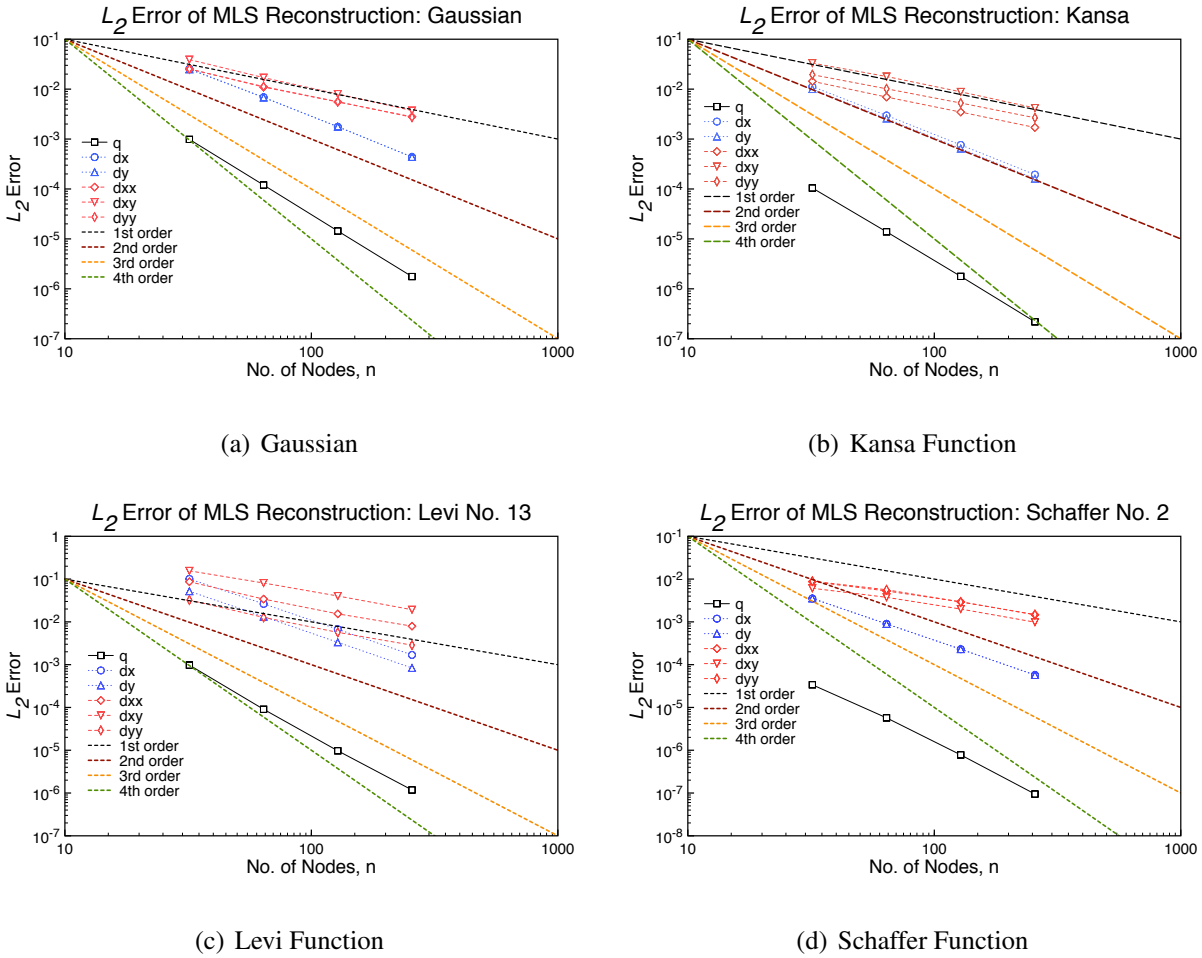
(d) Schaffer Function

Figure 6.24: Error Norms on Unstructured Grids: 2^{nd}-Order MLS with Anisotropic Weights.

From Figure 6.24, the function is reconstructed approximately second-order, and the first derivatives are reconstructed approximately first-order. Table 6.25 summarizes the computed order of accuracy for each equation on unstructured grids using anisotropic weights. As with the isotropic case, the reconstruction produces second-order results for the function and first-order results for the first derivatives. There are some slight differences between the isotropic weight results summarized in Table 6.22, but that is expected since the weights are different in the unstructured case. The reconstruction of the first-derivatives is best for the Gaussian and worse for the Schaffer No. 2 function. Third-order results are shown in Figure 6.25.

Table 6.25: Accuracy on Unstructured Grids: 2^{nd}-Order MLS with Anisotropic Weights.

Equation	Function		
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
Gaussian	2.00	0.94	0.92
Kansa	1.97	0.93	0.96
Levi	1.97	1.00	0.97
Schaffer	1.98	0.89	0.90



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.25: Error Norms on Unstructured Grids: 3^{rd}-Order MLS with Anisotropic Weights.

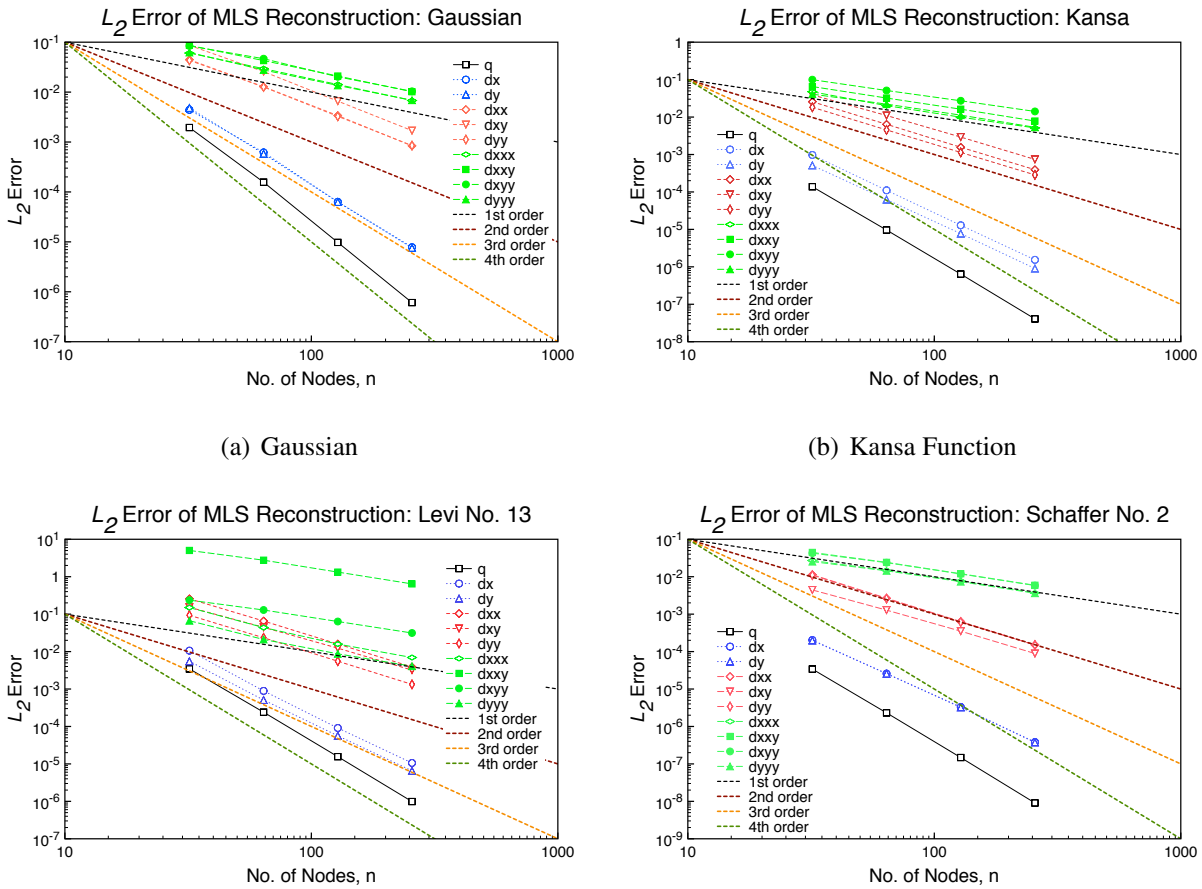
Figure 6.25 shows that the accuracy for the function is approximately third-order accurate,

and the first-derivatives second-order accurate, and second-derivatives first-order accurate. Table 6.26 summarizes the computed order of accuracy for each equation on unstructured grids using anisotropic weights. The results are again similar to the isotropic weighing results presented in Table 6.23, with the function obtaining approximately third-order accuracy, the first-derivatives are second-order, and the second-derivatives are roughly first-order. The reconstruction is best for the Gaussian function overall, but the reconstruction of the Levi is nearly as accurate. Fourth-order

Table 6.26: Accuracy on Unstructured Grids: 3^{rd}-Order MLS with Anisotropic Weights.

Equation	Function					
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
Gaussian	3.05	2.00	1.99	1.00	1.11	1.03
Kansa	3.01	1.97	2.00	1.02	1.05	0.97
Levi	3.14	1.99	2.00	1.07	1.05	1.10
Schaffer	2.97	1.98	1.99	0.94	0.96	0.98

results are shown in Figure 6.26.



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.26: Error Norms on Unstructured Grids: 4^{th}-Order MLS with Anisotropic Weights.

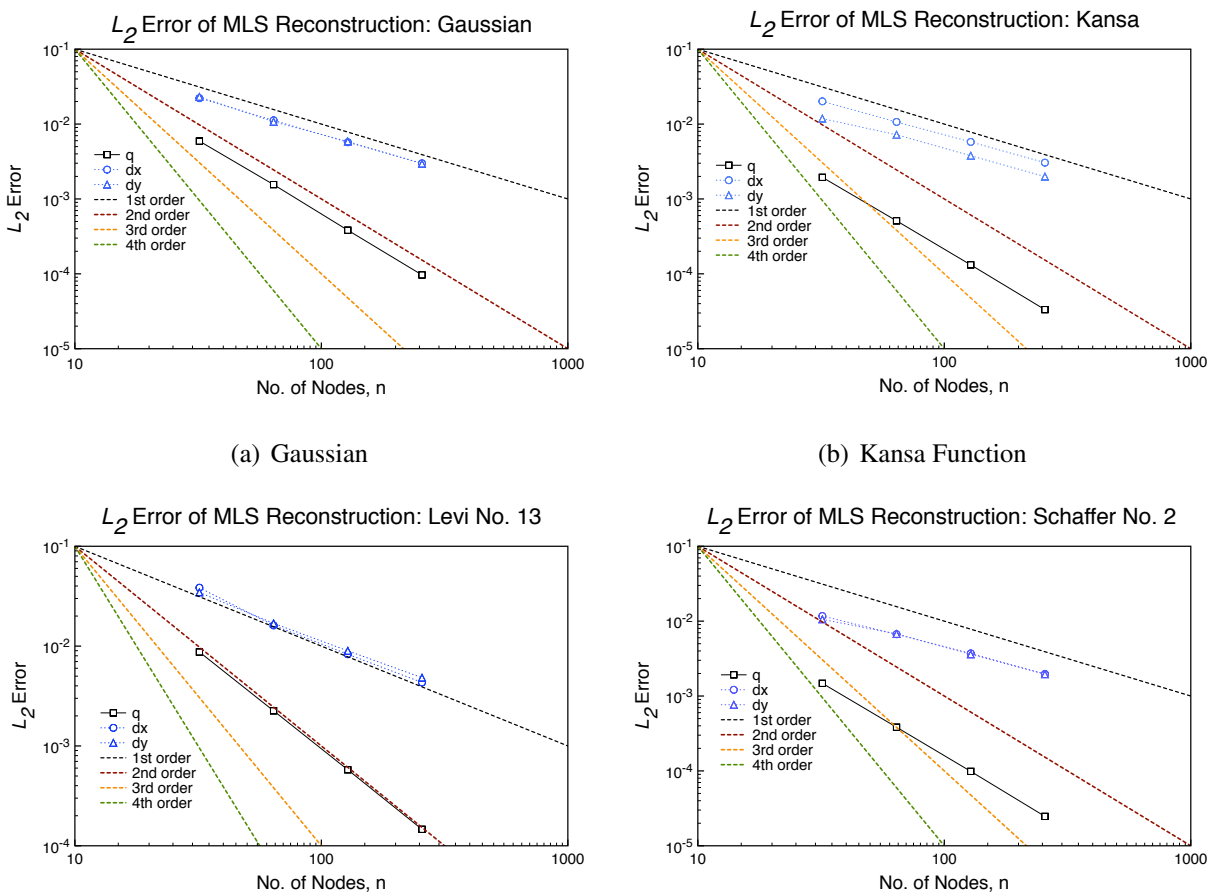
Figure 6.26 shows that the function is approximately fourth-order accurate, first-derivatives approximately third-order accurate, the second-derivatives second-order accurate, and third-derivatives first-order accurate. Table 6.27 summarizes the computed order of accuracy for each equation on unstructured grids using anisotropic weights. As with the second- and third-order reconstruction, the fourth-order reconstruction with anisotropic weights produces the same level of accuracy as the isotropic weighted MLS shown in Table 6.24, yielding a fourth-order accurate function, third-order accurate first-derivatives, second-order accurate second-derivatives, and first-order accurate third-derivatives. The best overall reconstruction is with the Gaussian function.

Table 6.27: Accuracy on Unstructured Grids: 4^{th}-Order MLS with Anisotropic Weights.

Equation	Function									
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial x y^2}$	$\frac{\partial^3 q}{\partial y^3}$
Gaussian	4.00	3.15	3.13	1.95	1.97	1.96	1.08	1.04	1.08	1.07
Kansa	3.93	3.07	3.05	2.06	1.99	2.04	1.03	1.09	1.00	1.08
Levi	3.94	3.21	3.14	2.09	2.05	2.12	1.47	1.34	1.20	1.30
Schaffer	3.98	3.05	3.04	2.12	1.96	2.11	1.06	1.10	1.08	1.10

6.3.2.3 Affine MLS

This subsection presents the results for Affine MLS reconstructions on unstructured grids using an anisotropic weighting scheme. The second-order results using Affine MLS with anisotropic weights are shown in Figure 6.27.



(a) Gaussian

(b) Kansa Function

(c) Levi Function

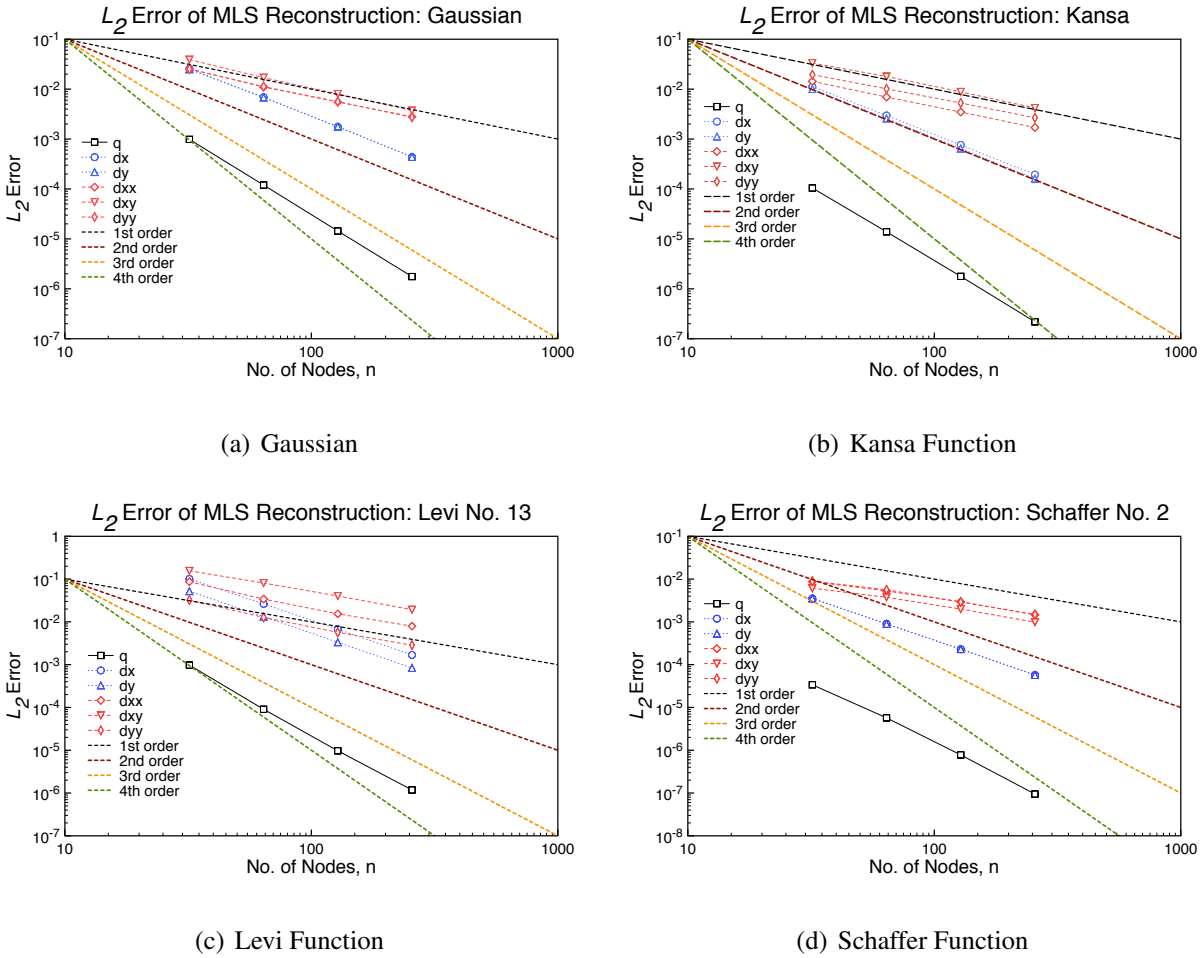
(d) Schaffer Function

Figure 6.27: Error Norms on Unstructured Grids: 2^{nd}-Order Affine MLS with Anisotropic Weights.

From Figure 6.27, the function is reconstructed approximately second-order, and the first derivatives are reconstructed approximately first-order. Table 6.28 summarizes the computed order of accuracy for each equation on unstructured grids using Affine MLS with isotropic weights. As with the previous two methods, Affine MLS in general produces second-order results for both the function and first-order first derivatives. Affine MLS produces the same results as the anisotropic weighted results in Table 6.25 since it uses the same base weighting scheme. The reconstruction of the first-derivatives is best for the Gaussian and worse for the Schaffer function. Third-order results are shown in Figure 6.28.

Table 6.28: Accuracy on Unstructured Grids: 2^{nd}-Order Affine MLS with Anisotropic Weights.

Equation	Function		
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
Gaussian	2.00	0.94	0.92
Kansa	1.97	0.93	0.96
Levi	1.97	1.00	0.97
Schaffer	1.98	0.89	0.90



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

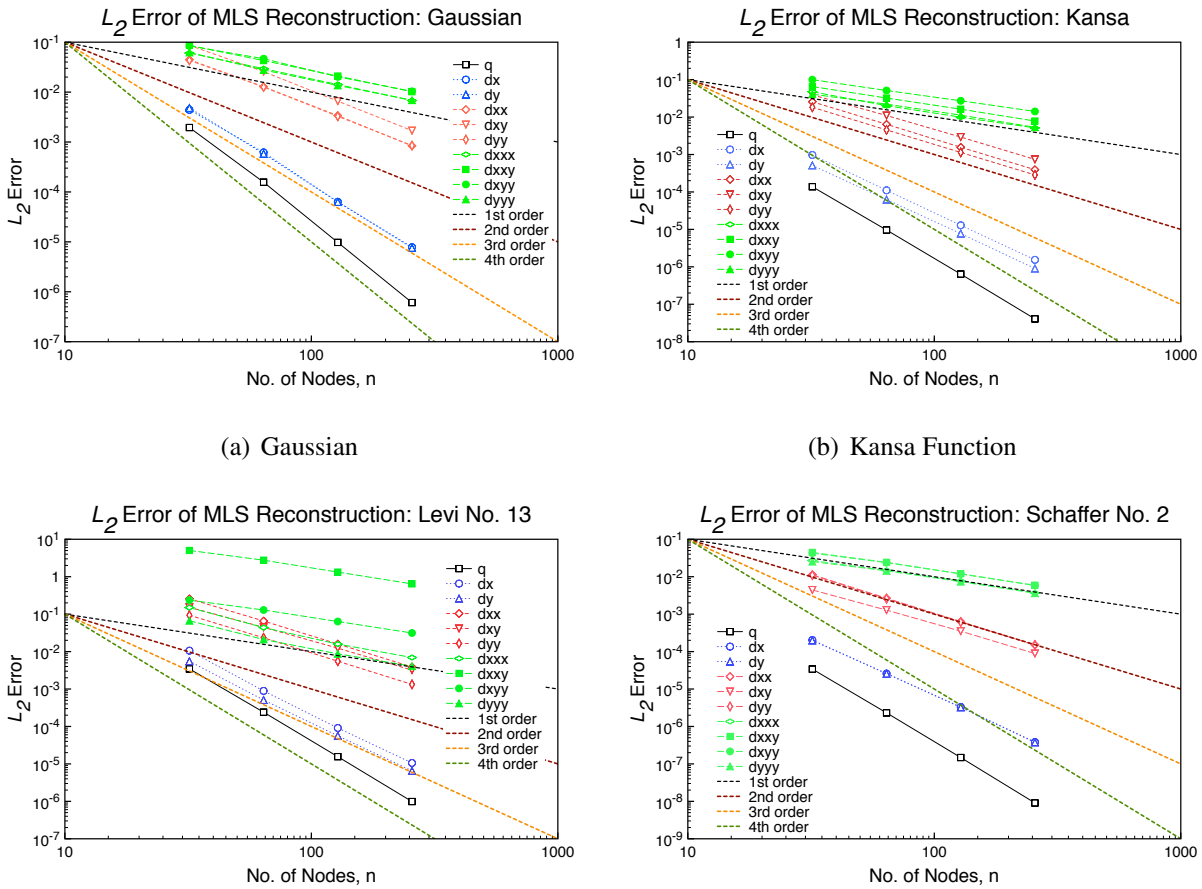
Figure 6.28: Error Norms on Unstructured Grids: 3^{rd}-Order Affine MLS with Anisotropic Weights.

Figure 6.28 shows that the accuracy for the function is approximately third-order accurate, and

the first-derivatives second-order accurate, and second-derivatives first-order accurate. Table 6.29 summarizes the computed order of accuracy for each equation using Affine MLS on unstructured grids with anisotropic weights. As with the second-order reconstruction, Affine MLS reproduces the same level of accuracy as the isotropic and anisotropic weighted MLS as shown in Tables 6.23 and 6.26, with the results matching the anisotropic weighted results since the Affine MLS uses the same base weighting scheme. The reconstruction is best for the Gaussian function overall, but the reconstruction of the Levi is nearly as accurate. Fourth-order results are shown in Figure 6.29.

Table 6.29: Accuracy on Unstructured Grids: 3^{rd}-Order Affine MLS with Anisotropic Weights.

Equation	Function					
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
Gaussian	3.05	2.00	1.99	1.00	1.11	1.03
Kansa	3.01	1.97	2.00	1.02	1.05	0.97
Levi	3.14	1.99	2.00	1.07	1.05	1.10
Schaffer	2.97	1.98	1.99	0.94	0.96	0.98



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.29: Error Norms on Unstructured Grids: 4^{th}-Order Affine MLS with Anisotropic Weights.

Figure 6.26 shows that the function is approximately fourth-order accurate, first-derivatives approximately third-order accurate, the second-derivatives second-order accurate, and third-derivatives first-order accurate. Table 6.30 summarizes the computed order of accuracy for each equation using Affine MLS on unstructured grids with anisotropic weights. Affine MLS reproduces the same level of accuracy as the anisotropic weighted MLS as shown in Table 6.27, since it uses the same base weighting scheme. The best overall reconstruction is with the Gaussian function.

Table 6.30: Accuracy on Unstructured Grids: 4^{th}-Order Affine MLS with Anisotropic Weights.

Equation	Function									
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial xy^2}$	$\frac{\partial^3 q}{\partial y^3}$
Gaussian	4.00	3.15	3.13	1.95	1.97	1.96	1.08	1.04	1.08	1.07
Kansa	3.93	3.07	3.05	2.06	1.99	2.04	1.03	1.09	1.00	1.08
Levi	3.94	3.21	3.14	2.09	2.05	2.12	1.47	1.34	1.20	1.30
Schaffer	3.98	3.05	3.04	2.12	1.96	2.11	1.06	1.10	1.08	1.10

6.3.2.4 Summary

After computing all of the reconstructions using the various weighting strategies, it is worth comparing the accuracy and error computed from each strategy for each function. Table 6.31 summarizes the order of accuracy for each of the functions using a second-order MLS reconstruction.

Table 6.31: Accuracy on Unstructured Grids: 2^{nd}-Order MLS Weight Comparison.

Equation	Weight	Function		
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
Gaussian	Iso.	2.00	0.95	0.93
	Aniso.	2.00	0.94	0.92
	Affine (2)	2.00	0.94	0.92
Kansa	Iso.	1.97	0.90	0.93
	Aniso.	1.97	0.93	0.96
	Affine (2)	1.97	0.93	0.96
Levi	Iso.	1.97	0.95	0.90
	Aniso.	1.97	1.00	0.97
	Affine (2)	1.97	1.00	0.97
Schaffer	Iso.	1.98	0.88	0.89
	Aniso.	1.98	0.89	0.90
	Affine (2)	1.98	0.89	0.90

Table 6.31 shows that the weighting strategy does have a slight effect on the orders of accuracy. For the Gaussian, the anisotropic weighted methods have a slightly lower accuracy for the

first derivatives, while the other functions using anisotropic weighting have a slightly higher accuracy for the first derivatives. There should be some differences between the strategies due to the description of the stencil through s. It is also easy to see here that the Affine MLS obtains the same accuracy as the base anisotropic weighting strategy, which is the desired result. It is also worthwhile to exam the error from the reconstructions. Table 6.32 shows the L_2 error for each of the functions on the finest mesh.

Table 6.32: L_2 Error on Finest Unstructured Grid: 2^{nd}-Order MLS Weight Comparison.

Equation	Weight	Function		
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
Gaussian	Iso.	9.66E-5	2.99E-3	2.96E-3
	Aniso.	9.69E-5	3.01E-3	2.97E-3
	Affine (2)	9.69E-5	3.01E-3	2.97E-3
Kansa	Iso.	3.32E-5	3.05E-3	1.99E-3
	Aniso.	3.33E-5	3.17E-3	2.08E-3
	Affine (2)	3.33E-5	3.17E-3	2.08E-3
Levi	Iso.	1.46E-4	4.33E-3	4.83E-3
	Aniso.	1.46E-4	5.65E-3	6.05E-3
	Affine (2)	1.46E-4	5.65E-3	6.05E-3
Schaffer	Iso.	2.47E-5	1.97E-3	1.95E-3
	Aniso.	2.48E-5	2.01E-3	2.00E-3
	Affine (2)	2.48E-5	2.01E-3	2.00E-3

Effectively, the differences in the computed error are negligible for the second-order reconstruction. The first derivatives have a slightly higher error, but 1E-3 or less will not make much

of a difference to the overall results. Table 6.33 summarizes the order of accuracy for each of the functions using a third-order MLS reconstruction.

Table 6.33: Accuracy on Unstructured Grids: 3^{rd}-Order MLS Weight Comparison.

Equation	Weight	Function					
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
Gaussian	Iso.	3.05	1.99	1.98	0.99	1.10	1.02
	Aniso.	3.05	2.00	1.99	1.00	1.11	1.03
	Affine (2)	3.05	2.00	1.99	1.00	1.11	1.03
Kansa	Iso.	2.99	1.97	2.00	1.01	1.05	0.96
	Aniso.	3.01	1.97	2.00	1.02	1.05	0.97
	Affine (2)	3.01	1.97	2.00	1.02	1.05	0.97
Levi	Iso.	3.13	1.98	1.99	1.05	1.04	1.08
	Aniso.	3.14	1.99	2.00	1.07	1.05	1.10
	Affine (2)	3.14	1.99	2.00	1.07	1.05	1.10
Schaffer	Iso.	2.96	1.98	1.99	0.92	0.97	0.96
	Aniso.	2.97	1.98	1.99	0.94	0.96	0.98
	Affine (2)	2.97	1.98	1.99	0.94	0.96	0.98

Table 6.33 shows that the weighting strategy does have an effect on the orders of accuracy. For each of the functions, the anisotropic weighting strategy has the better accuracy on the unstructured mesh. There should be some differences between the strategies due to the description of the stencil through s. It is also worthwhile to exam the error from the reconstructions. Table 6.34 shows the L_2 error for each of the functions on the finest mesh.

Table 6.34: L_2 Error on Finest Unstructured Grid: 3^{rd}-Order MLS Weight Comparison.

Equation	Weight	Function					
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
Gaussian	Iso.	1.76E-6	4.37E-4	4.38E-4	2.80E-3	3.72E-3	2.73E-3
	Aniso.	1.66E-6	4.45E-4	4.46E-4	3.05E-3	3.72E-3	2.99E-3
	Affine (2)	1.66E-6	4.45E-4	4.46E-4	3.05E-3	3.72E-3	2.99E-3
Kansa	Iso.	2.18E-7	1.94E-4	1.61E-4	1.71E-3	4.20E-3	2.67E-3
	Aniso.	2.10E-7	1.97E-4	1.64E-4	1.87E-3	4.34E-3	2.92E-3
	Affine (2)	2.10E-7	1.97E-4	1.64E-4	1.87E-3	4.34E-3	2.92E-3
Levi	Iso.	1.18E-6	1.69E-3	8.37E-4	7.94E-3	1.93E-2	2.86E-3
	Aniso.	1.19E-6	1.72E-3	8.52E-4	8.70E-3	2.14E-2	3.17E-3
	Affine (2)	1.19E-6	1.72E-3	8.52E-4	8.70E-3	2.15E-2	3.17E-3
Schaffer	Iso.	9.47E-8	5.74E-5	5.72E-5	1.47E-3	9.84E-4	1.48E-3
	Aniso.	8.84E-8	5.84E-5	5.81E-5	1.62E-3	9.90E-4	1.62E-3
	Affine (2)	8.84E-8	5.84E-5	5.81E-5	1.62E-3	9.91E-4	1.62E-3

As with the second-order results in Table 6.32, the difference in the error is negligible for the third-order reconstruction. Any differences are small, so any of the weighting strategies should produce the same results. Table 6.35 summarizes the order of accuracy for each of the functions using a third-order MLS reconstruction.

Table 6.35: Accuracy on Unstructured Grids: 4^{th}-Order MLS Weight Comparison.

Equation	Weight	Function									
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial xy^2}$	$\frac{\partial^3 q}{\partial y^3}$
Gaussian	Iso.	4.01	3.15	3.13	1.97	1.96	1.95	1.06	1.04	1.09	1.01
	Aniso.	4.00	3.15	3.13	1.95	1.97	1.96	1.08	1.04	1.08	1.07
	Affine (2)	4.00	3.15	3.13	1.95	1.97	1.96	1.08	1.04	1.08	1.07
Kansa	Iso.	3.94	3.08	3.06	2.01	1.95	2.00	0.99	1.02	0.93	1.03
	Aniso.	3.93	3.07	3.05	2.06	1.99	2.04	1.03	1.09	1.00	1.08
	Affine (2)	3.93	3.07	3.05	2.06	1.99	2.04	1.03	1.09	1.00	1.08
Levi	Iso.	3.96	3.20	3.14	2.04	1.90	2.07	1.33	1.05	1.02	1.19
	Aniso.	3.94	3.21	3.14	2.09	2.05	2.12	1.47	1.34	1.20	1.30
	Affine (2)	3.94	3.21	3.14	2.09	2.05	2.12	1.47	1.34	1.20	1.30
Schaffer	Iso.	4.00	3.04	3.05	2.07	1.93	2.06	1.00	1.02	1.01	0.99
	Aniso.	3.98	3.05	3.04	2.12	1.96	2.11	1.06	1.10	1.08	1.10
	Affine (2)	3.98	3.05	3.04	2.12	1.96	2.11	1.06	1.10	1.08	1.10

Table 6.35 shows that the weighting strategy again has an effect on the orders of accuracy. For each of the functions, the anisotropic weighting strategy has better accuracy for the higher-derivative terms, while the isotropic weighting has better accuracy for the function and first derivatives. These differences are probably due to the variations in the stencils that naturally arise from the mesh topology, where the anisotropic weighting better describes the stencils for the higher derivative terms. The L_2 error of the reconstructions on the finest mesh shown in Tables 6.36 and 6.37 may shed some additional light on this.

Table 6.36: L_2 Error on Finest Unstructured Grid: 4^{th}-Order MLS Weight Comparison, State and First and Second Derivatives.

Equation	Weight	Function					
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
Gaussian	Iso.	6.08E-7	7.82E-6	7.57E-6	8.38E-4	1.69E-3	8.49E-4
	Aniso	5.99E-7	7.77E-6	7.55E-6	9.03E-4	1.78E-3	8.96E-4
	Affine (2)	5.99E-7	7.77E-6	7.55E-6	9.03E-4	1.78E-3	8.96E-4
Kansa	Iso.	4.07E-8	1.54E-6	9.02E-7	3.93E-4	7.40E-4	2.85E-4
	Aniso.	4.07E-8	1.54E-6	9.02E-7	3.93E-4	7.40E-4	2.85E-4
	Affine (2)	4.07E-8	1.54E-6	9.02E-7	3.93E-4	7.40E-4	2.85E-4
Levi	Iso.	9.88E-7	1.06E-5	6.54E-6	3.89E-3	3.21E-3	1.33E-3
	Aniso.	9.88E-7	1.06E-5	6.54E-6	3.89E-3	3.21E-3	1.33E-3
	Affine (2)	9.88E-7	1.06E-5	6.54E-6	3.89E-3	3.21E-3	1.33E-3
Schaffer	Iso.	9.04E-9	3.83E-7	3.75E-7	1.51E-4	8.88E-5	1.50E-4
	Aniso.	9.06E-9	3.83E-7	3.75E-7	1.51E-4	8.88E-5	1.50E-4
	Affine (2)	9.06E-9	3.83E-7	3.75E-7	1.51E-4	8.88E-5	1.50E-4

Table 6.37: L_2 Error on Finest Unstructured Grid: 4^{th}-Order MLS Weight Comparison, Third Derivatives.

Equation	Weight	Function			
		$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial x y^2}$	$\frac{\partial^3 q}{\partial y^3}$
Gaussian	Iso.	6.79E-3	1.02E-2	1.04E-2	6.88E-3
	Aniso.	7.07E-3	1.08E-2	1.11E-2	6.98E-3
	Affine (2)	7.07E-3	1.08E-2	1.11E-2	6.98E-3
Kansa	Iso.	5.13E-3	7.84E-3	1.41E-2	5.33E-3
	Aniso.	5.13E-3	7.84E-3	1.41E-2	5.33E-3
	Affine (2)	5.13E-3	7.84E-3	1.41E-2	5.33E-3
Levi	Iso.	6.94E-3	0.64	3.14E-2	4.07E-3
	Aniso.	6.94E-3	0.64	3.14E-2	4.07E-3
	Affine (2)	6.94E-3	0.64	3.14E-2	4.07E-3
Schaffer	Iso.	3.68E-3	5.92E-3	5.81E-3	3.62E-3
	Aniso.	3.68-3	5.92E-3	5.81E-3	3.62E-3
	Affine (2)	3.68E-3	5.92E-3	5.81E-3	3.62E-3

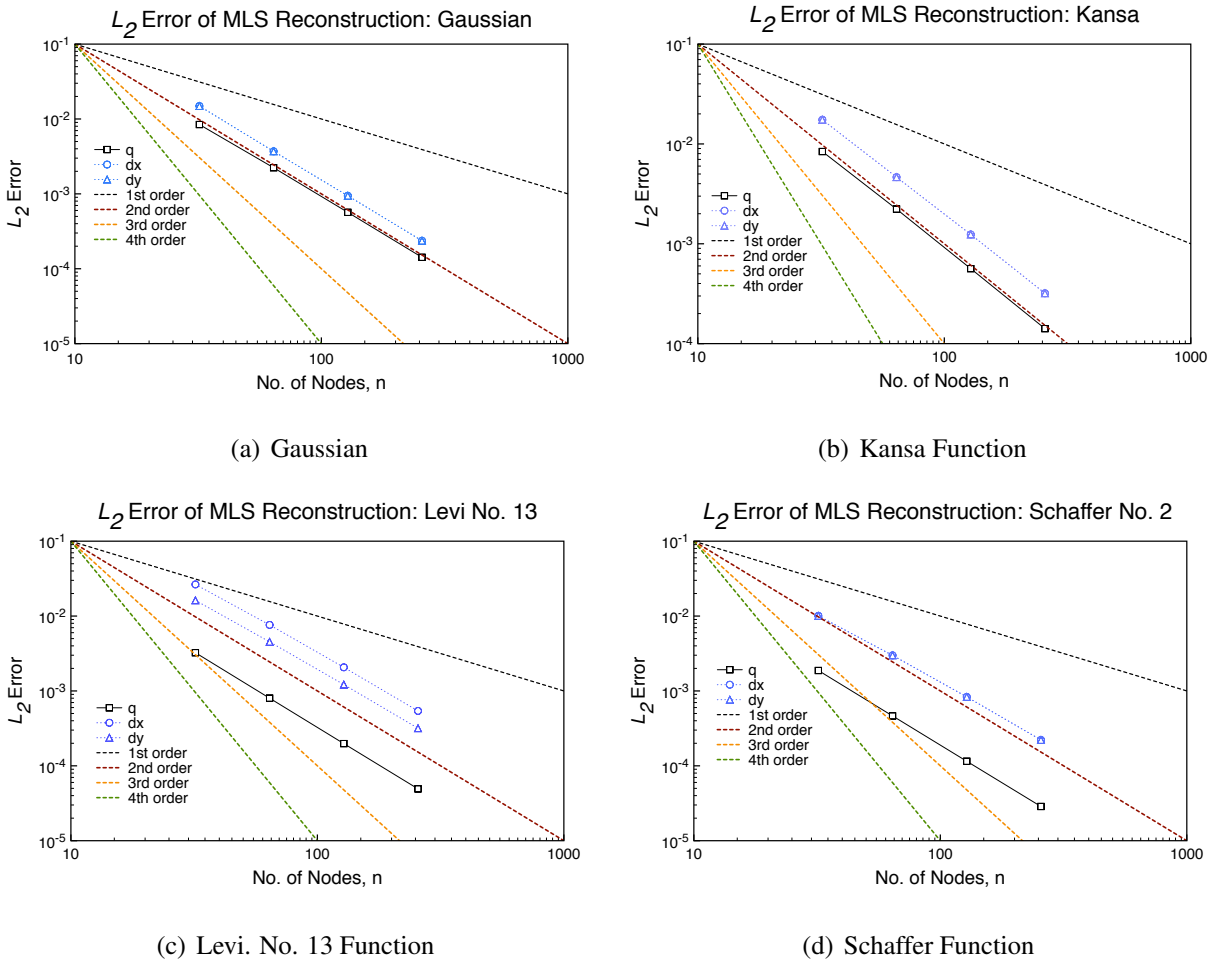
Unlike the previous results in Tables 6.32 and 6.34, the error results for the non-Gaussian functions are the same for each of the weighting strategies (and to a higher precision than shown here). For the Gaussian function, the differences in the error are small, at most less than $1E-4$, so there again should be no difference in the results. The results show that for any of the orders there is little difference in the weighting strategies, with some slight variations in the order or error. The condition number for each of the methods will complete the picture for the unstructured mesh, as will be shown in Section 6.6.

6.3.3 Stretched Grids

This section presents the results of order of accuracy of the MLS derivatives on stretched grids. First, the MLS derivatives using isotropic weights is presented. Next, the MLS derivatives using anisotropic weights are shown. Unlike the previous two sections, the Affine MLS with isotropic and anisotropic weights are evaluated as well.

6.3.3.1 Isotropic Weights

This subsection presents the results for MLS reconstructions on stretched grids using an isotropic weighting scheme. The second-order results using isotropic weights are shown in Figure 6.30.



(a) Gaussian

(b) Kansa Function

(c) Levi. No. 13 Function

(d) Schaffer Function

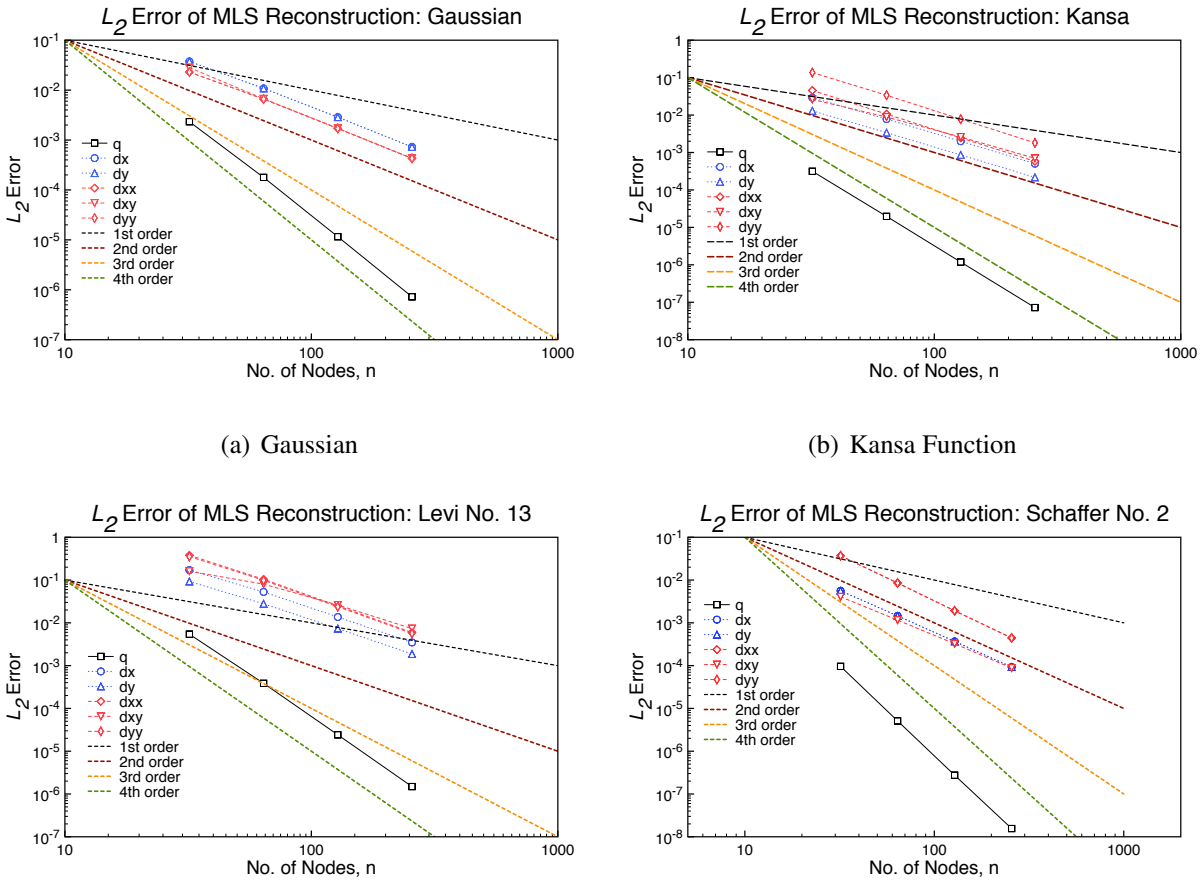
Figure 6.30: Error Norms on Stretched Grids: 2^{nd}-Order MLS with Isotropic Weights.

Figure 6.30 shows that the function and first derivative achieve approximately second-order accuracy. Table 6.38 summarizes the computed order of accuracy from the data. There appears to be no significant change in the order of accuracy based on which function is used. Third-order

Table 6.38: Accuracy on Stretched Grids: 2^{nd}-Order MLS with Isotropic Weights.

Equation	Function		
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
Gaussian	1.99	1.98	1.98
Kansa	2.01	2.02	2.01
Levi	2.00	1.92	1.90
Schaffer	2.01	1.88	1.88

results are shown in Figure 6.31.



(a) Gaussian

(b) Kansa Function

(c) Levi Function

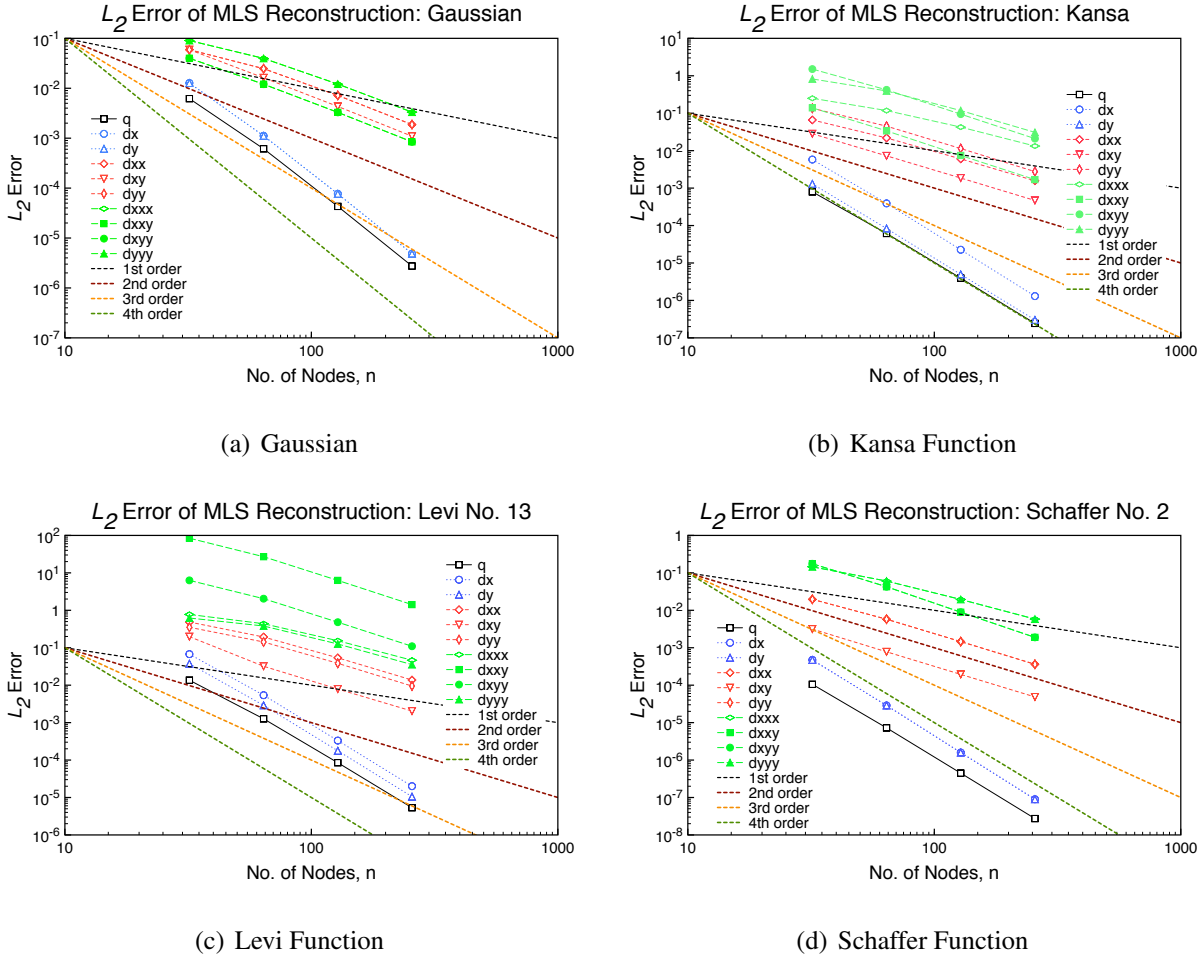
(d) Schaffer Function

Figure 6.31: Error Norms on Stretched Grids: 3rd-Order MLS with Isotropic Weights.

Figure 6.31 shows that the function achieves approximately fourth-order accuracy and the first- and second-derivative achieve approximately second-order accuracy. Table 6.39 summarizes the computed order of accuracy for each equation on stretched grids using isotropic weights. The best accuracy is achieved when reconstructing the Schaffer function and worse for the Gaussian. Fourth-order results are shown in Figure 6.32.

Table 6.39: Accuracy on Stretched Grids: 3^{rd}-Order MLS with Isotropic Weights.

Equation	Function					
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
Gaussian	3.97	1.95	1.95	1.97	1.98	1.97
Kansa	4.05	1.98	1.98	2.10	1.83	2.11
Levi	4.01	1.96	1.95	2.07	1.72	2.06
Schaffer	4.18	1.98	1.98	2.13	1.86	2.13



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.32: Error Norms on Stretched Grids: 4^{th}-Order MLS with Isotropic Weights.

Figure 6.32 shows that the function and first-derivatives achieve approximately fourth-order

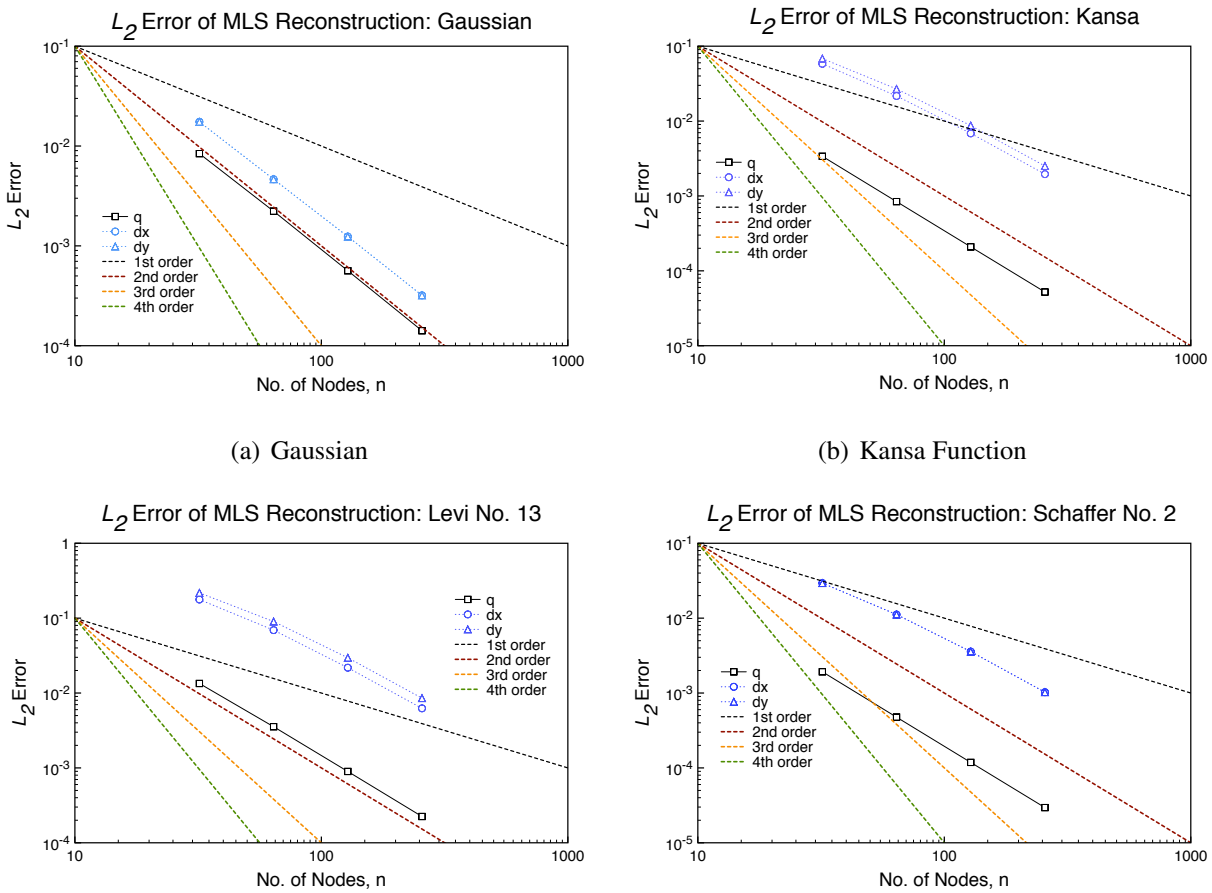
accuracy, while the second- and third-derivatives are approximately second-order accurate. Table 6.40 summarizes the computed order of accuracy for each equation on stretched grids using isotropic weights. As with the third-order solution, the reconstruction performs best with the Schaffer function and worse with the Gaussian.

Table 6.40: Accuracy on Stretched Grids: 4^{th}-Order MLS with Isotropic Weights.

Equation	Function									
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial xy^2}$	$\frac{\partial^3 q}{\partial y^3}$
Gaussian	3.90	3.92	3.92	1.86	1.94	1.86	1.79	1.92	1.92	1.79
Kansa	3.99	4.11	4.06	1.90	1.98	2.02	1.57	2.16	2.16	1.84
Levi	3.94	4.04	4.05	1.91	1.99	1.95	1.62	2.12	2.11	1.71
Schaffer	4.02	4.16	4.16	2.00	2.00	2.00	1.69	2.25	2.25	1.69

6.3.3.2 Anisotropic Weights

This subsection presents the results for MLS reconstructions on stretched grids using an anisotropic weighting scheme. The second-order results using anisotropic weights are shown in Figure 6.33.



(a) Gaussian

(b) Kansa Function

(c) Levi Function

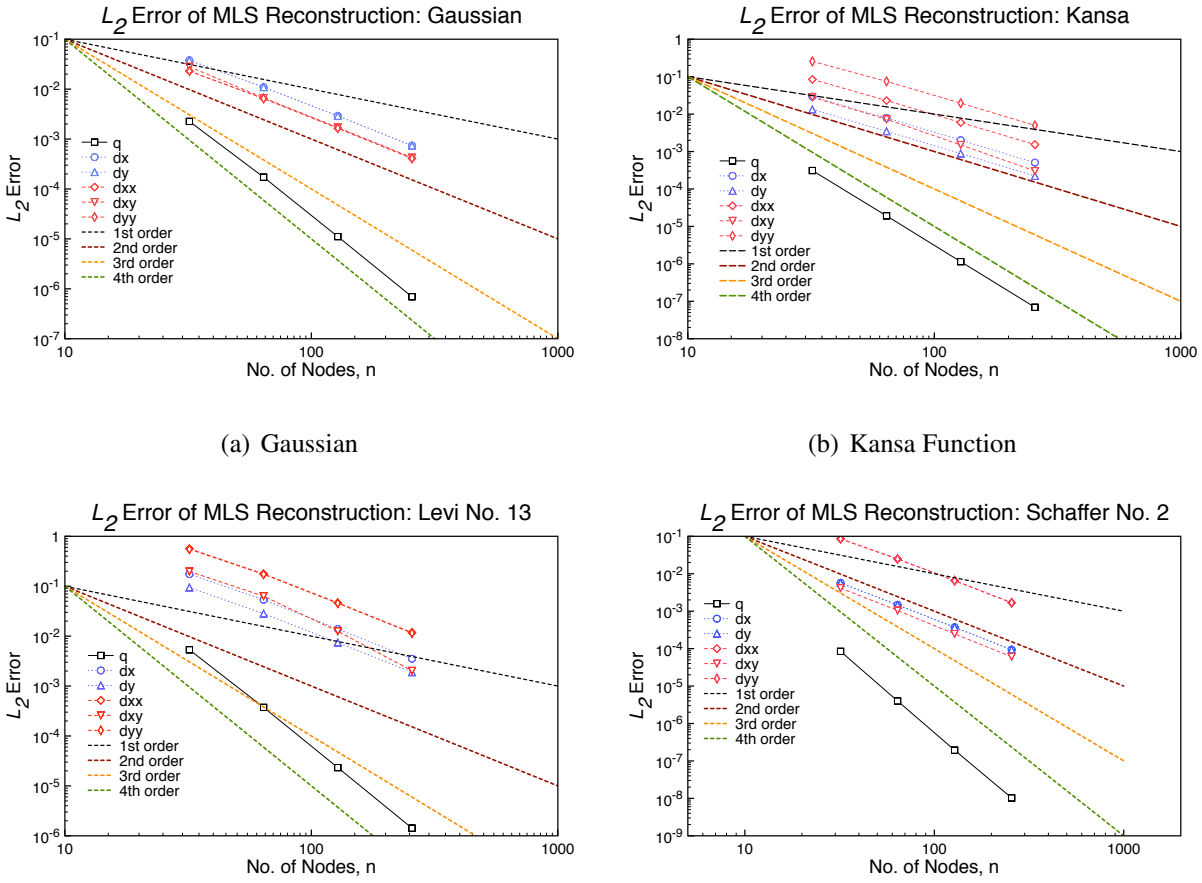
(d) Schaffer Function

Figure 6.33: Error Norms on Stretched Grids: 2^{nd}-Order MLS with Anisotropic Weights.

Figure 6.33 shows that the function and first derivative achieve approximately second-order accuracy. Table 6.41 summarizes the computed order of accuracy for each equation on stretched grids using anisotropic weights. Unlike the structured and unstructured meshes, the anisotropic weighted results show a slight decrease in the order for the derivatives compared the isotropic weights shown in Table 6.38. This is probably due to the difficulty in correctly modeling the derivatives of C for the elliptical weights. As with the isotropic case, the reconstruction performs similar for all the equations. Third-order results are shown in Figure 6.34.

Table 6.41: Accuracy on Stretched Grids: 2nd-Order MLS with Anisotropic Weights.

Equation	Function		
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
Gaussian	1.99	1.93	1.93
Kansa	2.01	1.73	1.71
Levi	1.99	1.73	1.70
Schaffer	2.01	1.72	1.72



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

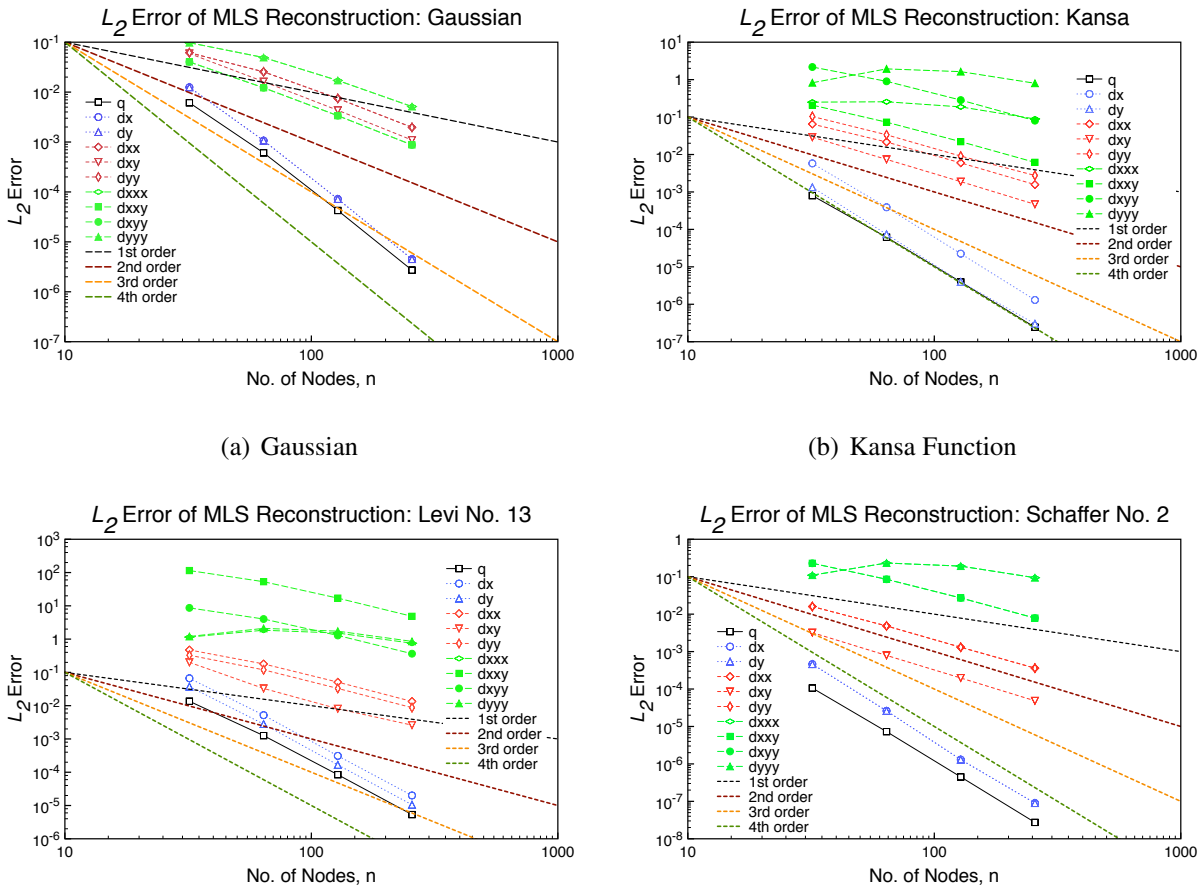
Figure 6.34: Error Norms on Stretched Grids: 3rd-Order MLS with Anisotropic Weights.

Figure 6.34 shows that the function achieves approximately fourth-order accuracy and the first-

and second-derivative achieve approximately second-order accuracy. Table 6.42 summarizes the computed order of accuracy for each equation on stretched grids using anisotropic weights. As with the second-order reconstruction, the third-order reconstruction with anisotropic weights have a slightly lower level of accuracy than the isotropic weights shown in Table 6.39, due to the ability to model the second derivative of C. The reconstruction performs the best for the Schaffer function, and the reconstruction is worse for the Gaussian. Fourth-order results are shown in Figure 6.35.

Table 6.42: Accuracy on Stretched Grids: 3^{rd}-Order MLS with Anisotropic Weights.

Equation	Function					
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
Gaussian	3.98	1.94	1.94	1.99	1.99	1.99
Kansa	4.06	1.98	1.98	1.95	2.29	1.95
Levi	4.02	1.96	1.95	1.95	2.48	1.95
Schaffer	4.30	1.99	1.99	1.94	2.06	1.94



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.35: Error Norms on Stretched Grids: 4^{th}-Order MLS with Anisotropic Weights.

Figure 6.35 shows that the function and first-derivatives achieve approximately fourth-order accuracy, while the second- and third-derivatives are approximately second-order accurate for the Gaussian. The same is true for the other functions, except for the third-derivatives. What appears to be happening is that the third-derivative of C is not accurately being modeled, so the third-derivatives of the function are not well captured. If these terms are the fault of the error, the diffuse derivative tests should show the issue removed. Table 6.43 summarizes the computed order of accuracy for each equation on stretched grids using anisotropic weights. The same trends from the second- and third-order results are seen here with a slight decrease in the order for each derivatives as compared to the isotropic weighted results. .As with the second- and third-order reconstruction,

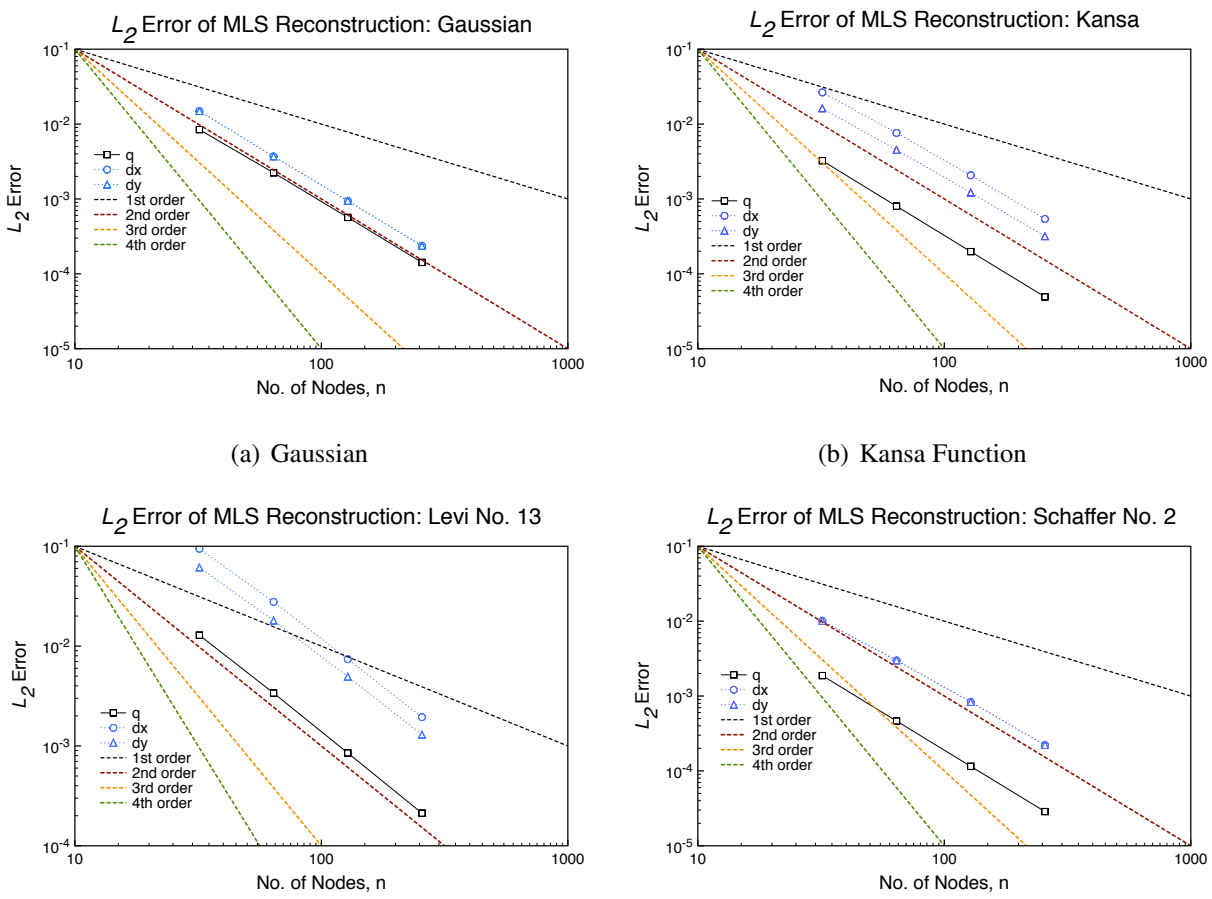
the fourth-order reconstruction with anisotropic weights produces the same level of accuracy as the isotropic weighted MLS shown in Table 6.40.

Table 6.43: Accuracy on Stretched Grids: 4^{th}-Order MLS with Anisotropic Weights.

Equation	Function									
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial xy^2}$	$\frac{\partial^3 q}{\partial y^3}$
Gaussian	3.90	3.94	3.94	1.85	1.94	1.85	1.63	1.90	1.90	1.63
Kansa	3.99	4.12	3.94	1.89	1.98	1.90	0.76	1.78	1.75	0.63
Levi	3.95	4.07	4.06	1.87	2.00	1.89	0.65	1.73	1.73	0.65
Schaffer	4.03	4.29	4.29	1.91	2.00	1.91	0.65	1.72	1.72	0.65

6.3.3.3 Affine MLS: Isotropic Weights

This subsection presents the results for Affine MLS reconstructions on stretched grids using an isotropic weighting scheme. The second-order results using Affine MLS with isotropic weights are shown in Figure 6.36.



(a) Gaussian

(b) Kansa Function

(c) Levi Function

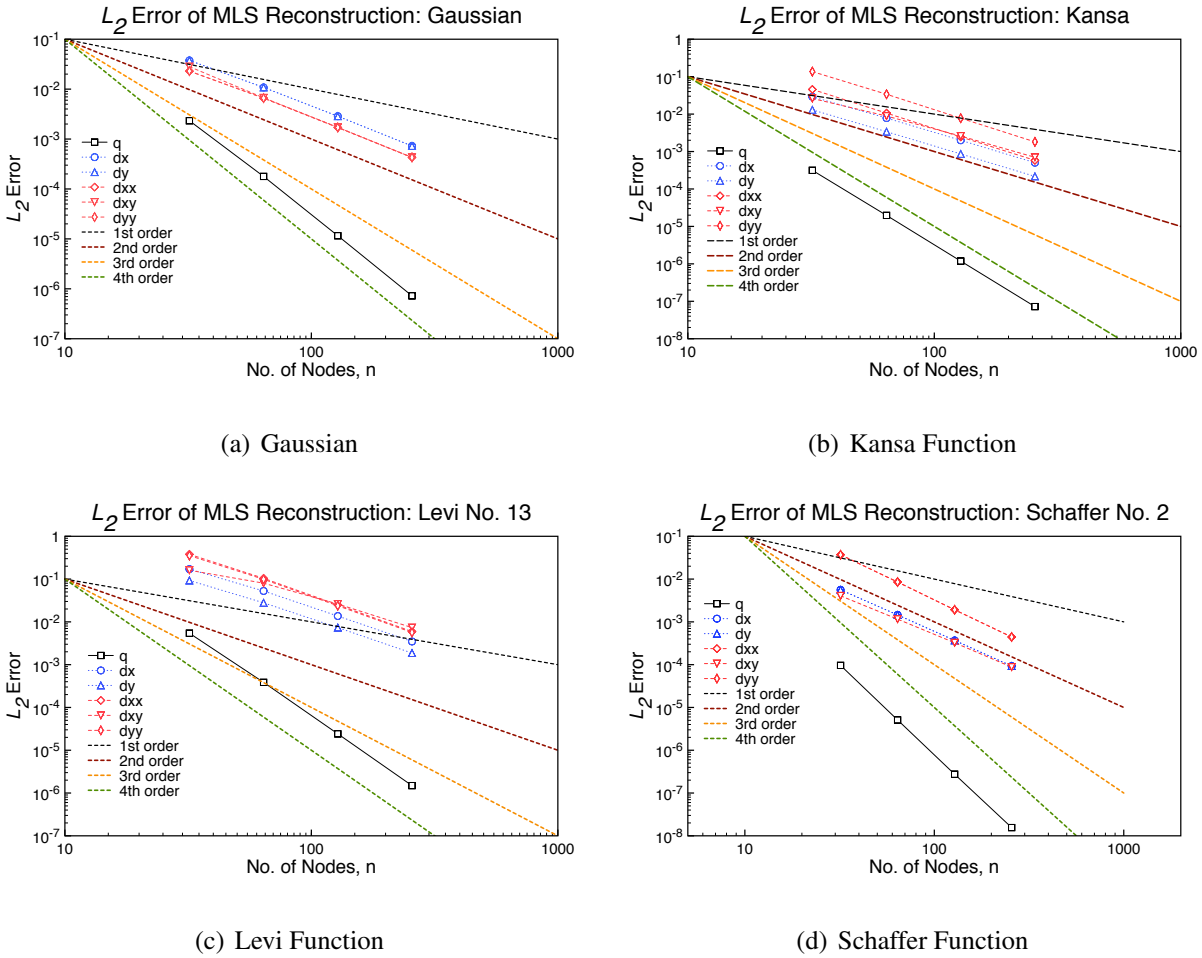
(d) Schaffer Function

Figure 6.36: Error Norms on Stretched Grids: 2^{nd}-Order Affine MLS with Isotropic Weights.

Figure 6.36 shows that the reconstruction and first-derivatives achieve approximately second-order accuracy. Table 6.44 summarizes the computed order of accuracy for each equation on stretched grids using Affine MLS with isotropic weights. As with the previous two methods, Affine MLS in general produces second-order results for both the function and first derivatives. Affine MLS also produces the same results as general method using isotropic weights as presented in Tables 6.38. As with the previous two sections, the reconstruction performs similarly for all equations. Third-order results are shown in Figure 6.37.

Table 6.44: Accuracy on Stretched Grids: 2^{nd}-Order Affine MLS with Isotropic Weights.

Equation	Function		
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
Gaussian	1.99	1.98	1.98
Kansa	2.01	1.91	1.92
Levi	2.00	1.92	1.90
Schaffer	2.01	1.88	1.88



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.37: Error Norms on Stretched Grids: 3^{rd}-Order Affine MLS with Isotropic Weights.

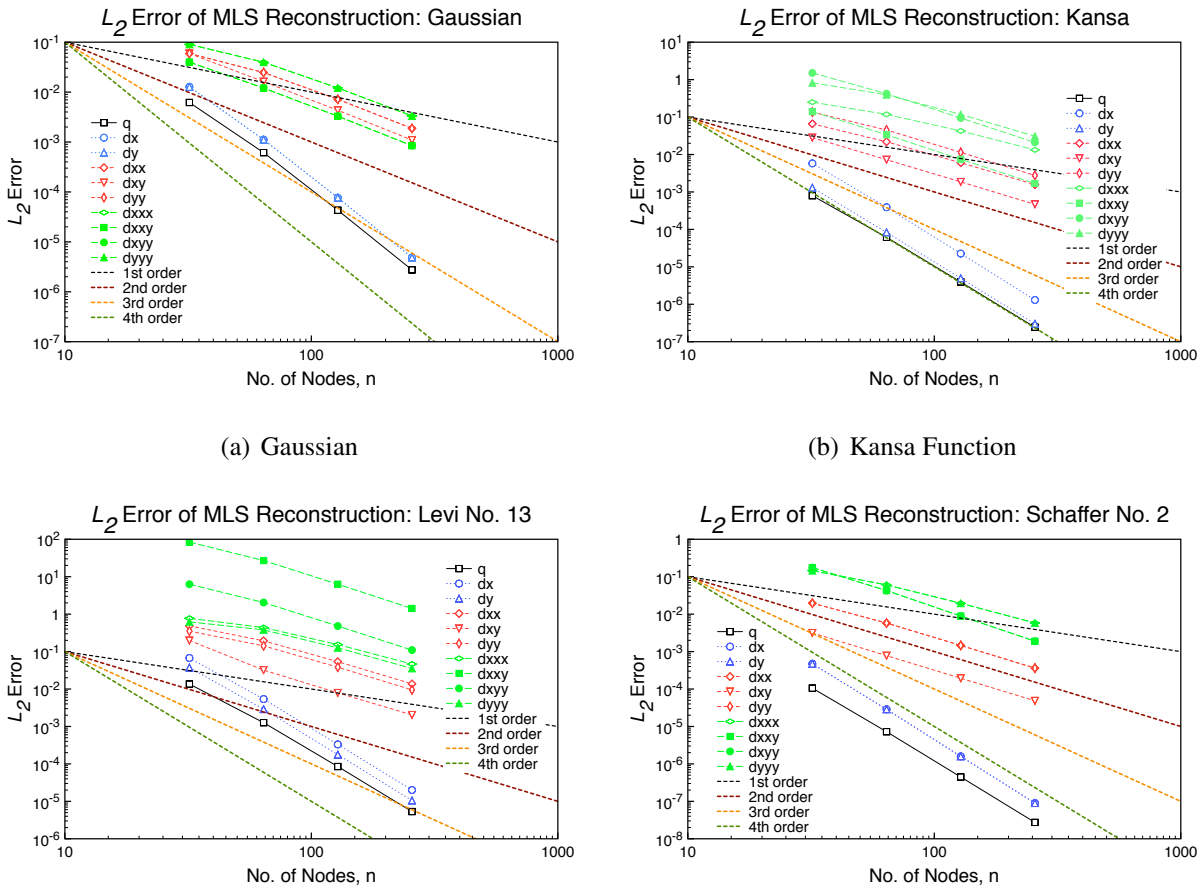
Figure 6.37 shows that the function is approximately fourth-order accurate, while the first-

and second-order derivatives are approximately second-order accurate. Table 6.45 summarizes the computed order of accuracy for each equation using Affine MLS on stretched grids with isotropic weights. As with the second-order reconstruction, Affine MLS reproduces the same level of accuracy as the isotropic weighted MLS as shown in Table 6.39. The reconstruction performs best for the Schaffer function, and it performs worse for the Gaussian. Fourth-order results are shown in

Table 6.45: Accuracy on Stretched Grids: 3^{rd}-Order Affine MLS with Isotropic Weights.

Equation	Function					
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
Gaussian	3.97	1.95	1.95	1.97	1.98	1.97
Kansa	4.05	1.98	1.98	2.10	1.83	2.11
Levi	4.01	1.96	1.95	2.07	1.72	2.06
Schaffer	4.18	1.98	1.98	2.13	1.86	2.13

Figure 6.38.



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.38: Error Norms on Stretched Grids: 4^{th}-Order Affine MLS with Isotropic Weights.

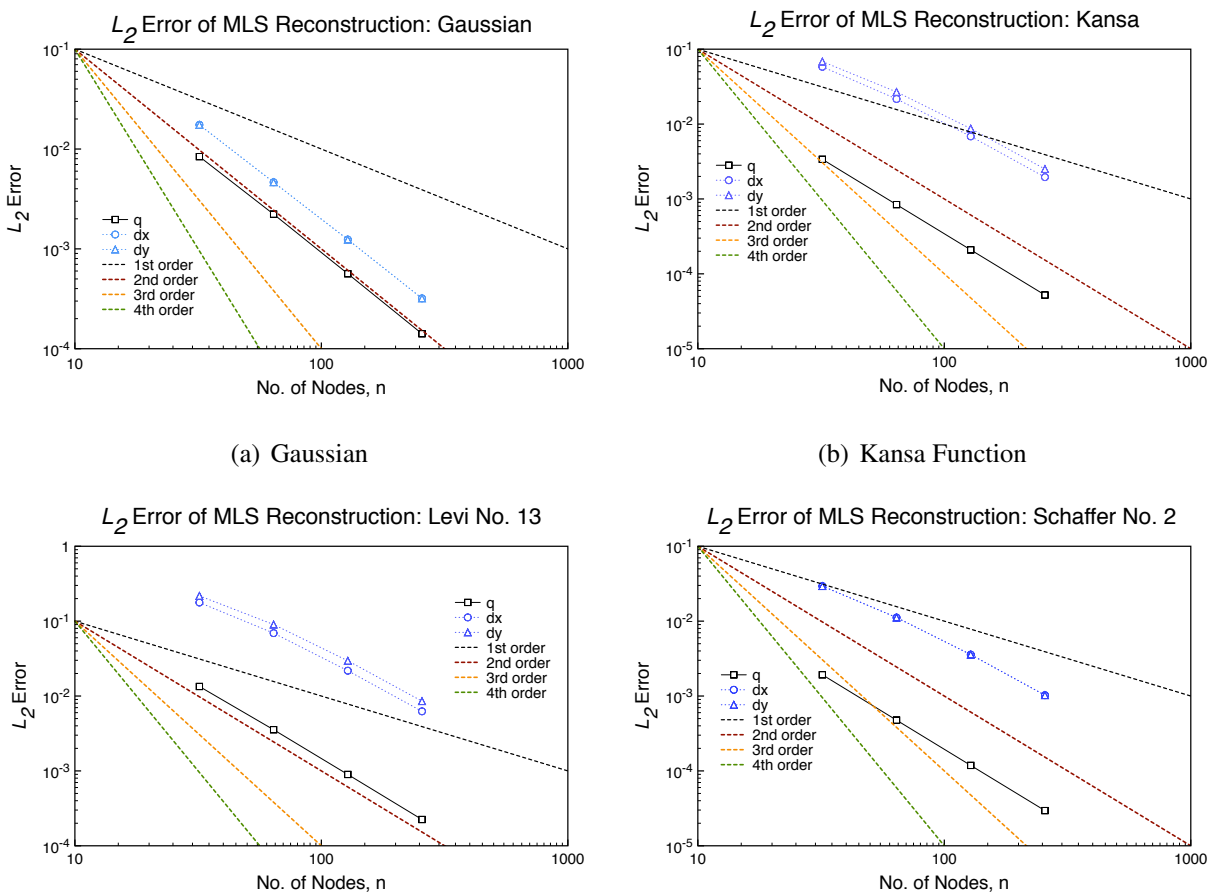
Figure 6.38 shows that the reconstruction of the function and first-derivatives achieves approximately fourth-order accuracy, while the second- and third-derivatives are approximately second-order accurate. Table 6.46 summarizes the computed order of accuracy for each equation using Affine MLS on stretched grids with isotropic weights. Affine MLS reproduces the same level of accuracy as the isotropic weighted MLS as shown in Table 6.40. As with the previous sections, the reconstruction performs best for the Schaffer function, and it performs worse for the Gaussian.

Table 6.46: Accuracy on Stretched Grids: 4^{th}-Order Affine MLS with Isotropic Weights.

Equation	Function									
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial x y^2}$	$\frac{\partial^3 q}{\partial y^3}$
Gaussian	3.90	3.92	3.92	1.86	1.94	1.86	1.79	1.92	1.92	1.79
Kansa	3.99	4.11	4.06	1.90	1.98	2.02	1.57	2.16	2.16	1.84
Levi	3.94	4.04	4.05	1.91	1.99	1.95	1.62	2.12	2.11	1.71
Schaffer	4.02	4.16	4.16	2.00	2.00	2.00	1.69	2.25	2.25	1.69

6.3.3.4 Affine MLS: Anisotropic Weights

This subsection presents the results for Affine MLS reconstructions on stretched grids using an anisotropic weighting scheme. The second-order results using Affine MLS with anisotropic weights are shown in Figure 6.39.



(a) Gaussian

(b) Kansa Function

(c) Levi Function

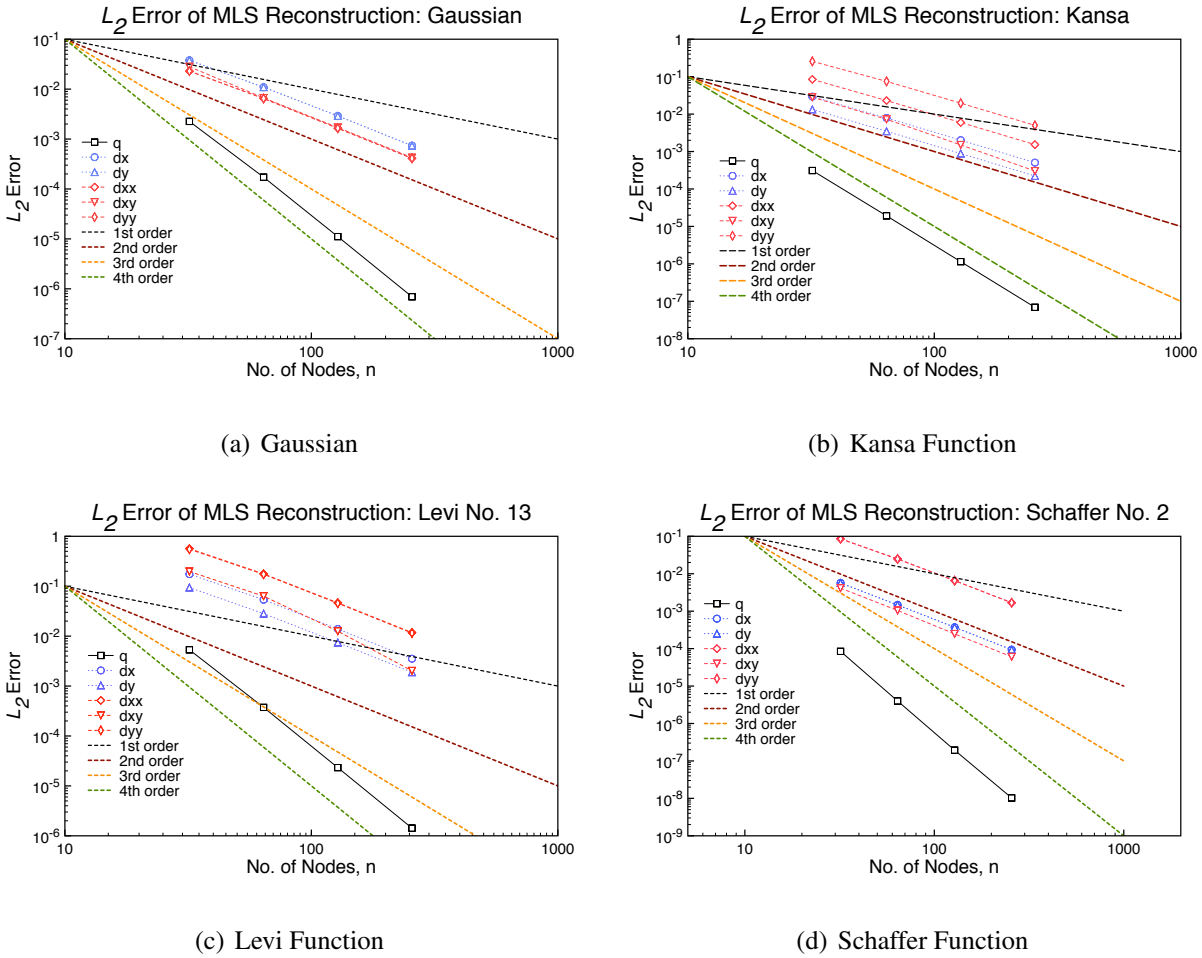
(d) Schaffer Function

Figure 6.39: Error Norms on Stretched Grids: 2^{nd}-Order Affine MLS with Anisotropic Weights.

Figure 6.39 shows that the reconstruction and first-derivatives achieve approximately second-order accuracy. Table 6.47 summarizes the computed order of accuracy for each equation on stretched grids using Affine MLS with anisotropic weights. As with the previous methods, Affine MLS in general produces second-order results for both the function and first derivatives. Affine MLS also produces the same results as general method using anisotropic weights as presented in Table 6.41. As with the previous two sections, the reconstruction performs similarly for all equations. Third-order results are shown in Figure 6.40.

Table 6.47: Accuracy on Stretched Grids: 2^{nd}-Order Affine MLS with Anisotropic Weights.

Equation	Function		
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
Gaussian	1.99	1.93	1.93
Kansa	2.01	1.73	1.71
Levi	1.99	1.73	1.70
Schaffer	2.01	1.72	1.72



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.40: Error Norms on Stretched Grids: 3^{rd}-Order Affine MLS with Anisotropic Weights.

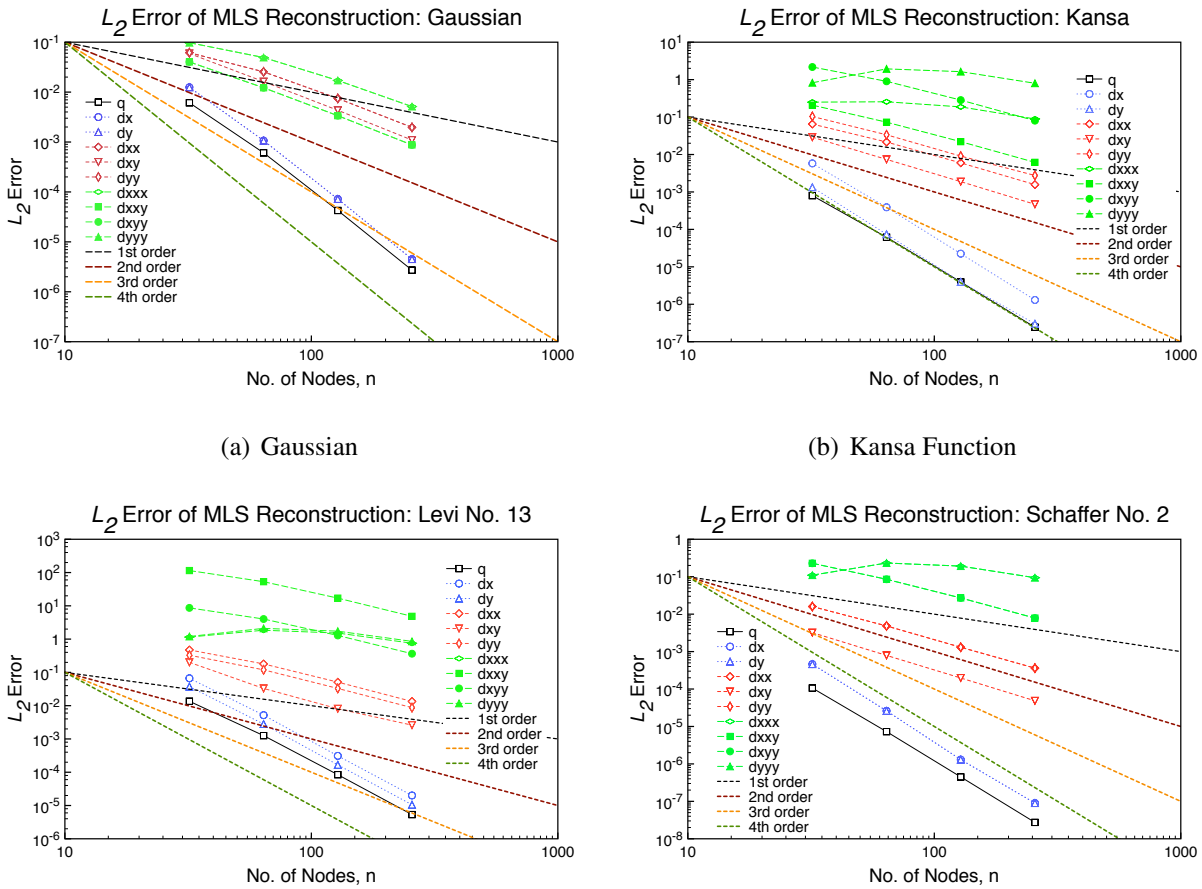
Figure 6.40 shows that the function is approximately fourth-order accurate, while the first-

and second-order derivatives are approximately second-order accurate. Table 6.48 summarizes the computed order of accuracy for each equation using Affine MLS on stretched grids with anisotropic weights. As with the second-order reconstruction, Affine MLS reproduces the same level of accuracy as the anisotropic weighted MLS as shown in Table 6.42. The reconstruction performs best for the Schaffer function, and it performs worse for the Gaussian. Fourth-order results are shown

Table 6.48: Accuracy on Stretched Grids: 3^{rd}-Order Affine MLS with Anisotropic Weights.

Equation	Function					
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
Gaussian	3.98	1.94	1.94	1.99	1.99	1.99
Kansa	4.06	1.98	1.98	1.95	2.29	1.95
Levi	4.02	1.96	1.95	1.95	2.48	1.95
Schaffer	4.30	1.99	1.99	1.94	2.06	1.94

in Figure 6.41.



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.41: Error Norms on Stretched Grids: 4^{th}-Order Affine MLS with Anisotropic Weights.

Figure 6.41 shows that the reconstruction of the function and first-derivatives achieves approximately fourth-order accuracy, while the second- and third-derivatives are approximately second-order accurate for the Gaussian, and the same issues presented for the other functions in Figure 6.35 are seen here. It should be noted then that the Affine MLS method will only produce a more accurate inverse, not necessarily more accurate derivatives. Table 6.49 summarizes the computed order of accuracy for each equation using Affine MLS on stretched grids with anisotropic weights. Affine MLS reproduces the same level of accuracy as the anisotropic weighted MLS as shown in Table 6.40.

Table 6.49: Accuracy on Stretched Grids: 4^{th}-Order Affine MLS with Anisotropic Weights.

Equation	Function									
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial xy^2}$	$\frac{\partial^3 q}{\partial y^3}$
Gaussian	3.90	3.94	3.94	1.85	1.94	1.85	1.63	1.90	1.90	1.63
Kansa	3.99	4.12	3.94	1.89	1.98	1.90	0.76	1.78	1.75	0.63
Levi	3.95	4.07	4.06	1.87	2.00	1.89	0.65	1.73	1.73	0.65
Schaffer	4.03	4.29	4.29	1.91	2.00	1.91	0.65	1.72	1.72	0.65

6.3.3.5 Summary

After computing all of the reconstructions using the various weighting strategies, it is worth comparing the accuracy and error computed from each strategy for each function. Table 6.50 summarizes the order of accuracy for each of the functions using a second-order MLS reconstruction.

Table 6.50: Accuracy on Stretched Grids: 2^{nd}-Order MLS Weight Comparison.

Equation	Weight	Function		
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
Gaussian	Iso.	1.99	1.98	1.98
	Aniso.	1.99	1.93	1.93
	Affine (1)	1.99	1.98	1.98
	Affine (2)	1.99	1.93	1.93
Kansa	Iso.	2.01	2.02	2.01
	Aniso.	2.01	1.73	1.71
	Affine (1)	2.01	1.91	1.92
	Affine (2)	2.01	1.73	1.71
Levi	Iso.	2.00	1.92	1.90
	Aniso.	1.99	1.73	1.70
	Affine (1)	2.00	1.92	1.90
	Affine (2)	1.99	1.73	1.70
Schaffer	Iso.	2.01	1.88	1.88
	Aniso.	2.01	1.72	1.72
	Affine (1)	2.01	1.88	1.88
	Affine (2)	2.01	1.72	1.72

Table 6.50 shows that the weighting strategy has a noticeable effect on the accuracy for the first derivatives. The first derivatives using the anisotropic weighting strategy have a lower order of accuracy than the isotropically weighted reconstructions. Additionally, it is worth pointing out that the reconstruction of the Levi function varies based on the weighting strategy, but only slightly so. Table 6.51 shows the L_2 error for each of the functions on the finest mesh. Table 6.51

Table 6.51: L_2 Error on Finest Stretched Grid: 2^{nd}-Order MLS Weight Comparison.

Equation	Weight	Function		
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
Gaussian	Iso.	1.42E-4	2.36E-4	2.36E-4
	Aniso.	1.41E-4	3.19E-4	3.19E-4
	Affine (1)	1.42E-4	2.36E-4	2.36E-4
	Affine (2)	1.41E-4	3.19E-4	3.19E-4
Kansa	Iso.	4.92E-5	5.41E-4	3.17E-4
	Aniso.	5.21E-5	1.95E-3	2.50E-3
	Affine (1)	4.92E-5	5.41E-4	3.17E-4
	Affine (2)	5.21E-5	1.95E-3	2.50E-3
Levi	Iso.	2.12E-4	1.94E-3	1.29E-3
	Aniso.	2.24E-4	6.24E-3	8.59E-3
	Affine (1)	2.12E-4	1.94E-3	1.29E-3
	Affine (2)	2.24E-4	6.24E-3	8.53E-3
Schaffer	Iso.	2.86E-5	2.21E-4	2.21E-4
	Aniso.	2.95E-5	1.02E-3	1.02E-3
	Affine (1)	2.86E-5	2.21E-4	2.21E-4
	Affine (2)	2.95E-5	1.02E-3	1.02E-3

shows that the anisotropic errors are higher for all of the functions except the Gaussian, but the difference is only noticeable for the first derivatives, where the difference is almost an order of magnitude. However, this difference is still only in the 1E-3 range, which is not significantly large for second-order schemes. Table 6.52 summarizes the order of accuracy for each of the functions using a third-order MLS reconstruction. Table 6.52 shows that the weighting strategy does have a noticeable effect on the accuracy of the reconstruction. For all the functions, the functions are

Table 6.52: Accuracy on Stretched Grids: 3^{rd}-Order MLS Weight Comparison.

Equation	Weight	Function					
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
Gaussian	Iso.	3.97	1.95	1.95	1.97	1.98	1.97
	Aniso.	3.98	1.94	1.94	1.99	1.99	1.99
	Affine (1)	3.97	1.95	1.95	1.97	1.98	1.97
	Affine (2)	3.98	1.94	1.94	1.99	1.99	1.99
Kansa	Iso.	4.05	1.98	1.98	2.10	1.83	2.11
	Aniso.	4.06	1.98	1.98	1.95	2.29	1.95
	Affine (1)	4.05	1.98	1.98	2.10	1.83	2.11
	Affine (2)	4.06	1.98	1.98	1.95	2.29	1.95
Levi	Iso.	4.01	1.96	1.95	2.07	1.72	2.06
	Aniso.	4.02	1.96	1.95	1.95	2.48	1.95
	Affine (1)	4.01	1.96	1.95	2.07	1.72	2.06
	Affine (2)	4.02	1.96	1.95	1.95	2.48	1.95
Schaffer	Iso.	4.18	1.98	1.98	2.13	1.86	2.13
	Aniso.	4.30	1.99	1.99	1.94	2.06	1.94
	Affine (1)	4.18	1.98	1.98	2.13	1.86	2.13
	Affine (2)	4.30	1.99	1.99	1.94	2.06	1.94

reconstructed better while the first derivatives are of the same order as the isotropic weighted ones, though for the Gaussian they are slightly lower. The second derivatives show some variation, but which is better is difficult to discern from these results. The exception to this is the Gaussian, where the anisotropically weighted reconstructions have a higher accuracy. Table 6.53 shows the L_2 error for each of the functions on the finest mesh.

Table 6.53: L_2 Error on Finest Stretched Grid: 3^{rd}-Order MLS Weight Comparison.

Equation	Weight	Function					
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
Gaussian	Iso.	7.23E-7	7.26E-4	7.26E-4	4.28E-4	4.29E-4	4.28E-4
	Aniso.	6.93E-7	7.37E-4	7.37E-4	4.10E-4	4.25E-4	4.10E-4
	Affine (1)	7.23E-7	7.26E-4	7.26E-4	4.28E-4	4.29E-4	4.28E-4
	Affine (2)	6.93E-7	7.37E-4	7.37E-4	4.10E-4	4.25E-4	4.10E-4
Kansa	Iso.	7.21E-8	5.02E-4	2.17E-4	5.81E-4	7.00E-4	1.81E-3
	Aniso.	6.91E-8	5.05E-4	2.22E-4	1.53E-3	3.02E-4	4.98E-3
	Affine (1)	7.21E-8	5.02E-4	2.17E-4	5.81E-4	7.00E-4	1.81E-3
	Affine (2)	6.91E-8	5.05E-4	2.22E-4	1.53E-3	3.02E-4	4.98E-3
Levi	Iso.	1.49E-6	3.47E-3	1.86E-3	5.94E-3	7.34E-3	5.66E-3
	Aniso.	1.42E-6	3.53E-3	1.88E-3	1.17E-2	2.02E-3	1.15E-2
	Affine (1)	1.49E-6	3.47E-3	1.86E-3	5.94E-3	7.34E-3	5.66E-3
	Affine (2)	1.42E-6	3.53E-3	1.88E-3	1.17E-2	2.02E-3	1.15E-2
Schaffer	Iso.	1.55E-8	9.28E-5	9.28E-5	4.43E-4	8.77E-5	4.43E-4
	Aniso.	1.02E-8	9.36E-5	9.36E-5	1.67E-3	6.04E-5	1.67E-3
	Affine (1)	1.55E-8	9.28E-5	9.28E-5	4.43E-4	8.77E-5	4.43E-4
	Affine (2)	1.02E-8	9.36E-5	9.36E-5	1.67E-3	6.04E-5	1.67E-3

The error on the finest grid shows that the anisotropic produces lower error for the function alone, but slightly higher error for the derivatives, at most 5E-3 for the second-derivatives. As with the second-order results in Table 6.51, this difference is low. Table 6.54 summarizes the order of accuracy for each of the functions using a third-order MLS reconstruction. Table 6.54 shows that the weighting strategy does again have a noticeable effect on the accuracy of the reconstruction. Generally, the functions and derivatives are reconstructed to a higher accuracy when using

Table 6.54: Accuracy on Stretched Grids: 4^{th}-Order MLS Weight Comparison.

Equation	Weight	Function									
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial xy^2}$	$\frac{\partial^3 q}{\partial y^3}$
Gaussian	Iso.	3.90	3.92	3.92	1.86	1.94	1.86	1.79	1.92	1.92	1.79
	Aniso.	3.90	3.94	3.94	1.85	1.94	1.85	1.63	1.90	1.90	1.63
	Affine (1)	3.90	3.92	3.92	1.86	1.94	1.86	1.79	1.92	1.92	1.79
	Affine (2)	3.90	3.94	3.94	1.85	1.94	1.85	1.63	1.90	1.90	1.63
Kansa	Iso.	3.99	4.11	4.06	1.90	1.98	2.02	1.57	2.16	2.16	1.84
	Aniso.	3.99	4.12	3.94	1.89	1.98	1.90	0.76	1.78	1.75	0.63
	Affine (1)	3.99	4.11	4.06	1.90	1.98	2.02	1.57	2.16	2.16	1.84
	Affine (2)	3.99	4.12	3.94	1.89	1.98	1.90	0.76	1.78	1.75	0.63
Levi	Iso.	3.94	4.04	4.05	1.91	1.99	1.95	1.62	2.12	2.11	1.71
	Aniso.	3.95	4.07	4.06	1.87	2.00	1.89	0.65	1.73	1.73	0.65
	Affine (1)	3.94	4.04	4.05	1.91	1.99	1.95	1.62	2.12	2.11	1.71
	Affine (2)	3.95	4.07	4.06	1.87	2.00	1.89	0.65	1.73	1.73	0.65
Schaffer	Iso.	4.02	4.16	4.16	2.00	2.00	2.00	1.69	2.25	2.25	1.69
	Aniso.	4.03	4.29	4.29	1.91	2.00	1.91	0.65	1.72	1.72	0.65
	Affine (1)	4.02	4.16	4.16	2.00	2.00	2.00	1.69	2.25	2.25	1.69
	Affine (2)	4.03	4.29	4.29	1.91	2.00	1.91	0.65	1.72	1.72	0.65

the anisotropic weighting strategy. For the higher derivatives, generally the isotropic weighted reconstructions perform better. Tables 6.55 and 6.56 show the L_2 error. Table 6.55 shows that the anisotropic weighted reconstructions produce lower error for the function and first derivatives. For the second derivatives, it depends on the function as to which weighing strategy does best. For the third derivatives, Table 6.56 shows that the isotropically weighted approach always has the lowest error. However, the difference is not large for the Gaussian and Schaffer functions, and with the exception of the $\frac{\partial^3 q}{\partial y^3}$, the Kansa function as well. The third derivatives of the Levi function using the anisotropic weighting strategy, however, are always an order of magnitude worse than the isotropically weighted one. As speculated in the previous section, is this most likely due to the difficultly in accurately computing the highest derivatives of $\mathbf{C}$. Overall, though, the differences for the highest derivatives should not overly affect the solution of a fourth-order scheme, but it is difficult on the stretched grid to say which weighing strategy is better. To make a final judgment on the best weighting strategy for the stretched mesh, the condition numbers will be computed in

Table 6.55: L_2 Error on Finest Stretched Grid: 4^{th}-Order MLS Weight Comparison, State and First and Second Derivatives.

Equation	Weight	Function					
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
Gaussian	Iso.	2.74E-6	4.86E-6	4.86E-6	1.87E-3	1.09E-3	1.87E-3
	Aniso.	2.71E-6	4.51E-6	4.51E-6	1.96E-3	1.09E-3	1.96E-3
	Affine (1)	2.74E-6	4.86E-6	4.86E-6	1.87E-3	1.09E-3	1.87E-3
	Affine (2)	2.71E-6	4.51E-6	4.51E-6	1.96E-3	1.09E-3	1.96E-3
Kansa	Iso.	2.43E-7	1.30E-6	3.01E-7	1.57E-3	4.67E-4	2.75E-3
	Aniso.	2.43E-7	1.29E-6	3.17E-7	1.56E-3	4.74E-4	2.40E-3
	Affine (1)	2.43E-7	1.30E-6	3.01E-7	1.57E-3	4.67E-4	2.75E-3
	Affine (2)	2.43E-7	1.29E-6	3.17E-7	1.56E-3	4.74E-4	2.40E-3
Levi	Iso.	5.34E-6	2.00E-5	1.05E-5	1.37E-2	2.03E-3	9.50E-3
	Aniso.	5.29E-6	1.85E-5	1.00E-5	1.34E-2	2.06E-3	8.72E-3
	Affine (1)	5.34E-6	2.00E-5	1.05E-5	1.37E-2	2.03E-3	9.50E-3
	Affine (2)	5.29E-6	1.85E-5	1.00E-5	1.34E-2	2.06E-3	8.72E-3
Schaffer	Iso.	2.74E-8	8.98E-8	8.98E-8	3.62E-4	4.85E-5	3.61E-4
	Aniso.	2.74E-8	6.85E-8	6.854E-8	3.44E-4	4.92E-5	3.44E-4
	Affine (1)	2.74E-8	8.98E-8	8.98E-8	3.62E-4	4.85E-5	3.62E-4
	Affine (2)	2.74E-8	6.85E-8	6.854E-8	3.44E-4	4.92E-5	3.44E-4

Section 6.6.

6.4 MLS Diffusive Derivatives

This section presents the order of accuracy of the various diffusive MLS derivatives described in Sections 5.1.5.2 and 5.1.5.3. Timings of the reconstruction methods are given when evaluating the semi- and fully-diffusive derivatives, but only for when evaluating the Gaussian function, since the computation of the shape functions is independent of the function reconstructed. The timings are performed on a Mac Pro with 24 GB of memory and a 2.26 GHz Quad-Core Intel Xeon processor. All of the weights are used. The scaling parameter in this section is $k = 0.8$, and the polynomial basis is the standard monomial basis. The inversion method of choice for this study is the Pivoting QR method, since it always produced the minimum condition number for $\mathbf{M}$ for each of the grids, as shown in Section 6.2. The fully and semi-diffuse derivative orders of accuracy are directly compared to the full derivatives for each of the meshes and orders of accuracy when a

Table 6.56: L_2 Error on Finest Stretched Grid: 4^{th}-Order MLS Weight Comparison, Third Derivatives.

Equation	Weight	Function			
		$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial x y^2}$	$\frac{\partial^3 q}{\partial y^3}$
Gaussian	Iso.	3.32E-3	8.44E-4	8.44E-4	3.32E-3
	Aniso.	5.13E-3	8.74E-4	8.74E-4	5.13E-3
	Affine (1)	3.32E-3	8.44E-4	8.44E-4	3.32E-3
	Affine (2)	5.13E-3	8.74E-4	8.74E-4	5.13E-3
Kansa	Iso.	1.32E-2	1.68E-3	2.09E-2	3.08E-2
	Aniso.	8.91E-2	6.18E-3	8.01E-2	0.80
	Affine (1)	1.32E-2	1.68E-3	2.09E-2	3.08E-2
	Affine (2)	8.91E-2	6.18E-3	8.01E-2	0.80
Levi	Iso.	4.64E-2	1.42	0.10	3.54E-2
	Aniso.	0.75	4.83	0.36	0.85
	Affine (1)	4.64E-2	1.42	0.10	3.54E-2
	Affine (2)	0.75	4.83	0.36	0.85
Schaffer	Iso.	5.83E-3	1.89E-3	1.89E-3	5.83E-3
	Aniso.	9.40E-2	7.85E-3	7.85E-3	9.40E-2
	Affine (1)	5.83E-3	1.89E-3	1.89E-3	5.83E-3
	Affine (2)	9.40E-2	7.85E-3	7.85E-3	9.40E-2

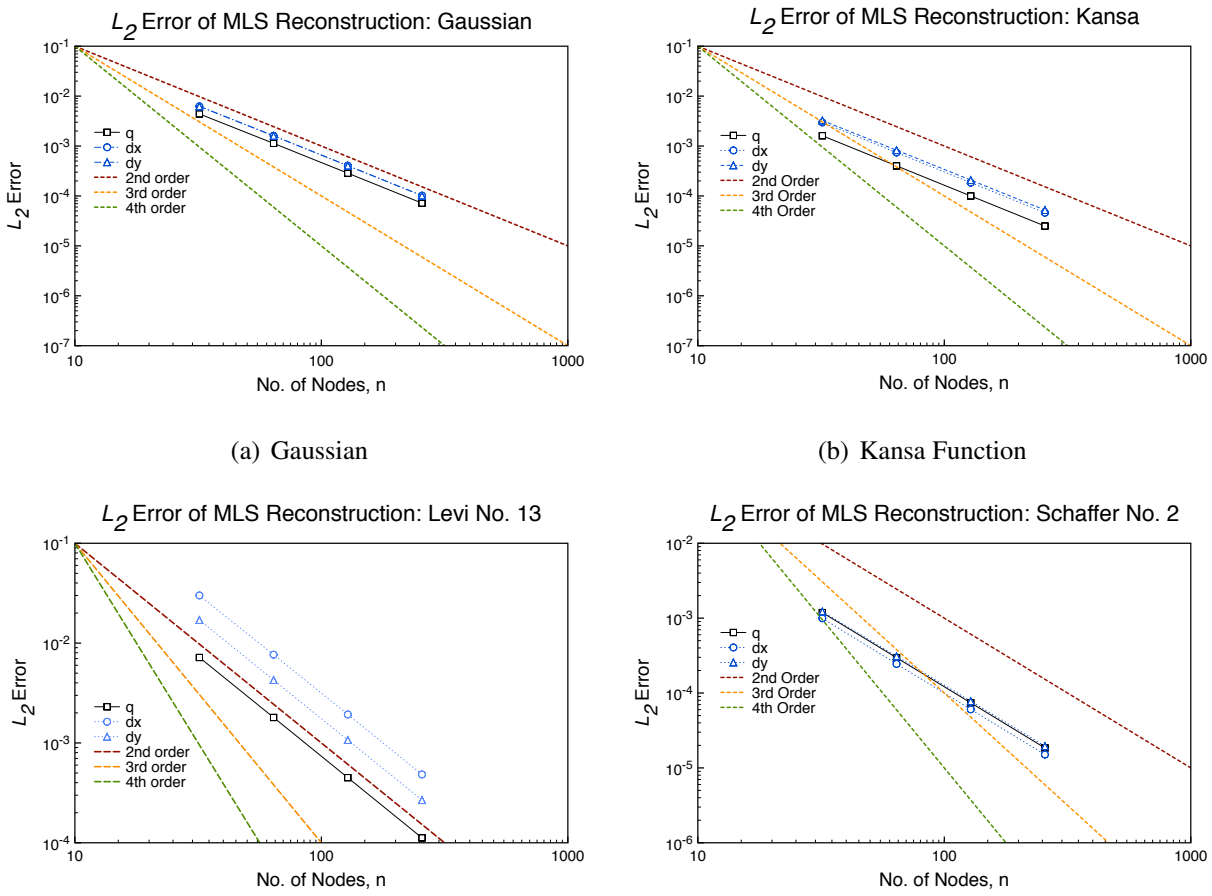
difference is apparent.

6.4.1 Structured Grids

This section presents the results of order of accuracy of the diffusive MLS derivatives on structured grids. For the structured grids, only the isotropic weights are used, as the more advanced weighting methods degenerate to the isotropic case.

6.4.1.1 Second-Order

The second-order MLS reconstruction of the derivatives can be in the full form (5.47) or in the fully-diffuse form (5.72). The full derivatives, reconstructed using isotropic weights, were shown in Table 6.10. The fully-diffuse derivatives reconstructed with isotropic weights are shown in Figure 6.42.



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.42: Error Norms on Structured Grids: 2^{nd}-Order Fully-Diffuse MLS with Isotropic Weights.

Figure 6.42 shows that the reconstruction achieves second-order accuracy for the function and first-derivatives. Table 6.57 summarizes the computed order of accuracy for each equation on structured grids using isotropic weights. The fully-diffuse reconstruction produces second-order results for both the function and first derivatives, which matches the order observed using the full reconstruction shown in Table 6.10.

Table 6.57: Accuracy on Structured Grids: 2^{nd}-Order Fully-Diffuse MLS with Isotropic Weights.

Equation	Function		
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
Gaussian	1.98	1.98	1.98
Kansa	2.00	2.00	1.99
Levi	2.00	1.99	2.00
Schaffer	2.00	2.02	1.99

It is also constructive to look at the timing for the full and fully-diffuse derivatives. Table 6.58 shows the timing for the reconstruction of the Gaussian. As seen from Table 6.58, the second-order

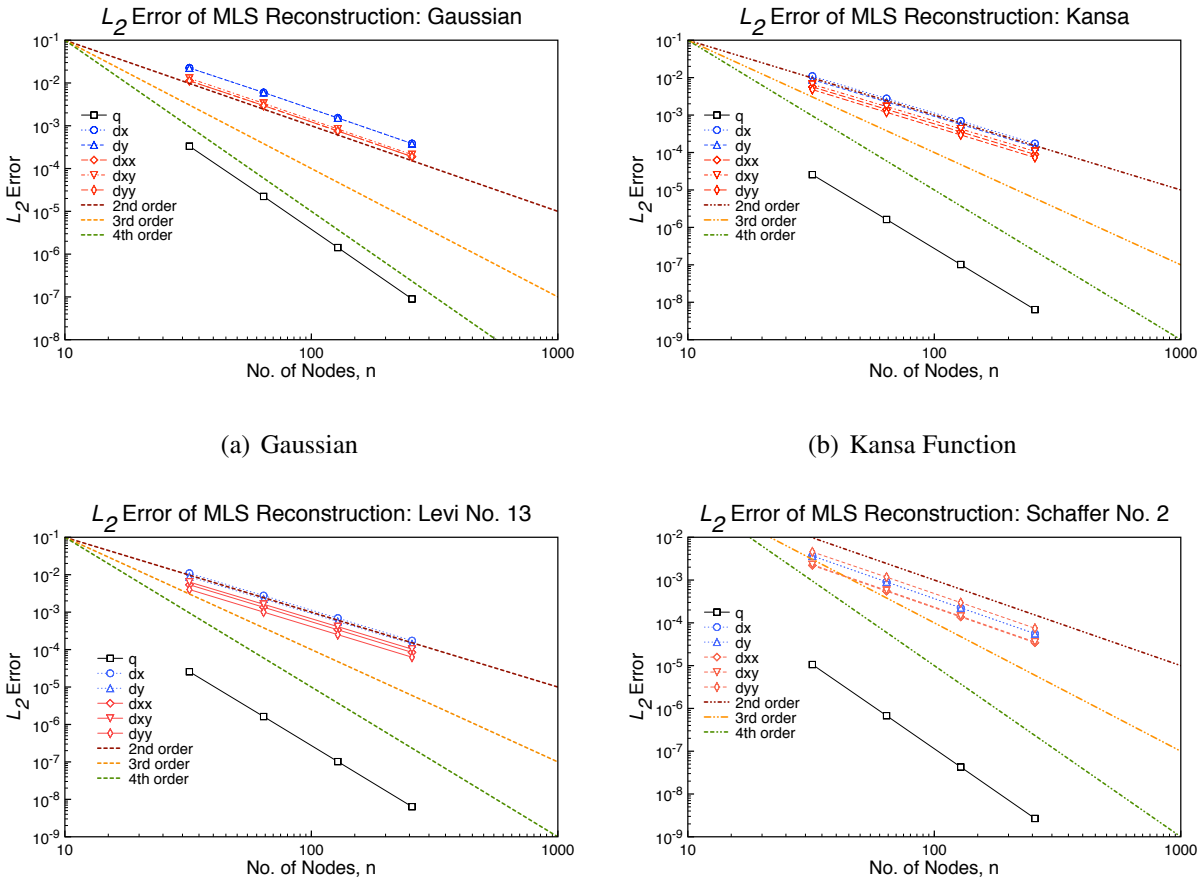
Table 6.58: Timing on Structured Grids: 2^{nd}-Order MLS with Isotropic Weights, in Seconds.

Mesh Size	Type	
	Full	Full-Diffuse
32	0.038	0.032
64	0.136	0.120
128	0.537	0.457
256	2.106	1.836

fully-diffuse is only slightly faster than the full reconstruction, so it is not overly beneficial to use the fully-diffuse.

6.4.1.2 Third-Order

The third-order MLS reconstruction of the derivatives can be in the full form (5.47), semi-diffuse (5.68), or in the fully-diffuse form (5.72). The semi-diffuse derivatives reconstructed with isotropic weights are shown in Figure 6.43.



(a) Gaussian

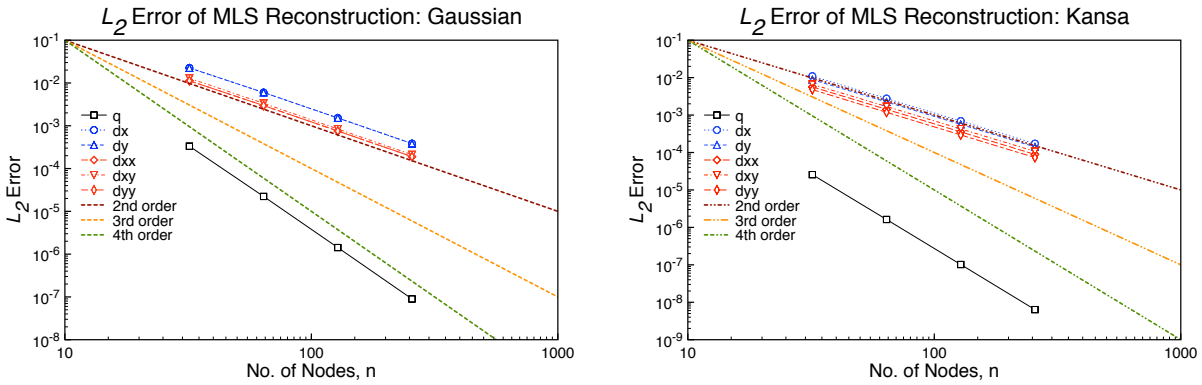
(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

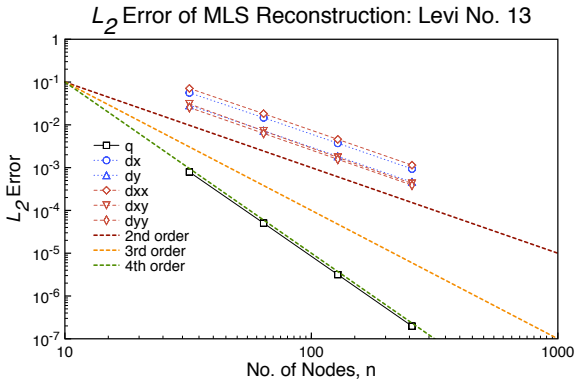
Figure 6.43: Error Norms on Structured Grids: 3^{rd}-Order Semi-Diffuse MLS with Isotropic Weights.

Figure 6.43 shows that the accuracy for the function is approximately fourth-order accurate, and the first- and second-order derivatives is second-order accurate. The fully-diffuse derivatives reconstructed with isotropic weights are shown in Figure 6.44.

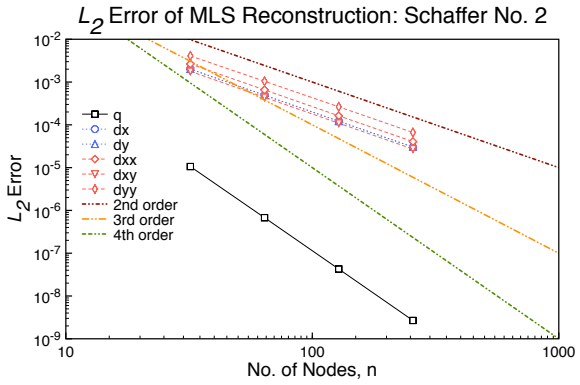


(a) Gaussian

(b) Kansa Function



(c) Levi Function



(d) Schaffer Function

Figure 6.44: Error Norms on Structured Grids: 3rd-Order Fully-Diffuse MLS with Isotropic Weights.

Figure 6.44 shows that the accuracy for the function is approximately fourth-order accurate, and the first- and second-order derivatives is second-order accurate. Table 6.59 summarizes the computed order of accuracy for each of the diffuse derivative methods. The accuracy for each of the derivative methods is the same for the function and first derivatives, and slightly different for the second derivatives, but not noticeably so. It is also constructive to look at the timing for the full and fully-diffuse derivatives. Table 6.66 shows the timing for the reconstruction of the Gaussian. As seen from Table 6.66, the fully-diffuse is nearly twice as fast at the highest level of refinement. The fully-diffuse as accurate as the full reconstruction, it makes sense to use the fully-diffuse to

Table 6.59: Accuracy on Structured Grids: 3^{rd}-Order Diffuse MLS with Isotropic Weights. SD=Semi-Diffuse, FD=Fully-Diffuse.

Equation	Type	Function					
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
Gaussian	Full	3.95	1.96	1.96	1.96	1.98	1.96
	SD	3.95	1.96	1.96	1.96	1.98	1.96
	FD	3.95	1.97	1.97	1.96	1.98	1.96
Kansa	Full	3.99	1.99	1.99	2.00	1.98	2.01
	SD	3.99	1.99	1.99	2.00	1.98	2.01
	FD	3.99	2.00	1.99	2.00	1.99	2.00
Levi	Full	3.99	1.96	1.98	1.99	1.97	2.07
	SD	3.99	1.99	1.99	2.00	1.98	2.00
	FD	3.99	1.97	1.99	1.99	2.06	2.01
Schaffer	Full	3.98	2.02	2.01	2.00	2.00	1.97
	SD	3.98	2.02	2.01	2.00	2.00	1.97
	FD	3.98	2.02	2.02	2.01	2.01	1.98

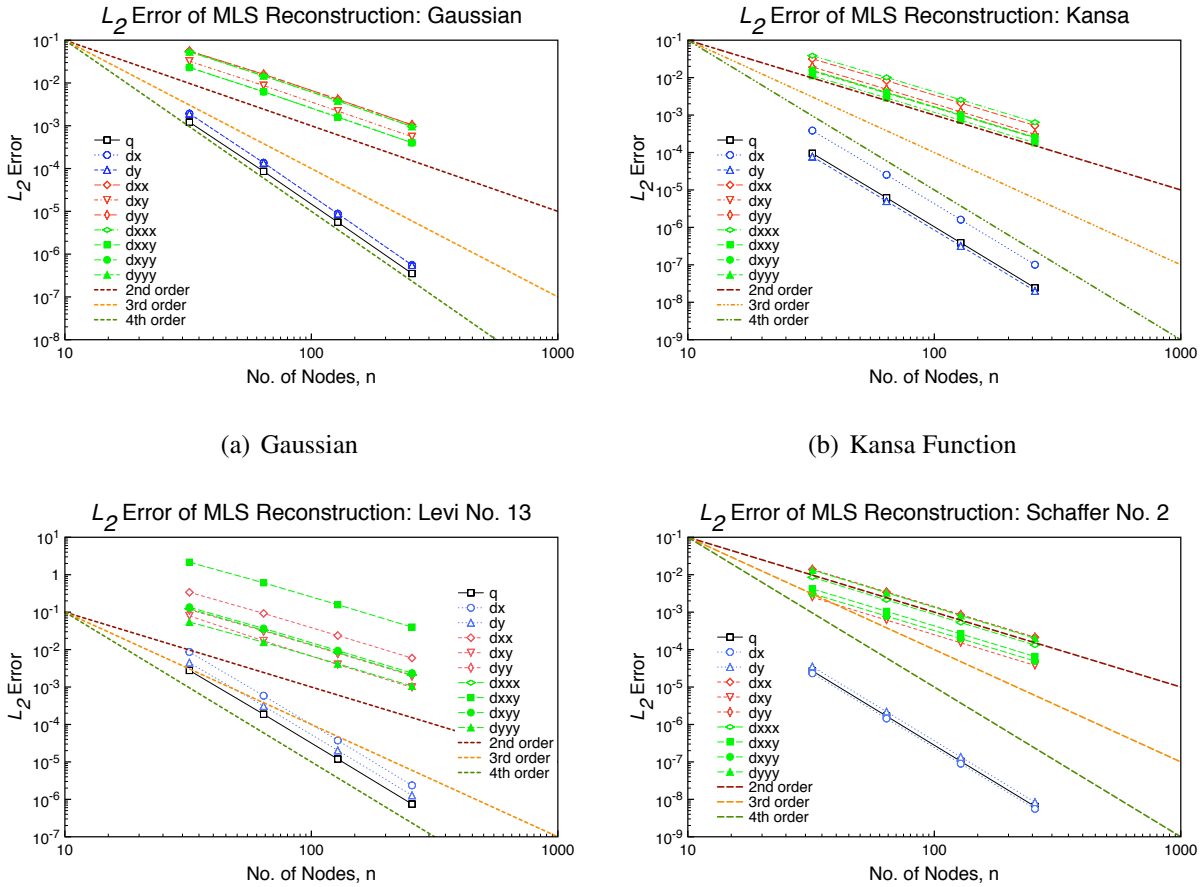
Table 6.60: Timing on Structured Grids: 3^{rd}-Order MLS with Isotropic Weights, in Seconds.

Mesh Size	Type		
	Full	Semi-Diffuse	Full-Diffuse
32	0.121	0.091	0.082
64	0.457	0.347	0.308
128	1.843	1.373	1.233
256	7.180	5.425	4.675

reduce computational time for larger systems.

6.4.1.3 Fourth-Order

The fourth-order MLS reconstruction of the derivatives can be in the full form (5.47), semi-diffuse (5.68) (with or without the second derivatives of $\mathbf{C}$), or in the fully-diffuse form (5.72). The semi-diffuse derivatives, including the second-derivatives of $\mathbf{C}$, reconstructed with isotropic weights are shown in Figure 6.45.



(a) Gaussian

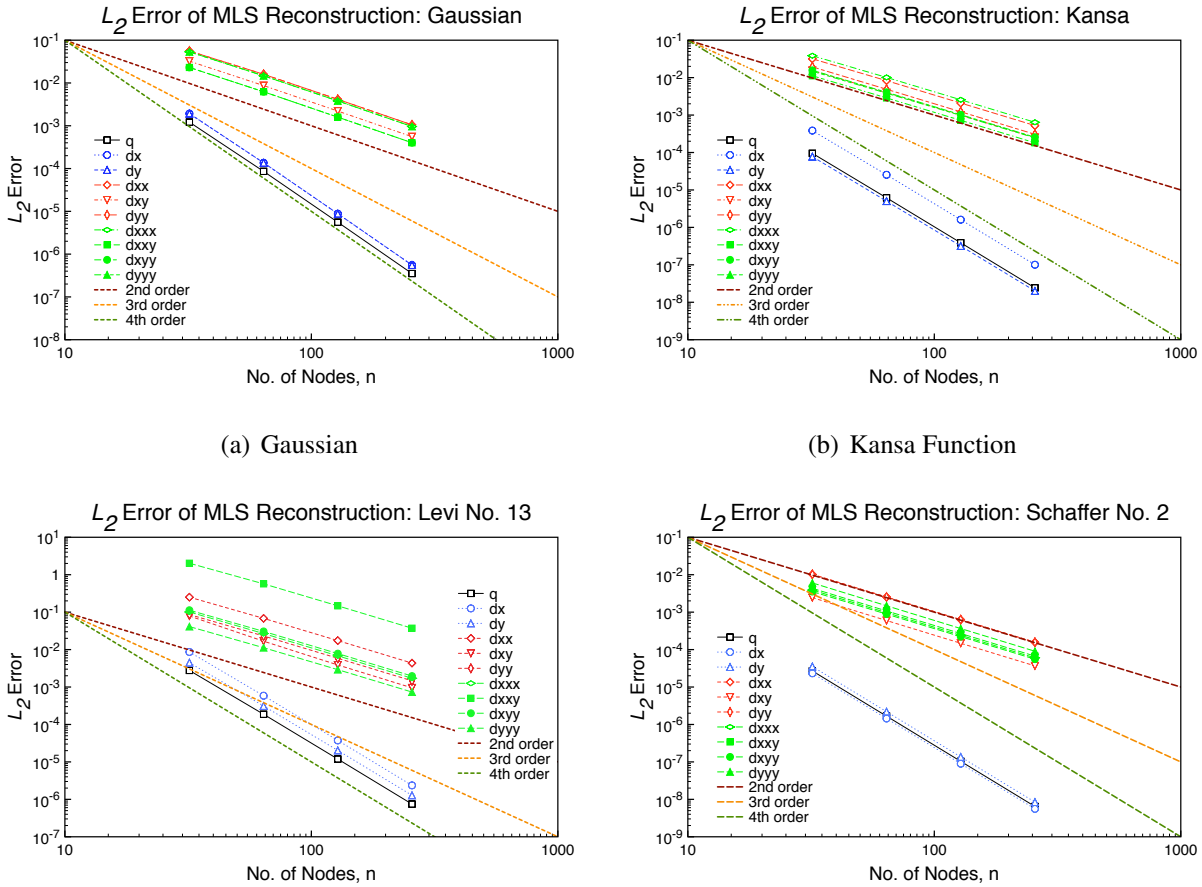
(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.45: Error Norms on Structured Grids: 4^{th}-Order Semi-Diffuse (1) MLS with Isotropic Weights.

Figure 6.45 shows that the function and first-derivatives is approximately fourth-order, and the second- and third-derivatives is approximately second-order. The semi-diffuse derivatives, without the second-derivatives of $\mathbf{C}$, reconstructed with isotropic weights are shown in Figure 6.46.



(a) Gaussian

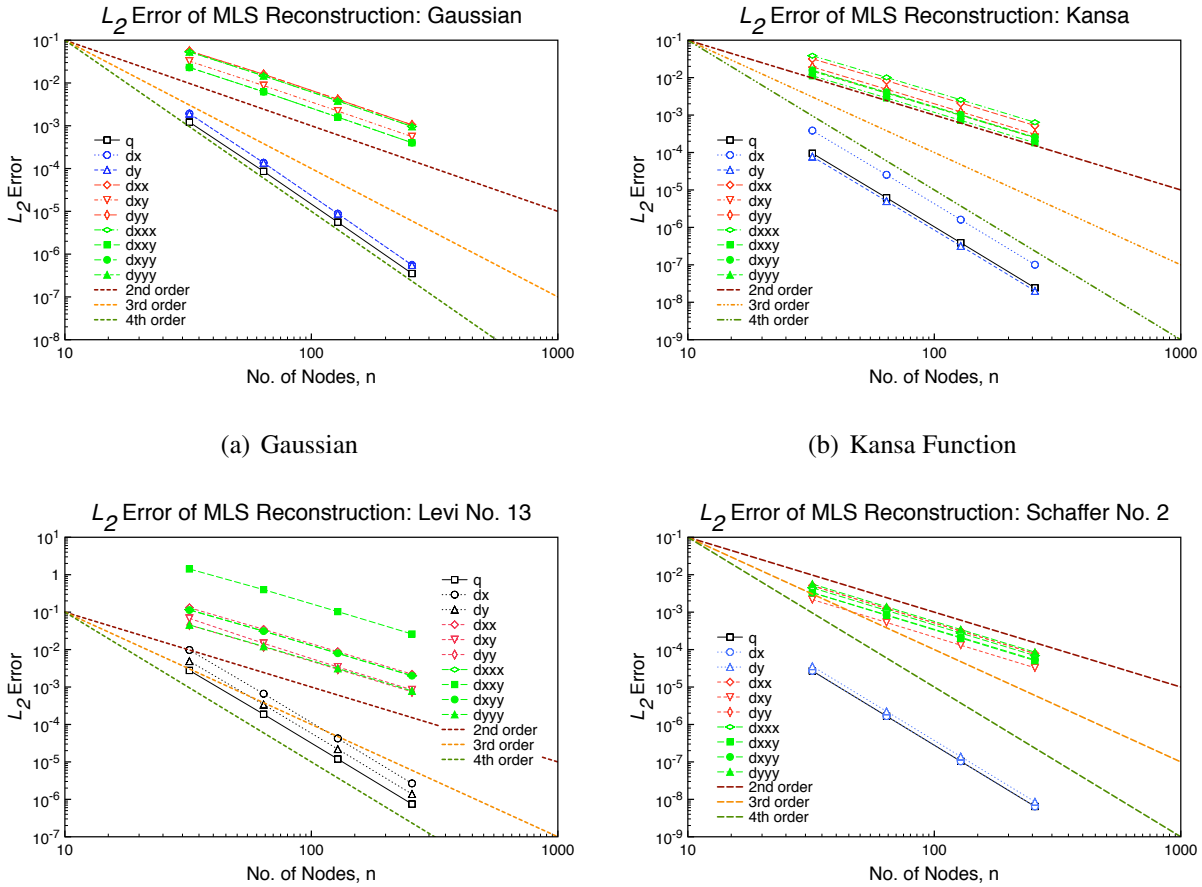
(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.46: Error Norms on Structured Grids: 4^{th}-Order Semi-Diffuse (2) MLS with Isotropic Weights.

Figure 6.46 shows that the function and first-derivatives is approximately fourth-order, and the second- and third-derivatives is approximately second-order. The fully-diffuse derivatives reconstructed with isotropic weights are shown in Figure 6.47.



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.47: Error Norms on Structured Grids: 4^{th}-Order Fully-Diffuse MLS with Isotropic Weights.

Figure 6.47 shows that the function and first-derivatives is approximately fourth-order, and the second- and third-derivatives is approximately second-order. Table 6.61 compares the computed order of accuracy for each equation on structured grids using isotropic weights for each level of diffusivity. As with the third-order results shown in Table 6.61, the orders of accuracy for the function and lower derivatives (first and second) the same between the levels of diffusivity. For the third-derivatives, the methods produce only slight differences in the order. It is also constructive to look at the timing for the varying levels of diffusivity. Table 6.62 shows the timing for the reconstruction of the Gaussian. Note that in the table, semi-diffuse (1) refers to the semi-diffuse

Table 6.61: Accuracy on Structured Grids: 4^{th}-Order Diffuse MLS with Isotropic Weights. SD1=Semi-Diffuse 1, SD2=Semi-Diffuse 2, FD=Fully-Diffuse.

Equation	Type	Function									
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial xy^2}$	$\frac{\partial^3 q}{\partial y^3}$
Gaussian	Full	3.92	3.92	3.92	1.91	1.95	1.91	1.93	1.95	1.95	1.93
	SD1	3.92	3.92	3.92	1.91	1.95	1.91	1.94	1.95	1.95	1.94
	SD2	3.92	3.92	3.92	1.92	1.95	1.92	1.95	1.95	1.95	1.95
	FD	3.92	3.92	3.92	1.94	1.96	1.94	1.95	1.95	1.95	1.95
Kansa	Full	3.98	3.97	3.97	1.97	1.98	1.99	1.97	1.98	1.98	1.97
	SD1	3.98	3.967	3.97	1.97	1.98	1.99	1.98	1.98	1.98	1.96
	SD2	3.98	3.97	3.97	1.98	1.98	1.99	1.98	1.98	1.99	1.98
	FD	3.98	3.97	3.98	1.99	1.98	1.99	1.98	1.98	1.99	1.98
Levi	Full	3.95	3.95	3.92	1.94	2.12	1.95	1.95	1.92	1.95	1.92
	SD1	3.95	3.95	3.92	1.94	2.12	1.95	1.96	1.92	1.95	1.89
	SD2	3.95	3.95	3.92	1.95	2.13	1.96	1.96	1.92	1.95	1.94
	FD	3.95	3.95	3.93	1.97	2.11	1.98	1.96	1.92	1.94	1.95
Schaffer	Full	4.01	4.01	4.01	2.01	2.02	2.01	2.01	2.00	1.99	2.01
	SD1	4.01	4.01	4.01	2.01	2.02	2.01	2.01	2.00	2.00	2.02
	SD2	4.01	4.01	4.01	2.01	2.02	2.01	2.00	2.00	2.00	2.02
	FD	4.01	4.01	4.01	2.01	2.02	2.01	2.01	2.01	1.99	2.02

results that include the second-derivative of $\mathbf{C}$ while semi-diffuse (2) does not include the second-derivative of $\mathbf{C}$. As seen from Table 6.62, the reduction in computation time is nearly two from

Table 6.62: Timing on Structured Grids: 4^{th}-Order MLS with Isotropic Weights, in Seconds.

Mesh Size	Type			
	Full	Semi-Diffuse (1)	Semi-Diffuse (2)	Full-Diffuse
32	0.716	0.413	0.302	0.268
64	2.081	1.480	1.259	1.029
128	11.037	6.219	4.498	4.067
256	43.758	24.471	17.921	15.497

full to semi-diffuse (with $\mathbf{C}$), and from full to full-diffuse the reduction is nearly three. Therefore, it is beneficial to use the diffuse derivatives when possible.

6.4.2 Unstructured Grids

This section presents the results of order of accuracy of the diffusive MLS derivatives on unstructured grids. First, the diffuse MLS derivatives using isotropic weights is presented. Next, the diffuse MLS derivatives using anisotropic weights are shown. The section concludes with the diffuse Affine MLS derivatives using anisotropic weights.

6.4.2.1 Isotropic Weights

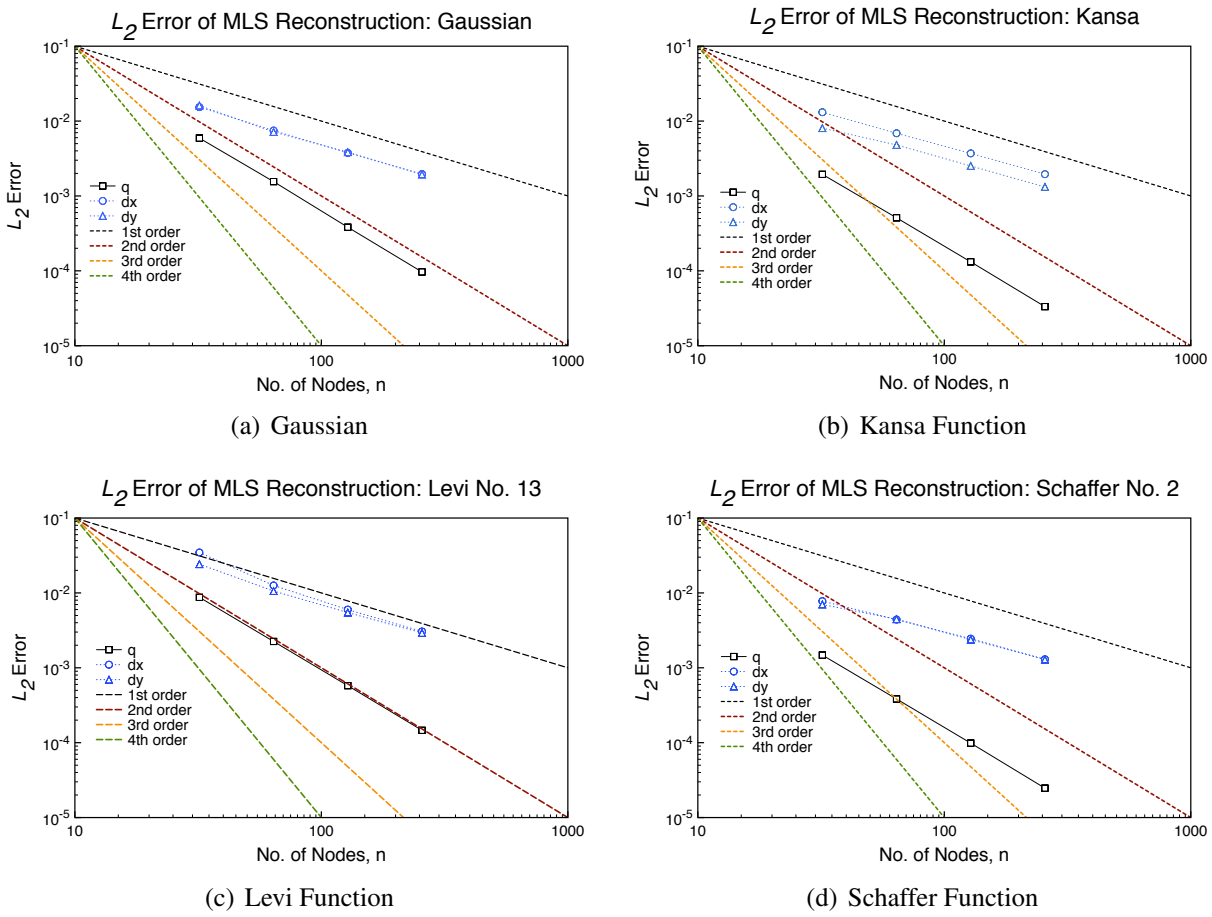
This subsection presents diffuse derivatives using isotropic weights on unstructured grids.

6.4.2.1.1 Second-Order The second-order MLS reconstruction of the derivatives can be in the full form (5.47) or in the fully-diffuse form (5.72). The fully-diffuse derivatives reconstructed with isotropic weights are shown in Figure 6.48. From Figure 6.48, the function is reconstructed approximately second-order, and the first derivatives are reconstructed approximately first-order. Table 6.63 compares the order of accuracy between the full and fully-diffuse derivatives. From

Table 6.63: Accuracy on Unstructured Grids: 2^{nd}-Order Diffuse MLS with Isotropic Weights. FD=Fully-Diffuse.

Equation	Type	Function		
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
Gaussian	Full	2.00	0.95	0.93
	FD	2.00	0.97	0.94
Kansa	Full	1.97	0.90	0.93
	FD	1.97	0.91	0.93
Levi	Full	1.97	0.95	0.90
	FD	1.97	1.02	0.93
Schaffer	Full	1.98	0.88	0.89
	FD	1.98	0.88	0.89

Table 6.63, there is no difference in the accuracy of the function based on the diffusivity of the derivatives. The first derivatives are either the same or slightly better when using the fully-diffuse



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

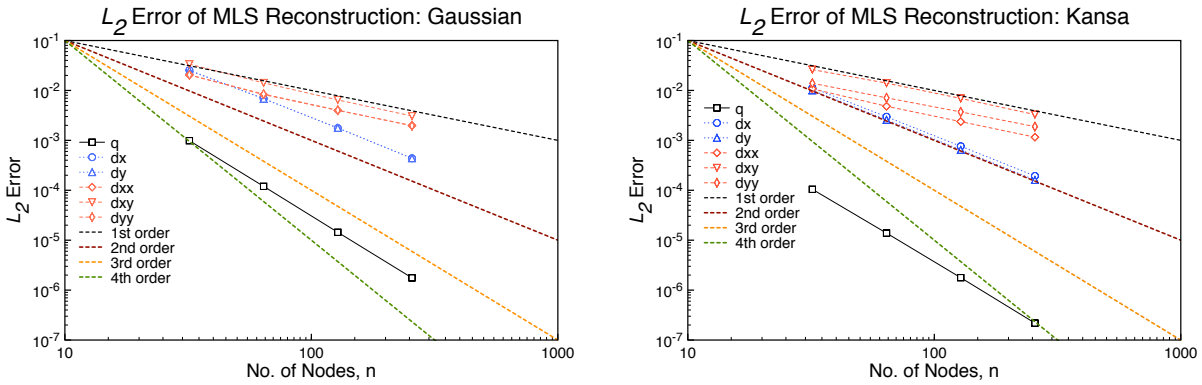
Figure 6.48: Error Norms on Unstructured Grids: 2^{nd}-Order Fully-Diffuse MLS with Isotropic Weights.

derivatives. This is probably because the less accurate derivatives of $\mathbf{C}$ are not computed with the fully-diffuse method. It is also constructive to look at the timing for the full and fully-diffuse derivatives. Table 6.64 shows the timing for the reconstruction of the Gaussian. As seen from Table 6.64, the second-order fully-diffuse is only slightly faster than the full reconstruction, so it is not overly beneficial to use the fully-diffuse from a timing standpoint.

6.4.2.1.2 Third-Order The third-order MLS reconstruction of the derivatives can be in the full form (5.47), semi-diffuse (5.68), or in the fully-diffuse form (5.72). The full derivatives, reconstructed using isotropic weights, were shown in Table 6.11. The semi-diffuse derivatives reconstructed with isotropic weights are shown in Figure 6.49.

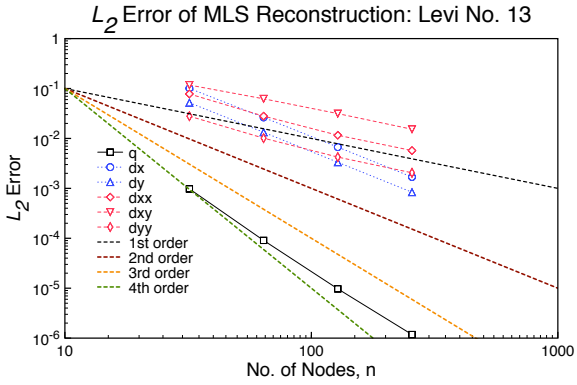
Table 6.64: Timing on Unstructured Grids: 2^{nd}-Order MLS with Isotropic Weights, in Seconds.

Mesh Size	Type	
	Full	Full-Diffuse
32	0.055	0.050
64	0.226	0.121
128	0.925	0.821
256	3.733	3.337

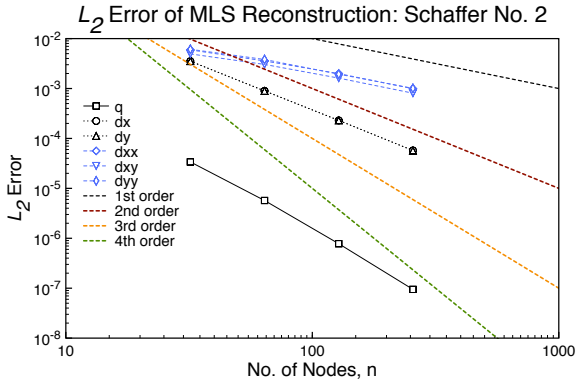


(a) Gaussian

(b) Kansa Function



(c) Levi Function

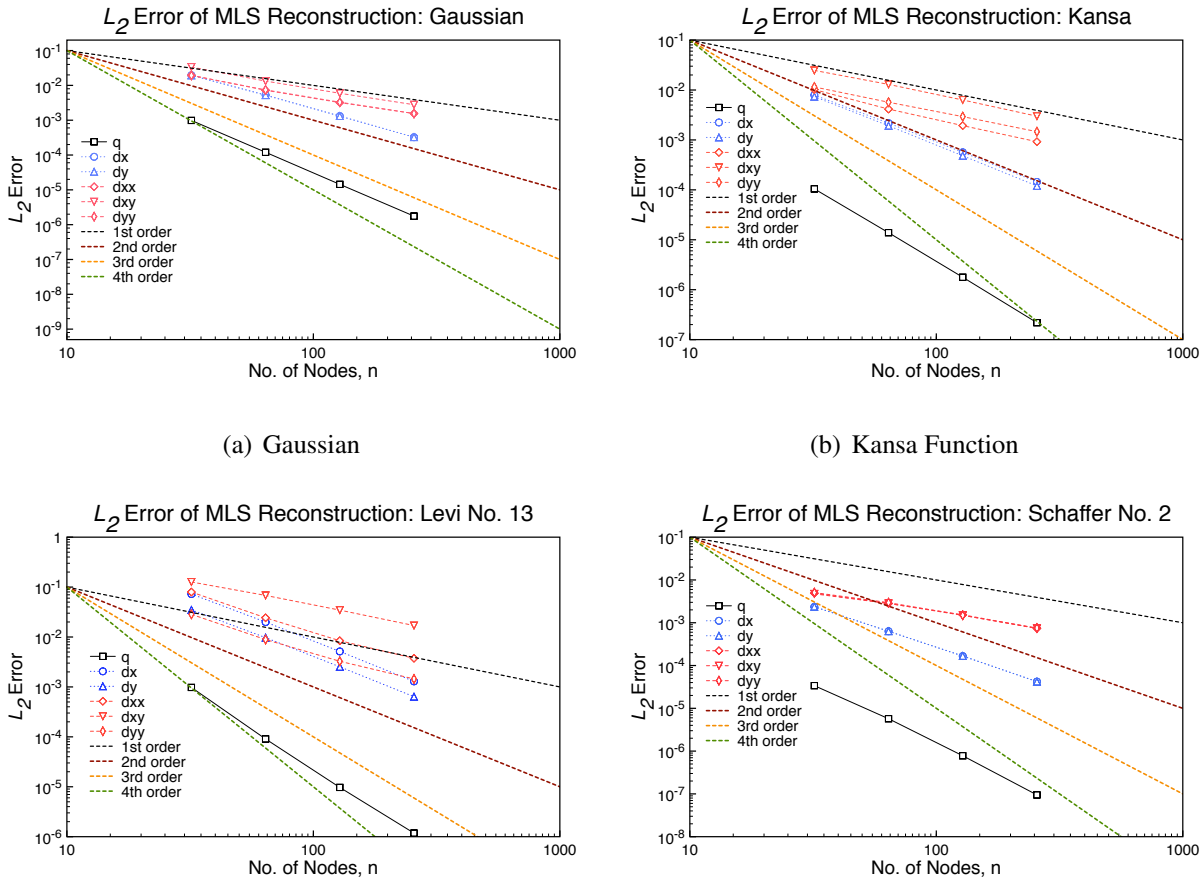


(d) Schaffer Function

Figure 6.49: Error Norms on Unstructured Grids: 3^{rd}-Order Semi-Diffuse MLS with Isotropic Weights.

Figure 6.43 shows that the accuracy for the function is approximately third-order accurate, and the first-derivatives second-order accurate, and second-derivatives first-order accurate. The

fully-diffuse derivatives reconstructed with isotropic weights are shown in Figure 6.50.



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.50: Error Norms on Unstructured Grids: 3^{rd}-Order Fully-Diffuse MLS with Isotropic Weights.

Figure 6.50 shows that the accuracy for the function is approximately third-order accurate, and the first-derivatives second-order accurate, and second-derivatives first-order accurate. Table 6.65 compares the orders of accuracy for the each level of diffusivity. The functions have the same accuracy for any of the derivative methods, and the full and semi-diffuse have the same first derivative accuracy as shown in Table 6.65. The fully-diffuse first derivatives are slightly better for the Gaussian, but slightly worse for the other functions. The fully-diffuse he second derivatives are more

Table 6.65: Accuracy on Unstructured Grids: 3^{rd}-Order Diffuse MLS with Isotropic Weights. SD=Semi-Diffuse, FD=Fully-Diffuse.

Equation	Type	Function					
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
Gaussian	Full	3.05	1.99	1.98	0.99	1.10	1.02
	SD	3.05	1.99	1.98	1.03	1.10	1.06
	FD	3.05	2.01	2.00	1.11	1.13	1.13
Kansa	Full	2.99	1.97	2.00	1.01	1.05	0.96
	SD	2.99	1.97	2.00	1.03	1.05	0.96
	FD	2.99	1.96	2.00	1.09	1.06	0.98
Levi	Full	3.13	1.98	1.99	1.05	1.04	1.08
	SD	3.13	1.98	1.99	1.15	1.01	1.15
	FD	3.13	1.97	1.96	1.35	1.00	1.31
Schaffer	Full	2.96	1.98	1.99	0.92	0.97	0.96
	SD	2.96	1.98	1.99	0.92	0.96	0.96
	FD	2.96	1.95	1.96	0.97	0.97	0.99

accurate than the full derivatives. As with the second-order results, this is probably due to the less accurate derivatives of $\mathbf{C}$ are not computed with the fully-diffuse method. It is also constructive to look at the timing for the full and fully-diffuse derivatives. Table 6.66 shows the timing for the reconstruction of the Gaussian. As seen from Table 6.66, the fully-diffuse is nearly twice as fast at

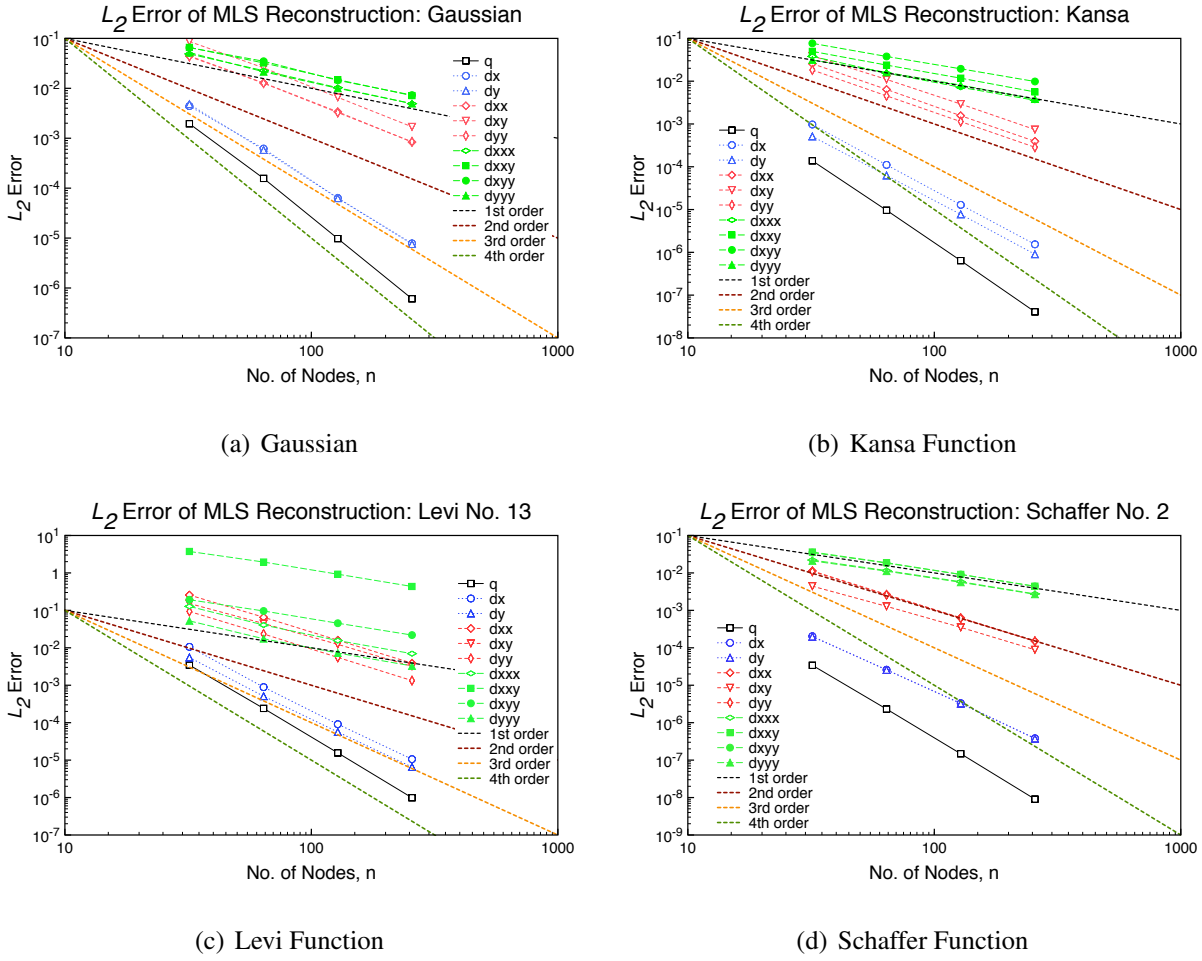
Table 6.66: Timing on Unstructured Grids: 3^{rd}-Order MLS with Isotropic Weights, in Seconds.

Mesh Size	Type		
	Full	Semi-Diffuse	Full-Diffuse
32	0.225	0.168	0.142
64	0.978	0.709	0.604
128	4.114	2.935	2.607
256	17.234	11.923	10.322

the highest level of refinement. Considering the fully-diffuse as accurate as the full reconstruction, it makes sense to use the fully-diffuse to reduce computational time for larger systems. Additionally, the semi-diffuse is nearly as fast as the fully-diffuse, so the additional terms are not much

more expensive.

6.4.2.1.3 Fourth-Order The fourth-order MLS reconstruction of the derivatives can be in the full form (5.47), semi-diffuse (5.68) (with or without the second derivatives of C), or in the fully-diffuse form (5.72). The full derivatives, reconstructed using isotropic weights, were shown in Table 6.24. The semi-diffuse derivatives, including the second-derivatives of C, reconstructed with isotropic weights are shown in Figure 6.51.



(a) Gaussian

(b) Kansa Function

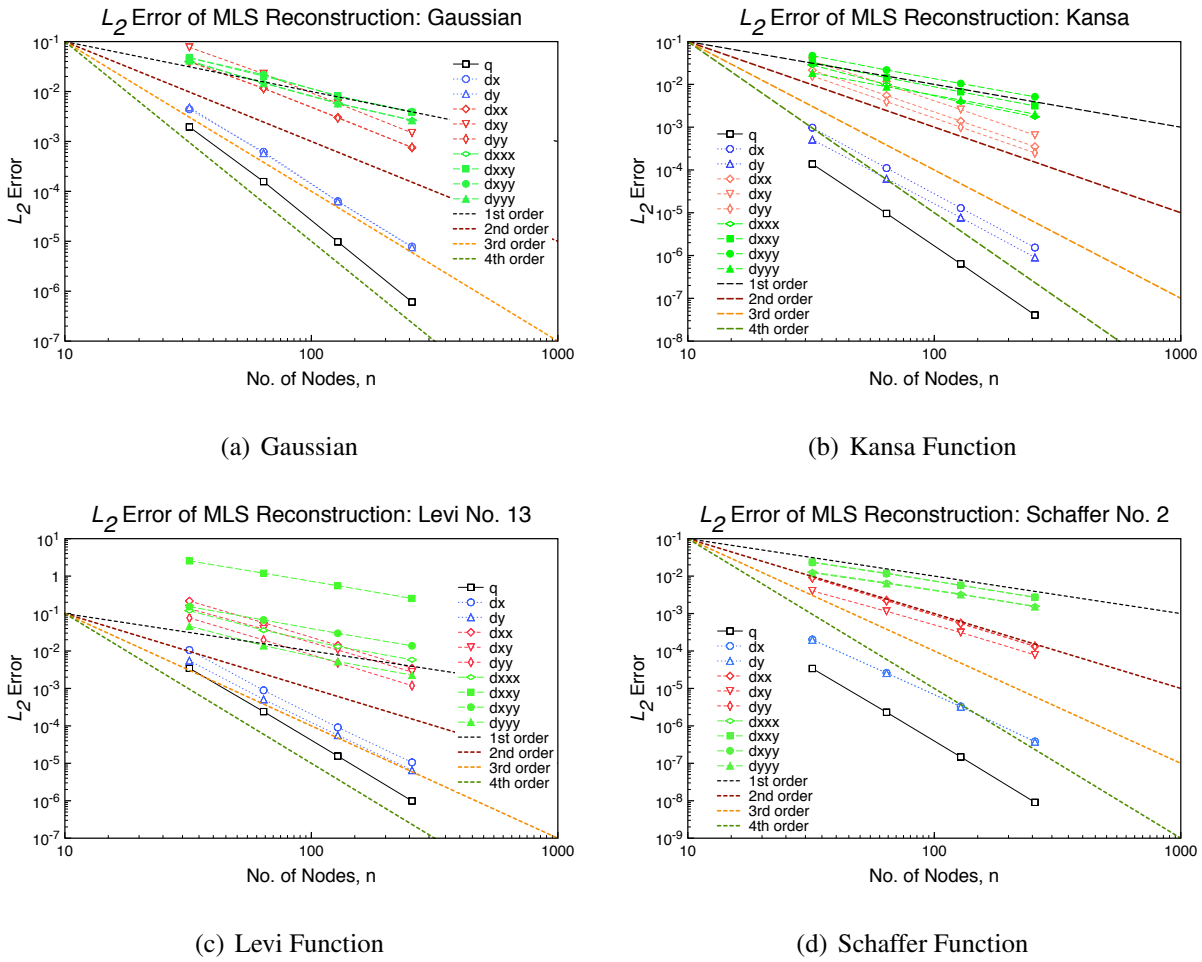
(c) Levi Function

(d) Schaffer Function

Figure 6.51: Error Norms on Unstructured Grids: 4^{th}-Order Semi-Diffuse (1) MLS with Isotropic Weights.

Figure 6.51 shows that the function is approximately fourth-order accurate, first-derivatives ap-

proximately third-order accurate, the second-derivatives second-order accurate, and third-derivatives first-order accurate. The semi-diffuse derivatives, without the second-derivatives of C, reconstructed with isotropic weights are shown in Figure 6.52.



(a) Gaussian

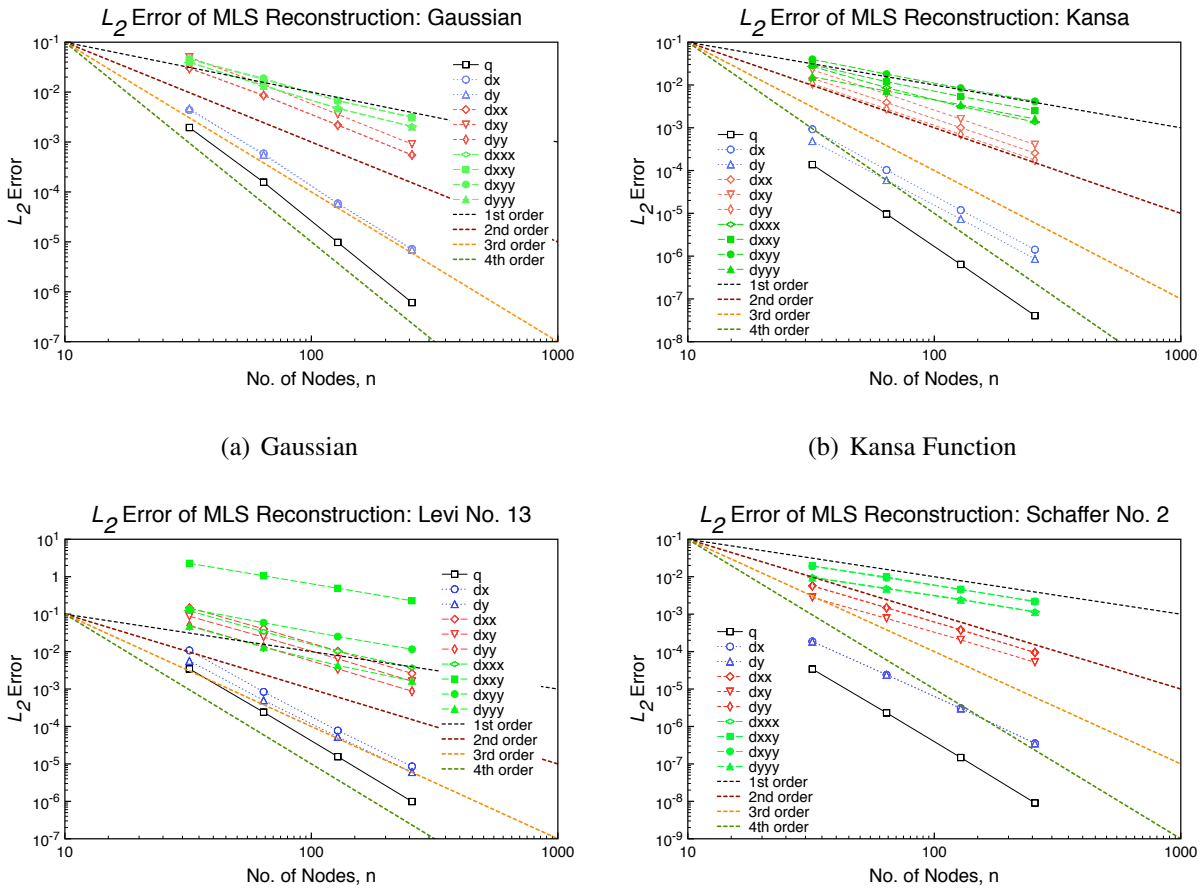
(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.52: Error Norms on Unstructured Grids: 4^{th}-Order Semi-Diffuse (2) MLS with Isotropic Weights.

Figure 6.52 shows that the function is approximately fourth-order accurate, first-derivatives approximately third-order accurate, the second-derivatives second-order accurate, and third-derivatives first-order accurate. The fully-diffuse derivatives reconstructed with isotropic weights are shown in Figure 6.53.



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.53: Error Norms on Unstructured Grids: 4^{th}-Order Fully-Diffuse MLS with Isotropic Weights.

Figure 6.53 shows that the function is approximately fourth-order accurate, first-derivatives approximately third-order accurate, the second-derivatives second-order accurate, and third-derivatives first-order accurate. Table 6.67 compares the accuracy for each of the diffusivity levels. The results in Table 6.67 are similar to the third-order results in Table 6.65, where the methods produce effectively the same accuracy except for the second and third derivatives. Interestingly, the semi-diffuse (2) method does not produce better second derivatives, as removing the second-derivatives of $\mathbf{C}$ is probably detrimental to the accuracy. The fully-diffuse produces the highest level of accuracy, as was the case for the lower orders. It is also constructive to look at the timing for the varying levels

Table 6.67: Accuracy on Unstructured Grids: 4^{th}-Order Fully-Diffuse MLS with Isotropic Weights. SD1=Semi-Diffuse 1, SD2=Semi-Diffuse 2, FD=Fully-Diffuse.

Equation	Type	Function									
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial xy^2}$	$\frac{\partial^3 q}{\partial y^3}$
Gaussian	Full	4.01	3.15	3.13	1.97	1.96	1.95	1.06	1.04	1.09	1.01
	SD1	4.01	3.15	3.13	1.97	1.96	1.95	1.10	1.09	1.13	1.06
	SD2	4.01	3.15	3.13	1.97	1.97	1.96	1.25	1.21	1.23	1.24
	FD	4.01	3.18	3.16	1.99	1.98	1.98	1.36	1.27	1.29	1.36
Kansa	Full	3.94	3.08	3.06	2.01	1.95	2.00	0.99	1.02	0.93	1.03
	SD1	3.94	3.08	3.06	2.01	1.95	2.00	1.05	1.04	0.97	1.04
	SD2	3.94	3.08	3.06	1.99	1.95	1.99	1.25	1.09	1.04	1.06
	FD	3.94	3.09	3.06	1.96	1.97	1.98	1.33	1.12	1.06	1.08
Levi	Full	3.96	3.20	3.14	2.04	1.90	2.07	1.33	1.05	1.02	1.19
	SD1	3.96	3.20	3.14	2.04	1.90	2.07	1.27	1.08	1.07	1.21
	SD2	3.96	3.20	3.14	2.01	1.91	2.03	1.31	1.12	1.15	1.31
	FD	3.96	3.31	3.19	1.97	1.96	1.97	1.58	1.11	1.18	1.50
Schaffer	Full	4.00	3.04	3.05	2.07	1.93	2.06	1.00	1.02	1.01	0.99
	SD1	4.00	3.04	3.05	2.07	1.93	2.06	1.04	1.04	1.03	1.03
	SD2	4.00	3.04	3.05	2.03	1.93	2.03	1.04	1.07	1.05	1.03
	FD	4.00	3.04	3.05	1.99	1.95	1.98	1.04	1.08	1.06	1.03

of diffusivity. Table 6.68 shows the timing for the reconstruction of the Gaussian. As seen from

Table 6.68: Timing on Unstructured Grids: 4^{th}-Order MLS with Isotropic Weights, in Seconds.

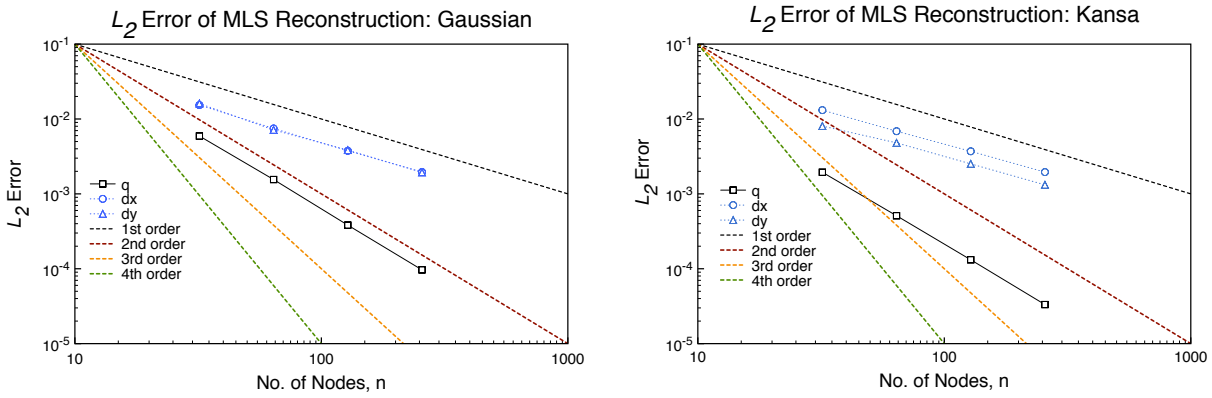
Mesh Size	Type			
	Full	Semi-Diffuse (1)	Semi-Diffuse (2)	Full-Diffuse
32	1.779	0.879	0.564	0.521
64	7.982	4.035	2.701	2.252
128	34.135	16.524	11.297	9.437
256	140.772	69.860	44.479	37.756

Table 6.62, the reduction in computation time is nearly two from full to semi-diffuse (with $\mathbf{C}$), and from full to full-diffuse the reduction is nearly four. Therefore, it is beneficial to use the diffuse derivatives when possible.

6.4.2.2 *Anisotropic Weights*

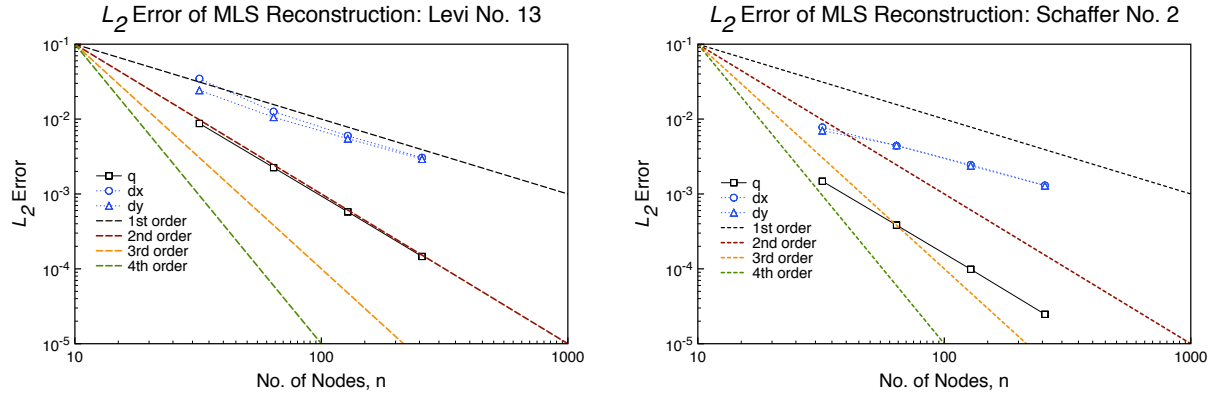
This subsection presents diffuse derivatives using anisotropic weights on unstructured grids.

6.4.2.2.1 Second-Order The second-order MLS reconstruction of the derivatives can be in the full form (5.47) or in the fully-diffuse form (5.72). The fully-diffuse derivatives reconstructed with anisotropic weights are shown in Figure 6.54.



(a) Gaussian

(b) Kansa Function



(c) Levi Function

(d) Schaffer Function

Figure 6.54: Error Norms on Unstructured Grids: 2^{nd}-Order Fully-Diffuse MLS with Anisotropic Weights.

From Figure 6.54, the function is reconstructed approximately second-order, and the first

derivatives are reconstructed approximately first-order. Table 6.69 compares the accuracy for the different diffusivity levels. From Table 6.63, there is no difference in the accuracy of the function

Table 6.69: Accuracy on Unstructured Grids: 2^{nd}-Order Diffuse MLS with Anisotropic Weights. FD=Fully-Diffuse.

Equation	Type	Function		
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
Gaussian	Full	2.00	0.94	0.92
	FD	2.00	0.97	0.95
Kansa	Full	1.97	0.93	0.96
	FD	1.97	0.91	0.93
Levi	Full	1.97	1.00	0.97
	FD	1.97	1.02	0.93
Schaffer	Full	1.98	0.89	0.90
	FD	1.98	0.89	0.90

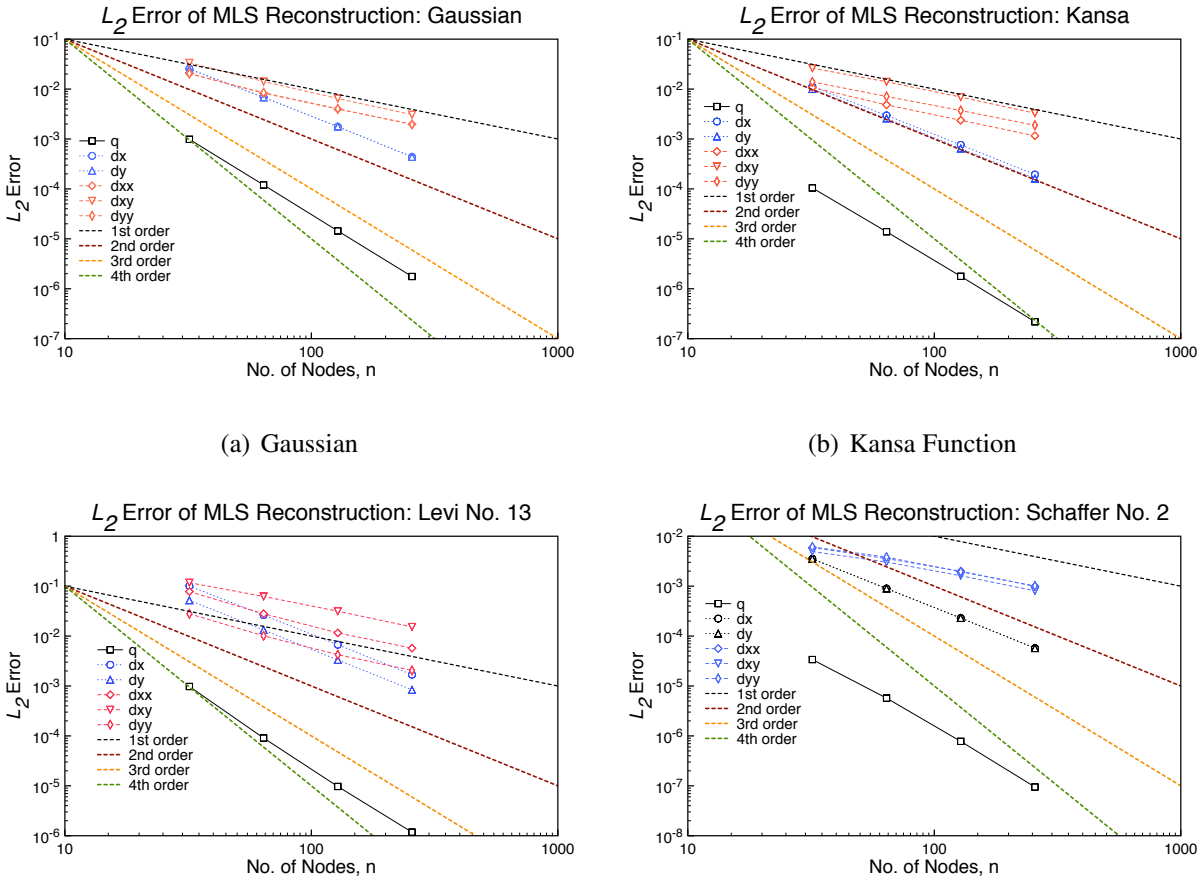
based on the diffusivity of the derivatives. The first derivatives are either the same or slightly better when using the fully-diffuse derivatives. This is probably because the less accurate derivatives of $\mathbf{C}$ are not computed with the fully-diffuse method. Table 6.70 shows the timing for the reconstruction of the Gaussian. As seen from Table 6.70, the second-order fully-diffuse is only slightly faster

Table 6.70: Timing on Unstructured Grids: 2^{nd}-Order MLS with Anisotropic Weights, in Seconds.

Mesh Size	Type	
	Full	Full-Diffuse
32	0.054	0.050
64	0.223	0.121
128	0.925	0.851
256	3.871	3.469

than the full reconstruction, so it is not overly beneficial to use the fully-diffuse from a timing standpoint.

6.4.2.2.2 Third-Order The third-order MLS reconstruction of the derivatives can be in the full form (5.47), semi-diffuse (5.68), or in the fully-diffuse form (5.72). The semi-diffuse derivatives reconstructed with anisotropic weights are shown in Figure 6.55.



(a) Gaussian

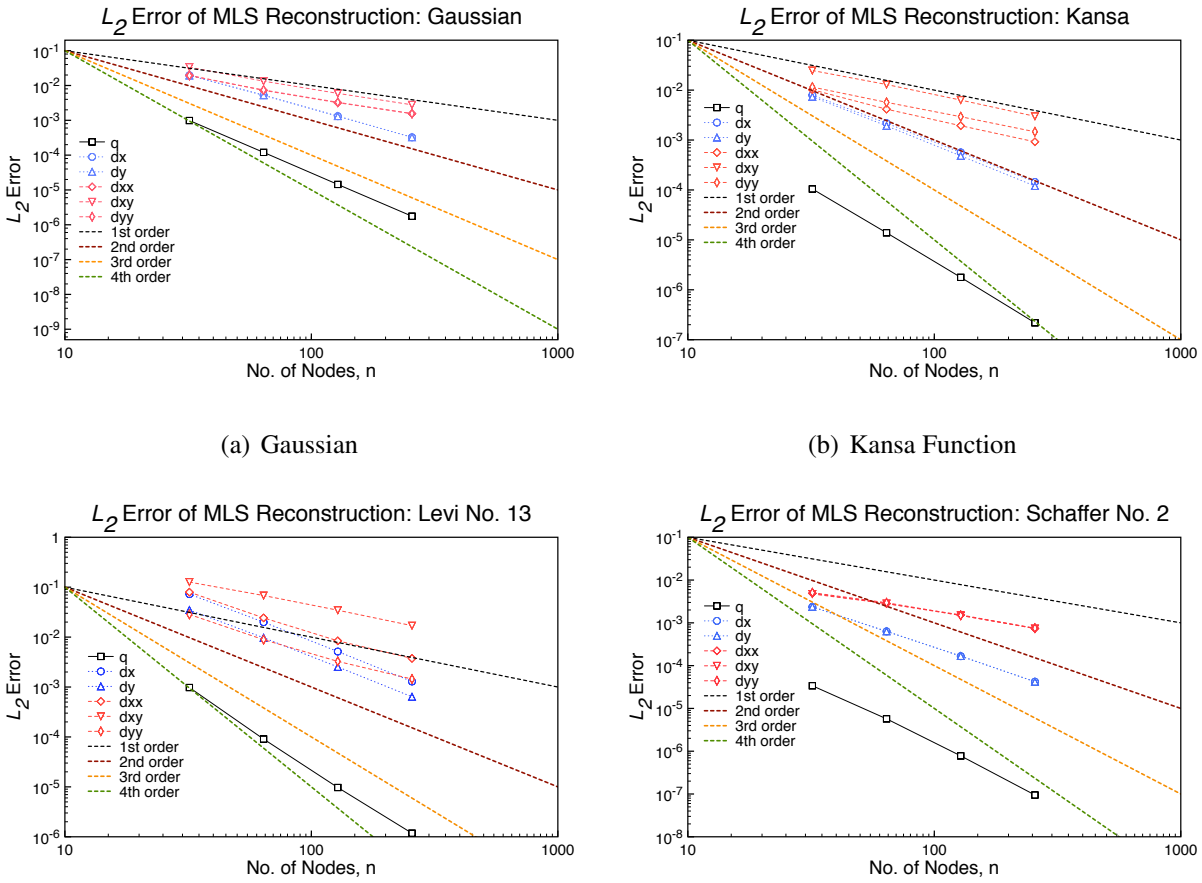
(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.55: Error Norms on Unstructured Grids: 3^{rd}-Order Semi-Diffuse MLS with Anisotropic Weights.

Figure 6.55 shows that the accuracy for the function is approximately third-order accurate, and the first-derivatives second-order accurate, and second-derivatives first-order accurate. The fully-diffuse derivatives reconstructed with anisotropic weights are shown in Figure 6.56.



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.56: Error Norms on Unstructured Grids: 3rd-Order Fully-Diffuse MLS with Anisotropic Weights.

Figure 6.56 shows that the accuracy for the function is approximately third-order accurate, and the first-derivatives second-order accurate, and second-derivatives first-order accurate. Table 6.71 compares the computed order of accuracy for each equation on unstructured grids using anisotropic weights for the diffusivity of the derivatives. The functions have the same accuracy for any of the derivative methods, and the full and semi-diffuse have the same first derivative accuracy as shown in Table 6.71. The fully-diffuse first derivatives are slightly better for the Gaussian, but slightly worse for the other functions. The fully-diffuse second derivatives are more accurate than the full derivatives. As with the second-order results, this is probably due to the less accurate

Table 6.71: Accuracy on Unstructured Grids: 3rd-Order Diffuse MLS with Anisotropic Weights. SD=Semi-Diffuse, FD=Fully-Diffuse.

Equation	Type	Function					
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
Gaussian	Full	3.05	2.00	1.99	1.00	1.11	1.03
	SD	3.05	2.00	1.99	1.03	1.11	1.08
	FD	3.05	2.01	2.00	1.11	1.14	1.14
Kansa	Full	3.01	1.97	2.00	1.02	1.05	0.97
	SD	3.01	1.97	2.00	1.07	1.06	0.98
	FD	3.01	1.95	2.00	1.11	1.07	0.99
Levi	Full	3.14	1.99	2.00	1.07	1.05	1.10
	SD	3.14	1.99	2.00	1.15	1.01	1.16
	FD	3.14	1.97	1.96	1.35	1.01	1.33
Schaffer	Full	2.97	1.98	1.99	0.94	0.96	0.98
	SD	2.97	1.98	1.99	0.95	0.97	1.01
	FD	2.97	1.95	1.95	0.98	0.98	1.00

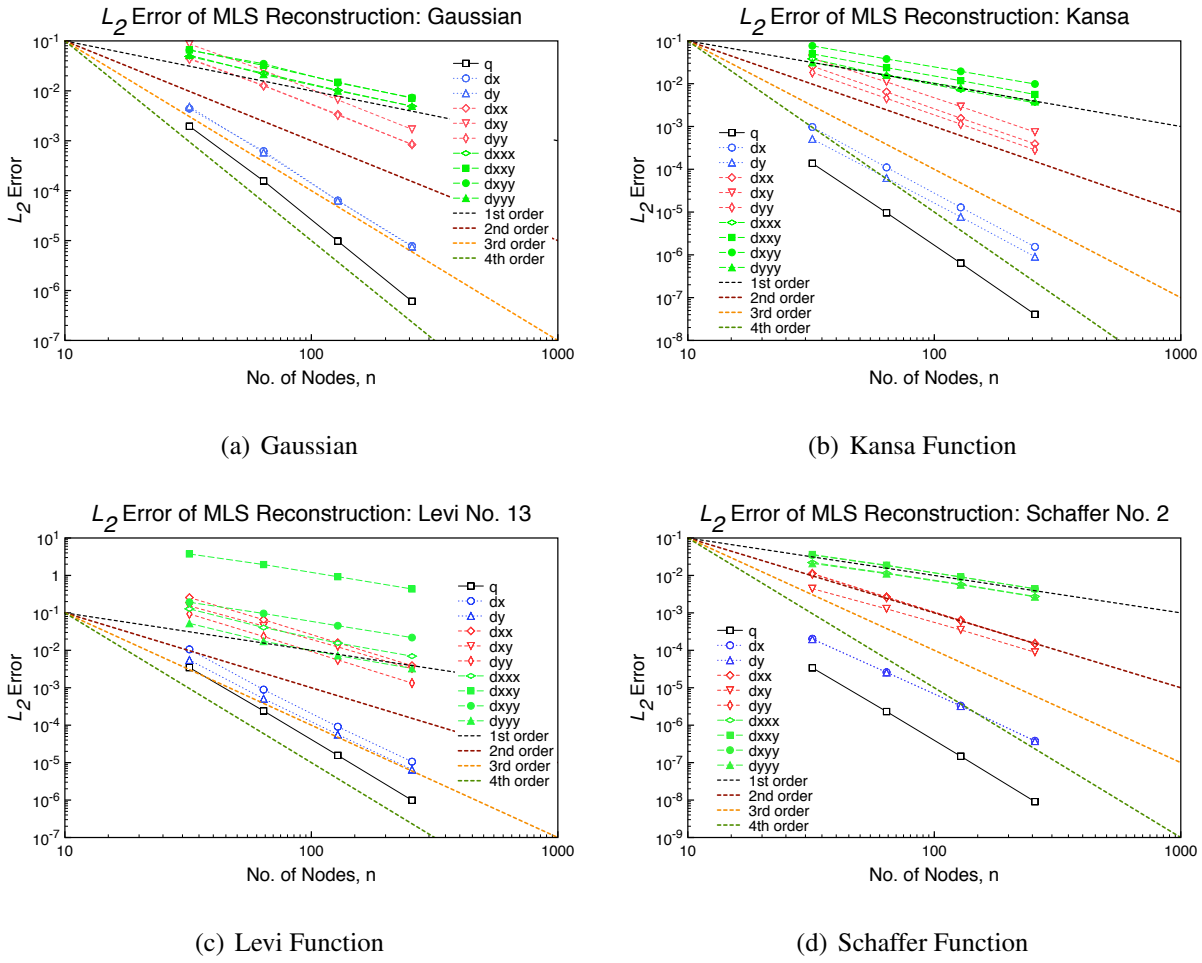
derivatives of $\mathbf{C}$ are not computed with the fully-diffuse method. Table 6.72 shows the timing for the reconstruction of the Gaussian. As seen from Table 6.72, the fully-diffuse is nearly twice

Table 6.72: Timing on Unstructured Grids: 3rd-Order MLS with Anisotropic Weights, in Seconds.

Mesh Size	Type		
	Full	Semi-Diffuse	Full-Diffuse
32	0.221	0.169	0.144
64	0.987	0.707	0.622
128	4.218	2.966	2.577
256	17.246	12.300	10.704

as fast at the highest level of refinement. Considering the fully-diffuse as accurate as the full reconstruction, it makes sense to use the fully-diffuse to reduce computational time for larger systems. Additionally, the semi-diffuse is nearly as fast as the fully-diffuse, so the additional terms are not much more expensive.

6.4.2.2.3 Fourth-Order The fourth-order MLS reconstruction of the derivatives can be in the full form (5.47), semi-diffuse (5.68) (with or without the second derivatives of C), or in the fully-diffuse form (5.72). The semi-diffuse derivatives, including the second-derivatives of C, reconstructed with anisotropic weights are shown in Figure 6.57.



(a) Gaussian

(b) Kansa Function

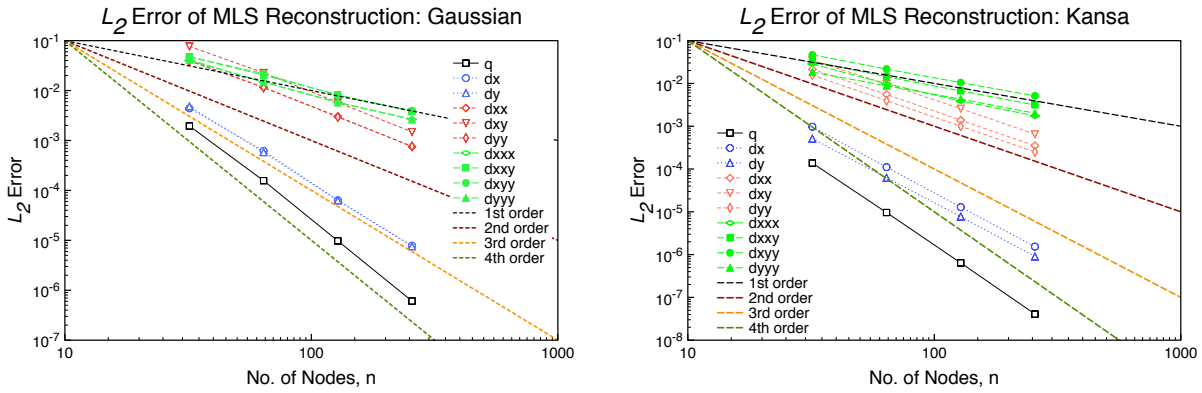
(c) Levi Function

(d) Schaffer Function

Figure 6.57: Error Norms on Unstructured Grids: 4^{th}-Order Semi-Diffuse (1) MLS with Anisotropic Weights.

Figure 6.57 shows that the function is approximately fourth-order accurate, first-derivatives approximately third-order accurate, the second-derivatives second-order accurate, and third-derivatives

first-order accurate. The semi-diffuse derivatives, without the second-derivatives of C, reconstructed with anisotropic weights are shown in Figure 6.58.



(a) Gaussian

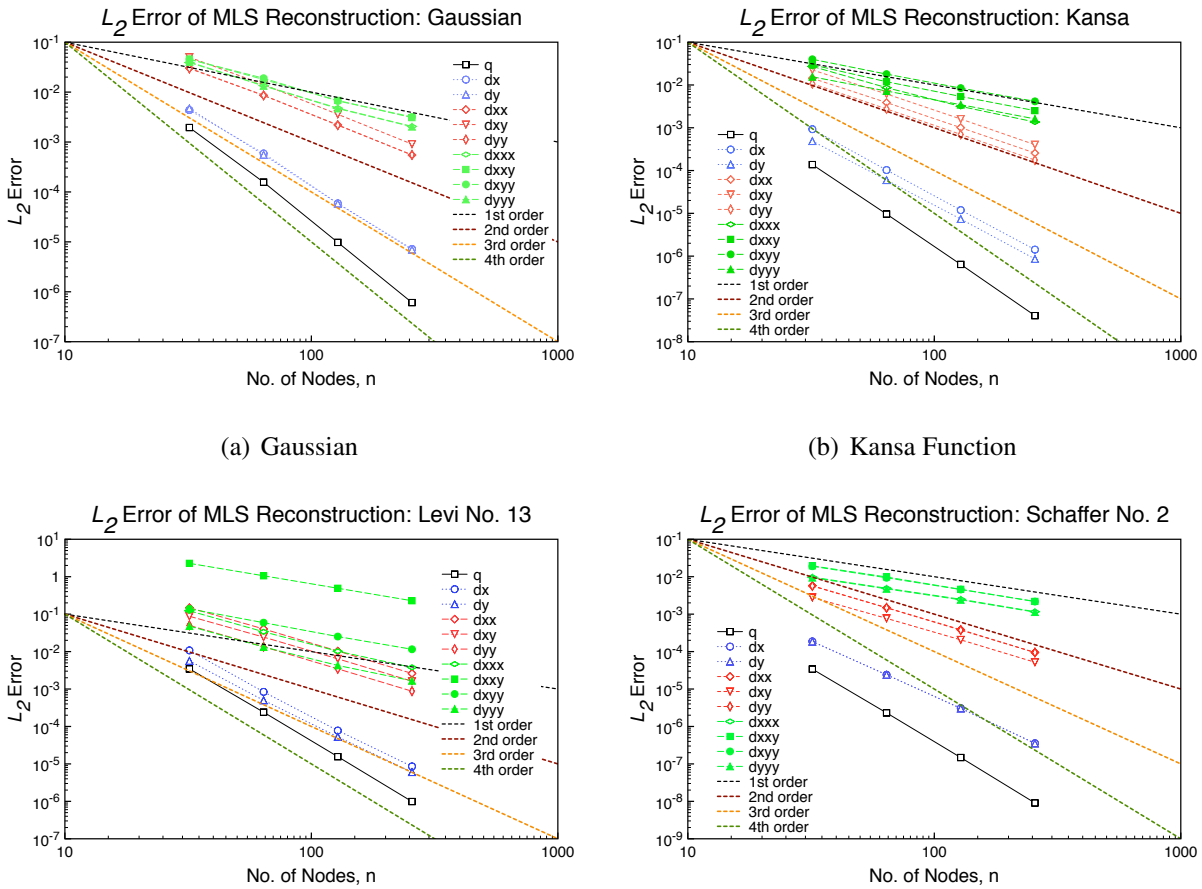
(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.58: Error Norms on Unstructured Grids: 4^{th}-Order Semi-Diffuse (2) MLS with Anisotropic Weights.

Figure 6.58 shows that the function is approximately fourth-order accurate, first-derivatives approximately third-order accurate, the second-derivatives second-order accurate, and third-derivatives first-order accurate. The fully-diffuse derivatives reconstructed with anisotropic weights are shown in Figure 6.59.



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.59: Error Norms on Unstructured Grids: 4^{th}-Order Fully-Diffuse MLS with Anisotropic Weights. SD1=Semi-Diffuse 1, SD2=Semi-Diffuse 2, FD=Fully-Diffuse.

Figure 6.59 shows that the function is approximately fourth-order accurate, first-derivatives approximately third-order accurate, the second-derivatives second-order accurate, and third-derivatives first-order accurate. Table 6.73 compares the computed order of accuracy for each equation on unstructured grids using anisotropic weights for each diffusivity level of the derivatives. The results in Table 6.73 are similar to the third-order results in Table 6.71, where the methods produce effectively the same accuracy except for the second and third derivatives. As with the isotropic weighted results, the semi-diffuse (2) method does not produce better second derivatives, as removing the second-derivatives of C is probably detrimental to the accuracy. The fully-diffuse produces the

Table 6.73: Accuracy on Unstructured Grids: 4^{th}-Order Fully-Diffuse MLS with Anisotropic Weights. SD1=Semi-Diffuse 1, SD2=Semi-Diffuse 2, FD=Fully-Diffuse

Equation	Type	Function									
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial xy^2}$	$\frac{\partial^3 q}{\partial y^3}$
Gaussian	Full	4.00	3.15	3.13	1.95	1.97	1.96	1.08	1.04	1.08	1.07
	SD1	4.00	3.15	3.13	1.95	1.97	1.96	1.09	1.08	1.13	1.08
	SD2	4.00	3.15	3.13	1.97	1.97	1.97	1.21	1.20	1.23	1.22
	FD	4.00	3.17	3.15	1.99	1.98	1.98	1.36	1.26	1.28	1.36
Kansa	Full	3.93	3.07	3.05	2.06	1.99	2.04	1.03	1.09	1.00	1.08
	SD1	3.94	3.08	3.06	2.03	1.95	2.00	1.09	1.05	0.98	1.04
	SD2	3.94	3.08	3.06	2.00	1.96	1.99	1.24	1.09	1.04	1.06
	FD	3.94	3.08	3.06	1.96	1.97	1.98	1.34	1.12	1.05	1.08
Levi	Full	3.94	3.21	3.14	2.09	2.05	2.12	1.47	1.34	1.20	1.30
	SD1	3.96	3.15	3.12	2.03	1.97	2.06	1.15	1.11	1.08	1.16
	SD2	3.96	3.15	3.12	2.00	1.95	2.03	1.22	1.12	1.15	1.24
	FD	3.96	3.29	3.18	1.97	1.96	1.97	1.56	1.11	1.18	1.49
Schaffer	Full	3.98	3.05	3.04	2.12	1.96	2.11	1.06	1.10	1.08	1.10
	SD1	4.00	3.05	3.04	2.05	1.92	2.04	1.03	1.04	1.03	1.01
	SD2	4.00	3.05	3.04	2.02	1.93	2.02	1.05	1.07	1.05	1.00
	FD	4.00	3.05	3.05	1.99	1.95	1.98	1.05	1.09	1.06	1.02

highest level of accuracy, as was the case for the lower orders. What is interesting is the non-Gaussian functions have different orders for the function reconstruction. It is unclear why this difference exists, as the function should be unaffected by the diffusive derivatives, as is the case with the Gaussian. This may warrant some further study, or it could be as simple as a truncation issue when the orders were computed, which seems most likely the cause of the difference. It is also constructive to look at the timing for the varying levels of diffusivity. Table 6.74 shows the timing for the reconstruction of the Gaussian. As seen from Table 6.74, the reduction in computation time is nearly two from full to semi-diffuse (with $\mathbf{C}$), and from full to full-diffuse the reduction is nearly four. Therefore, it is beneficial to use the diffuse derivatives when possible.

Table 6.74: Timing on Unstructured Grids: 4^{th}-Order MLS with Anisotropic Weights, in Seconds.

Mesh Size	Type			
	Full	Semi-Diffuse (1)	Semi-Diffuse (2)	Full-Diffuse
32	1.835	0.917	0.607	0.510
64	8.268	4.060	2.721	2.305
128	33.992	16.719	11.104	9.388
256	139.593	71.298	45.362	39.004

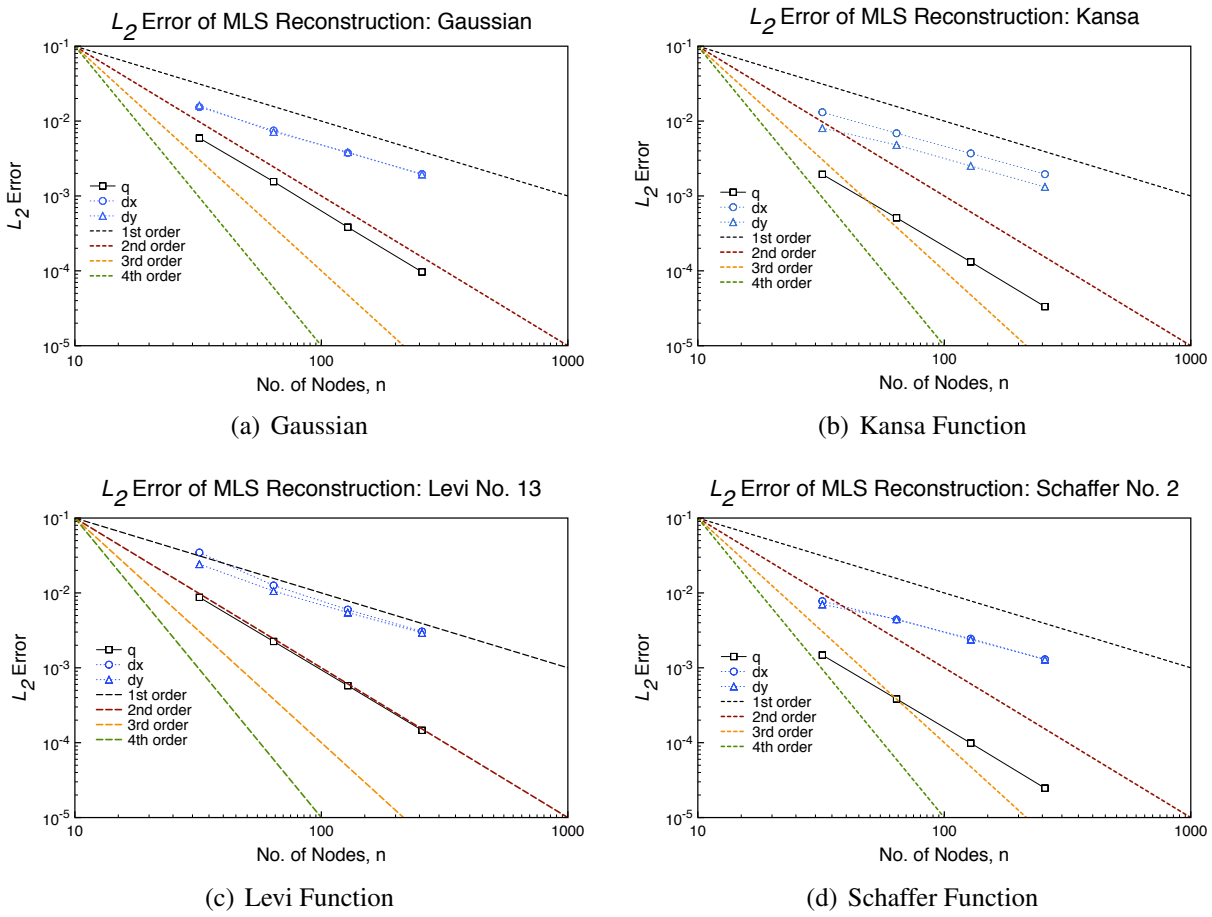
6.4.2.3 Affine MLS

This subsection presents the diffuse derivatives on unstructured grids using Affine MLS with anisotropic weights.

6.4.2.3.1 Second-Order The second-order MLS reconstruction of the derivatives can be in the full form (5.47) or in the fully-diffuse form (5.72). The Affine MLS fully-diffuse derivatives reconstructed with anisotropic weights are shown in Figure 6.60. From Figure 6.60, the function is reconstructed approximately second-order, and the first derivatives are reconstructed approximately first-order. Table 6.75 compares the computed order of accuracy for each equation on unstructured grids using Affine MLS with anisotropic weights for the diffusivity of the derivatives. The functions have the same accuracy for any of the derivative methods, as seen in Table 6.75. The

Table 6.75: Accuracy on Unstructured Grids: 2^{nd}-Order Diffuse Affine MLS with Anisotropic Weights. FD=Fully-Diffuse.

Equation		Function		
	Type	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
Gaussian	Full	2.00	0.94	0.92
	FD	2.00	0.97	0.95
Kansa	Full	1.97	0.93	0.96
	FD	1.97	0.91	0.93
Levi	Full	1.97	1.00	0.97
	FD	1.97	1.02	0.93
Schaffer	Full	1.98	0.89	0.90
	FD	1.98	0.89	0.90



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

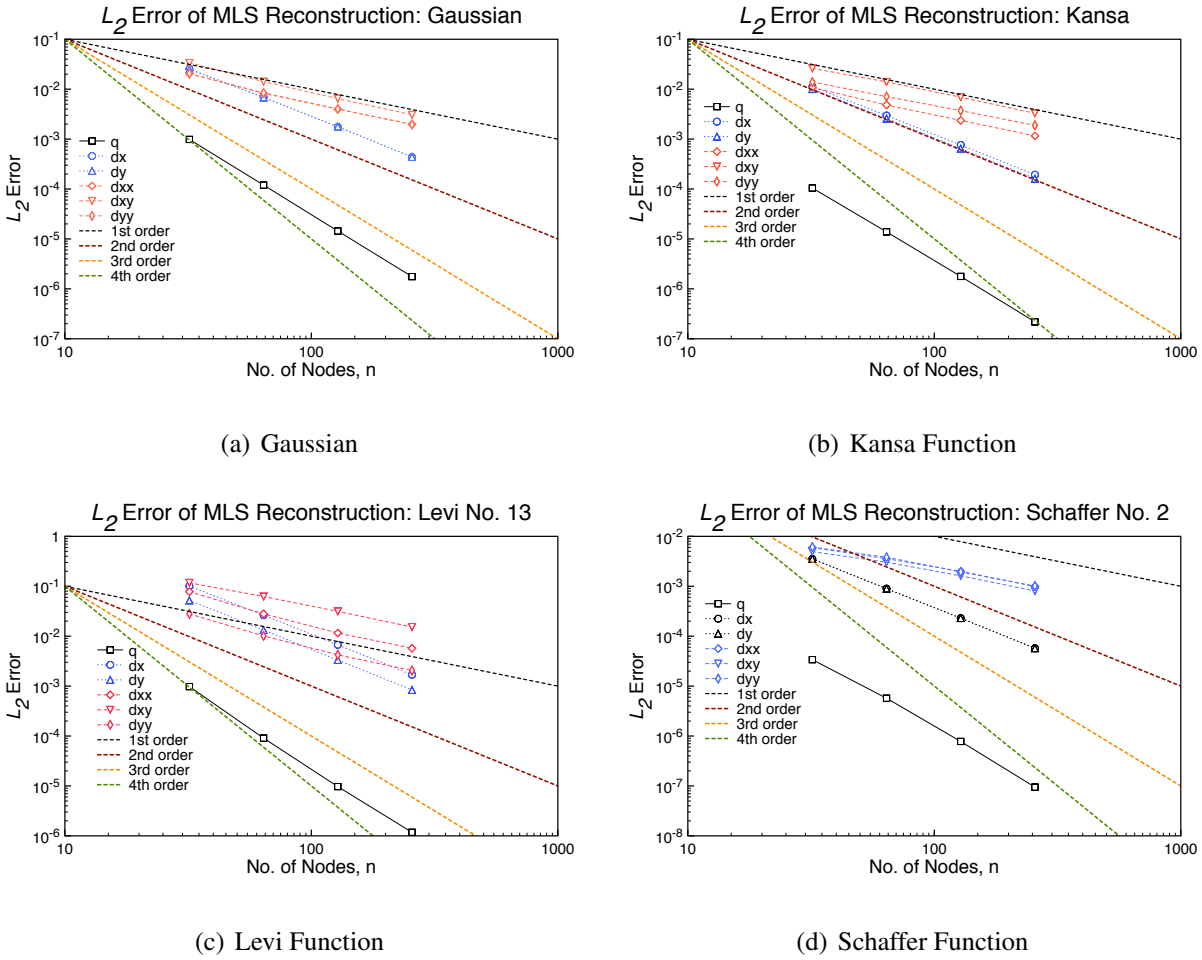
Figure 6.60: Error Norms on Unstructured Grids: 2^{nd}-Order Fully-Diffuse Affine MLS with Anisotropic Weights.

first derivatives are generally better for the fully-diffuse derivatives. This is probably due to the less accurate derivatives of $\mathbf{C}$ being neglected with the fully-diffuse method. It is also constructive to look at the timing for the full and fully-diffuse derivatives. Table 6.76 shows the timing for the reconstruction of the Gaussian. As seen from Table 6.76, the second-order fully-diffuse is only slightly faster than the full reconstruction, so it is not overly beneficial to use the fully-diffuse from a timing standpoint.

6.4.2.3.2 Third-Order The third-order MLS reconstruction of the derivatives can be in the full form (5.47), semi-diffuse (5.68), or in the fully-diffuse form (5.72). The semi-diffuse derivatives reconstructed with anisotropic weights are shown in Figure 6.61.

Table 6.76: Timing on Unstructured Grids: 2nd-Order Affine MLS with Anisotropic Weights, in Seconds.

Mesh Size	Type	
	Full	Full-Diffuse
32	0.054	0.052
64	0.241	0.133
128	0.994	0.906
256	4.027	3.577



(a) Gaussian

(b) Kansa Function

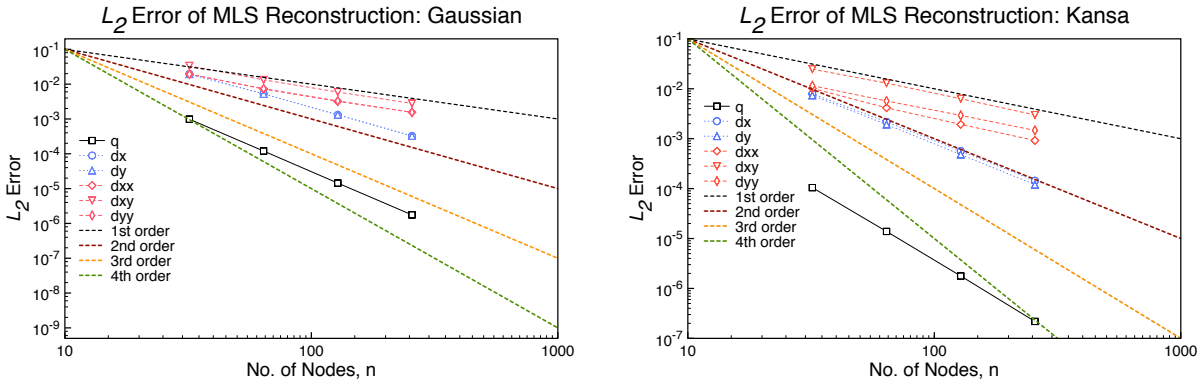
(c) Levi Function

(d) Schaffer Function

Figure 6.61: Error Norms on Unstructured Grids: 3rd-Order Semi-Diffuse Affine MLS with Anisotropic Weights.

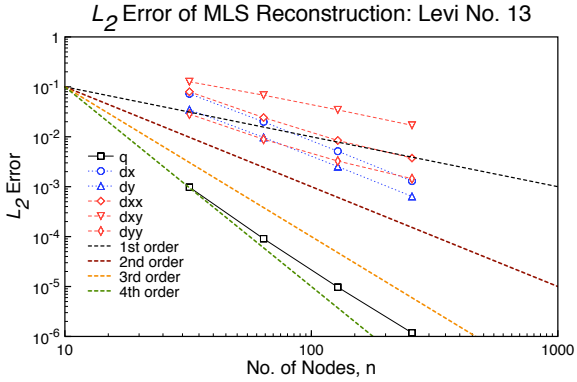
Figure 6.61 shows that the accuracy for the function is approximately third-order accurate, and

the first-derivatives second-order accurate, and second-derivatives first-order accurate. The fully-diffuse derivatives computed using Affine MLS with anisotropic weights are shown in Figure 6.62.

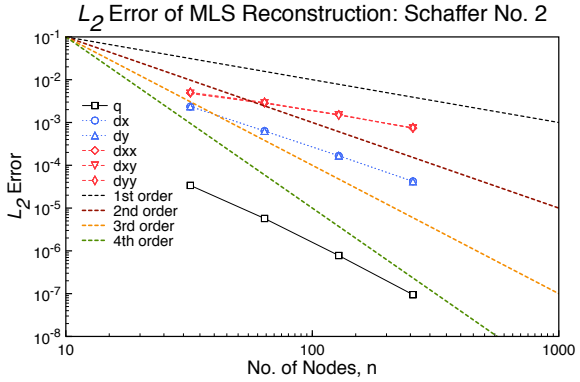


(a) Gaussian

(b) Kansa Function



(c) Levi Function



(d) Schaffer Function

Figure 6.62: Error Norms on Unstructured Grids: 3^{rd}-Order Fully-Diffuse Affine MLS with Anisotropic Weights.

Figure 6.62 shows that the accuracy for the function is approximately third-order accurate, and the first-derivatives second-order accurate, and second-derivatives first-order accurate. Table 6.77 compares the computed order of accuracy for each equation on unstructured grids using Affine MLS with anisotropic weights for the diffusivity of the derivatives. The functions have the same accuracy for any of the derivative methods, and the full and semi-diffuse have the same first deriva-

Table 6.77: Accuracy on Unstructured Grids: 3^{rd}-Order Diffuse Affine MLS with Anisotropic Weights. SD=Semi-Diffuse, FD=Fully-Diffuse.

Equation	Type	Function					
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
Gaussian	Full	3.05	2.00	1.99	1.00	1.11	1.03
	SD	3.05	2.00	1.99	1.03	1.11	1.08
	FD	3.05	2.01	2.00	1.11	1.14	1.14
Kansa	Full	3.01	1.97	2.00	1.02	1.05	0.97
	SD	3.01	1.97	2.00	1.07	1.06	0.98
	FD	3.01	1.95	2.00	1.11	1.07	0.99
Levi	Full	3.14	1.99	2.00	1.07	1.05	1.10
	SD	3.14	1.99	2.00	1.15	1.01	1.16
	FD	3.14	1.97	1.96	1.35	1.01	1.33
Schaffer	Full	2.97	1.98	1.99	0.94	0.96	0.98
	SD	2.97	1.98	1.99	0.95	0.97	1.01
	FD	2.97	1.95	1.95	0.98	0.98	1.00

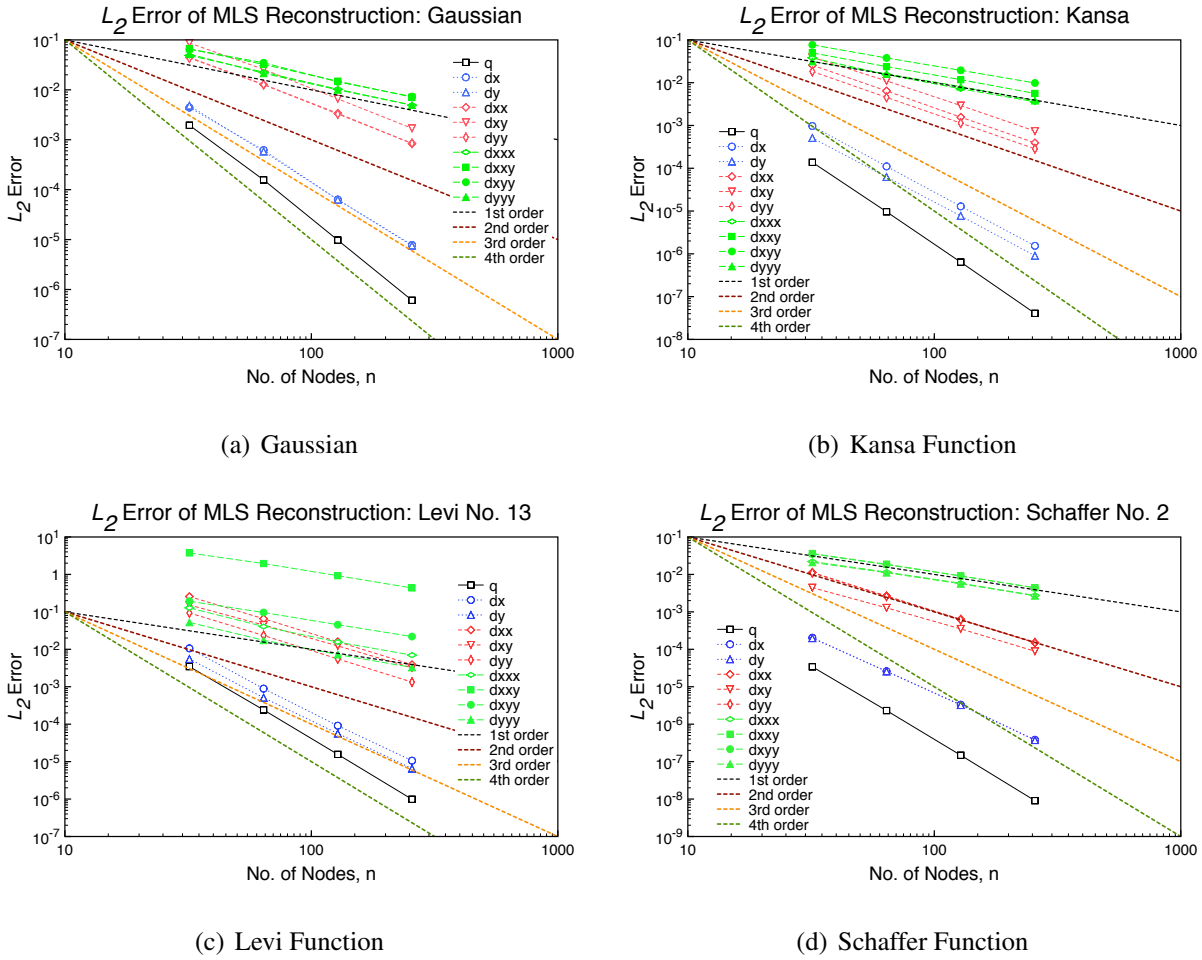
tive accuracy as shown in Table 6.65. The fully-diffuse first derivatives are slightly better for the Gaussian, but slightly worse for the other functions. The fully-diffuse second derivatives are more accurate than the full derivatives. As with the second-order results, this is probably due to the less accurate derivatives of $\mathbf{C}$ are not computed with the fully-diffuse method. It is also constructive to look at the timing for the full and fully-diffuse derivatives. Table 6.78 shows the timing for the reconstruction of the Gaussian. As seen from Table 6.78, the fully-diffuse is nearly twice as fast at

Table 6.78: Timing on Unstructured Grids: 3^{rd}-Order Affine MLS with Anisotropic Weights, in Seconds.

Mesh Size	Type		
	Full	Semi-Diffuse	Full-Diffuse
32	0.273	0.180	0.157
64	1.079	0.781	0.707
128	4.559	3.341	2.956
256	18.380	13.727	12.067

the highest level of refinement. Considering the fully-diffuse as accurate as the full reconstruction, it makes sense to use the fully-diffuse to reduce computational time for larger systems. Additionally, the semi-diffuse is nearly as fast as the fully-diffuse, so the additional terms are not much more expensive.

6.4.2.3.3 Fourth-Order The fourth-order MLS reconstruction of the derivatives can be in the full form (5.47), semi-diffuse (5.68) (with or without the second derivatives of $\mathbf{C}$), or in the fully-diffuse form (5.72). The semi-diffuse derivatives, including the second-derivatives of $\mathbf{C}$, reconstructed with anisotropic weights are shown in Figure 6.63.



(a) Gaussian

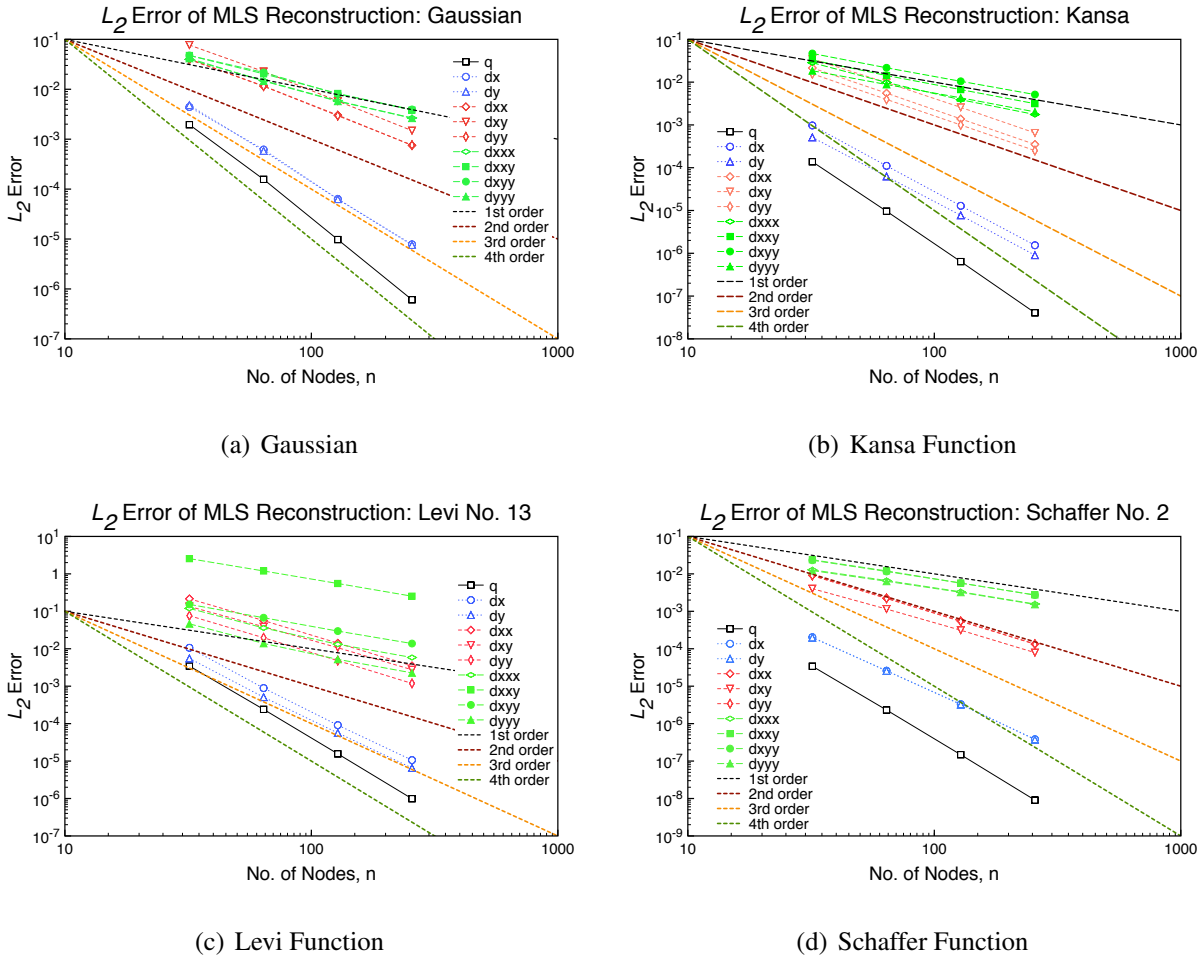
(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.63: Error Norms on Unstructured Grids: 4^{th}-Order Semi-Diffuse (1) Affine MLS with Anisotropic Weights.

Figure 6.63 shows that the function is approximately fourth-order accurate, first-derivatives approximately third-order accurate, the second-derivatives second-order accurate, and third-derivatives first-order accurate. The semi-diffuse derivatives, without the second-derivatives of C, reconstructed with Affine MLS using anisotropic weights are shown in Figure 6.64.



(a) Gaussian

(b) Kansa Function

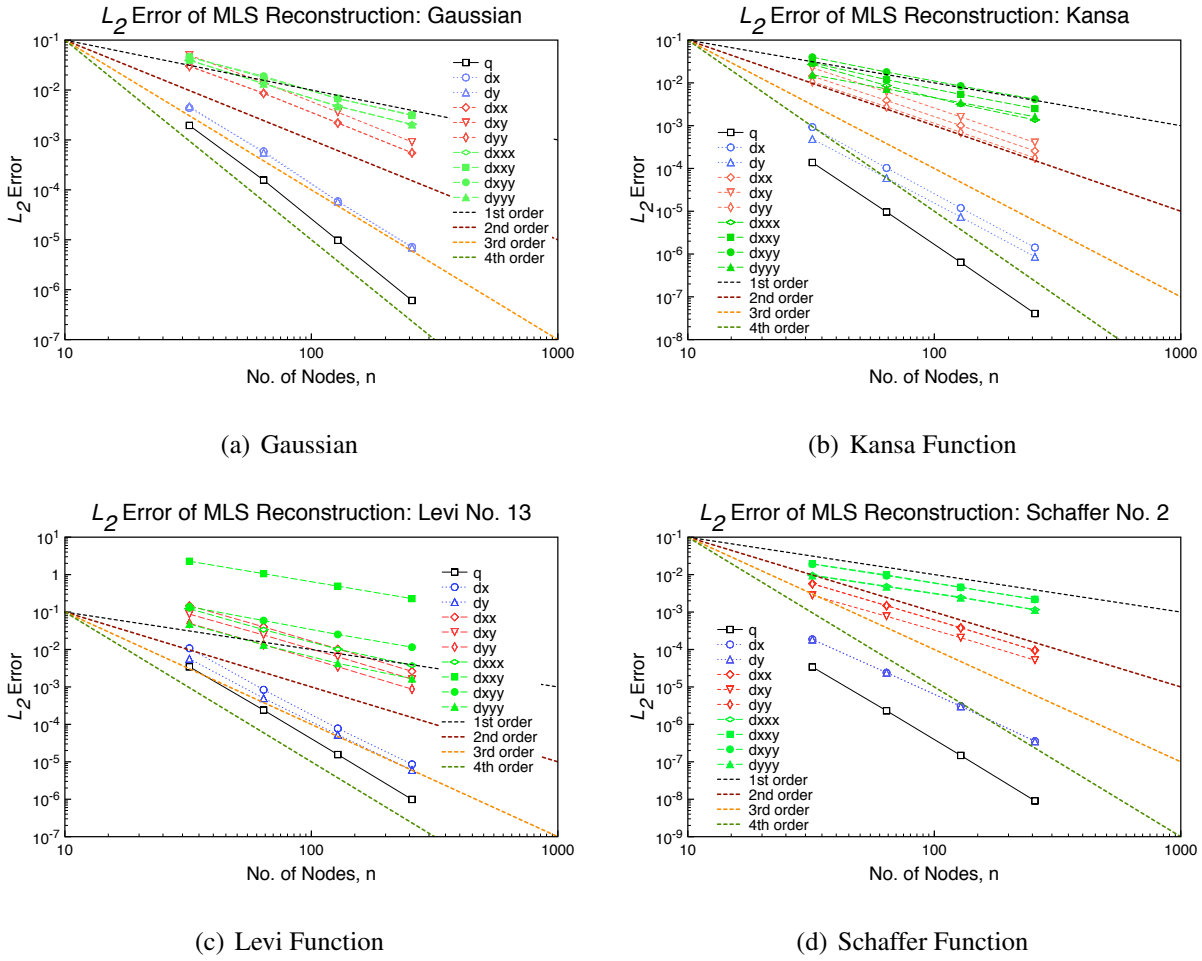
(c) Levi Function

(d) Schaffer Function

Figure 6.64: Error Norms on Unstructured Grids: 4^{th}-Order Semi-Diffuse (2) Affine MLS with Anisotropic Weights.

Figure 6.64 shows that the function is approximately fourth-order accurate, first-derivatives approximately third-order accurate, the second-derivatives second-order accurate, and third-derivatives first-order accurate.The fully-diffuse derivatives reconstructed with Affine MLS using anisotropic

weights are shown in Figure 6.65.



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.65: Error Norms on Unstructured Grids: 4^{th}-Order Fully-Diffuse Affine MLS with Anisotropic Weights.

Figure 6.65 shows that the function is approximately fourth-order accurate, first-derivatives approximately third-order accurate, the second-derivatives second-order accurate, and third-derivatives first-order accurate. Table 6.79 compares the computed order of accuracy for each equation on unstructured grids using Affine MLS with anisotropic weights for each diffusivity level of the derivatives. The results in Table 6.79 are similar to the third-order results in Table 6.77, where the methods produce effectively the same accuracy except for the second and third derivatives. As

Table 6.79: Accuracy on Unstructured Grids: 4^{th}-Order Diffuse Affine MLS with Anisotropic Weights.

Equation	Type	Function									
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial xy^2}$	$\frac{\partial^3 q}{\partial y^3}$
Gaussian	Full	4.00	3.15	3.13	1.95	1.97	1.96	1.08	1.04	1.08	1.07
	SD1	4.00	3.15	3.13	1.95	1.97	1.96	1.09	1.08	1.13	1.08
	SD2	4.00	3.15	3.13	1.97	1.97	1.97	1.21	1.20	1.23	1.22
	FD	4.00	3.17	3.15	1.99	1.98	1.98	1.36	1.26	1.28	1.36
Kansa	Full	3.93	3.07	3.05	2.06	1.99	2.04	1.03	1.09	1.00	1.08
	SD1	3.94	3.08	3.06	2.03	1.95	2.00	1.09	1.05	0.98	1.04
	SD2	3.94	3.08	3.06	2.00	1.96	1.99	1.24	1.09	1.04	1.06
	FD	3.94	3.08	3.06	1.96	1.97	1.98	1.34	1.12	1.05	1.08
Levi	Full	3.94	3.21	3.14	2.09	2.05	2.12	1.47	1.34	1.20	1.30
	SD1	3.96	3.15	3.12	2.03	1.97	2.06	1.15	1.11	1.08	1.16
	SD2	3.96	3.15	3.12	2.00	1.95	2.03	1.22	1.12	1.15	1.24
	FD	3.96	3.29	3.18	1.97	1.96	1.97	1.56	1.11	1.18	1.49
Schaffer	Full	3.98	3.05	3.04	2.12	1.96	2.11	1.06	1.10	1.08	1.10
	SD1	4.00	3.05	3.04	2.05	1.92	2.04	1.03	1.04	1.03	1.01
	SD2	4.00	3.05	3.04	2.02	1.93	2.02	1.05	1.07	1.05	1.00
	FD	4.00	3.05	3.05	1.99	1.95	1.98	1.05	1.09	1.06	1.02

with the isotropic weighted results, the semi-diffuse (2) method does not produce better second derivatives, as removing the second-derivatives of $\mathbf{C}$ is probably detrimental to the accuracy. The fully-diffuse produces the highest level of accuracy, as was the case for the lower orders. Additionally, the Affine MLS reconstructions for the non-Gaussian functions is more accurate for the function, which is as noted earlier probably a truncation issue when computing the error. Table 6.80 shows the timing for the reconstruction of the Gaussian. As seen from Table 6.80, the reduction in computation time is nearly two from full to semi-diffuse (with $\mathbf{C}$), and from full to full-diffuse the reduction is nearly four. Therefore, it is beneficial to use the diffuse derivatives when possible.

6.4.3 Stretched Grids

This section presents the results of order of accuracy of the diffusive MLS derivatives on stretched grids. First, the diffuse MLS derivatives using isotropic weights is presented. Next,

Table 6.80: Timing on Unstructured Grids: 4^{th}-Order Affine MLS with Anisotropic Weights, in Seconds.

Mesh Size	Type			
	Full	Semi-Diffuse (1)	Semi-Diffuse (2)	Full-Diffuse
32	1.835	0.945	0.642	0.555
64	8.265	4.193	2.817	2.435
128	35.241	17.737	11.877	10.686
256	143.068	71.096	46.561	40.553

the diffuse MLS derivatives using anisotropic weights are shown. The section concludes with the diffuse Affine MLS derivatives using anisotropic weights.

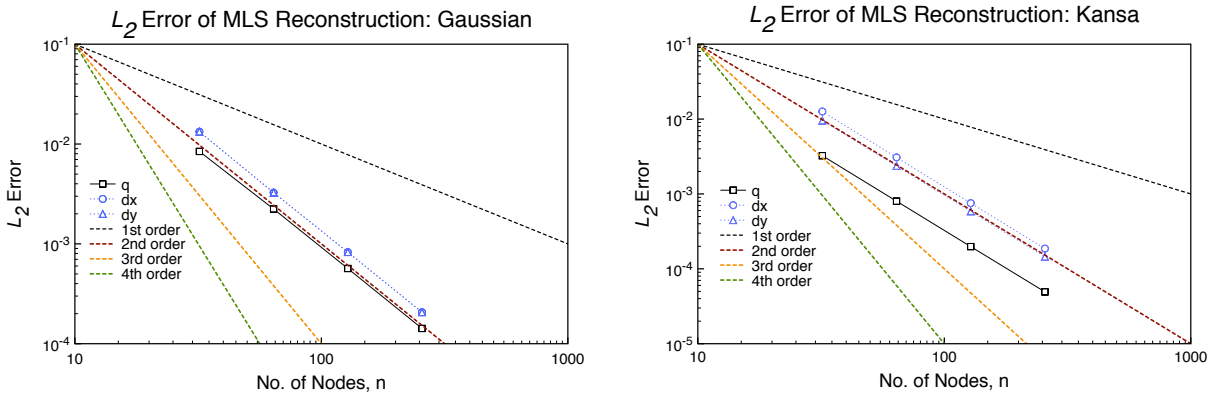
6.4.3.1 Isotropic Weights

This subsection presents diffuse derivatives using isotropic weights on stretched grids.

6.4.3.1.1 Second-Order The second-order MLS reconstruction of the derivatives can be in the full form (5.47) or in the fully-diffuse form (5.72). The fully-diffuse derivatives reconstructed with isotropic weights are shown in Figure 6.66. Figure 6.66 shows that the reconstruction achieves second-order accuracy for the function and first-derivatives. Table 6.81 compares the accuracy for the different diffusivity levels. From Table 6.81, there is no difference in the accuracy of the func-

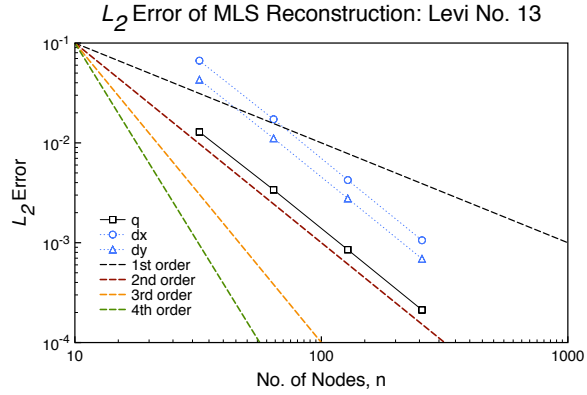
Table 6.81: Accuracy on Stretched Grids: 2^{nd}-Order Fully-Diffuse MLS with Isotropic Weights.FD=Fully-Diffuse.

Equation	Type	Function		
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
Gaussian	Full	1.99	1.98	1.98
	FD	1.99	1.99	1.99
Kansa	Full	2.01	2.02	2.01
	FD	2.01	2.02	2.01
Levi	Full	2.00	1.92	1.90
	FD	2.00	2.01	2.00
Schaffer	Full	2.01	1.88	1.88
	FD	2.01	2.02	2.02

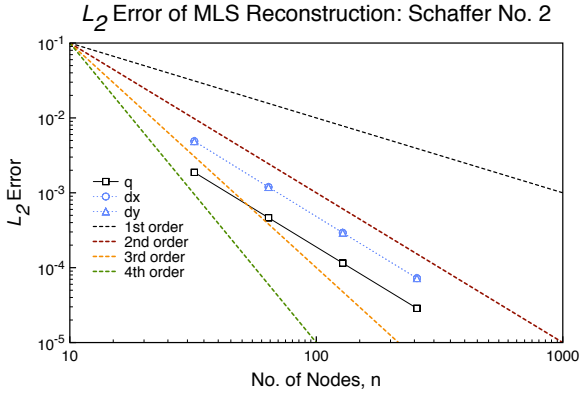


(a) Gaussian

(b) Kansa Function



(c) Levi Function



(d) Schaffer Function

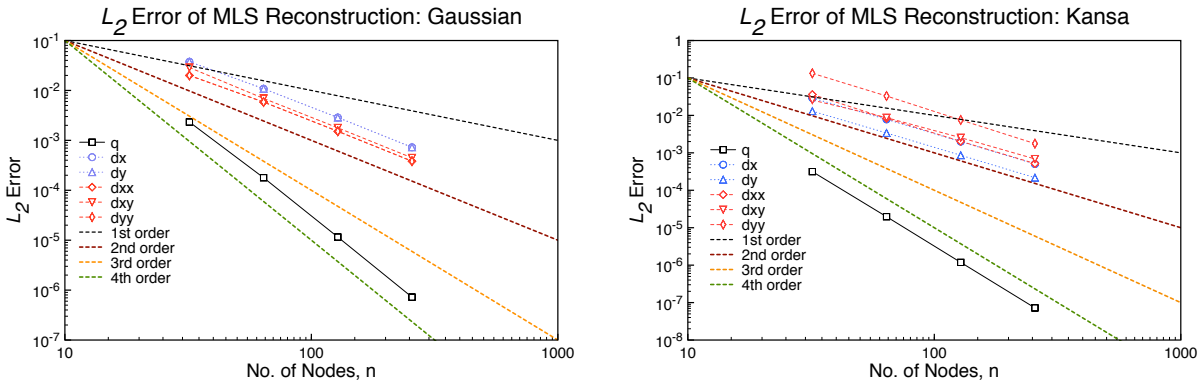
Figure 6.66: Error Norms on Stretched Grids: 2^{nd}-Order Fully-Diffuse MLS with Isotropic Weights.

tion based on the diffusivity of the derivatives. The first derivatives are either the same or slightly better when using the fully-diffuse derivatives. This is probably because the less accurate derivatives of $\mathbf{C}$ are not computed with the fully-diffuse method. It is also constructive to look at the timing for the full and fully-diffuse derivatives. Table 6.82 shows the timing for the reconstruction of the Gaussian. As seen from Table 6.82, the second-order fully-diffuse is only slightly faster than the full reconstruction, so it is not overly beneficial to use the fully-diffuse.

6.4.3.1.2 Third-Order The third-order MLS reconstruction of the derivatives can be in the full form (5.47), semi-diffuse (5.68), or in the fully-diffuse form (5.72). The semi-diffuse derivatives reconstructed with isotropic weights are shown in Figure 6.67.

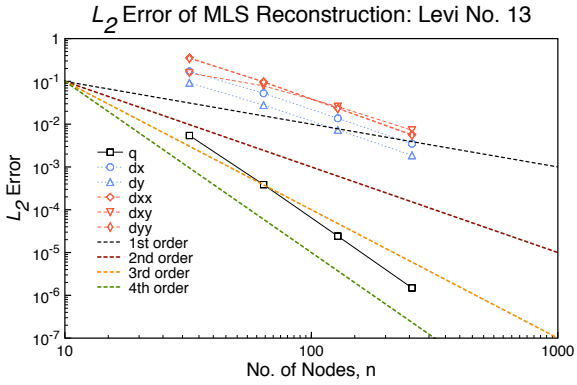
Table 6.82: Timing on Stretched Grids: 2^{nd}-Order MLS with Isotropic Weights, in Seconds.

Mesh Size	Type	
	Full	Full-Diffuse
32	0.068	0.061
64	0.262	0.253
128	1.023	0.933
256	4.051	3.747

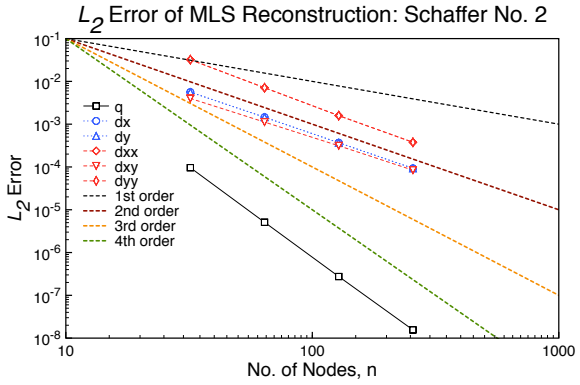


(a) Gaussian

(b) Kansa Function



(c) Levi Function

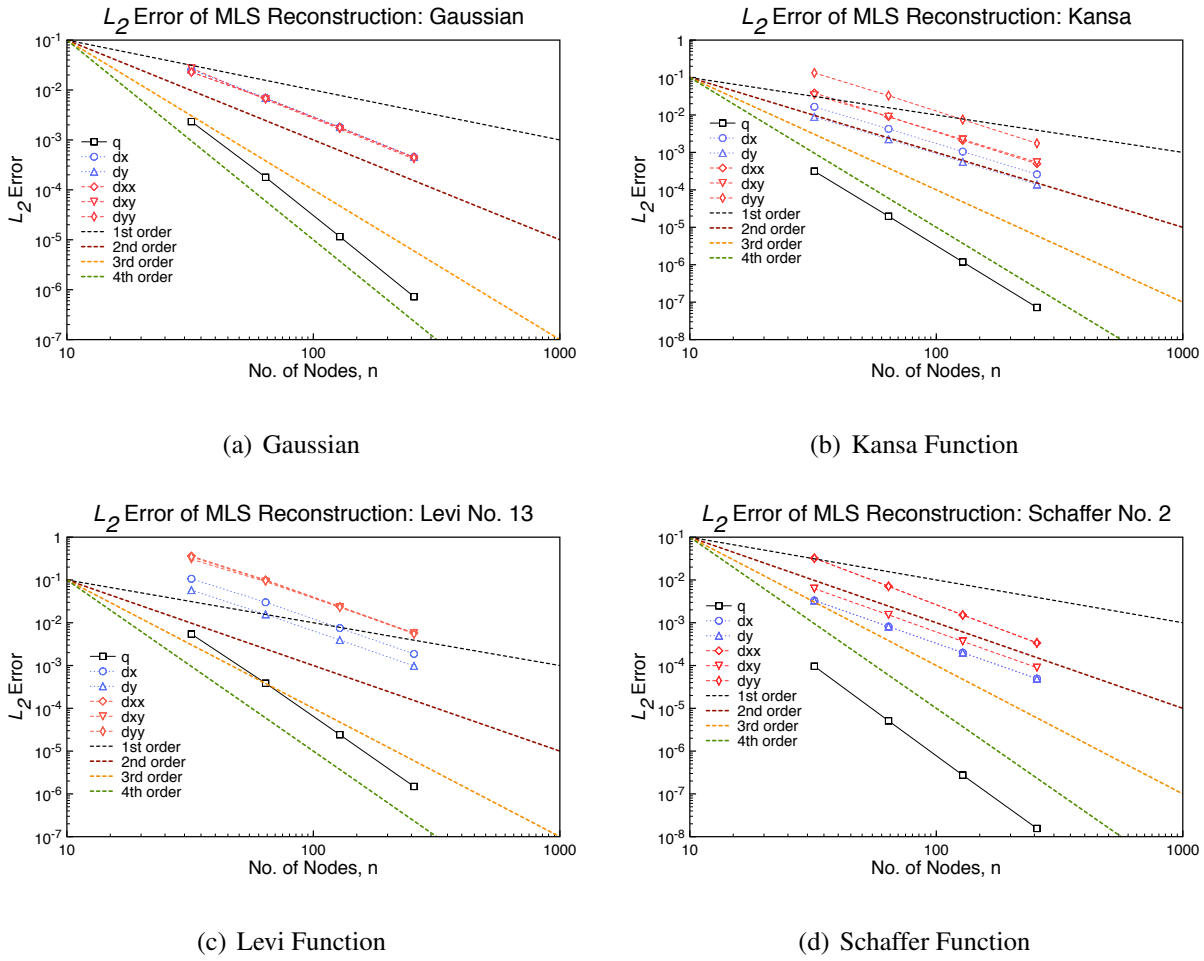


(d) Schaffer Function

Figure 6.67: Error Norms on Stretched Grids: 3^{rd}-Order Semi-Diffuse MLS with Isotropic Weights.

Figure 6.67 shows that the accuracy for the function is approximately fourth-order accurate, and the first- and second-order derivatives is second-order accurate. The fully-diffuse derivatives

reconstructed with isotropic weights are shown in Figure 6.68.



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.68: Error Norms on Stretched Grids: 3^{rd}-Order Fully-Diffuse MLS with Isotropic Weights.

Figure 6.68 shows that the accuracy for the function is approximately fourth-order accurate, and the first- and second-order derivatives is second-order accurate. Table 6.83 compares the accuracy for each of the diffusivity levels on stretched grids using isotropic weights. The functions have the same accuracy for any of the derivative methods, and the full and semi-diffuse have the same first derivative accuracy as shown in Table 6.83. The fully-diffuse first derivatives are slightly better for the Gaussian, but slightly worse for the other functions. The fully-diffuse second derivatives

Table 6.83: Accuracy on Stretched Grids: 3^{rd}-Order Diffuse MLS with Isotropic Weights. SD=Semi-Diffuse, FD=Fully-Diffuse.

Equation	Type	Function					
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
Gaussian	Full	3.97	1.95	1.95	1.97	1.98	1.97
	SD	3.97	1.95	1.95	1.97	1.98	1.97
	FD	3.97	1.97	1.97	1.96	1.98	1.96
Kansa	Full	4.05	1.98	1.98	2.10	1.83	2.11
	SD	4.05	1.98	1.98	2.00	1.83	2.11
	FD	4.05	2.02	2.01	2.09	2.02	2.12
Levi	Full	4.01	1.96	1.95	2.07	1.72	2.06
	SD	4.01	1.96	1.95	2.05	1.72	2.06
	FD	4.01	2.00	2.00	2.06	2.00	2.07
Schaffer	Full	4.18	1.98	1.98	2.13	1.86	2.13
	SD	4.18	1.98	1.98	2.12	1.87	2.12
	FD	4.18	2.02	2.02	2.20	2.05	2.20

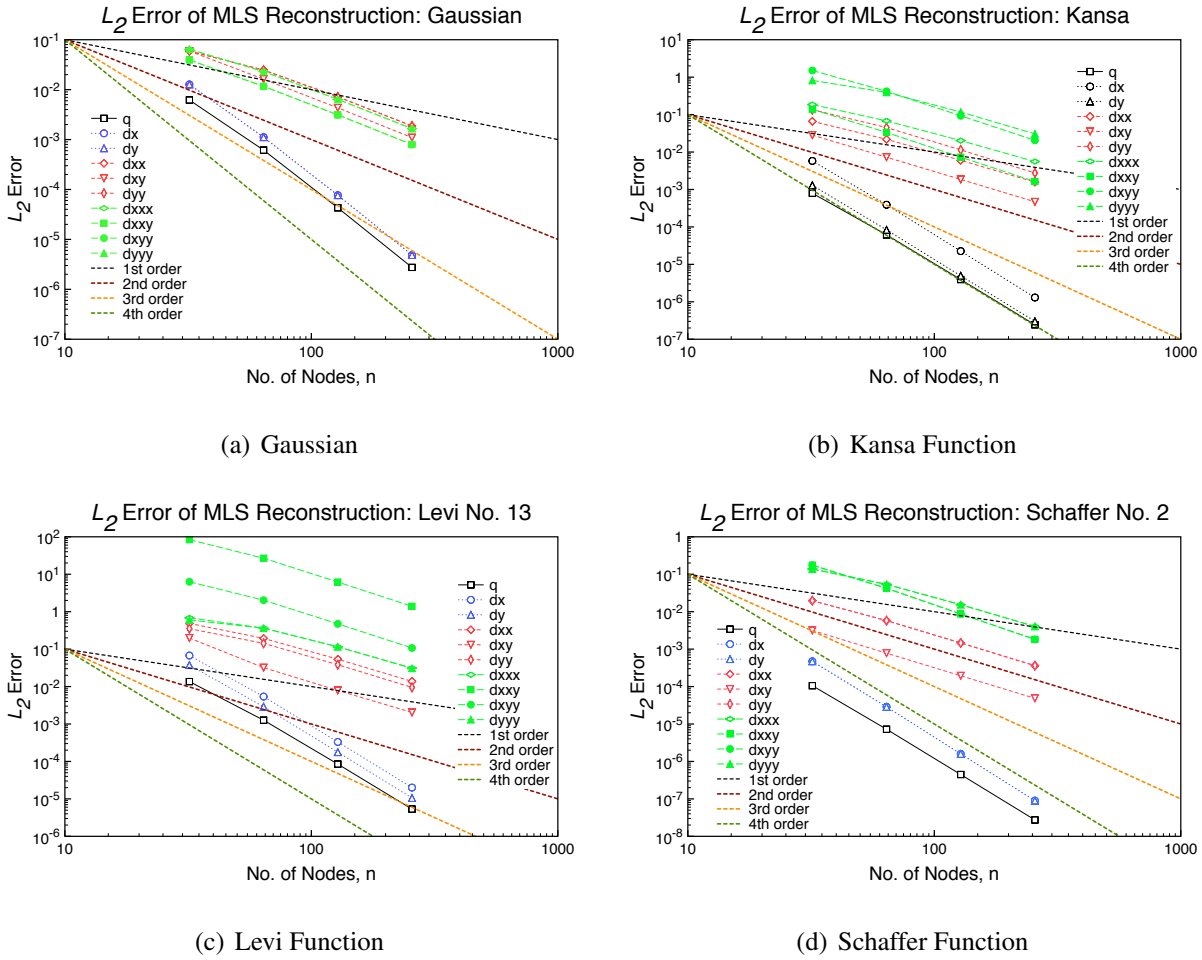
are more accurate than the full derivatives. As with the second-order results, this is probably due to the less accurate derivatives of $\mathbf{C}$ are not computed with the fully-diffuse method. It is also constructive to look at the timing for the full and fully-diffuse derivatives. Table 6.84 shows the timing for the reconstruction of the Gaussian. As seen from Table 6.84, the fully-diffuse is nearly

Table 6.84: Timing on Stretched Grids: 3^{rd}-Order MLS with Isotropic Weights, in Seconds.

Mesh Size	Type		
	Full	Semi-Diffuse	Full-Diffuse
32	0.210	0.160	0.154
64	0.857	0.651	0.573
128	3.338	2.544	2.260
256	12.97	9.976	8.950

twice as fast at the highest level of refinement. Considering the fully-diffuse as accurate as the full reconstruction, it makes sense to use the fully-diffuse to reduce computational time for larger systems.

6.4.3.1.3 Fourth-Order The fourth-order MLS reconstruction of the derivatives can be in the full form (5.47), semi-diffuse (5.68) (with or without the second derivatives of C), or in the fully-diffuse form (5.72). The semi-diffuse derivatives, including the second-derivatives of C, reconstructed with isotropic weights are shown in Figure 6.69.



(a) Gaussian

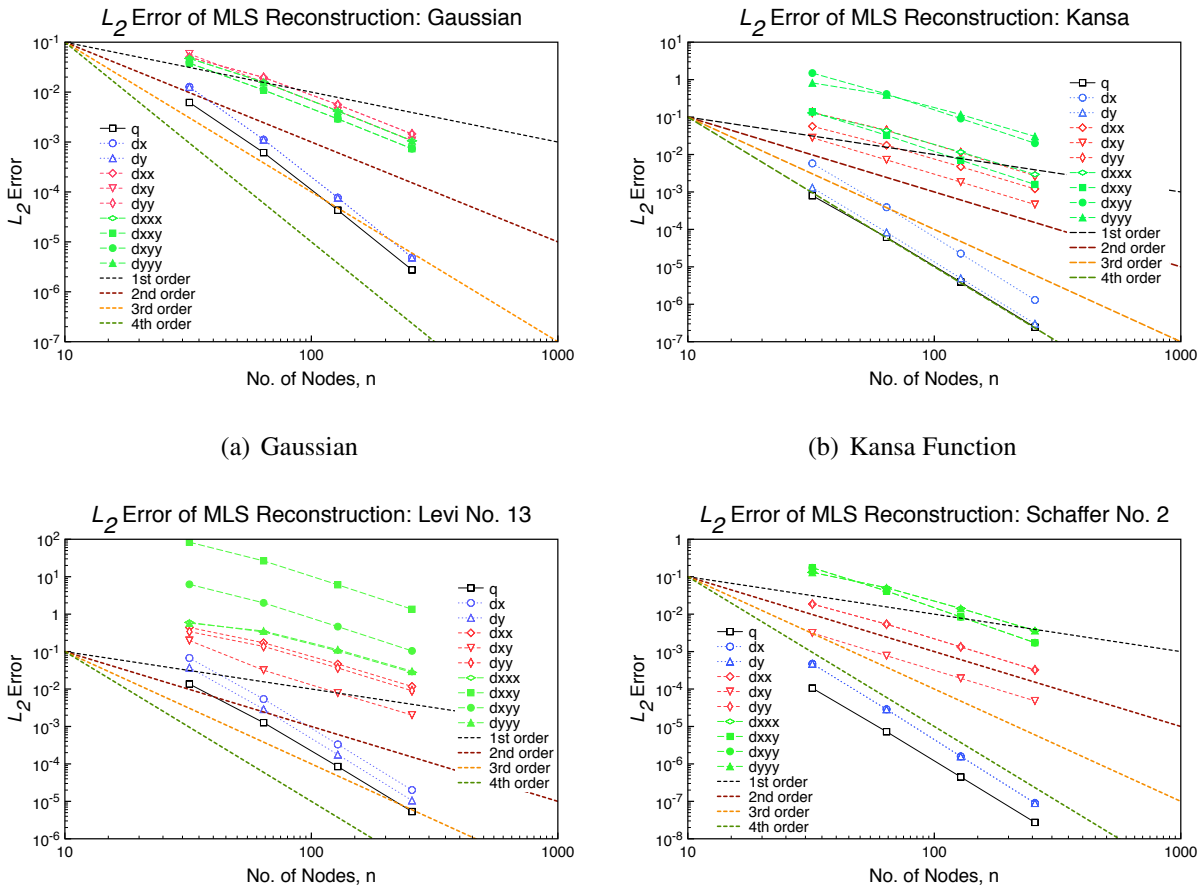
(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.69: Error Norms on Stretched Grids: 4^{th}-Order Semi-Diffuse (1) MLS with Isotropic Weights.

Figure 6.69 shows that the function and first-derivatives is approximately fourth-order, and the second- and third-derivatives is approximately second-order. The semi-diffuse derivatives, without the second-derivatives of C, reconstructed with isotropic weights are shown in Figure 6.70.



(a) Gaussian

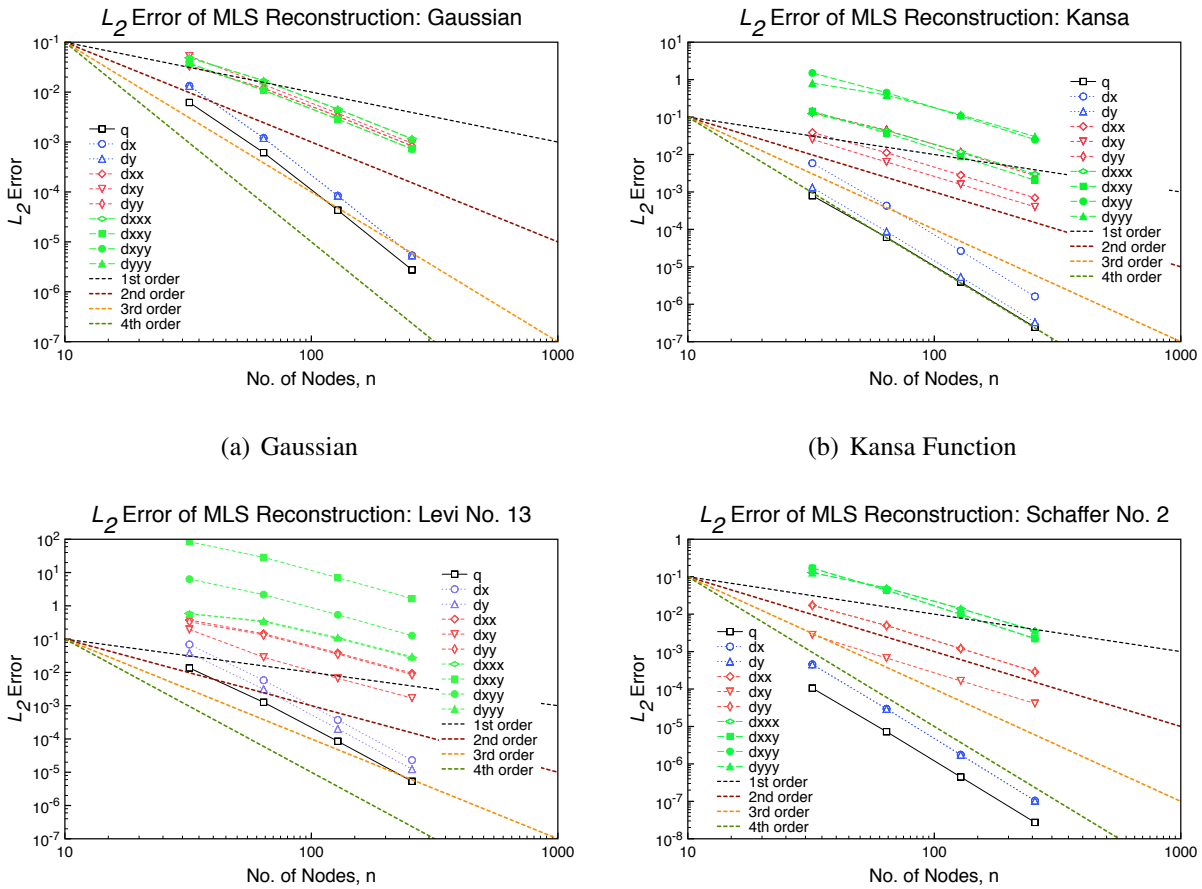
(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.70: Error Norms on Stretched Grids: 4^{th}-Order Semi-Diffuse (2) MLS with Isotropic Weights.

Figure 6.70 shows that the function and first-derivatives is approximately fourth-order, and the second- and third-derivatives is approximately second-order. The fully-diffuse derivatives reconstructed with isotropic weights are shown in Figure 6.71.



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.71: Error Norms on Stretched Grids: 4^{th}-Order Fully-Diffuse MLS with Isotropic Weights.

Figure 6.71 shows that the function and first-derivatives is approximately fourth-order, and the second- and third-derivatives is approximately second-order. Table 6.85 compares the accuracy for each of the diffusivity levels. The results in Table 6.85 are similar to the third-order results in Table 6.83, where the methods produce effectively the same accuracy except for the second and third derivatives. The fully-diffuse produces the highest level of accuracy, as was the case for the lower orders. It is also constructive to look at the timing for the varying levels of diffusivity. Table 6.86 shows the timing for the reconstruction of the Gaussian. Note that in the table, semi-diffuse (1) refers to the semi-diffuse results that include the second-derivative of C while semi-

Table 6.85: Accuracy on Stretched Grids: 4^{th}-Order Diffuse MLS with Isotropic Weights. SD1=Semi-Diffuse 1, SD2=Semi-Diffuse 2, FD=Fully-Diffuse

Equation	Type	Function									
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial xy^2}$	$\frac{\partial^3 q}{\partial y^3}$
Gaussian	Full	3.90	3.92	3.92	1.86	1.94	1.86	1.79	1.92	1.92	1.79
	SD1	3.90	3.92	3.92	1.86	1.94	1.86	1.90	1.93	1.93	1.90
	SD2	3.90	3.92	3.92	1.88	1.95	1.88	1.95	1.95	1.95	1.95
	FD	3.90	3.91	3.91	1.92	1.95	1.92	1.93	1.95	1.95	1.93
Kansa	Full	3.99	4.11	4.06	1.90	1.98	2.02	1.57	2.16	2.16	1.84
	SD1	3.99	4.11	4.06	1.90	1.98	2.02	1.80	2.18	2.17	1.84
	SD2	3.99	4.11	4.06	1.94	1.98	2.02	1.94	2.18	2.18	1.84
	FD	3.99	4.03	4.02	2.00	1.99	2.02	1.92	2.09	2.09	1.84
Levi	Full	3.94	4.04	4.05	1.91	1.99	1.95	1.62	2.12	2.11	1.71
	SD1	3.94	4.04	4.05	1.91	1.99	1.95	1.78	2.13	2.12	1.78
	SD2	3.94	4.04	4.05	1.94	1.99	1.97	1.80	2.15	2.13	1.78
	FD	3.94	3.99	3.99	1.97	2.04	1.97	1.79	2.06	2.05	1.78
Schaffer	Full	4.02	4.16	4.16	2.00	2.00	2.00	1.69	2.25	2.25	1.69
	SD1	4.02	4.16	4.16	2.00	2.00	2.00	1.87	2.27	2.27	1.87
	SD2	4.02	4.16	4.16	2.03	2.00	2.03	1.90	2.29	2.29	1.90
	FD	4.02	4.07	4.07	2.05	2.02	2.05	1.90	2.15	2.15	1.90

diffuse (2) does not include the second-derivative of $\mathbf{C}$. As seen from Table 6.86, the reduction in

Table 6.86: Timing on Stretched Grids: 4^{th}-Order MLS with Isotropic Weights, in Seconds.

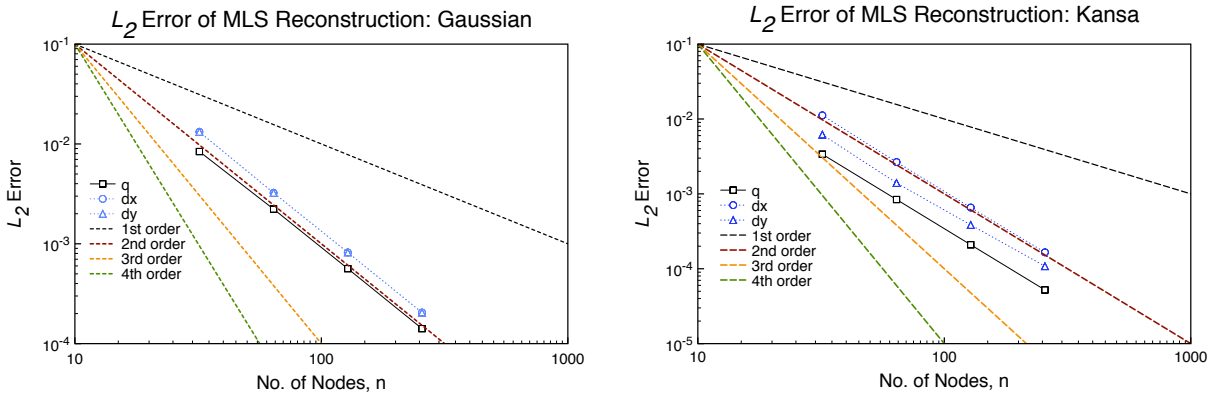
Mesh Size	Type			
	Full	Semi-Diffuse (1)	Semi-Diffuse (2)	Full-Diffuse
32	1.271	0.750	0.549	0.470
64	4.931	2.868	2.106	1.801
128	19.25	11.09	7.988	7.028
256	75.96	43.10	30.92	27.45

computation time is nearly two from full to semi-diffuse (with $\mathbf{C}$), and from full to full-diffuse the reduction is nearly three. Therefore, it is beneficial to use the diffuse derivatives when possible.

6.4.3.2 *Anisotropic Weights*

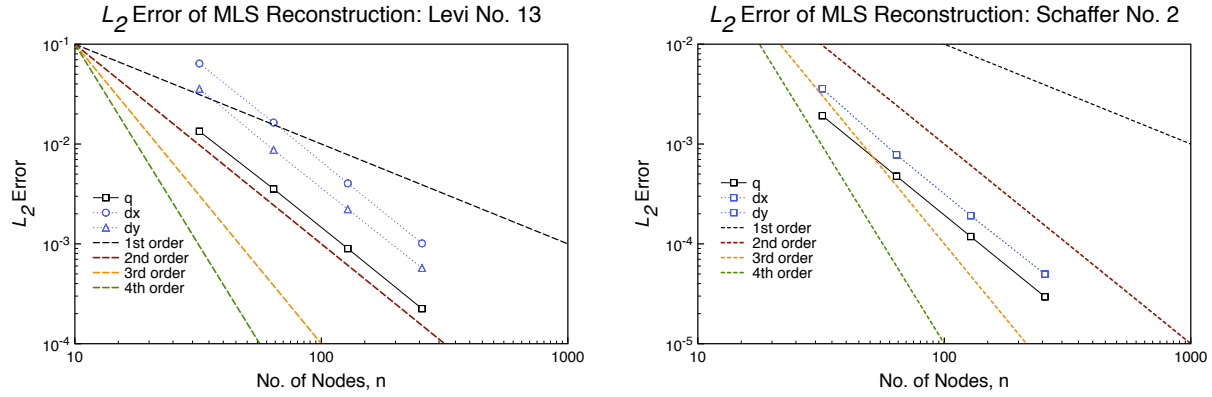
This subsection presents diffuse derivatives using anisotropic weights on stretched grids.

6.4.3.2.1 Second-Order The second-order MLS reconstruction of the derivatives can be in the full form (5.47) or in the fully-diffuse form (5.72). The fully-diffuse derivatives reconstructed with anisotropic weights are shown in Figure 6.72.



(a) Gaussian

(b) Kansa Function



(c) Levi Function

(d) Schaffer Function

Figure 6.72: Error Norms on Stretched Grids: 2^{nd}-Order Fully-Diffuse MLS with Anisotropic Weights.

From Figure 6.72, the function and first derivatives are approximately second-order. Table 6.87

compares the accuracy for each level of diffusivity. The functions have the same accuracy for any

Table 6.87: Accuracy on Stretched Grids: 2^{nd}-Order Diffuse MLS with Anisotropic Weights. FD=Fully-Diffuse.

Equation	Type	Function		
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
Gaussian	Full	1.99	1.93	1.93
	FD	1.99	1.99	1.99
Kansa	Full	2.01	1.73	1.71
	FD	2.01	2.00	1.85
Levi	Full	1.99	1.73	1.70
	FD	1.99	2.01	1.97
Schaffer	Full	2.01	1.72	1.72
	FD	2.01	1.88	1.88

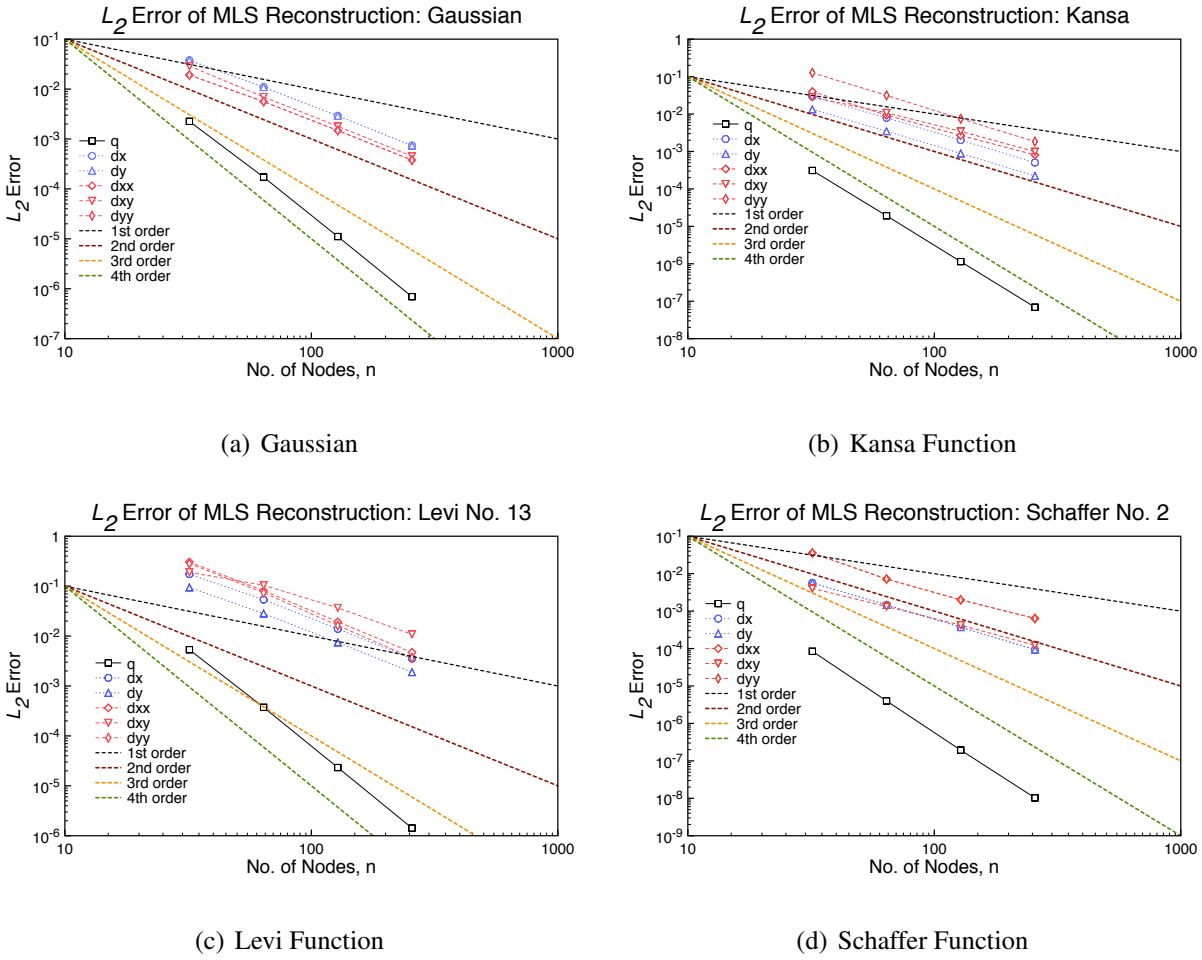
of the derivative methods, as seen in Table 6.87. The first derivatives are generally better for the fully-diffuse derivatives. This is probably due to the less accurate derivatives of C being neglected with the fully-diffuse method. It is also constructive to look at the timing for the full and fully-diffuse derivatives. Table 6.88 shows the timing for the reconstruction of the Gaussian. As seen

Table 6.88: Timing on Stretched Grids: 2^{nd}-Order MLS with Anisotropic Weights, in Seconds.

Mesh Size	Type	
	Full	Full-Diffuse
32	0.065	0.061
64	0.257	0.239
128	1.028	0.940
256	4.150	3.750

from Table 6.88, the second-order fully-diffuse is only slightly faster than the full reconstruction, so it is not overly beneficial to use the fully-diffuse from a timing standpoint.

6.4.3.2.2 Third-Order The third-order MLS reconstruction of the derivatives can be in the full form (5.47), semi-diffuse (5.68), or in the fully-diffuse form (5.72). The semi-diffuse derivatives reconstructed with anisotropic weights are shown in Figure 6.73.



(a) Gaussian

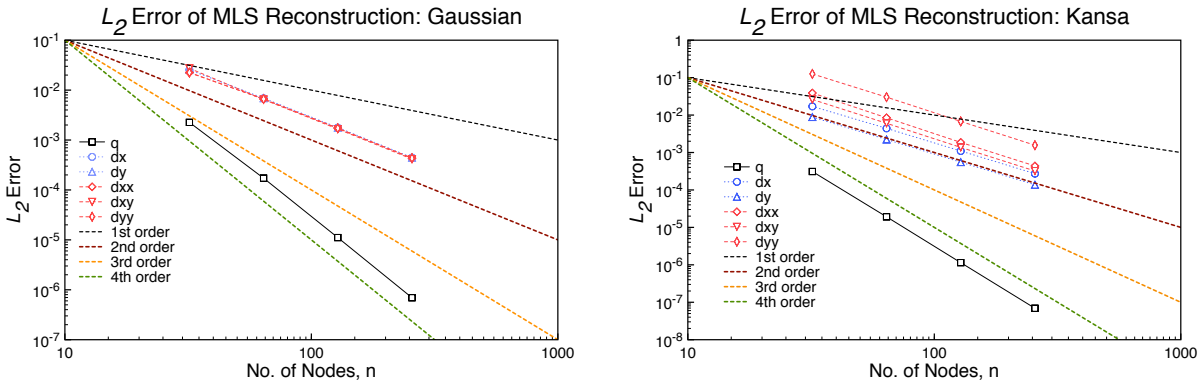
(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

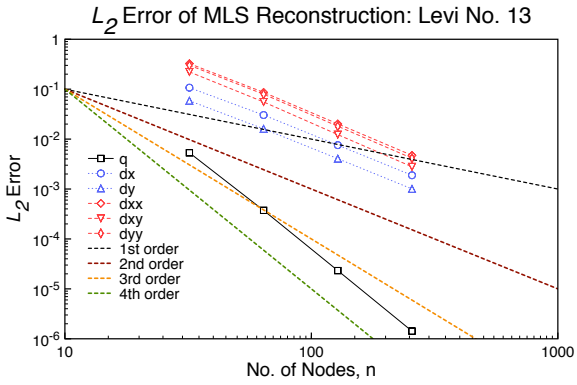
Figure 6.73: Error Norms on Stretched Grids: 3^{rd}-Order Semi-Diffuse MLS with Anisotropic Weights.

Figure 6.73 shows that the accuracy for the function is approximately fourth-order accurate, and the first- and second-derivatives are approximately second-order accurate. The fully-diffuse derivatives reconstructed with anisotropic weights are shown in Figure 6.74.

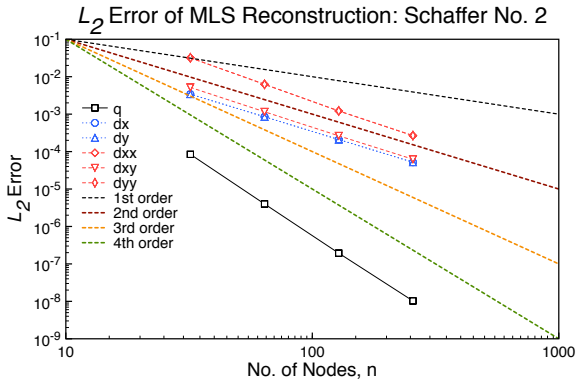


(a) Gaussian

(b) Kansa Function



(c) Levi Function



(d) Schaffer Function

Figure 6.74: Error Norms on Stretched Grids: 3^{rd}-Order Fully-Diffuse MLS with Anisotropic Weights.

Figure 6.74 shows that the accuracy for the function is approximately fourth-order accurate, and the first- and second-derivatives are approximately second-order accurate. Table 6.89 compares the order of accuracy for each level of diffusivity. The functions have the same accuracy for any of the derivative methods, and the full and semi-diffuse have the same first derivative accuracy as shown in Table 6.83. The fully-diffuse first derivatives are slightly better for the Gaussian, but slightly worse for the other functions. The fully-diffuse second derivatives are more accurate than the full derivatives. As with the second-order results, this is probably due to the less accurate derivatives of C are not computed with the fully-diffuse method. It is also constructive to look at the timing

Table 6.89: Accuracy on Stretched Grids: 3^{rd}-Order Diffuse MLS with Anisotropic Weights. SD=Semi-Diffuse, FD=Fully-Diffuse.

Equation	Type	Function					
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
Gaussian	Full	3.98	1.94	1.94	1.99	1.99	1.99
	SD	3.98	1.94	1.94	1.95	1.97	1.95
	FD	3.98	1.97	1.97	1.97	1.98	1.97
Kansa	Full	4.06	1.98	1.98	1.95	2.29	1.95
	SD	4.06	1.98	1.98	1.76	1.72	2.05
	FD	4.06	2.01	2.01	2.14	2.11	2.13
Levi	Full	4.02	1.96	1.95	1.95	2.48	1.95
	SD	4.02	1.96	1.95	2.06	1.63	2.16
	FD	4.02	2.00	2.00	2.10	2.16	2.13
Schaffer	Full	4.30	1.99	1.99	1.94	2.06	1.94
	SD	4.30	2.02	2.02	2.28	2.11	2.27
	FD	4.30	2.02	2.02	2.28	2.11	2.27

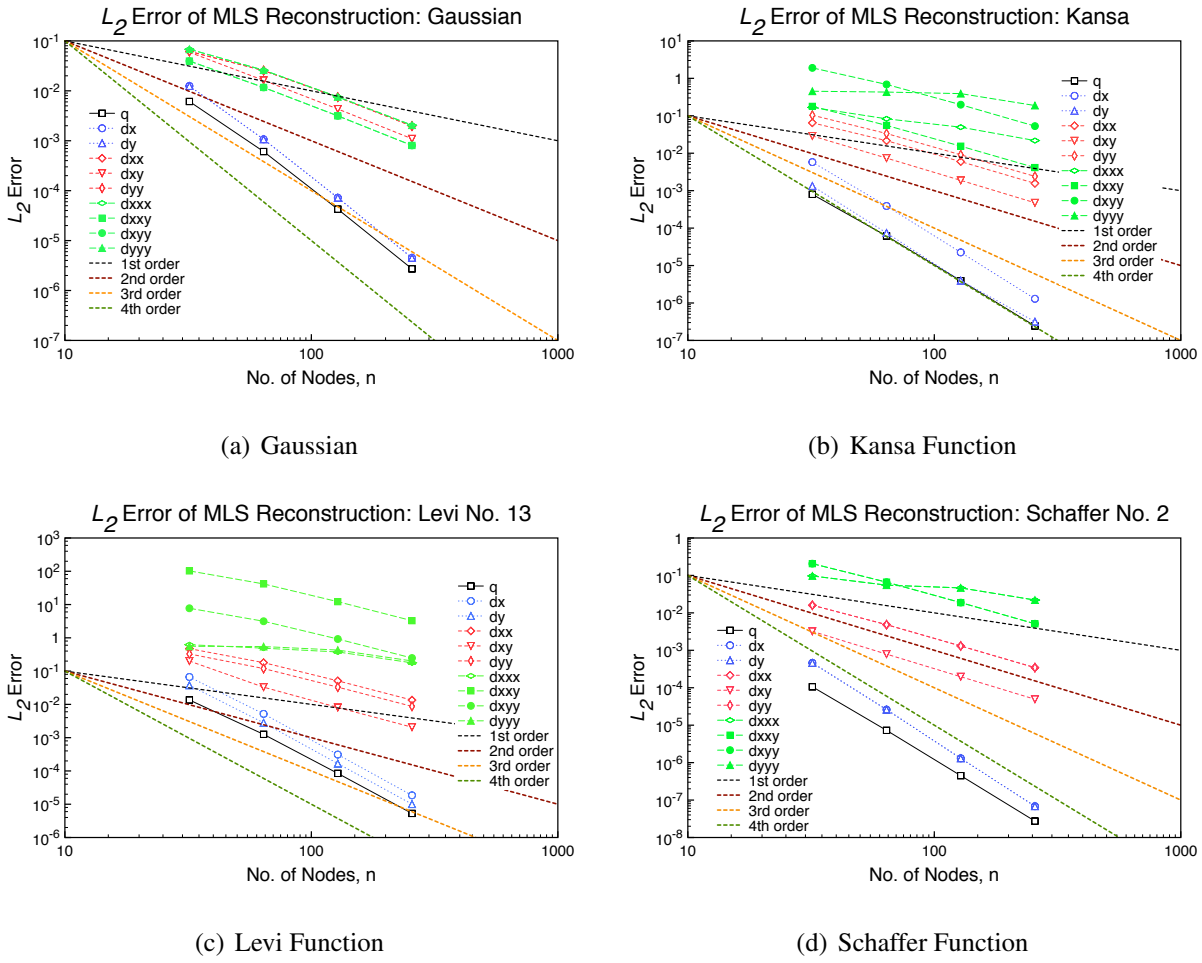
for the full and fully-diffuse derivatives. Table 6.90 shows the timing for the reconstruction of the Gaussian. As seen from Table 6.90, the fully-diffuse is nearly twice as fast at the highest level

Table 6.90: Timing on Stretched Grids: 3^{rd}-Order MLS with Anisotropic Weights, in Seconds.

Mesh Size	Type		
	Full	Semi-Diffuse	Full-Diffuse
32	0.216	0.161	0.151
64	0.841	0.654	0.584
128	3.290	2.567	2.276
256	13.10	10.01	9.002

of refinement. Considering the fully-diffuse as accurate as the full reconstruction, it makes sense to use the fully-diffuse to reduce computational time for larger systems. Additionally, the semi-diffuse is nearly as fast as the fully-diffuse, so the additional terms are not much more expensive.

6.4.3.2.3 Fourth-Order The fourth-order MLS reconstruction of the derivatives can be in the full form (5.47), semi-diffuse (5.68) (with or without the second derivatives of C), or in the fully-diffuse form (5.72). The semi-diffuse derivatives, including the second-derivatives of C, reconstructed with anisotropic weights are shown in Figure 6.75.



(a) Gaussian

(b) Kansa Function

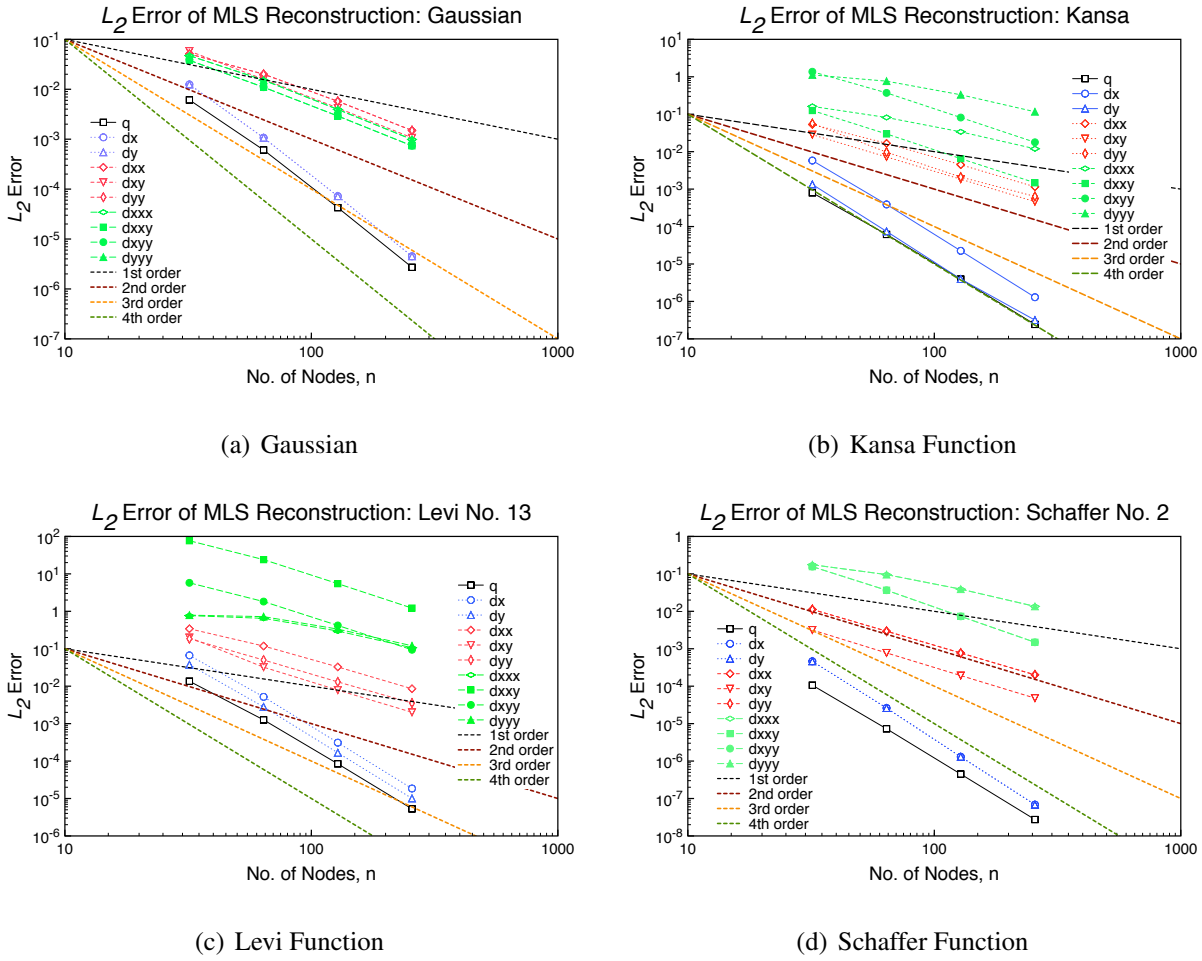
(c) Levi Function

(d) Schaffer Function

Figure 6.75: Error Norms on Stretched Grids: 4^{th}-Order Semi-Diffuse (1) MLS with Anisotropic Weights.

Figure 6.75 shows that the function and first-derivatives are approximately fourth-order accurate, and second- and third-derivatives are approximately second-order accurate. Compared to the

results for the full reconstruction shown in Figure 6.35, the third-derivatives now have decreasing error as the mesh is refined, showing the higher-order derivatives of C are harder to model in this case. The semi-diffuse derivatives, without the second-derivatives of C, reconstructed with anisotropic weights are shown in Figure 6.76.



(a) Gaussian

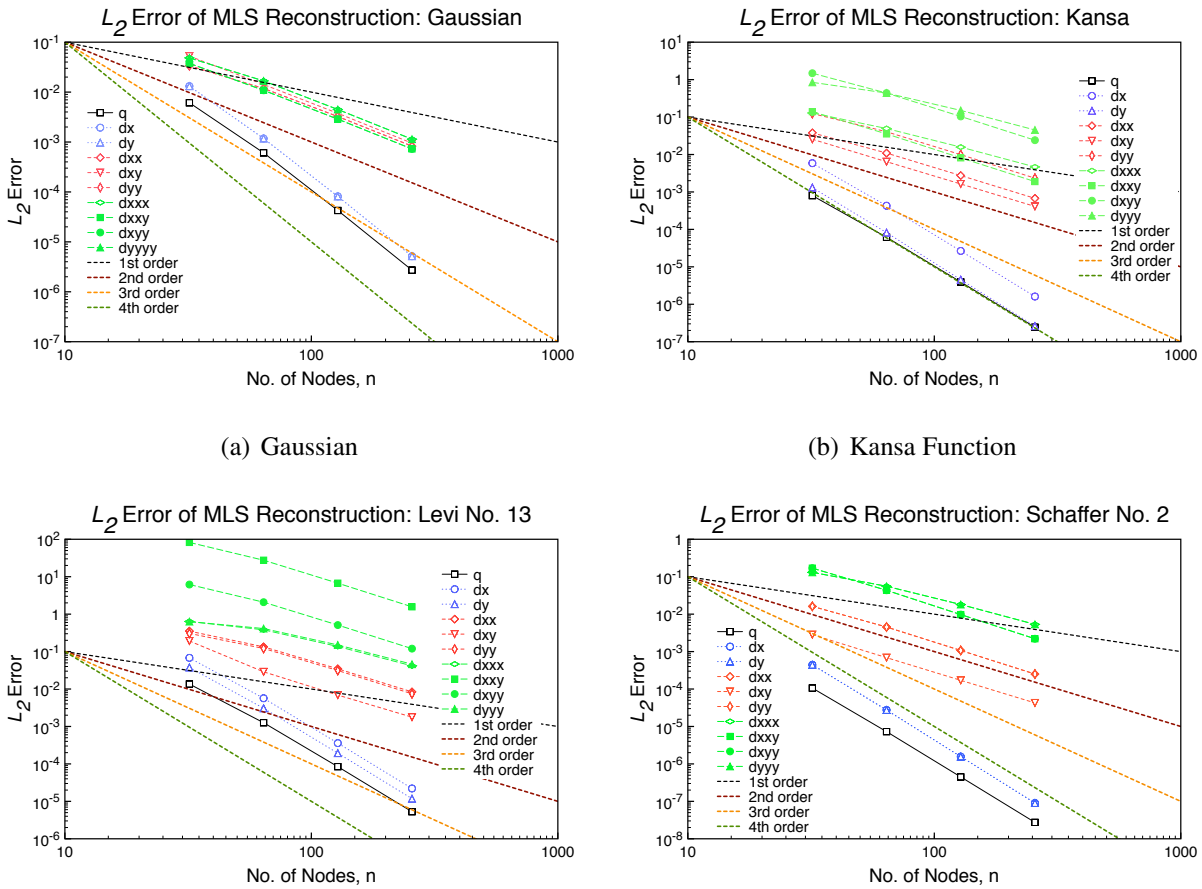
(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.76: Error Norms on Stretched Grids: 4^{th}-Order Semi-Diffuse (2) MLS with Anisotropic Weights.

Figure 6.76 shows that the function and first-derivatives are approximately fourth-order accurate, and second- and third-derivatives are approximately second-order accurate. The fully-diffuse derivatives reconstructed with anisotropic weights are shown in Figure 6.77.



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.77: Error Norms on Stretched Grids: 4^{th}-Order Fully-Diffuse MLS with Anisotropic Weights.

Figure 6.77 shows that the function and first-derivatives are approximately fourth-order accurate, and second- and third-derivatives are approximately second-order accurate. Table 6.91 compares the computed order of accuracy for each level of diffusivity. The results in Table 6.91 are similar to the third-order results in Table 6.89, where the methods produce effectively the same accuracy except for the second and third derivatives. The semi-diffuse (1) method produces the same level of accuracy as the full derivatives, while the semi-diffuse (2) and fully-diffuse derivatives produce more accurate higher-derivatives. The fully-diffuse produces the best higher-derivatives, most likely due to the elimination of the higher derivatives of $\mathbf{C}$. Table 6.92 shows the timing for

Table 6.91: Accuracy on Stretched Grids: 4^{th}-Order Diffuse MLS with Anisotropic Weights. SD1=Semi-Diffuse 1, SD2=Semi-Diffuse 2, FD=Fully-Diffuse.

Equation	Type	Function									
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial xy^2}$	$\frac{\partial^3 q}{\partial y^3}$
Gaussian	Full	3.90	3.94	3.94	1.85	1.94	1.85	1.83	1.93	1.93	1.83
	SD1	3.90	3.94	3.94	1.85	1.94	1.85	1.83	1.93	1.93	1.83
	SD2	3.90	3.94	3.94	1.87	1.95	1.87	1.98	1.95	1.95	1.98
	FD	3.90	3.92	3.92	1.92	1.95	1.92	1.94	1.95	1.95	1.94
Kansa	Full	3.99	4.12	3.94	1.89	1.98	1.90	0.96	1.87	1.85	0.60
	SD1	3.99	4.12	3.94	1.89	1.98	1.90	0.96	1.87	1.85	0.60
	SD2	3.99	4.12	3.94	1.93	1.98	1.97	1.39	2.17	2.19	1.36
	FD	3.99	4.03	4.15	2.01	1.98	2.06	1.70	2.11	2.09	1.64
Levi	Full	3.95	4.07	4.06	1.87	2.00	1.89	0.73	1.83	1.84	0.72
	SD1	3.95	4.07	4.06	1.87	2.00	1.89	0.73	1.83	1.84	0.72
	SD2	3.95	4.07	4.06	1.90	2.00	1.91	1.30	2.15	2.14	1.29
	FD	3.95	4.00	4.01	2.00	2.01	2.01	1.60	2.07	2.06	1.58
Schaffer	Full	4.03	4.29	4.29	1.91	2.00	1.91	0.66	1.85	1.85	0.66
	SD1	4.03	4.29	4.29	1.91	2.00	1.91	0.66	1.85	1.85	0.66
	SD2	4.03	4.29	4.29	1.94	2.01	1.94	1.41	2.31	2.31	1.41
	FD	4.03	4.13	4.13	2.09	2.01	2.09	1.69	2.15	2.15	1.69

the reconstruction of the Gaussian. As seen from Table 6.92, the reduction in computation time is

Table 6.92: Timing on Stretched Grids: 4^{th}-Order MLS with Anisotropic Weights, in Seconds.

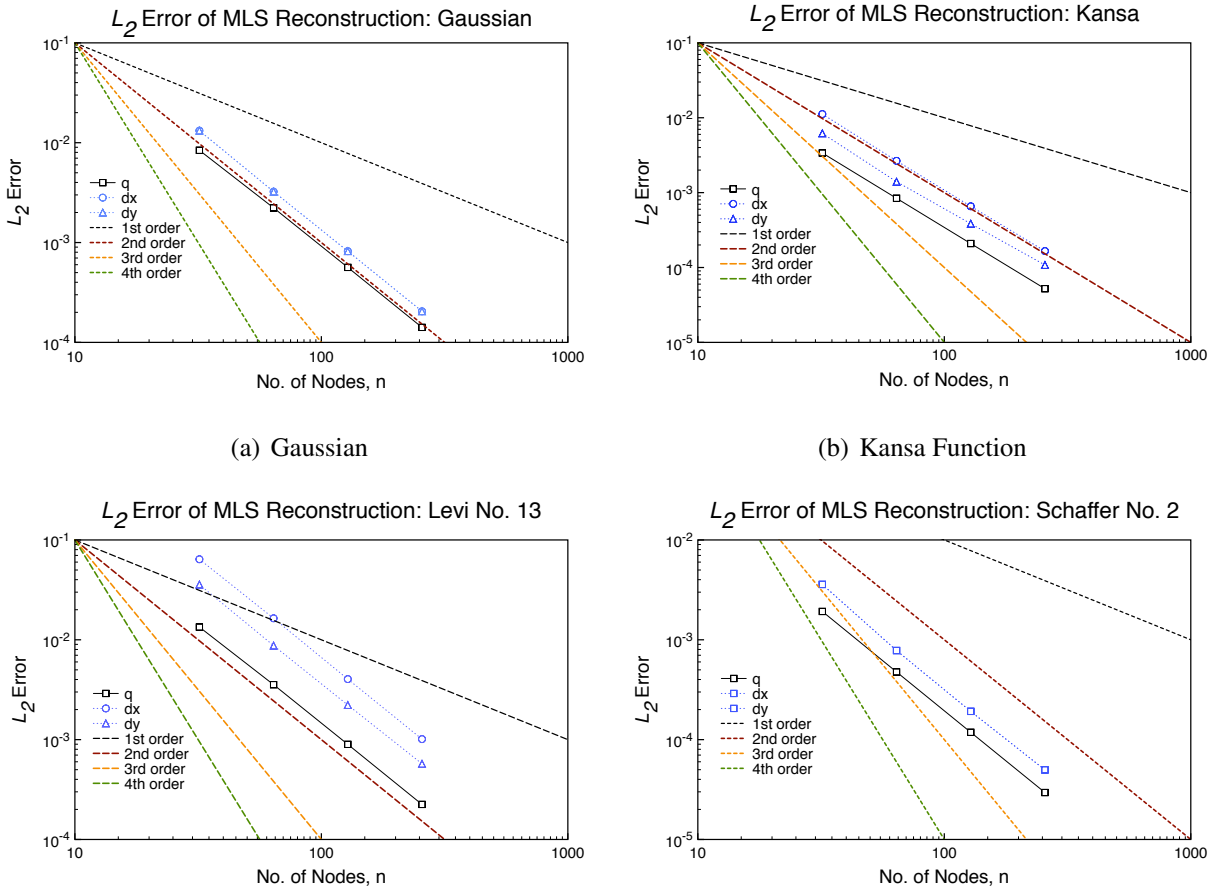
Mesh Size	Type			
	Full	Semi-Diffuse (1)	Semi-Diffuse (2)	Full-Diffuse
32	1.281	0.751	0.546	0.483
64	4.985	2.830	2.075	1.805
128	19.22	11.00	8.035	6.991
256	75.96	42.25	31.09	27.003

nearly two from full to semi-diffuse (with $\mathbf{C}$), and from full to full-diffuse the reduction is nearly four. Therefore, it is beneficial to use the diffuse derivatives when possible.

6.4.3.3 Affine MLS

This subsection presents the diffuse derivatives on stretched grids using Affine MLS with anisotropic weights.

6.4.3.3.1 Second-Order The second-order MLS reconstruction of the derivatives can be in the full form (5.47) or in the fully-diffuse form (5.72). The Affine MLS fully-diffuse derivatives reconstructed with anisotropic weights are shown in Figure 6.78.



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.78: Error Norms on Stretched Grids: 2^{nd}-Order Fully-Diffuse Affine MLS with Anisotropic Weights.

Table 6.93: Accuracy on Stretched Grids: 2^{nd}-Order Diffuse Affine MLS with Anisotropic Weights. FD=Fully-Diffuse.

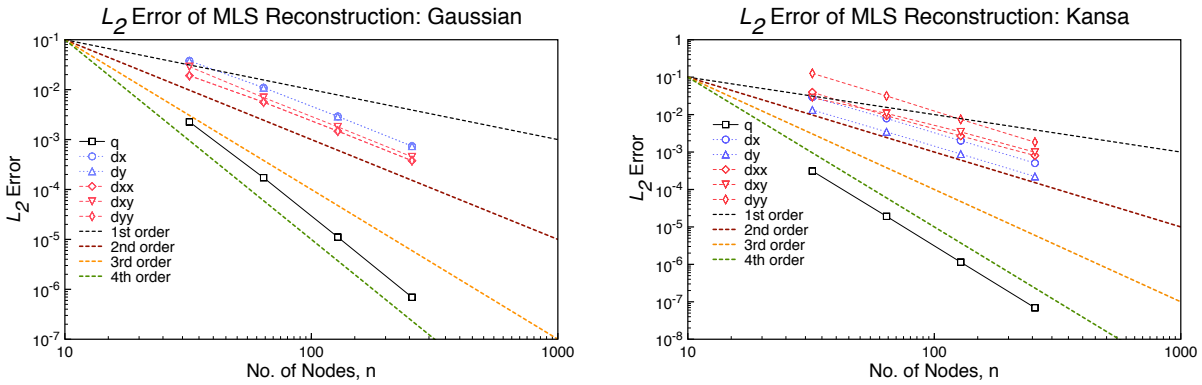
Equation	Type	Function		
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
Gaussian	Full	1.99	1.93	1.93
	FD	1.99	1.99	1.99
Kansa	Full	2.01	1.73	1.71
	FD	2.01	2.00	1.85
Levi	Full	1.99	1.73	1.70
	FD	1.99	2.01	1.97
Schaffer	Full	2.01	1.72	1.72
	FD	2.01	1.99	1.99

From Figure 6.78, the function and first-derivatives are reconstructed approximately second-order. Table 6.93 compares the accuracy for each level of diffusivity. The functions have the same accuracy for any of the derivative methods, as seen in Table 6.93. The first derivatives are generally better for the fully-diffuse derivatives. This is probably due to the less accurate derivatives of C being neglected with the fully-diffuse method. It is also constructive to look at the timing for the full and fully-diffuse derivatives. Table 6.94 shows the timing for the reconstruction of the Gaussian. As seen from Table 6.94, the second-order fully-diffuse is only slightly faster than the full reconstruction, so it is not overly beneficial to use the fully-diffuse from a timing standpoint.

Table 6.94: Timing on Stretched Grids: 2^{nd}-Order Affine MLS with Anisotropic Weights, in Seconds.

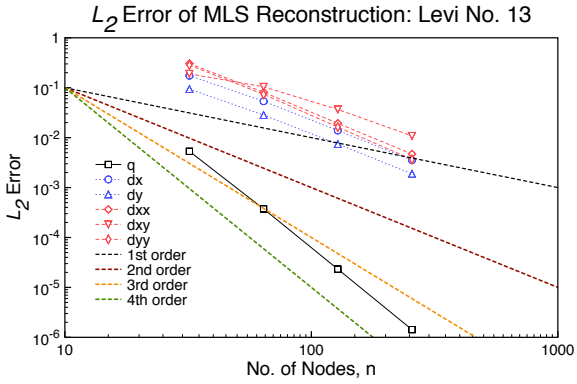
Mesh Size	Type	
	Full	Full-Diffuse
32	0.071	0.064
64	0.284	0.258
128	1.108	1.013
256	4.417	4.094

6.4.3.3.2 Third-Order The third-order MLS reconstruction of the derivatives can be in the full form (5.47), semi-diffuse (5.68), or in the fully-diffuse form (5.72). The full derivatives, using Affine MLS reconstruction with anisotropic weights, were shown in Table 6.48. The semi-diffuse derivatives reconstructed with anisotropic weights are shown in Figure 6.79.

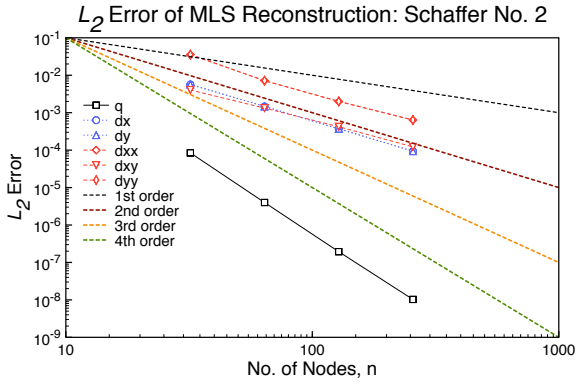


(a) Gaussian

(b) Kansa Function



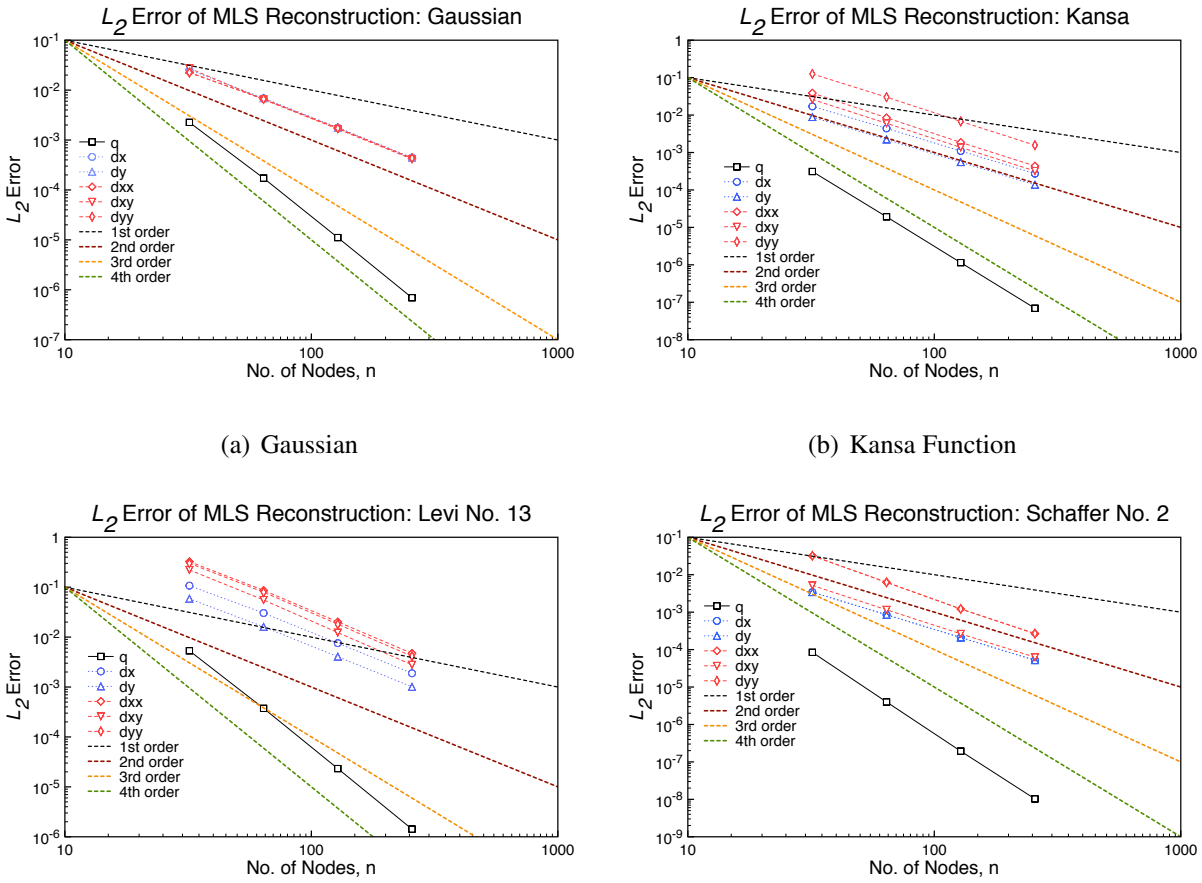
(c) Levi Function



(d) Schaffer Function

Figure 6.79: Error Norms on Stretched Grids: 3^{rd}-Order Semi-Diffuse Affine MLS with Anisotropic Weights.

Figure 6.79 shows that the accuracy for the function is approximately fourth-order accurate while the first- and second-derivatives are second-order accurate. The fully-diffuse derivatives computed using Affine MLS with anisotropic weights are shown in Figure 6.80.



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.80: Error Norms on Stretched Grids: 3rd-Order Fully-Diffuse Affine MLS with Anisotropic Weights.

Figure 6.80 shows that the accuracy for the function is approximately fourth-order accurate, and the first- and second-derivatives are approximately second-order accurate. Table 6.95 compares the accuracy for each level of diffusivity. The functions have the same accuracy for any of the derivative methods, and the full and semi-diffuse have the same first derivative accuracy as shown in Table 6.95. The fully-diffuse first derivatives are slightly better for the Gaussian, but slightly worse for the other functions. The fully-diffuse second derivatives are more accurate than the full derivatives. As with the second-order results, this is probably due to the less accurate derivatives of $\mathbf{C}$ are not computed with the fully-diffuse method. The semi-diffuse second derivatives are the least

Table 6.95: Accuracy on Stretched Grids: 3^{rd}-Order Diffuse Affine MLS with Anisotropic Weights. SD=Semi-Diffuse, FD=Fully-Diffuse.

Equation	Type	Function					
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
Gaussian	Full	3.98	1.94	1.94	1.99	1.99	1.99
	SD	3.98	1.94	1.94	1.95	1.97	1.95
	FD	3.98	1.97	1.97	1.97	1.98	1.97
Kansa	Full	4.06	1.98	1.98	1.95	2.29	1.95
	SD	4.06	1.98	1.98	1.76	1.72	2.05
	FD	4.06	2.01	2.01	2.14	2.11	2.13
Levi	Full	4.02	1.96	1.95	1.95	2.48	1.95
	SD	4.02	1.96	1.95	2.06	1.63	2.16
	FD	4.02	2.00	2.00	2.10	2.16	2.13
Schaffer	Full	4.30	1.99	1.99	1.94	2.06	1.94
	SD	4.30	1.99	1.99	1.75	1.72	1.75
	FD	4.30	2.02	2.02	2.28	2.11	2.27

accurate for each function, which shows keeping intermediate terms can make the reconstruction less accurate. It is also constructive to look at the timing for the full and fully-diffuse derivatives. Table 6.96 shows the timing for the reconstruction of the Gaussian. As seen from Table 6.96,

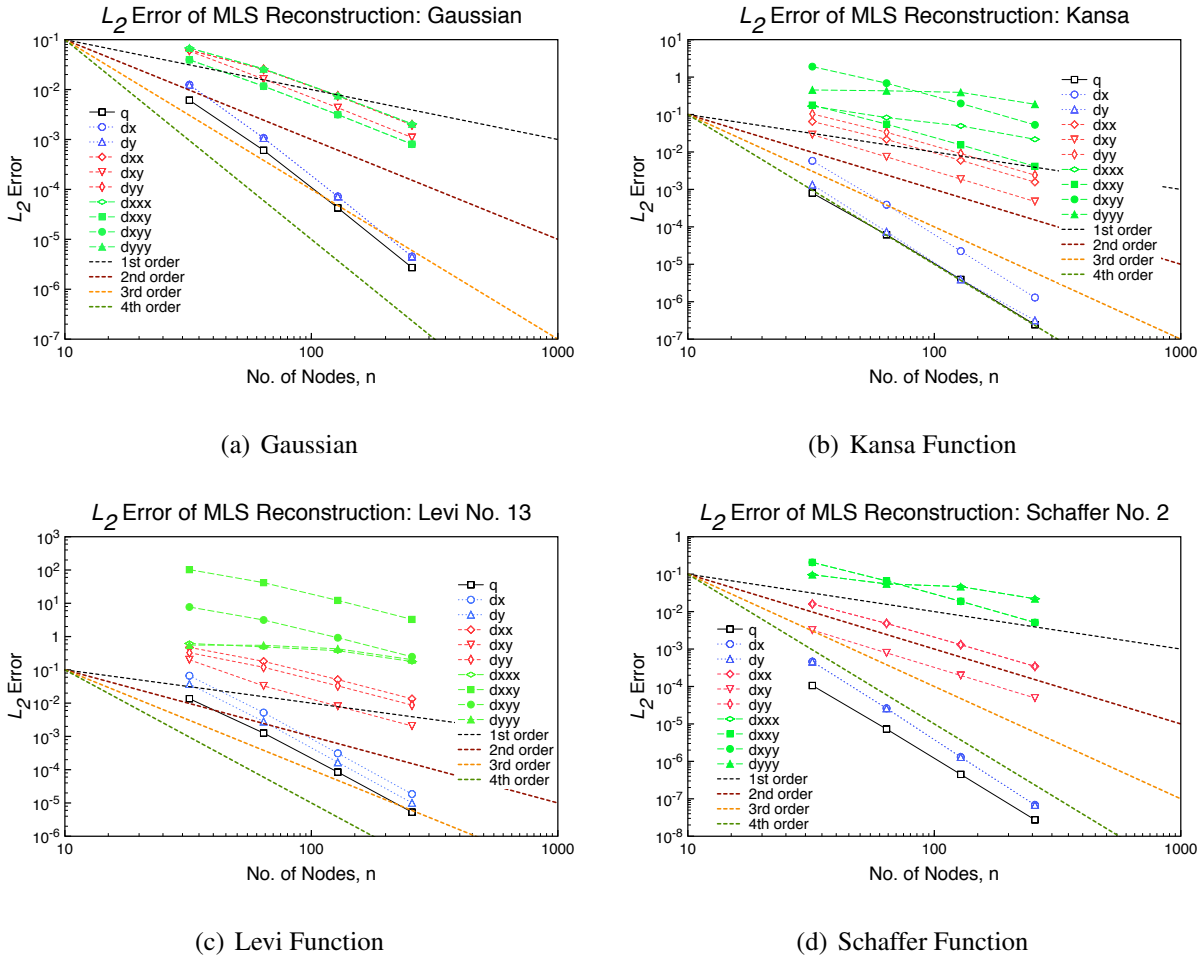
Table 6.96: Timing on Stretched Grids: 3^{rd}-Order Affine MLS with Anisotropic Weights, in Seconds.

Mesh Size	Type		
	Full	Semi-Diffuse	Full-Diffuse
32	0.231	0.179	0.163
64	0.930	0.729	0.658
128	3.591	2.851	2.596
256	14.12	11.17	10.05

the fully-diffuse is nearly twice as fast at the highest level of refinement. Considering the fully-diffuse as accurate as the full reconstruction, it makes sense to use the fully-diffuse to reduce computational time for larger systems. Additionally, the semi-diffuse is nearly as fast as the fully-

diffuse, so the additional terms are not much more expensive.

6.4.3.3.3 Fourth-Order The fourth-order MLS reconstruction of the derivatives can be in the full form (5.47), semi-diffuse (5.68) (with or without the second derivatives of $\mathbf{C}$), or in the fully-diffuse form (5.72). The semi-diffuse derivatives, including the second-derivatives of $\mathbf{C}$, reconstructed with anisotropic weights are shown in Figure 6.81.



(a) Gaussian

(b) Kansa Function

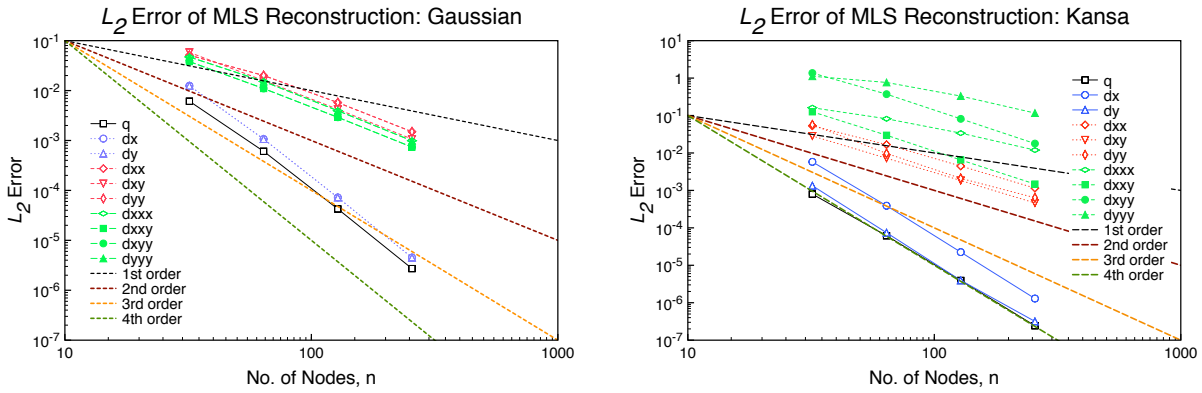
(c) Levi Function

(d) Schaffer Function

Figure 6.81: Error Norms on Stretched Grids: 4^{th}-Order Semi-Diffuse (1) Affine MLS with Anisotropic Weights.

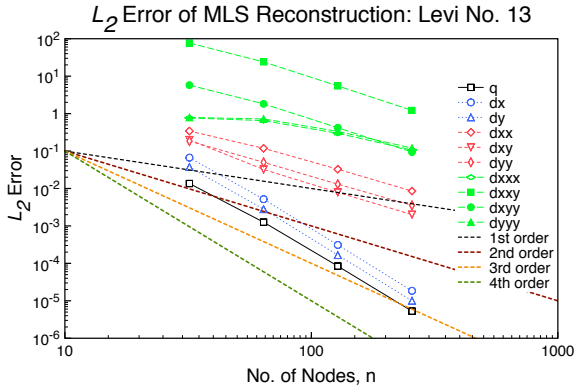
Figure 6.81 shows that the function and first-derivatives are approximately fourth-order accurate, and the second- and third-derivatives are approximately second-order accurate. The semi-

diffuse derivatives, without the second-derivatives of C, reconstructed with Affine MLS using anisotropic weights are shown in Figure 6.82.

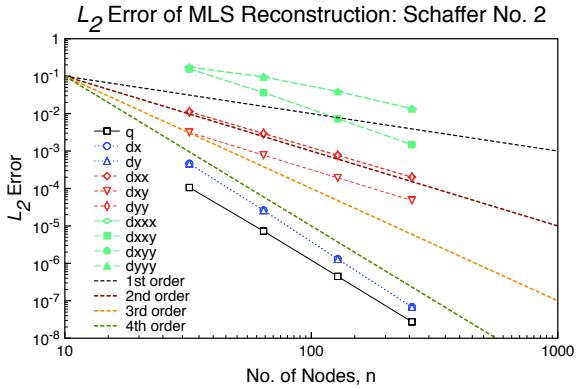


(a) Gaussian

(b) Kansa Function



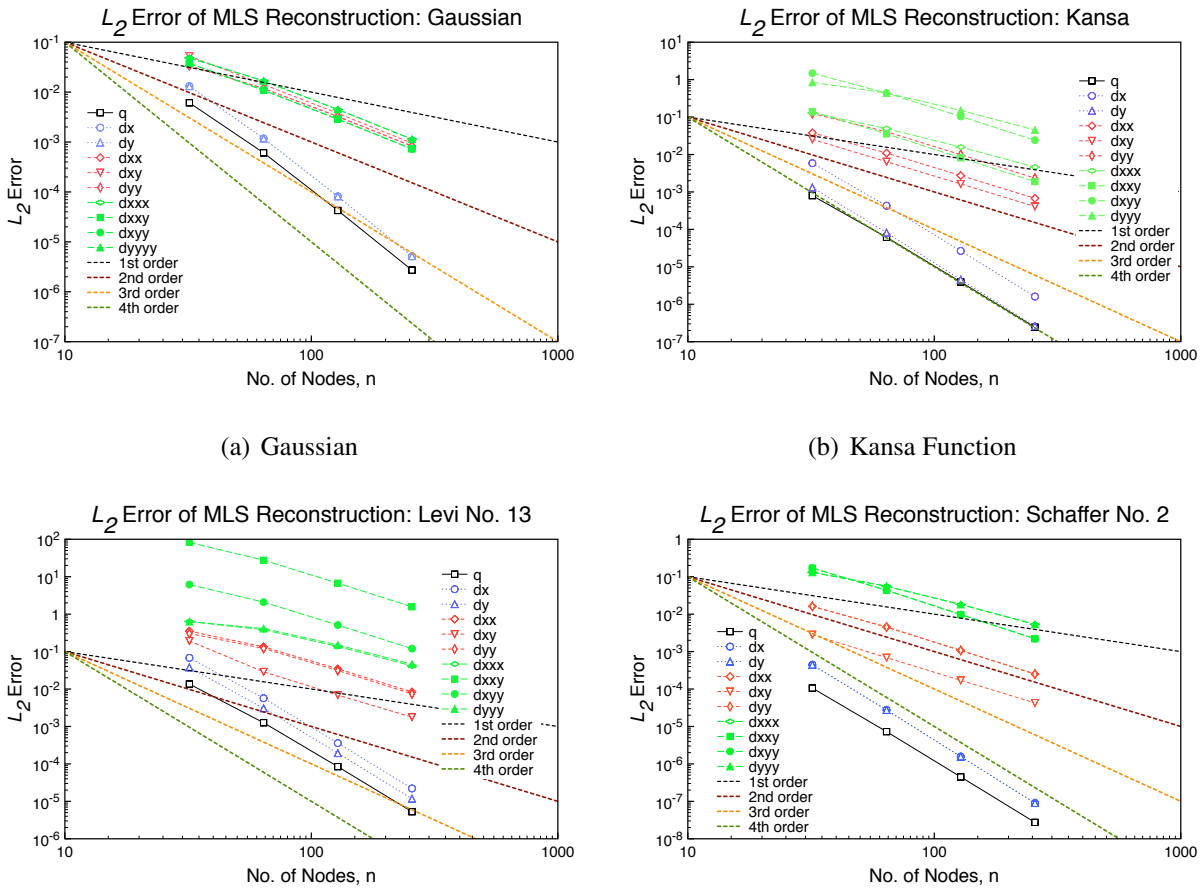
(c) Levi Function



(d) Schaffer Function

Figure 6.82: Error Norms on Stretched Grids: 4^{th}-Order Semi-Diffuse (2) Affine MLS with Anisotropic Weights.

Figure 6.82 shows that the function and first-derivatives are approximately fourth-order accurate, and the second- and third-derivatives are approximately second-order accurate. The fully-diffuse derivatives reconstructed with Affine MLS using anisotropic weights are shown in Figure 6.83.



(a) Gaussian

(b) Kansa Function

(c) Levi Function

(d) Schaffer Function

Figure 6.83: Error Norms on Stretched Grids: 4^{th}-Order Fully-Diffuse Affine MLS with Anisotropic Weights.

Figure 6.83 shows that the function and first-derivatives are approximately fourth-order accurate, and the second- and third-derivatives are approximately second-order accurate. Table 6.97 compares the accuracy for the different diffusivity levels. The results in Table 6.97 are similar to the third-order results in Table 6.95, where the methods produce effectively the same accuracy except for the second and third derivatives. As with the isotropic weighted results, the semi-diffuse (2) method does not produce better second derivatives, as removing the second-derivatives of $\mathbf{C}$ is probably detrimental to the accuracy. The fully-diffuse produces the highest level of accuracy, as was the case for the lower orders. The fully-diffuse first derivatives are the least accurate though,

Table 6.97: Accuracy on Stretched Grids: 4^{th}-Order Diffuse MLS with Anisotropic Weights. SD1=Semi-Diffuse 1, SD2=Semi-Diffuse 2, FD=Fully-Diffuse.

Equation	Type	Function									
		q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial xy^2}$	$\frac{\partial^3 q}{\partial y^3}$
Gaussian	Full	3.90	3.94	3.94	1.85	1.94	1.85	1.63	1.90	1.90	1.63
	SD1	3.90	3.94	3.94	1.85	1.94	1.85	1.83	1.93	1.93	1.83
	SD2	3.90	3.94	3.94	1.87	1.95	1.87	1.98	1.95	1.95	1.98
	FD	3.90	3.92	3.92	1.92	1.95	1.92	1.94	1.95	1.95	1.94
Kansa	Full	3.99	4.12	3.94	1.89	1.98	1.90	0.76	1.78	1.75	0.63
	SD1	3.99	4.12	3.94	1.89	1.98	1.90	0.96	1.87	1.85	0.60
	SD2	3.99	4.12	3.94	1.93	1.98	1.97	1.39	2.17	2.19	1.36
	FD	3.99	4.03	4.15	2.01	1.98	2.06	1.70	2.11	2.09	1.64
Levi	Full	3.95	4.07	4.06	1.87	2.00	1.89	0.65	1.73	1.73	0.65
	SD1	3.95	4.07	4.06	1.87	2.00	1.89	0.73	1.83	1.84	0.72
	SD2	3.95	4.07	4.06	1.90	2.00	1.91	1.30	2.15	2.14	1.29
	FD	3.95	4.00	4.01	2.00	2.01	2.01	1.60	2.07	2.06	1.58
Schaffer	Full	4.03	4.29	4.29	1.91	2.00	1.91	0.65	1.72	1.72	0.65
	SD1	4.03	4.29	4.29	1.91	2.00	1.91	0.66	1.85	1.85	0.66
	SD2	4.03	4.29	4.29	1.94	2.01	1.94	1.41	2.31	2.31	1.41
	FD	4.03	4.13	4.13	2.09	2.01	2.09	1.69	2.15	2.15	1.69

so excluding all of the derivatives of $\mathbf{C}$ may not be best. It is also constructive to look at the timing for the varying levels of diffusivity. Table 6.98 shows the timing for the reconstruction of the Gaussian. As seen from Table 6.98, the reduction in computation time is nearly two from full to semi-diffuse (with $\mathbf{C}$), and from full to full-diffuse the reduction is nearly four. Therefore, it is beneficial to use the diffuse derivatives when possible.

6.5 Comparison of Variable Scaling Parameter k

This section presents a detailed study on the effect of the scaling parameter k on the MLS reconstruction on each of the grids. In this section, k is varied from 0.1-1.1 as shown in Table 6.99. The basis used is the monomial basis. The Gaussian function is used in this study. Also, unlike the previous studies, the log-log of the error for each gradient is plotted separately to better illustrate the effect.

Table 6.98: Timing on Stretched Grids: 4^{th}-Order Affine MLS with Anisotropic Weights, in Seconds.

Mesh Size	Type			
	Full	Semi-Diffuse (1)	Semi-Diffuse (2)	Full-Diffuse
32	1.351	0.798	0.579	0.412
64	5.064	3.019	2.273	1.551
128	19.73	11.61	8.620	6.059
256	78.51	45.41	34.07	23.25

Table 6.99: Variation of Scaling Parameter k.

k	0.1	0.25	0.4	0.5	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95	1.0	1.05	1.1

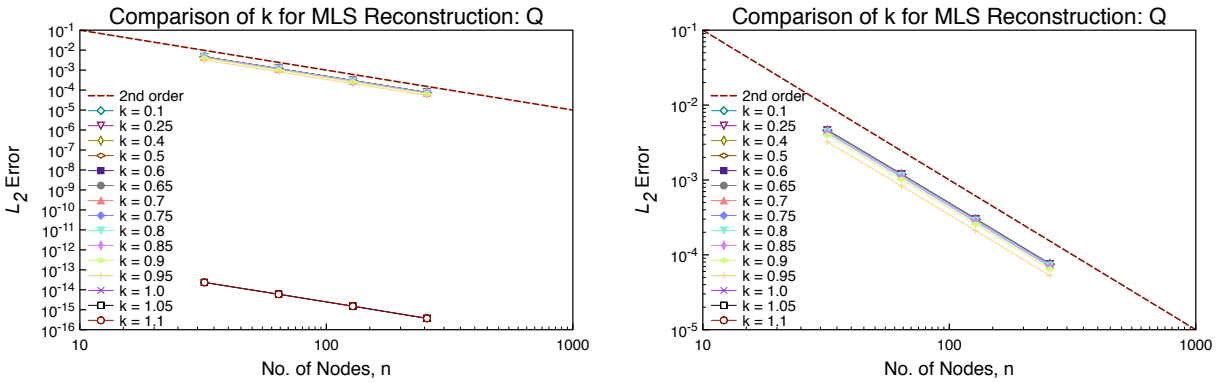
The inversion method of choice for this study is the Pivoting QR method, since it always produced the minimum condition number for $\mathbf{M}$ for each of the grids, as shown in Section 6.2.

6.5.1 Structured Grids

This section presents the detailed study on the effect of the scaling parameter k on the MLS reconstruction on structured grids. As observed earlier for structured grids, results obtained for the anisotropic and Affine MLS would produce the same results, so they are not tested here.

6.5.1.1 Second-Order

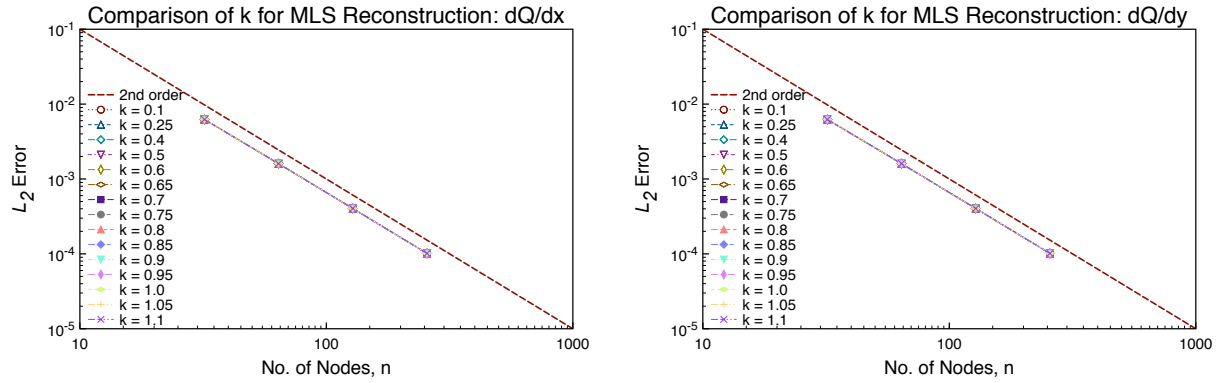
The error for the reconstruction of the field using a second-order MLS reconstruction is shown in Figure 6.84. At first, it would appear that the error for q for $k \in [0.1 - 0.95]$ is effectively the same. Figure 6.84(b) shows the error for these k only. Even from Figure 6.84(b), there is not much variation for the error of q for different k, though the least error occurs when using $k = 0.95$, which pushes the boundary of $\Omega_{\mathbf{x}_I}$ close to the outlying nodes in the stencil. Additionally, when looking at Figure 6.84 it would appear obvious that using $k \in [1.0 - 1.1]$ should always be used, since the error is on the order of $10^{-14} \rightarrow 10^{-16}$. These values of k perform best, since they effectively exclude the outlying nodes from the reconstruction and use only the central node $\mathbf{x}_I$.



(a) q (b) q, zoomed

Figure 6.84: Error Norms for State on Structured Grids: 2^{nd}-Order MLS with Isotropic Weights Varying k.

Therefore, when choosing the scaling parameter k, some care has to be made to ensure the central node is included in the stencil, otherwise, $k \in [1.0 - 1.1]$ will give incorrect answers. The first derivatives are shown in Figure 6.85. Examining both derivatives, it appears that the order of



(a) $\frac{\partial q}{\partial x}$ (b) $\frac{\partial q}{\partial y}$

Figure 6.85: Error Norms for First Derivatives on Structured Grids: 2^{nd}-Order MLS with Isotropic Weights Varying k.

accuracy and overall error do not vary with k. To further determine if one exists, Table 6.100

shows the computed order of accuracy for the varying k. From Table 6.100, the overall accuracy

Table 6.100: Accuracy on Structured Grids: 2^{nd}-Order MLS with Isotropic Weights Varying k.

	Function		
k	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
0.1	1.98	1.98	1.98
0.25	1.98	1.98	1.98
0.4	1.98	1.98	1.98
0.5	1.98	1.98	1.98
0.6	1.98	1.98	1.98
0.65	1.98	1.98	1.98
0.7	1.98	1.98	1.98
0.75	1.98	1.98	1.98
0.8	1.98	1.98	1.98
0.85	1.98	1.98	1.98
0.9	1.98	1.98	1.98
0.95	1.98	1.98	1.98
1.0	1.99	1.98	1.98
1.05	1.99	1.98	1.98
1.1	1.99	1.98	1.98

between any of the values k is effectively the same, with only slight improvement observed for $k \in [1.0 - 1.1]$ for the Gaussian and its derivatives. Therefore, as long as the $\mathbf{x}_I$ is included in the stencil $\Omega_{\mathbf{x}_I}$, then $k \in [1.0 - 1.1]$ should be used; otherwise, any k can be used in the reconstruction. All variations of k achieve second-order accuracy for the function and first-order accuracy for the first derivatives. Before determining which k might be best, the condition number of $\mathbf{M}$ should be checked. Table 6.101 shows the condition numbers for each k. The condition numbers from Table 6.101 show that while the order of accuracy may be 'best' for $k \in [1.0 - 1.1]$, by far those k have the poorest condition number. The poor conditioning of the $\mathbf{M}$ is due to the extreme points in the stencil $\Omega_{\mathbf{x}_I}$ having weights near zero by virtue of exclusion from the choice in k. Generally, the condition number decreases with increasing k for $k < 1.0$. Based upon this data, it would appear better to use $k < 1.0$ for reconstructing the function and its derivatives.

Table 6.101: Average Condition Number of M on Structured Grids: 2^{nd}-Order MLS with Isotropic Weights Varying k.

	Mesh Size			
k	32	64	128	256
0.1	15.85	15.82	15.82	15.82
0.25	6.36	6.35	6.35	6.35
0.4	3.99	3.98	3.98	3.98
0.5	3.19	3.19	3.19	3.19
0.6	2.67	2.66	2.66	2.66
0.65	2.47	2.46	2.46	2.46
0.7	2.30	2.29	2.29	2.29
0.75	2.16	2.15	2.15	2.15
0.8	2.04	2.04	2.04	2.04
0.85	1.95	1.95	1.95	1.95
0.9	1.91	1.91	1.91	1.91
0.95	2.00	2.00	2.00	2.00
1.0	$7.08E5$	$7.07E5$	$7.07E5$	$7.07E5$
1.05	$6.74E5$	$6.74E5$	$6.74E5$	$6.74E5$
1.1	$6.43E5$	$6.43E5$	$6.43E5$	$6.43E5$

6.5.1.2 Third-Order

The error for the reconstruction of the field using a third-order MLS reconstruction is shown in Figure 6.86.

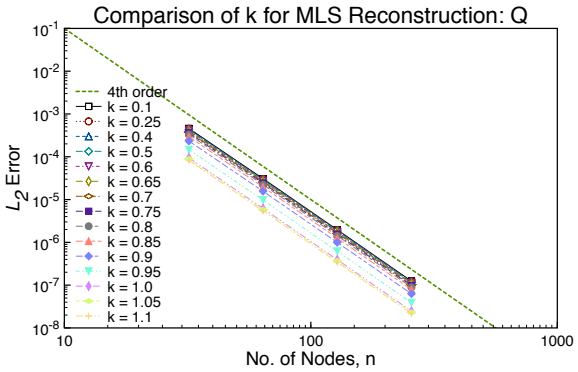
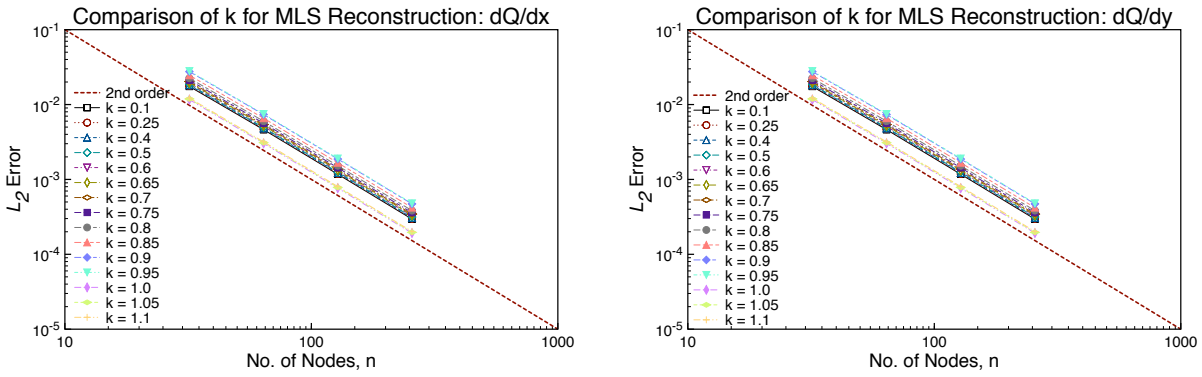


Figure 6.86: Error Norms for State on Structured Grids: 3^{rd}-Order MLS with Isotropic Weights Varying k.

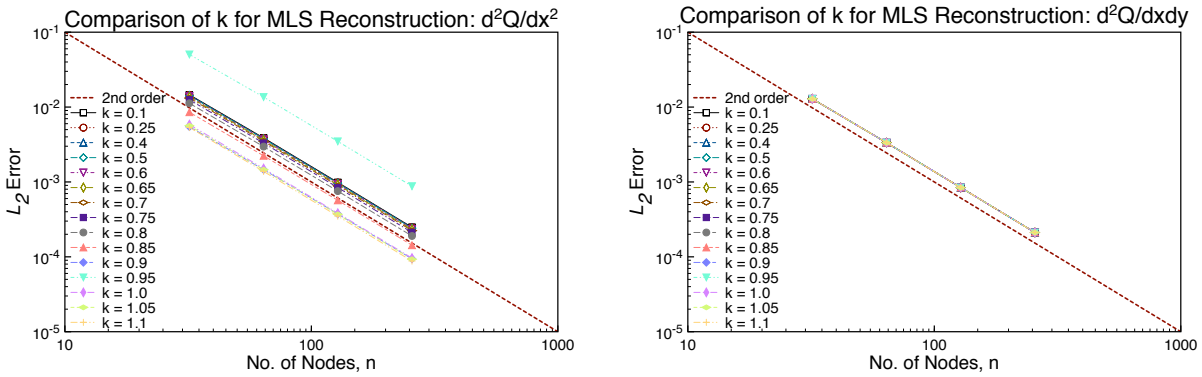
Unlike the second-order approximation, there is more spread in the error as a function of k for the reconstruction of the function. As k increases, the error drops in the reconstruction, with the lowest error for $k = 1.1$. For $k \in [1.0 - 1.1]$, the error is lowest since the 'furthest' stencil nodes are effectively excluded from the approximation. Otherwise, there is no major difference in the error for the reconstruction based on k for the function. The error for the reconstruction of the first derivatives is shown in Figure 6.87



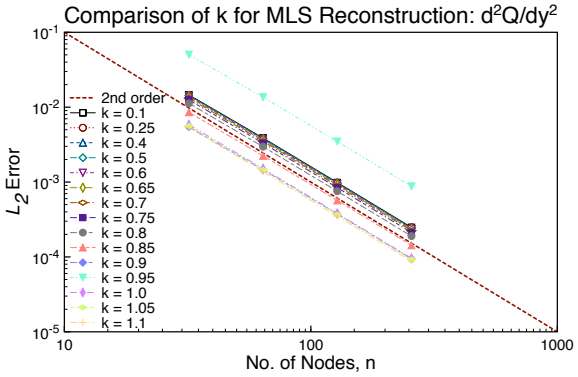
(a) $\frac{\partial q}{\partial x}$ (b) $\frac{\partial q}{\partial y}$

Figure 6.87: Error Norms for First Derivatives on Structured Grids: 3^{rd}-Order MLS with Isotropic Weights Varying k.

Again, the lowest error is observed for $k = [1.0 - 1.1]$. However, unlike the second-order reconstruction, the error increases with increasing k for $k = [0.1 - 0.95]$, though there is no significant difference in the error for $k < 0.85$. The error in the second-derivatives for varying k is shown in Figure 6.88



(a) $\frac{\partial^2 q}{\partial x^2}$ (b) $\frac{\partial^2 q}{\partial xy}$



(c) $\frac{\partial^2 q}{\partial y^2}$

Figure 6.88: Error Norms for Second Derivatives on Structured Grids: 3rd-Order MLS with Isotropic Weights Varying k.

As with the first-derivatives, the lowest error observed occurs when using $k = [1.0 - 1.1]$, with error decreasing as k increases from $k = [0.1 - 0.85]$ for both $\frac{\partial^2 q}{\partial x^2}$ and $\frac{\partial^2 q}{\partial y^2}$. There is also no observed variation in error with k for $\frac{\partial^2 q}{\partial xy}$. It is interesting to note that the largest error for $\frac{\partial^2 q}{\partial x^2}$ and $\frac{\partial^2 q}{\partial y^2}$ occurs with $k = 0.95$, and $k = 0.9$ has a larger error than would be expected considering the other trends. As to why $k = 0.95$ produces the largest error, one could speculate that the value of weight function for nodes near the stencil boundary is very small, which tends to produce poorer conditioned moment matrices, thus increasing the error in the reconstruction. The overall accuracy of the reconstruction for each k is presented in Table 6.102.

Table 6.102: Accuracy on Structured Grids: 3^{rd}-Order MLS with Isotropic Weights Varying k.

	Function					
k	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
0.1	3.95	1.96	1.96	1.96	1.98	1.96
0.25	3.95	1.96	1.96	1.96	1.98	1.96
0.4	3.95	1.96	1.96	1.96	1.98	1.96
0.5	3.95	1.96	1.96	1.96	1.98	1.96
0.6	3.95	1.96	1.96	1.98	1.98	1.96
0.65	3.95	1.96	1.96	1.96	1.98	1.96
0.7	3.95	1.96	1.96	1.96	1.98	1.96
0.75	3.95	1.96	1.96	1.96	1.98	1.96
0.8	3.95	1.96	1.96	1.96	1.98	1.96
0.85	3.95	1.96	1.96	1.97	1.98	1.97
0.9	3.95	1.96	1.96	1.95	1.98	1.95
0.95	3.96	1.96	1.96	1.95	1.98	1.95
1.0	3.97	1.97	1.97	1.97	1.98	1.97
1.05	3.97	1.97	1.97	1.97	1.98	1.97
1.1	3.97	1.97	1.97	1.98	1.98	1.98

From Table 6.102, the overall accuracy between any of the values k is effectively the same, with only slight improvement observed for $k \in [1.0 - 1.1]$, and slightly lower accuracy observed for $k \in [0.9 - 0.95]$. Overall, any $k \in [0.1 - 0.85]$ would be a reasonable choice for the MLS reconstruction, achieving nearly fourth-order accuracy for the function and second-order accuracy for the first- and second-derivatives. However, before making a final determination, the condition number of $\mathbf{M}$ should be checked. Table 6.109 shows the condition numbers for each k.

Table 6.103: Average Condition Number of M on Structured Grids: 3^{rd}-Order MLS with Isotropic Weights Varying k.

	Mesh Size			
k	32	64	128	256
0.1	725.58	725.00	724.86	724.82
0.25	119.16	119.05	119.02	119.01
0.4	48.63	48.58	48.56	48.56
0.5	32.24	32.20	32.19	32.19
0.6	23.21	23.18	23.17	23.17
0.65	20.11	20.09	20.08	20.08
0.7	17.61	17.59	17.58	17.58
0.75	15.54	15.52	15.52	15.52
0.8	13.82	13.79	13.79	13.78
0.85	12.35	12.32	12.32	12.32
0.9	11.12	11.10	11.09	11.09
0.95	10.93	10.90	10.89	10.89
1.0	10.82	10.80	10.79	10.79
1.05	10.09	10.07	10.06	10.06
1.1	9.50	9.47	9.47	9.47

The condition numbers from Table 6.103 show that for increasing k, the condition number decreases. In conjunction with data in Table 6.102, it is determined that choosing $k \in [0.6 - 0.8]$ produces the best overall results for the reconstruction.

6.5.1.3 Fourth-Order

The error for the reconstruction of the field using a fourth-order MLS reconstruction is shown in Figure 6.89.

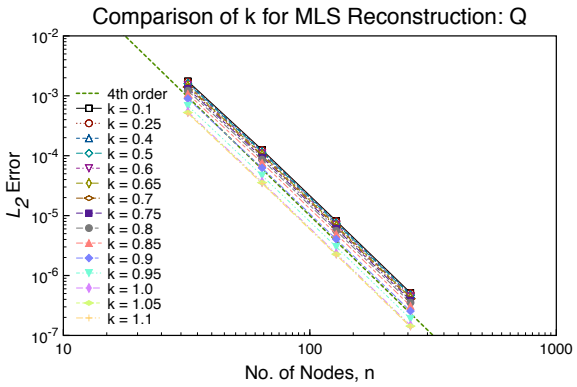
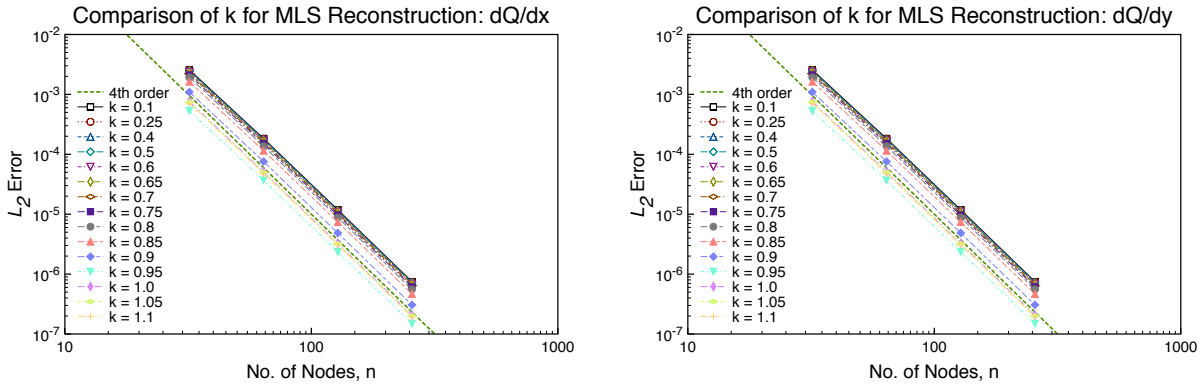


Figure 6.89: Error Norms for State on Structured Grids: 4^{th}-Order MLS with Isotropic Weights Varying k.

As with the third-order case, the fourth-order reconstruction has lower error as k is increased, with lowest error observed with $k = 1.1$. Otherwise, there is no major difference in the error for the reconstruction based on k for the function. The error for the reconstruction of the first derivatives is shown in Figure 6.90

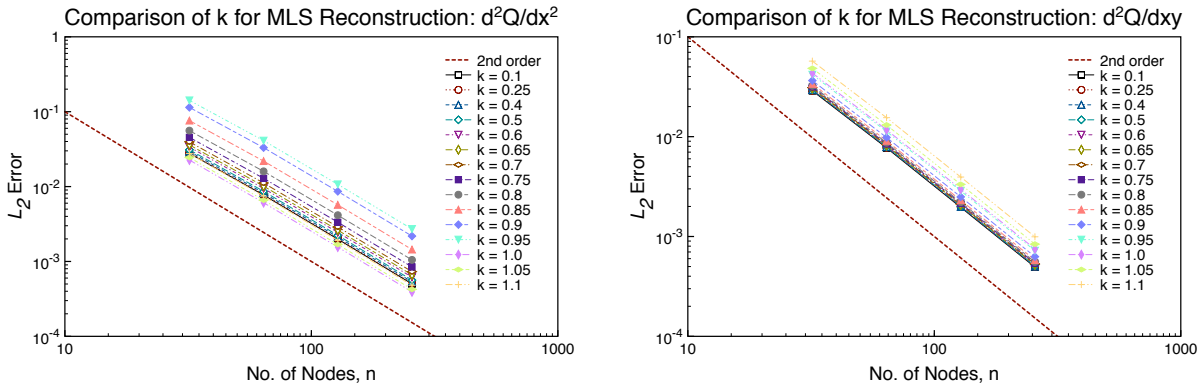


(a) $\frac{\partial q}{\partial x}$ (b) $\frac{\partial q}{\partial y}$

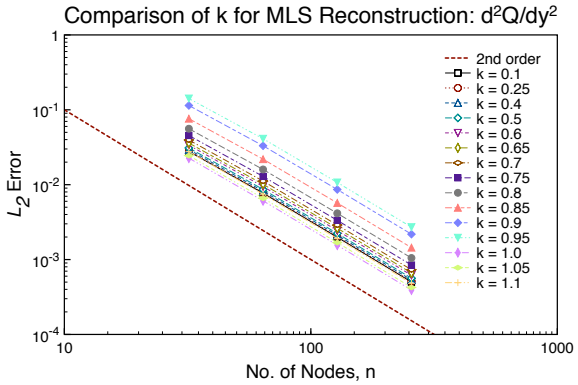
Figure 6.90: Error Norms for First Derivatives on Structured Grids: 4^{th}-Order MLS with Isotropic Weights Varying k.

Again, the lowest error is observed for $k = [1.0 - 1.1]$. The error also decreases as k increases,

though there is little variation in the overall error for $k \in [0.1 - 0.85]$. The error in the second-derivatives for varying k is shown in Figure 6.91



(a) $\frac{\partial^2 q}{\partial x^2}$ (b) $\frac{\partial^2 q}{\partial xy}$

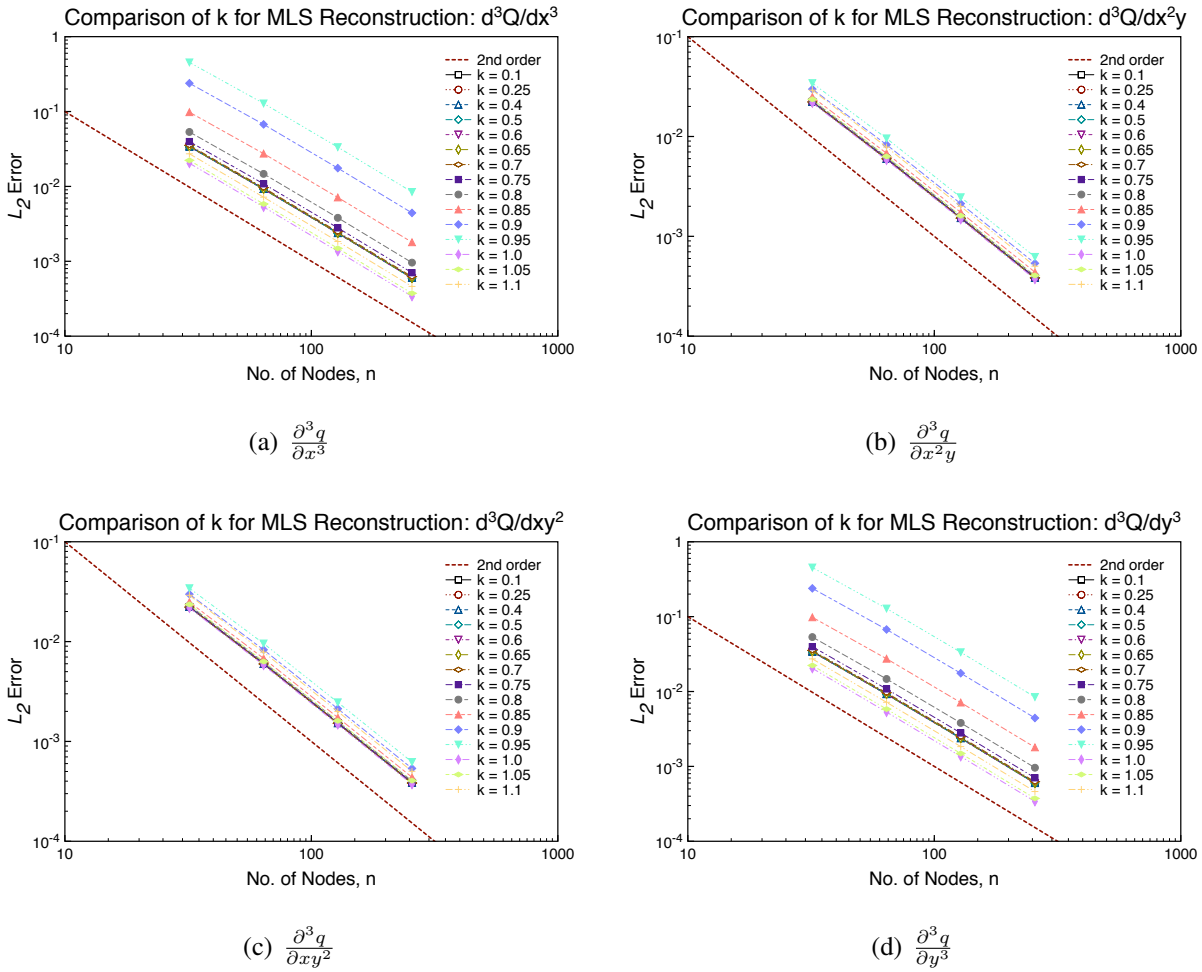


(c) $\frac{\partial^2 q}{\partial y^2}$

Figure 6.91: Error Norms for Second Derivatives on Structured Grids: 4^{th}-Order MLS with Isotropic Weights Varying k.

The results in Figure 6.91 are similar to those observed in the third-order reconstruction of the second-derivatives in Figure 6.88. As observed for the third-order reconstruction, the lowest error observed occurs when using $k = [1.0 - 1.1]$, with error decreasing as k increases from $k = [0.1 - 0.85]$ for both $\frac{\partial^2 q}{\partial x^2}$ and $\frac{\partial^2 q}{\partial y^2}$. There is also no observed variation in error with k for $\frac{\partial^2 q}{\partial xy}$. Also, as with the third-order reconstruction, the largest error for $\frac{\partial^2 q}{\partial x^2}$ and $\frac{\partial^2 q}{\partial y^2}$ occurs with $k = 0.95$,

and $k = 0.9$ has a larger error than would be expected considering the other trends. The error for the third-derivatives with varying k is shown in Figure 6.92.



(a) $\frac{\partial^3 q}{\partial x^3}$ (b) $\frac{\partial^3 q}{\partial x^2 y}$

(c) $\frac{\partial^3 q}{\partial x y^2}$ (d) $\frac{\partial^3 q}{\partial y^3}$

Figure 6.92: Error Norms for Third Derivatives on Structured Grids: 4^{th}-Order MLS with Isotropic Weights Varying k.

The lowest error for the third-derivatives is observed for $k \in [1.0 - 1.1]$ for all derivatives. For the intermediate derivatives, $\frac{\partial^3 q}{\partial x^2 y}$ and $\frac{\partial^3 q}{\partial x y^2}$, the largest error is observed for $k \in [0.9 - 0.95]$, while the other k have effectively the same error. For $\frac{\partial^3 q}{\partial x^3}$ and $\frac{\partial^3 q}{\partial y^3}$, the error increases with increasing k, with the largest error observed for $k = 0.95$. The other values of $k \in [0.1 - 0.8]$ produce roughly

the same amount of error. The overall accuracy of the reconstruction for each k is presented in Table 6.102.

Table 6.104: Accuracy on Structured Grids: 4^{th}-Order MLS with Isotropic Weights Varying k.

	Function									
k	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial xy^2}$	$\frac{\partial^3 q}{\partial y^3}$
0.1	3.91	3.91	3.91	1.93	1.96	1.93	1.94	1.95	1.95	1.94
0.25	3.91	3.91	3.91	1.93	1.96	1.93	1.94	1.95	1.95	1.94
0.4	3.91	3.91	3.91	1.93	1.96	1.93	1.94	1.95	1.95	1.94
0.5	3.91	3.92	3.92	1.93	1.96	1.93	1.94	1.95	1.95	1.94
0.6	3.91	3.92	3.92	1.92	1.96	1.93	1.94	1.95	1.95	1.94
0.65	3.91	3.92	3.92	1.92	1.96	1.92	1.94	1.95	1.95	1.94
0.7	3.91	3.92	3.92	1.92	1.96	1.92	1.94	1.95	1.95	1.94
0.75	3.92	3.92	3.92	1.92	1.95	1.92	1.94	1.95	1.95	1.94
0.8	3.92	3.92	3.92	1.91	1.95	1.91	1.93	1.95	1.95	1.93
0.85	3.92	3.92	3.92	1.91	1.95	1.90	1.92	1.94	1.94	1.92
0.9	3.93	3.94	3.94	1.90	1.95	1.90	1.92	1.93	1.93	1.92
0.95	3.94	3.93	3.93	1.90	1.95	1.90	1.92	1.93	1.93	1.92
1.0	3.95	3.95	3.95	1.95	1.95	1.95	1.97	1.96	1.96	1.97
1.05	3.95	3.95	3.95	1.95	1.95	1.95	1.97	1.95	1.95	1.97
1.1	3.95	3.95	3.95	1.95	1.95	1.95	1.96	1.95	1.95	1.96

From Table 6.104, the overall accuracy between any of the values k is effectively the same, with only slight improvement observed for $k \in [1.0 - 1.1]$, and slightly lower accuracy observed for $k \in [0.9 - 0.95]$. Overall, any $k \in [0.1 - 0.85]$ would be a reasonable choice for the MLS reconstruction. The accuracy associated with the derivatives is roughly second-order for second-

and third-derivatives and fourth-order for the first-derivatives and function.Table 6.111 shows the condition numbers for each k.

Table 6.105: Average Condition Number of $\mathbf{M}$ on Structured Grids: 4^{th}-Order MLS with Isotropic Weights Varying k.

	Mesh Size			
k	32	64	128	256
0.1	22330	22265	22249	22245
0.25	1463.8	1459.2	1458.0	1457.7
0.4	372.45	371.20	370.88	370.79
0.5	197.47	196.77	196.59	196.54
0.6	118.76	118.33	118.21	118.18
0.65	95.29	94.93	94.83	94.81
0.7	77.81	77.52	77.44	77.42
0.75	64.51	64.26	64.19	64.18
0.8	54.17	53.95	53.89	53.88
0.85	46.02	45.83	45.77	45.76
0.9	39.55	39.38	39.33	39.32
0.95	34.51	34.35	34.31	34.30
1.0	30.73	30.57	30.54	30.53
1.05	27.83	27.69	27.65	27.64
1.1	25.69	25.57	25.54	25.53

The condition numbers from Table 6.105 show that for increasing k, the condition number decreases. In conjunction with data in Table 6.104, it is determined that choosing $k \in [0.6 - 0.8]$ produces the best overall results for the reconstruction.

6.5.2 Unstructured Grids

This section presents a detailed study on the effect of the scaling parameter k on the MLS reconstruction on unstructured grids. For this study, the MLS reconstruction using anisotropic weights and Affine MLS with anisotropic weights are used.

6.5.2.1 Anisotropic Weights

This subsection presents the results of varying the scaling parameter k when using anisotropic weighting.

6.5.2.1.1 Second-Order The error for the reconstruction of the field using a second-order MLS reconstruction is shown in Figure 6.93. From Figure 6.93, there is not much variation for the error

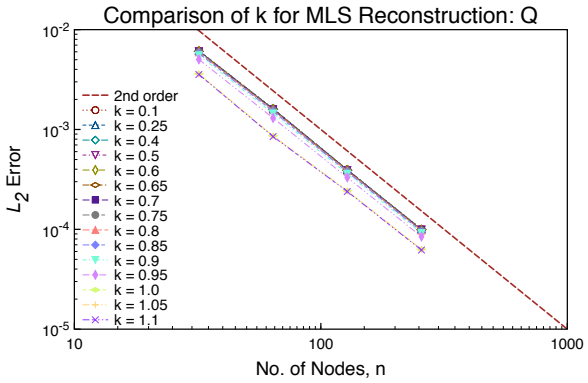
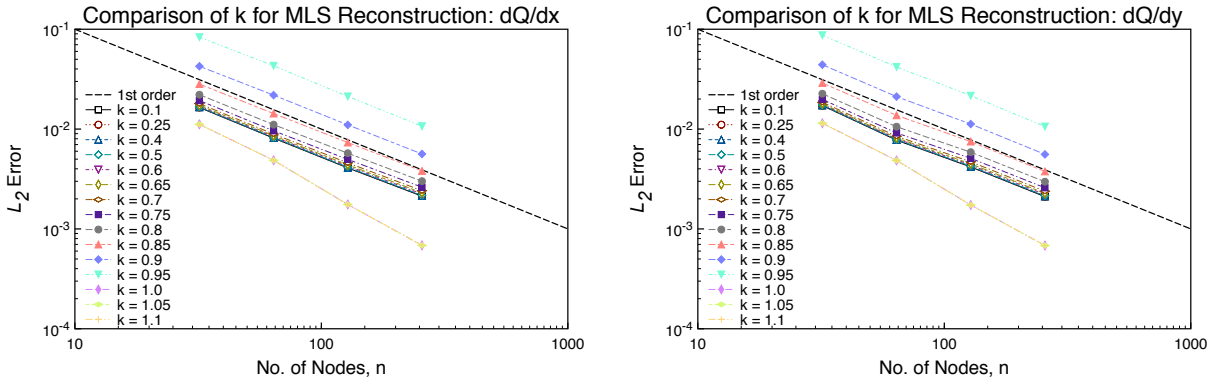


Figure 6.93: Error Norms for State on Unstructured Grids: 2^{nd}-Order MLS with Anisotropic Weights Varying k.

of q for $k \in [0.1 - 0.95]$, though the least error occurs when using $k = 0.95$, which pushes the boundary of $\Omega_{\mathbf{x}_I}$ close to the outlying nodes in the stencil. The lowest overall error of q is for $k \in [1.0 - 1.1]$, though not as dramatically for the structured results shown in Figure 6.84. These values of k perform best, since they effectively exclude the outlying nodes from the reconstruction and use only the central node $\mathbf{x}_I$. Therefore, when choosing the scaling parameter k, some care has to be made to ensure the central node is included in the stencil, otherwise, $k \in [1.0 - 1.1]$ will

give incorrect answers. The first derivatives are shown in Figure 6.94. Examining both derivatives,



(a) $\frac{\partial q}{\partial x}$ (b) $\frac{\partial q}{\partial y}$

Figure 6.94: Error Norms for First Derivatives on Unstructured Grids: 2^{nd}-Order MLS with Anisotropic Weights Varying k.

there is a broad variation with respect to k. The lowest error appears to be for $k \in [1.0 - 1.1]$, with the worse error occurring with $k = 0.95$. Table 6.106 shows the computed order of accuracy for the varying k. From Table 6.106, the accuracy of q decreases with increasing k, with the maximum accuracy being observed when $k < 0.7$. For the derivatives, the error increases with increasing k up to $k = 0.75$, then decreases again with a maximum accuracy for $k = [1.0 - 1.1]$. The approximate accuracy for the function is second-order, and the accuracy for the first derivatives is around first-order. As with the structured case, the error is lower for the function due to clipping the extreme nodes in the stencil $\Omega_{\mathbf{x}_I}$ for when $k \in [1.0 - 1.1]$. However, at these values of k, the exclusion of the extreme nodes causes the accuracy to suffer for the first-derivatives. Before determining which k might be best, the condition number of $\mathbf{M}$ should be checked. Table 6.107 shows the condition numbers for each k. The condition numbers from Table 6.107 show that while the order of accuracy may be 'best' for $k \in [1.0 - 1.1]$, by far those k have the poorest condition number. Generally, the condition number decreases with increasing k for $k < 1.0$. Based upon this data, it would appear better to use $k < 1.0$ for reconstructing the function and its derivatives.

Table 6.106: Accuracy on Unstructured Grids: 2^{nd}-Order MLS with Anisotropic Weights Varying k.

k	Function		
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
0.1	2.01	0.97	0.94
0.25	2.01	0.97	0.94
0.4	2.01	0.96	0.94
0.5	2.01	0.96	0.94
0.6	2.01	0.95	0.93
0.65	2.01	0.95	0.92
0.7	2.00	0.94	0.92
0.75	2.00	0.94	0.91
0.8	2.00	0.94	0.92
0.85	2.00	0.96	0.93
0.9	1.99	0.98	0.96
0.95	1.97	1.01	0.99
1.0	1.89	1.42	1.42
1.05	1.89	1.42	1.42
1.1	1.89	1.42	1.42

6.5.2.1.2 Third-Order The error for the reconstruction of the field using a third-order MLS reconstruction is shown in Figure 6.95.

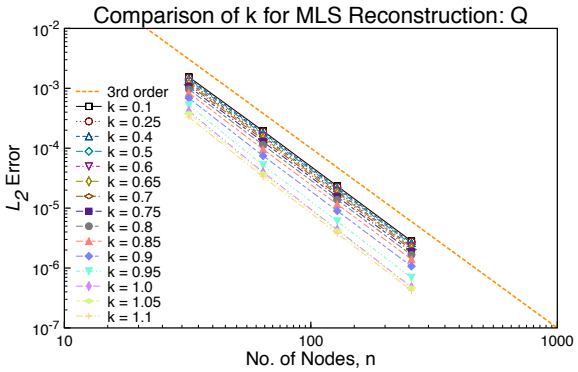
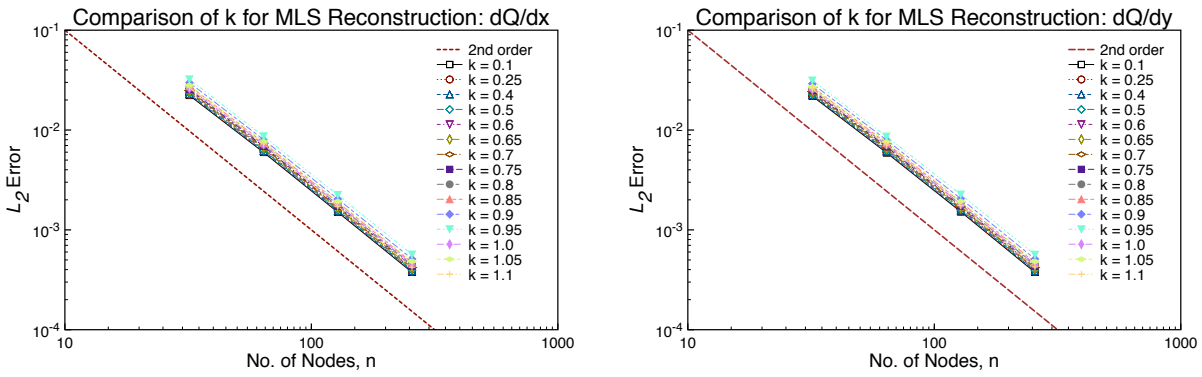


Figure 6.95: Error Norms for State on Unstructured Grids: 3^{rd}-Order MLS with Anisotropic Weights Varying k.

Table 6.107: Average Condition Number of M on Unstructured Grids: 2^{nd}-Order MLS with Anisotropic Weights Varying k.

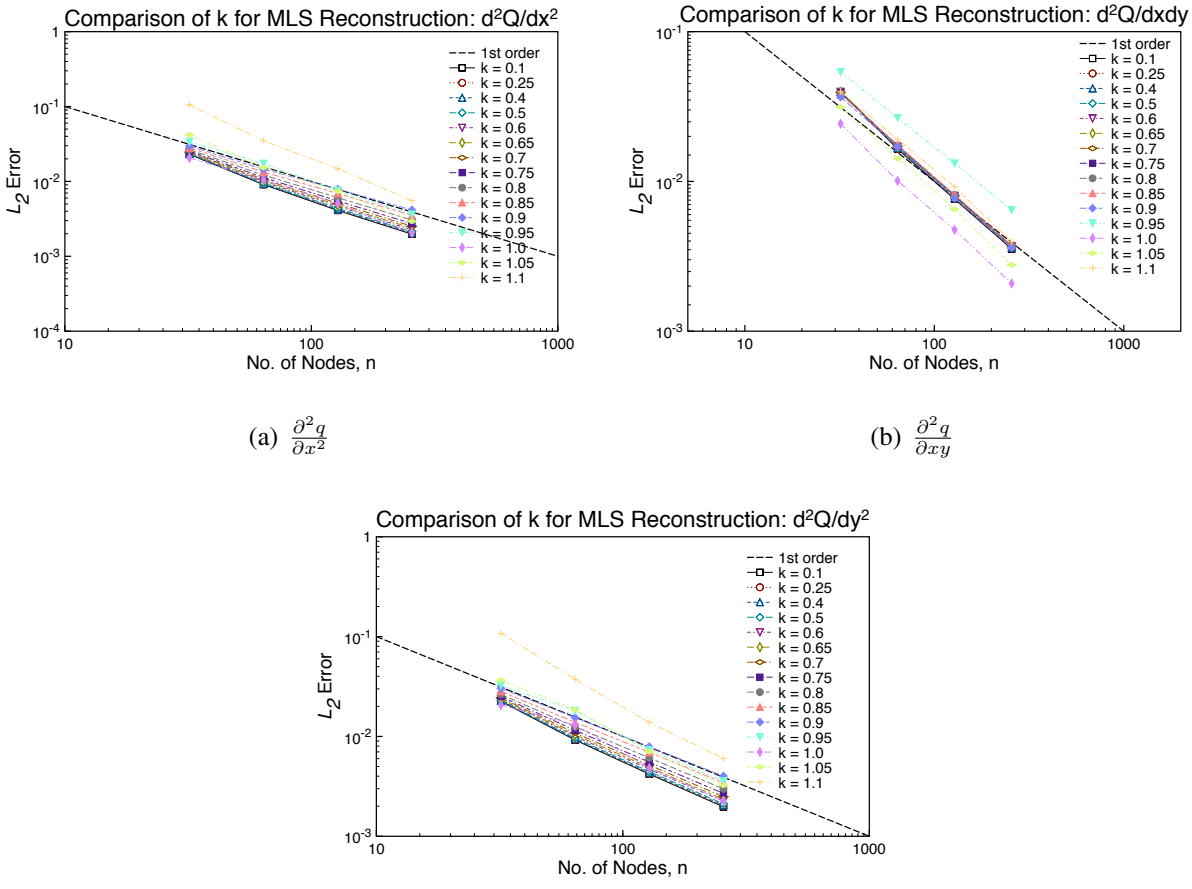
	Mesh Size			
k	32	64	128	256
0.1	20.10	21.03	21.71	21.74
0.25	8.92	8.59	8.88	8.90
0.4	5.52	5.48	5.68	5.69
0.5	4.23	4.45	4.60	4.62
0.6	3.57	3.75	3.89	3.90
0.65	3.32	3.49	3.61	3.63
0.7	3.10	3.26	3.38	3.40
0.75	2.91	3.06	3.18	3.19
0.8	2.75	2.89	2.99	3.01
0.85	2.62	2.75	2.84	2.85
0.9	2.52	2.62	2.71	2.71
0.95	2.51	2.58	2.64	2.62
1.0	$5.21E5$	$4.27E5$	$3.71E5$	$3.09E5$
1.05	$4.96E5$	$4.07E5$	$3.53E5$	$2.95E5$
1.1	$4.74E5$	$3.88E5$	$3.37E5$	$2.81E5$

As with the second-order approximation, there is only slight spread in the error as a function of k for the reconstruction of the function. As k increases, the error drops in the reconstruction, with the lowest error for $k = 1.1$. For $k \in [1.0 - 1.1]$, the error is lowest since the 'furthest' stencil nodes are effectively excluded from the approximation. Otherwise, there is no major difference in the error for the reconstruction based on k for the function. The error for the reconstruction of the first derivatives is shown in Figure 6.96 Again, the lowest error is observed for $k = [1.0 - 1.1]$. However, unlike the second-order reconstruction, the error increases with increasing k for $k = [0.1 - 0.95]$, though there is no significant difference in the error for $k < 0.85$. The error in the second-derivatives for varying k is shown in Figure 6.97



(a) $\frac{\partial q}{\partial x}$ (b) $\frac{\partial q}{\partial y}$

Figure 6.96: Error Norms for First Derivatives on Unstructured Grids: 3rd-Order MLS with Anisotropic Weights Varying k.



(a) $\frac{\partial^2 q}{\partial x^2}$ (b) $\frac{\partial^2 q}{\partial xy}$

(c) $\frac{\partial^2 q}{\partial y^2}$

Figure 6.97: Error Norms for Second Derivatives on Unstructured Grids: 3rd-Order MLS with Anisotropic Weights Varying k.

Unlike with the first-derivatives, the highest error observed occurs when using $k = [1.0 - 1.1]$, with error decreasing as k decreases for both $\frac{\partial^2 q}{\partial x^2}$ and $\frac{\partial^2 q}{\partial y^2}$. There is also no observed variation in error with k for $\frac{\partial^2 q}{\partial xy}$ except for $k = 0.95$ (highest) and $k = [1.0 - 1.05]$ (lowest). The overall accuracy of the reconstruction for each k is presented in Table 6.108.

Table 6.108: Accuracy on Unstructured Grids: 3^{rd}-Order MLS with Anisotropic Weights Varying k.

	Function					
k	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
0.1	3.05	1.99	1.99	1.10	1.11	1.11
0.25	3.05	1.99	1.99	1.10	1.11	1.11
0.4	3.05	1.99	1.99	1.09	1.11	1.10
0.5	3.05	2.00	1.99	1.08	1.10	1.09
0.6	3.05	2.00	1.99	1.06	1.10	1.08
0.65	3.05	2.00	1.99	1.05	1.10	1.07
0.7	3.05	2.00	1.99	1.04	1.10	1.06
0.75	3.05	2.00	1.99	1.02	1.11	1.05
0.8	3.05	2.00	1.99	1.00	1.11	1.03
0.85	3.05	1.99	1.99	0.96	1.12	1.01
0.9	3.06	1.99	1.98	0.90	1.12	0.97
0.95	3.12	1.97	1.97	1.13	1.02	1.17
1.0	3.19	1.99	1.99	1.17	1.14	1.26
1.05	3.17	1.99	1.99	1.19	1.18	1.26
1.1	3.16	2.05	2.03	1.33	1.14	1.32

From Table 6.108, the overall accuracy between any of the values k is effectively the same.

The accuracy for the function improves with increasing k up to $k = 0.95$, which has the highest accuracy and begins to decrease as k increase beyond $k = 1.0$. The average accuracy for the function for all k is third-order. For the first derivatives, all methods achieve second-order, so all k would produce similar accuracy as the mesh is refined. The second derivatives show decreasing accuracy as k is increased, with the minimum accuracy occurring with $k = 1.1$. Before determining which k might be best, the condition number of $\mathbf{M}$ should be checked. Table 6.109 shows the condition numbers for each k.

Table 6.109: Average Condition Number of $\mathbf{M}$ on Unstructured Grids: 3^{rd}-Order MLS with Anisotropic Weights Varying k.

	Mesh Size			
k	32	64	128	256
0.1	686.83	666.91	650.38	626.11
0.25	114.30	111.11	108.42	104.43
0.4	47.29	46.08	45.01	43.43
0.5	31.70	30.97	30.29	29.29
0.6	23.23	22.79	22.32	21.65
0.65	20.38	20.06	19.66	19.10
0.7	18.14	17.92	17.60	17.14
0.75	16.38	16.27	16.01	15.63
0.8	15.00	15.01	14.82	14.51
0.85	13.99	14.13	14.01	13.78
0.9	13.35	13.66	13.64	13.48
0.95	13.38	13.74	13.78	13.64
1.0	13.45	13.73	13.73	13.55
1.05	12.92	13.17	13.16	12.98
1.1	12.50	12.78	12.83	12.70

The condition numbers from Table 6.109 show that for increasing k, the condition number decreases. In conjunction with data in Table 6.108, it is determined that choosing $k \in [0.6 - 0.8]$ produces the best overall results for the reconstruction.

6.5.2.1.3 Fourth-Order The error for the reconstruction of the field using a fourth-order MLS reconstruction is shown in Figure 6.98.

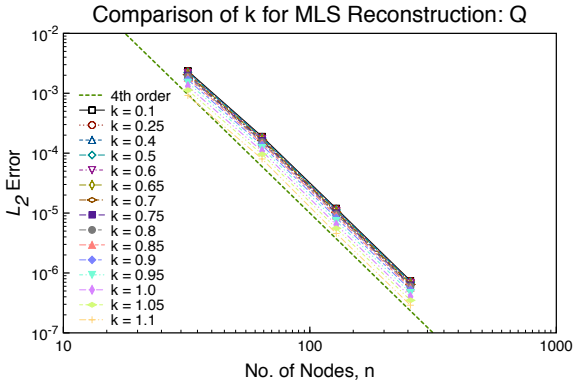
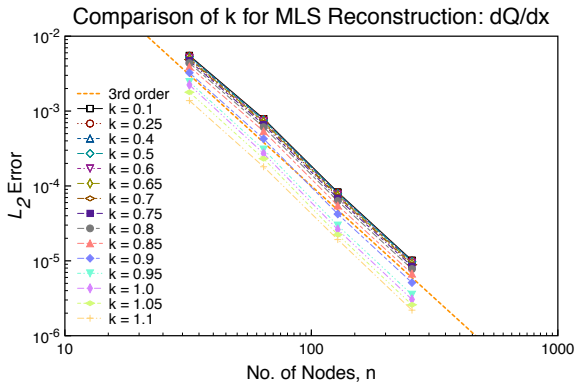
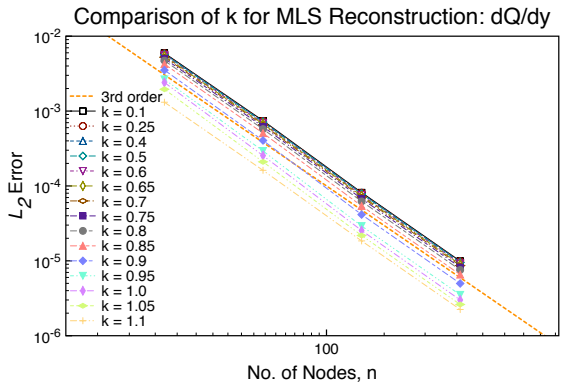


Figure 6.98: Error Norms for State on Unstructured Grids: 4^{th}-Order MLS with Anisotropic Weights Varying k.

As with the third-order case, the fourth-order reconstruction has lower error as k is increased, with lowest error observed with $k = 1.1$. Otherwise, there is no major difference in the error for the reconstruction based on k for the function. The error for the reconstruction of the first derivatives is shown in Figure 6.99 The lowest error is observed for $k = [1.0 - 1.1]$. The error also decreases as k increases, though there is little variation in the overall error for $k \in [0.1 - 0.85]$. The error in the second-derivatives for varying k is shown in Figure 6.100

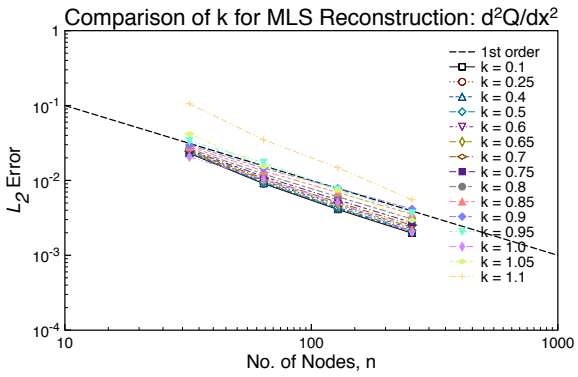


(a) $\frac{\partial q}{\partial x}$

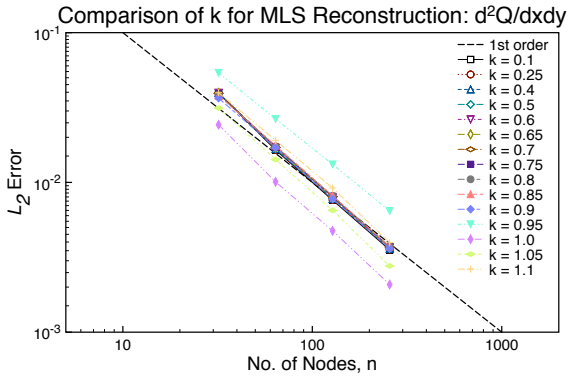


(b) $\frac{\partial q}{\partial y}$

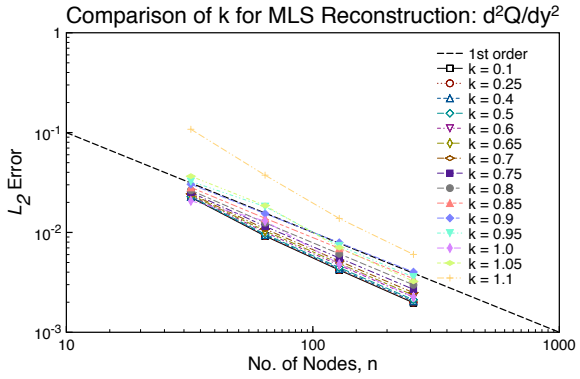
Figure 6.99: Error Norms for First Derivatives on Unstructured Grids: 4^{th}-Order MLS with Anisotropic Weights Varying k.



(a) $\frac{\partial^2 q}{\partial x^2}$



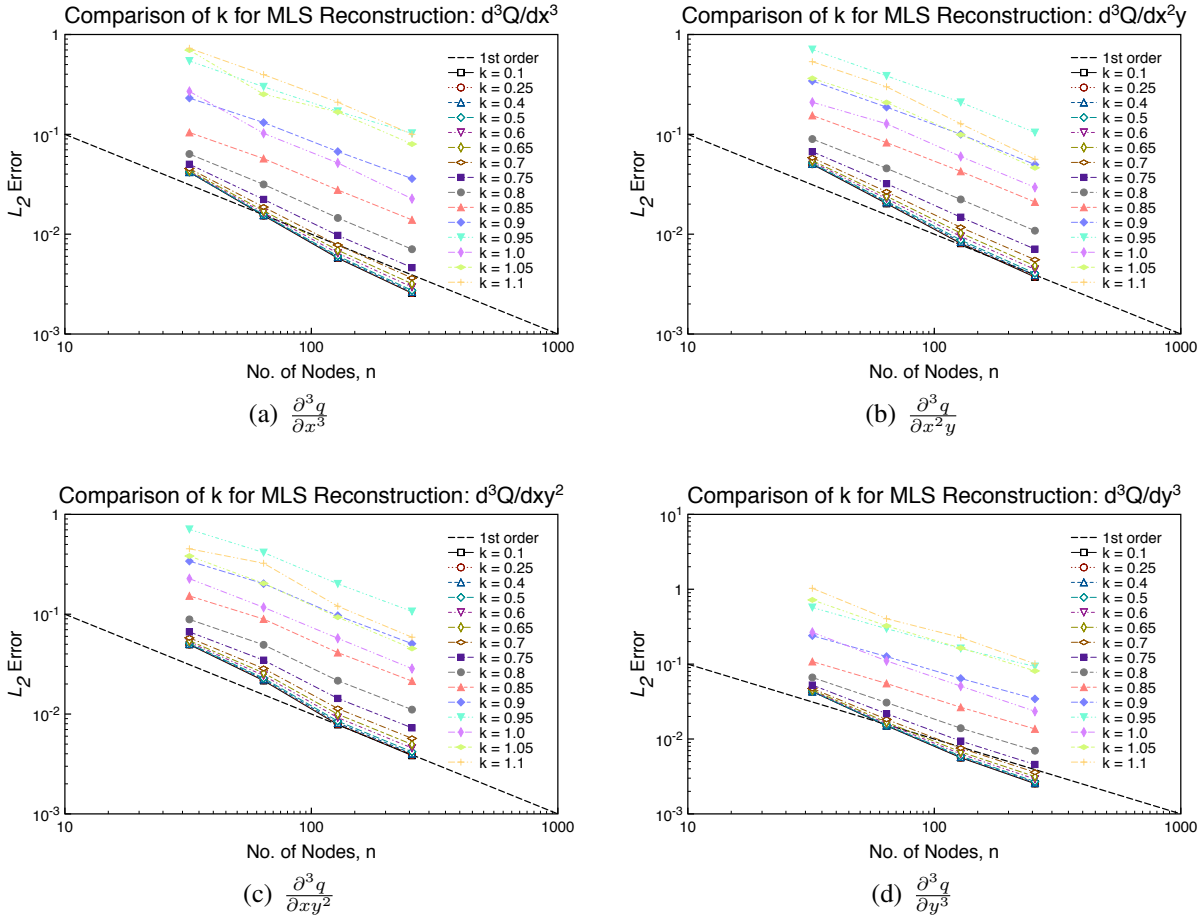
(b) $\frac{\partial^2 q}{\partial xy}$



(c) $\frac{\partial^2 q}{\partial y^2}$

Figure 6.100: Error Norms for Second Derivatives on Unstructured Grids: 4^{th}-Order MLS with Anisotropic Weights Varying k.

The results in Figure 6.100 show that the highest error observed occurs when using $k = [1.0 - 1.1]$, with error decreasing as k decreases for both $\frac{\partial^2 q}{\partial x^2}$ and $\frac{\partial^2 q}{\partial y^2}$. There is also no observed variation in error with k for $\frac{\partial^2 q}{\partial xy}$ except for $k = 0.95$ (highest) and $k = [1.0 - 1.05]$ (lowest). The error for the third-derivatives with varying k is shown in Figure 6.101. Figure 6.101 shows a large



(a) $\frac{\partial^3 q}{\partial x^3}$ (b) $\frac{\partial^3 q}{\partial x^2 y}$

(c) $\frac{\partial^3 q}{\partial x y^2}$ (d) $\frac{\partial^3 q}{\partial y^3}$

Figure 6.101: Error Norms for Third Derivatives on Unstructured Grids: 4^{th}-Order MLS with Anisotropic Weights Varying k.

spread in the error, with the general trend being increasing error as k increases. The highest error observed occurs when using $k = 1.1$ and $k = [0.95, 1.05]$, with another cluster of larger error for $k = 0.85 - 0.9, 1.0]$. For the intermediate derivatives, $\frac{\partial^3 q}{\partial x^2 y}$ and $\frac{\partial^3 q}{\partial x y^2}$, the largest error is observed

for $k \in [0.95]$, with the same clusters as the other derivatives. Generally, the error increases with increasing k for the third-derivatives, with the lowest error for $k = [0.1 - 0.65]$. The overall accuracy of the reconstruction for each k is presented in Table 6.108.

Table 6.110: Accuracy on Unstructured Grids: 4^{th}-Order MLS with Anisotropic Weights Varying k.

	Function									
k	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial xy^2}$	$\frac{\partial^3 q}{\partial y^3}$
0.1	4.00	3.14	3.12	1.98	1.98	1.97	1.29	1.22	1.25	1.28
0.25	4.00	3.14	3.12	1.98	1.98	1.97	1.29	1.22	1.24	1.28
0.4	4.00	3.14	3.11	1.98	1.97	1.97	1.28	1.21	1.24	1.27
0.5	4.00	3.14	3.11	1.98	1.97	1.97	1.26	1.20	1.23	1.25
0.6	4.00	3.14	3.12	1.97	1.97	1.97	1.24	1.18	1.21	1.23
0.65	4.00	3.14	3.12	1.97	1.97	1.97	1.21	1.16	1.19	1.21
0.7	4.00	3.14	3.12	1.97	1.97	1.97	1.18	1.13	1.16	1.17
0.75	4.00	3.14	3.12	1.96	1.97	1.97	1.13	1.09	1.13	1.13
0.8	4.00	3.15	3.13	1.95	1.97	1.96	1.08	1.04	1.08	1.07
0.85	4.00	3.16	3.14	1.94	1.96	1.96	1.02	0.99	1.03	1.01
0.9	4.01	3.19	3.17	1.90	1.96	1.94	0.93	0.96	1.00	0.94
0.95	4.02	3.22	3.19	1.84	1.96	1.91	0.77	0.94	0.98	0.84
1.0	4.04	3.24	3.20	1.93	1.98	1.95	1.09	1.06	1.02	1.12
1.05	4.05	3.24	3.16	1.95	1.99	1.95	0.83	1.09	1.08	1.00
1.1	4.06	3.18	3.09	2.01	1.97	2.01	0.99	1.21	1.23	1.00

From Table 6.110, the accuracy for the function increases as k increases, with maximum at $k = 1.1$, but on average the order of accuracy is roughly fourth-order. For the first derivatives, the

accuracy increases with increasing k up to $k = 0.9$, after which the accuracy decreases. Overall, the accuracy for the first-derivatives is approximately third-order. For the second-derivatives, the accuracy is approximately second-order, with the maximum obtained when $k = 0.8$, with slightly lower accuracy for other values of k. For the third-derivatives, the accuracy decreases as k is increased, with the minimum at $k = 1.1$, with a secondary maximum accuracy occurring for $k = 0.8$. The average error for the third-derivatives is first-order. Table 6.111 shows the condition numbers for each k.

Table 6.111: Average Condition Number of $\mathbf{M}$ on Unstructured Grids: 4^{th}-Order MLS with Anisotropic Weights Varying k.

	Mesh Size			
k	32	64	128	256
0.1	21663	20835	20583	19987
0.25	1437.6	1383.7	1366.7	1327.0
0.4	370.78	357.32	352.97	342.68
0.5	198.70	191.68	189.43	183.94
0.6	120.93	116.92	115.63	112.31
0.65	97.84	94.73	93.70	91.03
0.7	80.70	78.32	77.55	75.36
0.75	67.78	66.00	65.45	63.64
0.8	57.96	56.70	56.36	54.86
0.85	50.58	49.81	49.69	48.45
0.9	45.32	45.09	45.21	44.19
0.95	42.13	42.59	43.06	42.21
1.0	40.23	41.38	42.32	41.75
1.05	37.72	39.16	40.31	39.83
1.1	35.84	37.26	38.43	37.95

The condition numbers from Table 6.111 show that for increasing k, the condition number decreases. In conjunction with data in Table 6.110, it is determined that choosing $k \in [0.6 - 0.8]$ produces the best overall results for the reconstruction.

6.5.2.2 Affine MLS

This subsection presents the results of varying the scaling parameter k with Affine MLS reconstruction on unstructured grids using anisotropic weights.

6.5.2.2.1 Second-Order The error for the reconstruction of the field using a second-order MLS reconstruction is shown in Figure 6.102. From Figure 6.102, there is not much variation for the

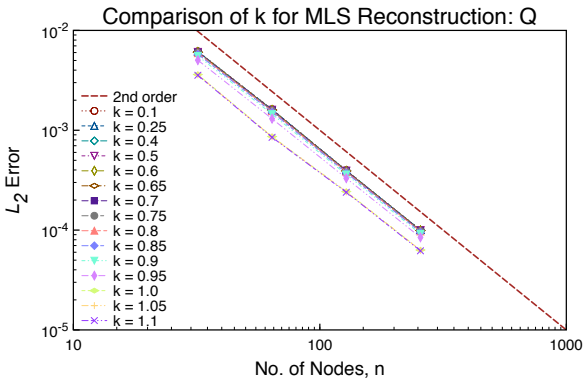
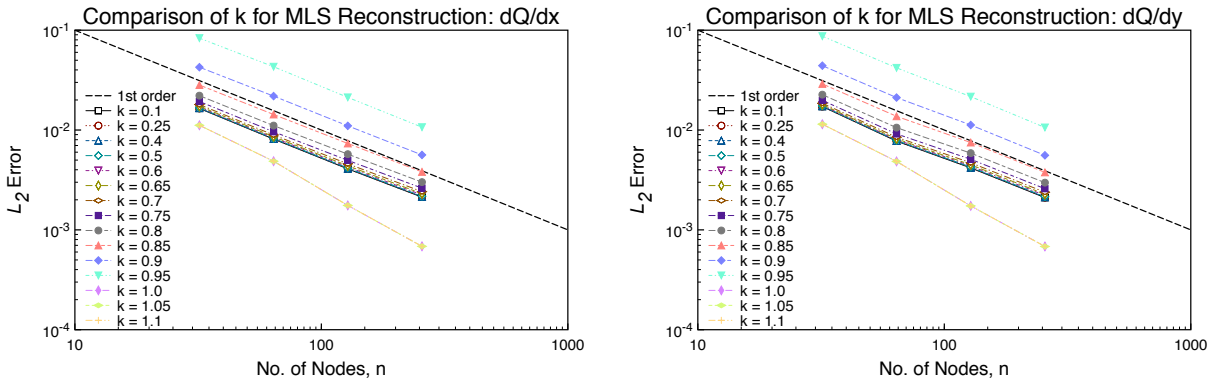


Figure 6.102: Error Norms for State on Unstructured Grids: 2^{nd}-Order Affine MLS with Anisotropic Weights Varying k.

error of q for different k, though the least error occurs when using $k = 0.95$, which pushes the boundary of $\Omega_{\mathbf{x}_I}$ close to the outlying nodes in the stencil. The lowest overall error of q is for $k \in [1.0 - 1.1]$, though not as dramatically for the structured results shown in Figure 6.84. These values of k perform best, since they effectively exclude the outlying nodes from the reconstruction and use only the central node $\mathbf{x}_I$. Therefore, when choosing the scaling parameter k, some care has to be made to ensure the central node is included in the stencil, otherwise, $k \in [1.0 - 1.1]$ will give incorrect answers. The first derivatives are shown in Figure 6.103.



(a) $\frac{\partial q}{\partial x}$ (b) $\frac{\partial q}{\partial y}$

Figure 6.103: Error Norms for First Derivatives on Unstructured Grids: 2^{nd}-Order Affine MLS with Anisotropic Weights Varying k.

Examining both derivatives, there is a broad variation with respect to k. The lowest error appears to be for $k \in [1.0 - 1.1]$, with the worse error occurring with $k = 0.95$. Table 6.112 shows the computed order of accuracy for the varying k.

Table 6.112: Accuracy on Unstructured Grids: 2^{nd}-Order Affine MLS with Anisotropic Weights Varying k.

	Function		
k	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
0.1	2.01	0.97	0.94
0.25	2.01	0.97	0.94
0.4	2.01	0.96	0.94
0.5	2.01	0.96	0.94
0.6	2.01	0.95	0.93
0.65	2.01	0.95	0.92
0.7	2.00	0.94	0.92
0.75	2.00	0.94	0.91
0.8	2.00	0.94	0.92
0.85	2.00	0.96	0.93
0.9	1.99	0.98	0.96
0.95	1.97	1.01	0.99
1.0	1.89	1.42	1.42
1.05	1.89	1.42	1.42
1.1	1.89	1.42	1.42

From Table 6.112, the overall accuracy between any of the values k is effectively the same, with only slight improvement observed for $k \in [1.0 - 1.1]$, with the accuracy just above second-order for the function. For the first-derivatives, the accuracy decreases as k is increased, with the minimum accuracy occurring for $k = 1.05$, though the average accuracy is still approximately first-order. Table 6.113 shows the condition numbers for each k.

Table 6.113: Average Condition Number of M on Unstructured Grids: 2^{nd}-Order Affine MLS with Anisotropic Weights Varying k.

	Mesh Size			
k	32	64	128	256
0.1	17.92	18.51	18.93	19.20
0.25	7.31	7.57	7.75	7.88
0.4	4.65	4.83	4.96	5.04
0.5	3.77	3.92	4.03	4.10
0.6	3.17	3.31	3.41	3.47
0.65	2.95	3.08	3.17	3.22
0.7	2.76	2.87	2.96	3.01
0.75	2.59	2.70	2.78	2.83
0.8	2.44	2.55	2.62	2.67
0.85	2.32	2.42	2.48	2.52
0.9	2.24	2.31	2.36	2.39
0.95	2.22	2.26	2.28	2.30
1.0	$4.32E5$	$3.26E5$	$2.61E5$	$2.09E5$
1.05	$4.11E5$	$3.11E5$	$2.48E5$	$1.99E5$
1.1	$3.93E5$	$2.97E5$	$2.37E5$	$1.90E5$

The condition numbers from Table 6.113 show that while the order of accuracy may be 'best' for $k \in [1.0 - 1.1]$, by far those k have the poorest condition number. Generally, the condition number decreases with increasing k for $k < 1.0$. Based upon this data, it would appear better to use $k < 1.0$ for reconstructing the function and its derivatives.

6.5.2.2.2 Third-Order The error for the reconstruction of the field using a third-order Affine MLS reconstruction is shown in Figure 6.104.

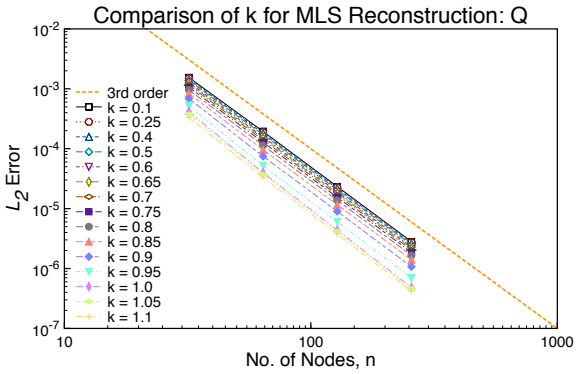
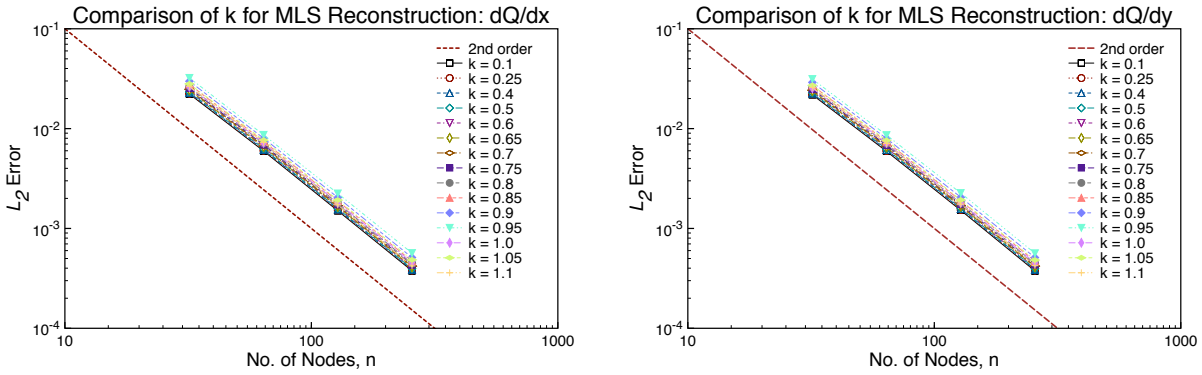


Figure 6.104: Error Norms for State on Unstructured Grids: 3^{rd}-Order Affine MLS with Anisotropic Weights Varying k.

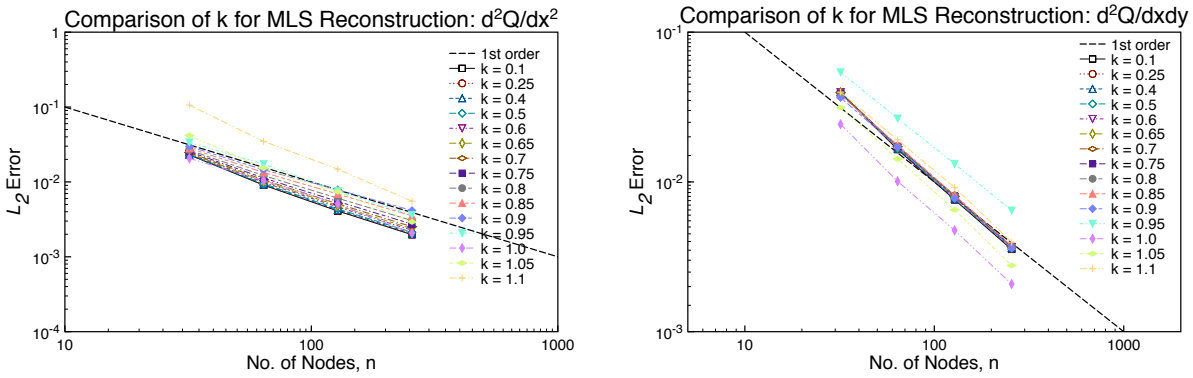
As with the second-order approximation, there is only slight spread in the error as a function of k for the reconstruction of the function. As k increases, the error drops in the reconstruction, with the lowest error for $k = 1.1$. For $k \in [1.0 - 1.1]$, the error is lowest since the 'furthest' stencil nodes are effectively excluded from the approximation. Otherwise, there is no major difference in the error for the reconstruction based on k for the function. The error for the reconstruction of the first derivatives is shown in Figure 6.105



(a) $\frac{\partial q}{\partial x}$ (b) $\frac{\partial q}{\partial y}$

Figure 6.105: Error Norms for First Derivatives on Unstructured Grids: 3^{rd}-Order Affine MLS with Anisotropic Weights Varying k.

Again, the lowest error is observed for $k = [1.0 - 1.1]$. However, unlike the second-order reconstruction, the error increases with increasing k for $k = [0.1 - 0.95]$, though there is no significant difference in the error for $k < 0.85$. The error in the second-derivatives for varying k is shown in Figure 6.106



(a) $\frac{\partial^2 q}{\partial x^2}$

(b) $\frac{\partial^2 q}{\partial xy}$

(c) $\frac{\partial^2 q}{\partial y^2}$

Figure 6.106: Error Norms for Second Derivatives on Unstructured Grids: 3^{rd}-Order Affine MLS with Anisotropic Weights Varying k.

Unlike with the first-derivatives, the highest error observed occurs when using $k = [1.0 - 1.1]$, with error decreasing as k decreases for both $\frac{\partial^2 q}{\partial x^2}$ and $\frac{\partial^2 q}{\partial y^2}$. There is also no observed variation in

error with k for $\frac{\partial^2 q}{\partial xy}$ except for $k = 0.95$ (highest) and $k = [1.0 - 1.05]$ (lowest). The overall accuracy of the reconstruction for each k is presented in Table 6.114.

Table 6.114: Accuracy on Unstructured Grids: 3^{rd}-Order Affine MLS with Anisotropic Weights Varying k.

	Function					
k	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
0.1	3.05	1.99	1.99	1.10	1.11	1.11
0.25	3.05	1.99	1.99	1.10	1.11	1.11
0.4	3.05	1.99	1.99	1.09	1.11	1.10
0.5	3.05	2.00	1.99	1.08	1.10	1.09
0.6	3.05	2.00	1.99	1.06	1.10	1.08
0.65	3.05	2.00	1.99	1.05	1.10	1.07
0.7	3.05	2.00	1.99	1.04	1.11	1.06
0.75	3.05	2.00	1.99	1.02	1.11	1.05
0.8	3.05	2.00	1.99	1.00	1.11	1.03
0.85	3.05	1.99	1.99	0.96	1.12	1.01
0.9	3.06	1.99	1.98	0.90	1.12	0.97
0.95	3.12	1.97	1.97	1.13	1.02	1.17
1.0	3.19	1.99	1.99	1.17	1.14	1.26
1.05	3.17	1.99	1.99	1.19	1.18	1.26
1.1	3.16	2.05	2.03	1.33	1.14	1.32

From Table 6.114, the overall accuracy between any of the values k is effectively the same. The accuracy for the function improves with increasing k up to $k = 0.95$, which has the highest accuracy and begins to decrease as k increase beyond $k = 1.0$. The average accuracy for the

function for all k is third-order. For the first derivatives, all methods achieve second-order, so all k would produce similar accuracy as the mesh is refined. The second derivatives show decreasing accuracy as k is increased, with the minimum accuracy occurring with $k = 1.1$. Table 6.115 shows the condition numbers for each k.

Table 6.115: Average Condition Number of $\mathbf{M}$ on Unstructured Grids: 3^{rd}-Order Affine MLS with Anisotropic Weights Varying k.

	Mesh Size			
k	32	64	128	256
0.1	595.62	565.41	542.34	525.14
0.25	99.30	94.39	90.60	87.75
0.4	41.23	39.31	37.82	36.68
0.5	27.71	26.49	25.56	24.84
0.6	20.33	19.52	18.89	18.42
0.65	17.84	17.18	16.67	16.29
0.7	15.87	15.35	14.95	14.64
0.75	14.31	13.93	13.62	13.37
0.8	13.08	12.83	12.62	12.44
0.85	12.15	12.05	11.92	11.81
0.9	11.55	11.58	11.57	11.53
0.95	11.50	11.57	11.59	11.59
1.0	11.37	11.41	11.41	11.41
1.05	10.80	10.89	10.92	10.93
1.1	10.55	10.71	10.80	10.84

The condition numbers from Table 6.115 show that for increasing k, the condition number

decreases. In conjunction with data in Table 6.114, it is determined that choosing $k \in [0.6 - 0.8]$ produces the best overall results for the reconstruction.

6.5.2.2.3 Fourth-Order The error for the reconstruction of the field using a fourth-order Affine MLS reconstruction is shown in Figure 6.107.

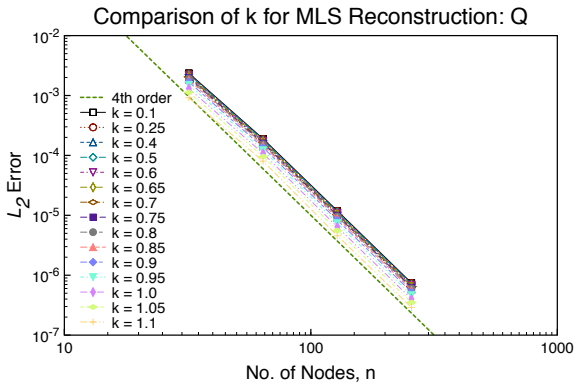
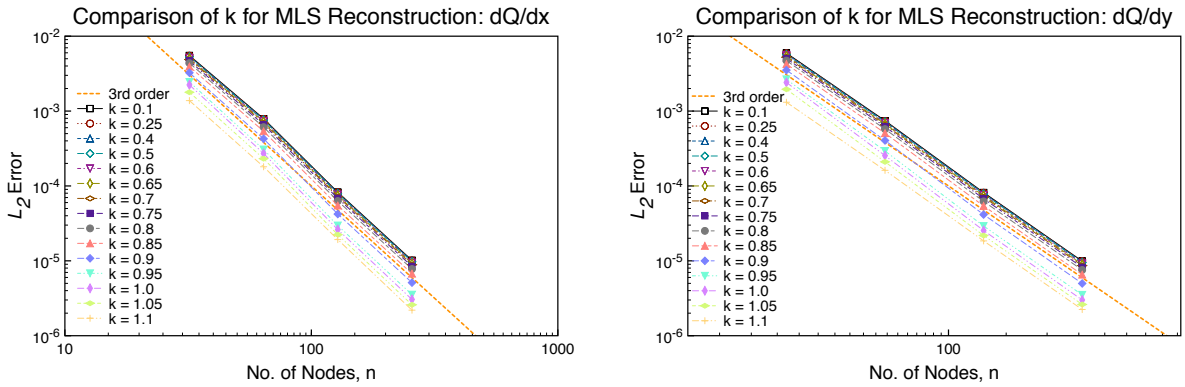


Figure 6.107: Error Norms for State on Unstructured Grids: 4^{th}-Order Affine MLS with Anisotropic Weights Varying k.

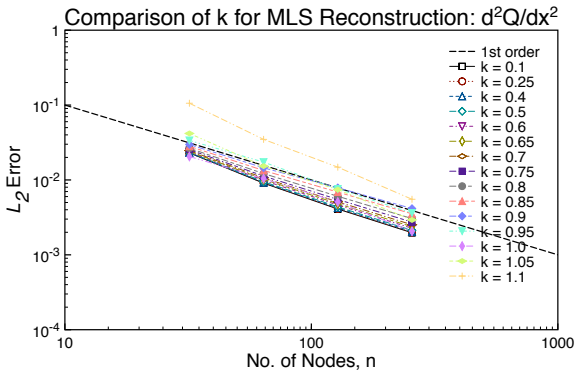
As with the third-order case, the fourth-order reconstruction has lower error as k is increased, with lowest error observed with $k = 1.1$. Otherwise, there is no major difference in the error for the reconstruction based on k for the function. The error for the reconstruction of the first derivatives is shown in Figure 6.108



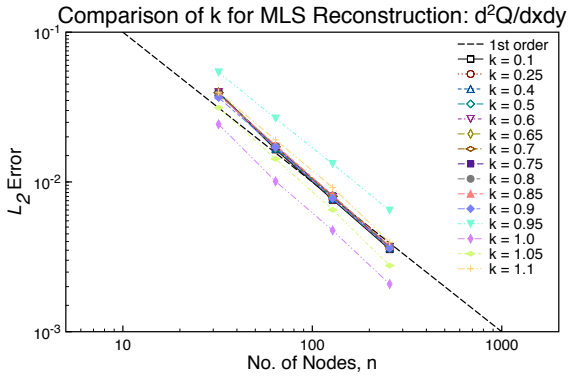
(a) $\frac{\partial q}{\partial x}$ (b) $\frac{\partial q}{\partial y}$

Figure 6.108: Error Norms for First Derivatives on Unstructured Grids: 4^{th}-Order Affine MLS with Anisotropic Weights Varying k.

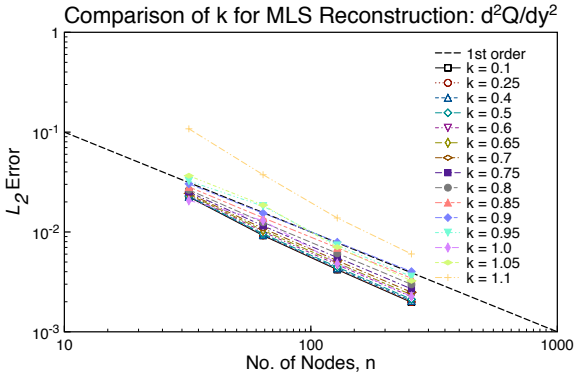
The lowest error is observed for $k = [1.0 - 1.1]$. The error also decreases as k increases, though there is little variation in the overall error for $k \in [0.1 - 0.85]$. The error in the second-derivatives for varying k is shown in Figure 6.109



(a) $\frac{\partial^2 q}{\partial x^2}$



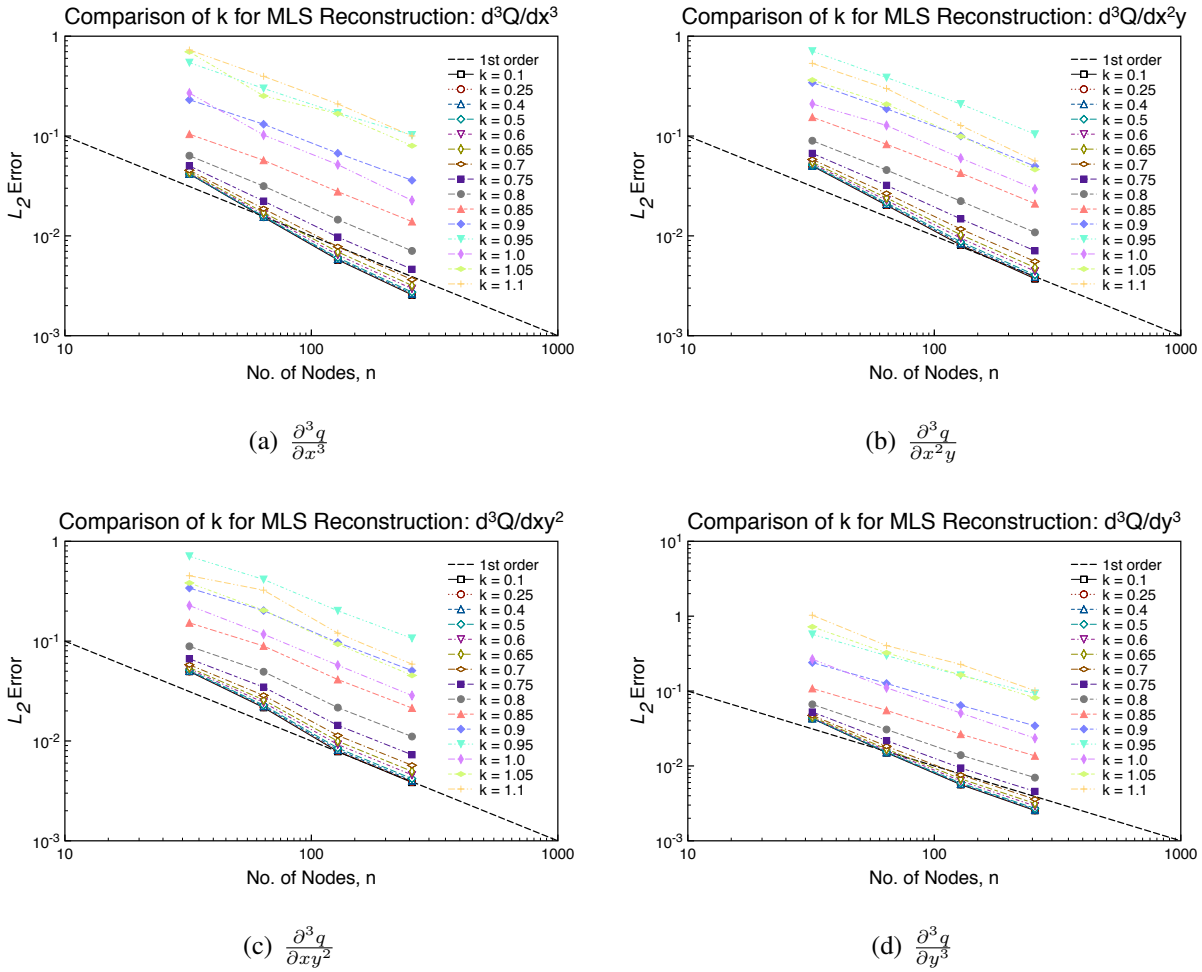
(b) $\frac{\partial^2 q}{\partial xy}$



(c) $\frac{\partial^2 q}{\partial y^2}$

Figure 6.109: Error Norms for Second Derivatives on Unstructured Grids: 4^{th}-Order Affine MLS with Anisotropic Weights Varying k.

The results in Figure 6.109 show that the highest error observed occurs when using $k = [1.0 - 1.1]$, with error decreasing as k decreases for both $\frac{\partial^2 q}{\partial x^2}$ and $\frac{\partial^2 q}{\partial y^2}$. There is also no observed variation in error with k for $\frac{\partial^2 q}{\partial xy}$ except for $k = 0.95$ (highest) and $k = [1.0 - 1.05]$ (lowest). The error for the third-derivatives with varying k is shown in Figure 6.110.



(a) $\frac{\partial^3 q}{\partial x^3}$ (b) $\frac{\partial^3 q}{\partial x^2 y}$

(c) $\frac{\partial^3 q}{\partial x y^2}$ (d) $\frac{\partial^3 q}{\partial y^3}$

Figure 6.110: Error Norms for Third Derivatives on Unstructured Grids: 4^{th}-Order Affine MLS with Anisotropic Weights Varying k.

Figure 6.110 shows a large spread in the error, with the general trend being increasing error as k increases. The highest error observed occurs when using $k = 1.1$ and $k = [0.95, 1.05]$, with another cluster of larger error for $k = 0.85 - 0.9, 1.0]$. For the intermediate derivatives, $\frac{\partial^3 q}{\partial x^2 y}$ and $\frac{\partial^3 q}{\partial x y^2}$, the largest error is observed for $k \in [0.95]$, with the same clusters as the other derivatives. Generally, the error increases with increasing k for the third-derivatives, with the lowest error for $k = [0.1 - 0.65]$. The overall accuracy of the reconstruction for each k is presented in Table 6.114.

Table 6.116: Accuracy on Unstructured Grids: 4^{th}-Order Affine MLS with Anisotropic Weights Varying k.

	Function									
k	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial xy^2}$	$\frac{\partial^3 q}{\partial y^3}$
0.1	4.00	3.14	3.12	1.98	1.98	1.97	1.29	1.22	1.25	1.28
0.25	4.00	3.14	3.12	1.98	1.98	1.97	1.29	1.22	1.24	1.28
0.4	4.00	3.14	3.11	1.98	1.97	1.97	1.28	1.21	1.24	1.27
0.5	4.00	3.14	3.11	1.98	1.97	1.97	1.26	1.20	1.23	1.25
0.6	4.00	3.14	3.12	1.97	1.97	1.97	1.24	1.18	1.21	1.23
0.65	4.00	3.14	3.12	1.97	1.97	1.97	1.21	1.16	1.19	1.21
0.7	4.00	3.14	3.12	1.97	1.97	1.97	1.18	1.13	1.16	1.17
0.75	4.00	3.14	3.12	1.96	1.97	1.97	1.13	1.09	1.13	1.13
0.8	4.00	3.15	3.13	1.95	1.97	1.96	1.08	1.04	1.08	1.07
0.85	4.00	3.16	3.14	1.94	1.96	1.96	1.02	0.99	1.03	1.01
0.9	4.01	3.19	3.17	1.90	1.96	1.94	0.93	0.96	1.00	0.94
0.95	4.02	3.22	3.19	1.84	1.96	1.91	0.77	0.94	0.98	0.84
1.0	4.04	3.24	3.20	1.93	1.98	1.95	1.09	1.06	1.02	1.12
1.05	4.05	3.24	3.16	1.95	1.99	1.95	0.83	1.09	1.08	1.00
1.1	4.06	3.18	3.09	2.01	1.97	2.01	0.99	1.21	1.23	1.00

From Table 6.116, the accuracy for the function increases as k increases, with maximum at $k = 1.1$, but on average the order of accuracy is roughly fourth-order. For the first derivatives, the accuracy increases with increasing k up to $k = 0.9$, after which the accuracy decreases. Overall, the accuracy for the first-derivatives is approximately third-order. For the second-derivatives, the accuracy is approximately second-order, with the maximum obtained when $k = 0.8$, with slightly lower accuracy for other values of k. For the third-derivatives, the accuracy decreases as k is

increased, with the minimum at $k = 1.1$, with a secondary maximum accuracy occurring for $k = 0.8$. The average error for the third-derivatives is first-order. Table 6.117 shows the condition numbers for each k.

Table 6.117: Average Condition Number of $\mathbf{M}$ on Unstructured Grids: 4^{th}-Order Affine MLS with Anisotropic Weights Varying k.

	Mesh Size			
k	32	64	128	256
0.1	17737	16566	15920	15545
0.25	1178.7	1105.6	2059.8	1035.0
0.4	305.05	285.62	274.68	268.35
0.5	163.79	153.42	147.61	144.13
0.6	99.89	93.62	90.11	87.93
0.65	80.85	75.83	73.01	71.23
0.7	66.64	62.60	60.30	58.80
0.75	55.89	52.59	50.72	49.44
0.8	47.70	45.02	43.48	42.38
0.85	41.50	39.34	38.10	37.18
0.9	36.99	35.31	34.36	33.62
0.95	34.04	32.87	32.16	31.61
1.0	31.97	31.21	30.74	30.33
1.05	29.62	29.15	28.89	28.57
1.1	28.28	28.25	28.14	28.03

The condition numbers from Table 6.117 show that for increasing k, the condition number decreases. In conjunction with data in Table 6.116, it is determined that choosing $k \in [0.6 - 0.8]$

produces the best overall results for the reconstruction.

6.5.3 Stretched Grids

This section presents a detailed study on the effect of the scaling parameter k on the MLS reconstruction on stretched grids. For this study, the MLS reconstruction using isotropic and anisotropic weights and Affine MLS with anisotropic weights are used.

6.5.3.1 Isotropic Weights

This subsection presents the results of varying the scaling parameter k when using isotropic weighting.

6.5.3.1.1 Second-Order The error for the reconstruction of the field using a second-order MLS reconstruction is shown in Figure 6.111.

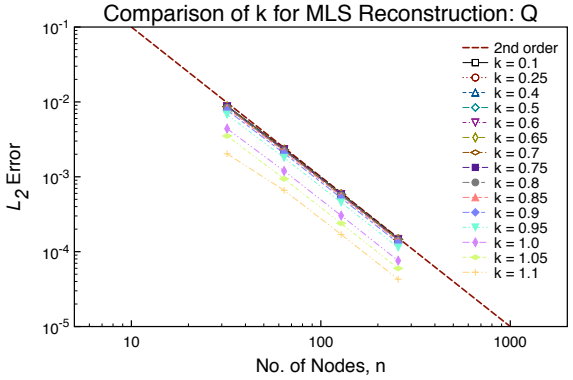
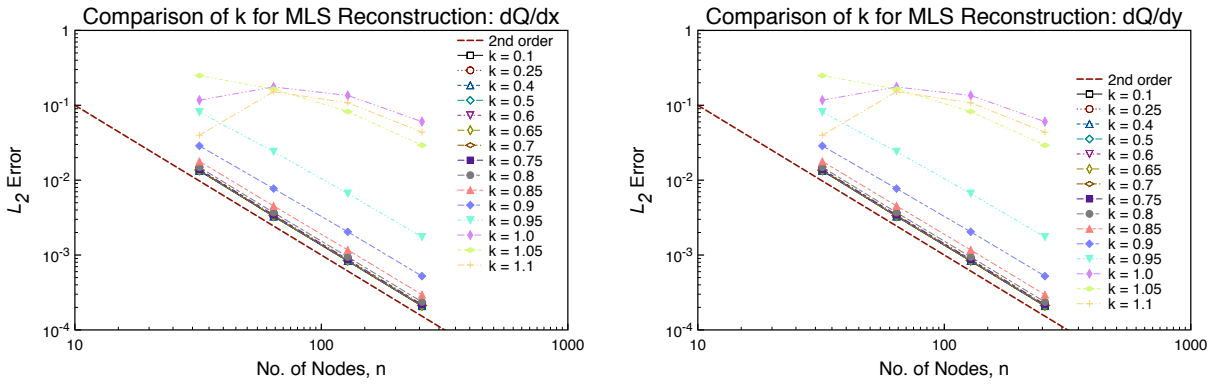


Figure 6.111: Error Norms for State on Stretched Grids: 2^{nd}-Order MLS with Isotropic Weights Varying k.

From Figure 6.111, there is not much variation for the error of q for $k \in [0.1 - 0.95]$, though the least error occurs when using $k = 0.95$, which pushes the boundary of $\Omega_{\mathbf{x}_I}$ close to the outlying nodes in the stencil. The lowest overall error of q is for $k \in [1.0 - 1.1]$, though not as dramatically for the structured results shown in Figure 6.84. These values of k perform best, since they effectively exclude the outlying nodes from the reconstruction and use only the central node $\mathbf{x}_I$.

Therefore, when choosing the scaling parameter k, some care has to be made to ensure the central node is included in the stencil, otherwise, $k \in [1.0 - 1.1]$ will give incorrect answers. The first derivatives are shown in Figure 6.112.



(a) $\frac{\partial q}{\partial x}$ (b) $\frac{\partial q}{\partial y}$

Figure 6.112: Error Norms for First Derivatives on Stretched Grids: 2^{nd}-Order MLS with Isotropic Weights Varying k.

Examining both derivatives, there is a broad variation with respect to k. The largest error appears to be for $k \in [1.0 - 1.1]$, which is not the case for both the structured and unstructured meshes. These large errors most likely results from the poor modeling of the derivatives of C. Table 6.118 shows the computed order of accuracy for the varying k.

Table 6.118: Accuracy on Stretched Grids: 2^{nd}-Order MLS with Isotropic Weights Varying k.

	Function		
k	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
0.1	1.99	1.99	1.99
0.25	1.99	1.99	1.99
0.4	1.99	1.99	1.99
0.5	1.99	1.99	1.99
0.6	1.99	1.99	1.99
0.65	1.99	1.99	1.99
0.7	1.99	1.99	1.99
0.75	1.99	1.99	1.99
0.8	1.99	1.98	1.98
0.85	1.99	1.97	1.97
0.9	1.99	1.94	1.94
0.95	1.99	1.89	1.89
1.0	1.99	0.77	0.77
1.05	1.99	1.25	1.25
1.1	1.97	0.90	0.90

From Table 6.118, the accuracy of q is the same for all k. For the derivatives, the error increases with increasing k. The approximate accuracy for the function and first-derivatives is second-order, except for $k \in [1.0 - 1.1]$. This case highlights the need to pick $k < 1.0$, since the exclusion of the extreme nodes causes large errors in the derivatives. To further prove this point, the condition numbers of $\mathbf{M}$ are shown in Table 6.119 for each k.

Table 6.119: Average Condition Number of $\mathbf{M}$ on Stretched Grids: 2^{nd}-Order MLS with Isotropic Weights Varying k.

	Mesh Size			
k	32	64	128	256
0.1	58.20	53.63	51.24	50.03
0.25	23.30	21.42	20.44	19.95
0.4	14.57	13.37	12.75	12.43
0.5	11.67	10.70	10.20	9.94
0.6	9.73	8.91	8.48	8.27
0.65	8.97	8.21	7.81	7.62
0.7	8.29	7.59	7.22	7.04
0.75	7.67	7.03	6.69	6.52
0.8	7.11	6.51	6.20	6.05
0.85	6.64	6.08	5.76	5.60
0.9	6.26	5.67	5.33	5.16
0.95	5.97	5.34	4.97	4.79
1.0	$7.38E4$	$3.61E4$	$1.78E4$	8888
1.05	$1.99E5$	$4.03E5$	$8.94E5$	$1.10E6$
1.1	$4.18E5$	$8.22E5$	$1.05E6$	$1.03E6$

The condition numbers from Table 6.119 clearly show that the condition numbers for $k \geq 1.0$ produce the poor accuracy in the results. Generally, the condition number decreases with increasing k for $k < 1.0$. Based upon this data, it would appear better to use $k < 1.0$ for reconstructing the function and its derivatives.

6.5.3.1.2 Third-Order The error for the reconstruction of the field using a third-order MLS reconstruction is shown in Figure 6.113.

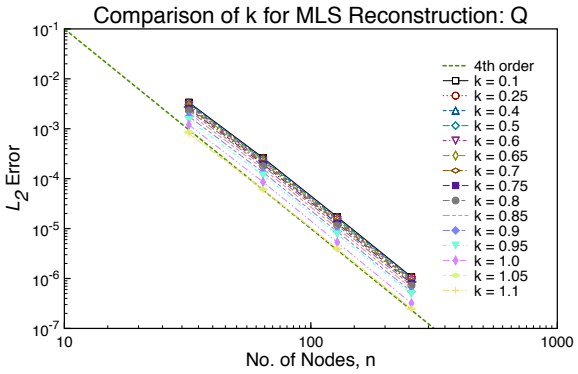
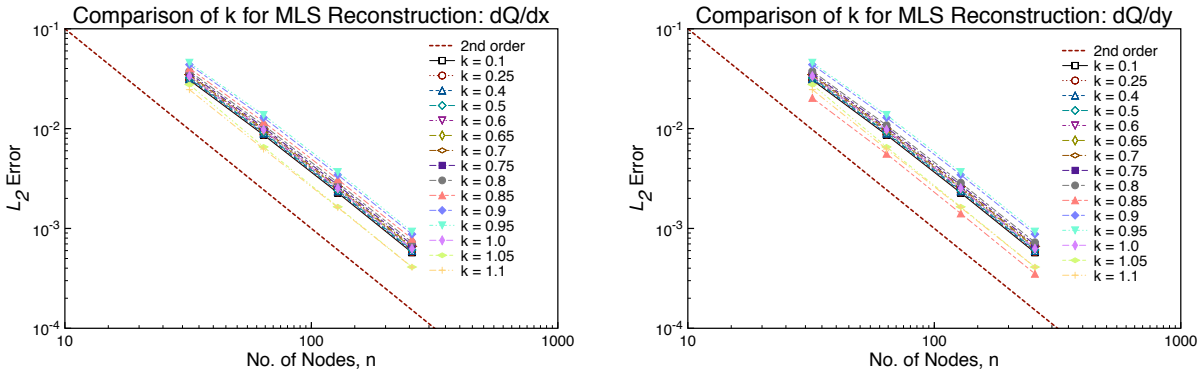


Figure 6.113: Error Norms for State on Stretched Grids: 3^{rd}-Order MLS with Isotropic Weights Varying k.

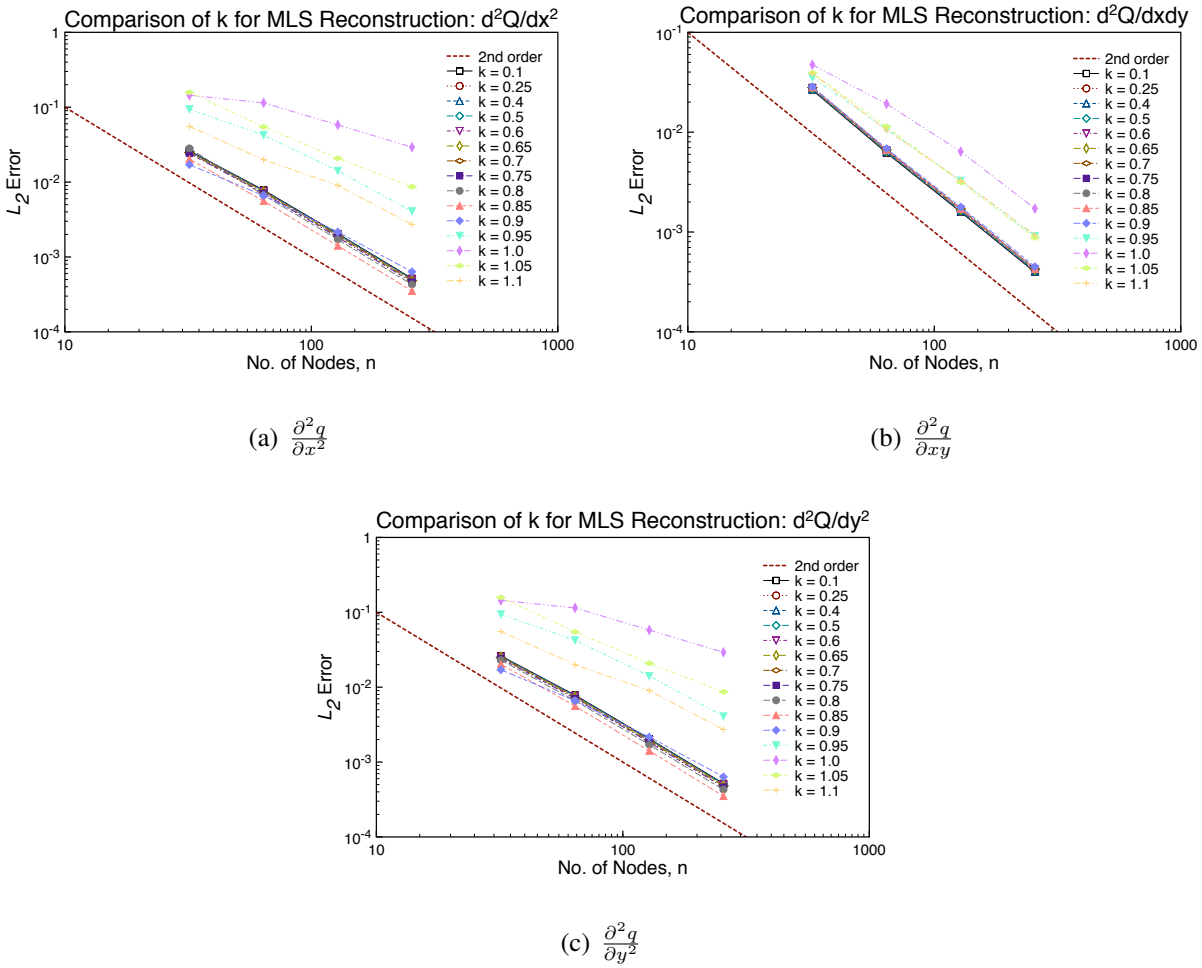
As with the second-order approximation, there is only slight spread in the error as a function of k for the reconstruction of the function. As k increases, the error drops in the reconstruction, with the lowest error for $k = 1.1$. For $k \in [1.0 - 1.1]$, the error is lowest since the 'furthest' stencil nodes are effectively excluded from the approximation. Otherwise, there is no major difference in the error for the reconstruction based on k for the function. The error for the reconstruction of the first derivatives is shown in Figure 6.114



(a) $\frac{\partial q}{\partial x}$ (b) $\frac{\partial q}{\partial y}$

Figure 6.114: Error Norms for First Derivatives on Stretched Grids: 3^{rd}-Order MLS with Isotropic Weights Varying k.

Again, the lowest error is observed for $k = [1.0 - 1.1]$. However, unlike the second-order reconstruction, the error increases with increasing k for $k = [0.1 - 0.95]$, though there is no significant difference in the error for $k < 0.85$. The error in the second-derivatives for varying k is shown in Figure 6.115



(a) $\frac{\partial^2 q}{\partial x^2}$

(b) $\frac{\partial^2 q}{\partial xy}$

(c) $\frac{\partial^2 q}{\partial y^2}$

Figure 6.115: Error Norms for Second Derivatives on Stretched Grids: 3^{rd}-Order MLS with Isotropic Weights Varying k.

Unlike with the first-derivatives, the highest error observed occurs when using $k = [0.95 - 1.1]$, with error decreasing as k decreases. The overall accuracy of the reconstruction for each k is presented in Table 6.120.

Table 6.120: Accuracy on Stretched Grids: 3rd-Order MLS with Isotropic Weights Varying k.

	Function					
k	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
0.1	3.96	1.96	1.96	1.96	1.99	1.96
0.25	3.96	1.96	1.96	1.96	1.99	1.96
0.4	3.96	1.96	1.96	1.96	1.98	1.96
0.5	3.96	1.95	1.95	1.96	1.98	1.96
0.6	3.96	1.95	1.95	1.96	1.98	1.96
0.65	3.96	1.95	1.95	1.96	1.98	1.96
0.7	3.97	1.95	1.95	1.96	1.98	1.96
0.75	3.97	1.95	1.95	1.97	1.98	1.97
0.8	3.97	1.95	1.95	1.97	1.98	1.97
0.85	3.98	1.94	1.94	2.00	1.98	2.00
0.9	3.97	1.94	1.94	1.69	1.96	1.69
0.95	3.95	1.94	1.94	1.69	1.78	1.69
1.0	4.03	1.98	1.98	0.98	1.74	0.98
1.05	3.98	2.00	2.00	1.33	1.85	1.33
1.1	3.96	1.97	1.97	1.44	1.74	1.44

From Table 6.120, the overall accuracy between any of the values k is effectively the same. The accuracy for the function increases with increasing k, with an average accuracy of fourth-order. For the first derivatives, all methods achieve nominal second-order accuracy, with the order decreasing as k is increased up to $k = 1.0$, where the order of accuracy increases. The second derivatives show decreasing accuracy as k is increased, with the minimum accuracy occurring with $k = 1.1$. Before determining which k might be best, the condition number of $\mathbf{M}$ should be checked. Table 6.121 shows the condition numbers for each k.

Table 6.121: Average Condition Number of $\mathbf{M}$ on Stretched Grids: 3^{rd}-Order MLS with Isotropic Weights Varying k.

	Mesh Size			
k	32	64	128	256
0.1	7238	6541	6006	5690
0.25	1171	1056	969.7	918.3
0.4	463.7	418.1	383.6	363.2
0.5	299.2	269.7	247.4	234.4
0.6	208.8	188.3	172.8	163.7
0.65	178.1	160.6	147.4	139.7
0.7	153.6	138.6	127.2	120.5
0.75	134.0	120.8	110.8	104.9
0.8	118.0	116.2	97.26	92.00
0.85	104.9	94.09	86.02	81.32
0.9	93.90	83.99	76.72	72.52
0.95	84.79	75.66	69.13	65.42
1.0	76.97	68.64	62.84	59.58
1.05	69.93	62.41	57.25	54.29
1.1	63.82	57.03	52.36	49.66

The condition numbers from Table 6.121 show that for increasing k, the condition number decreases. In conjunction with data in Table 6.120, it is determined that choosing $k \in [0.6 - 0.8]$ produces the best overall results for the reconstruction.

6.5.3.1.3 Fourth-Order The error for the reconstruction of the field using a fourth-order MLS reconstruction is shown in Figure 6.116.

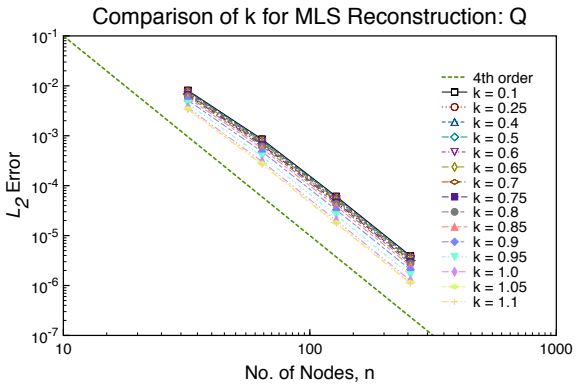
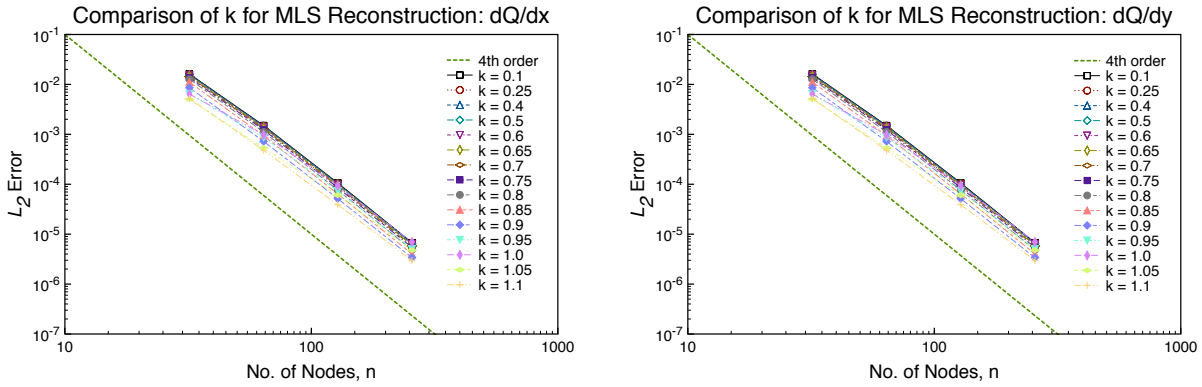


Figure 6.116: Error Norms for State on Stretched Grids: 4^{th}-Order MLS with Isotropic Weights Varying k.

As with the third-order case, the fourth-order reconstruction has lower error as k is increased, with lowest error observed with $k = 1.1$. Otherwise, there is no major difference in the error for the reconstruction based on k for the function. The error for the reconstruction of the first derivatives is shown in Figure 6.117

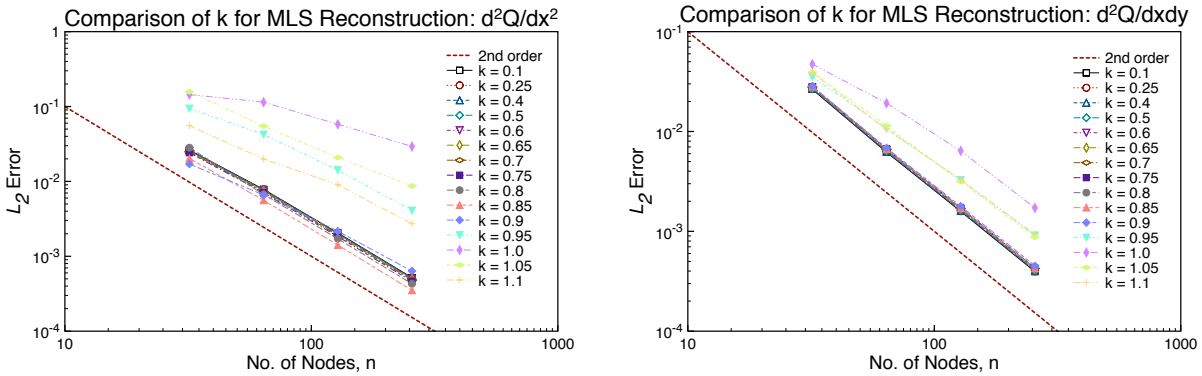


(a) $\frac{\partial q}{\partial x}$ (b) $\frac{\partial q}{\partial y}$

Figure 6.117: Error Norms for First Derivatives on Stretched Grids: 4^{th}-Order MLS with Isotropic Weights Varying k.

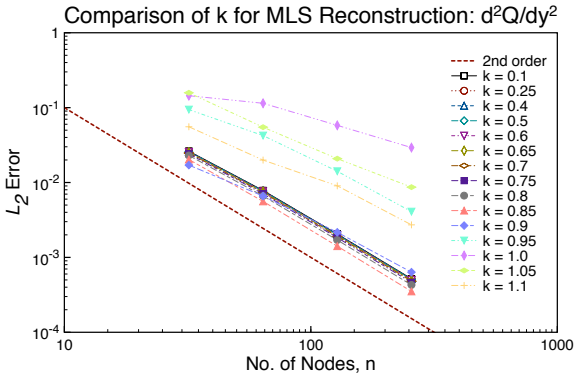
The lowest error is observed for $k = [1.0 - 1.1]$. The error also decreases as k increases, though

there is little variation in the overall error for $k \in [0.1 - 0.85]$. The error in the second-derivatives for varying k is shown in Figure 6.118



(a) $\frac{\partial^2 q}{\partial x^2}$

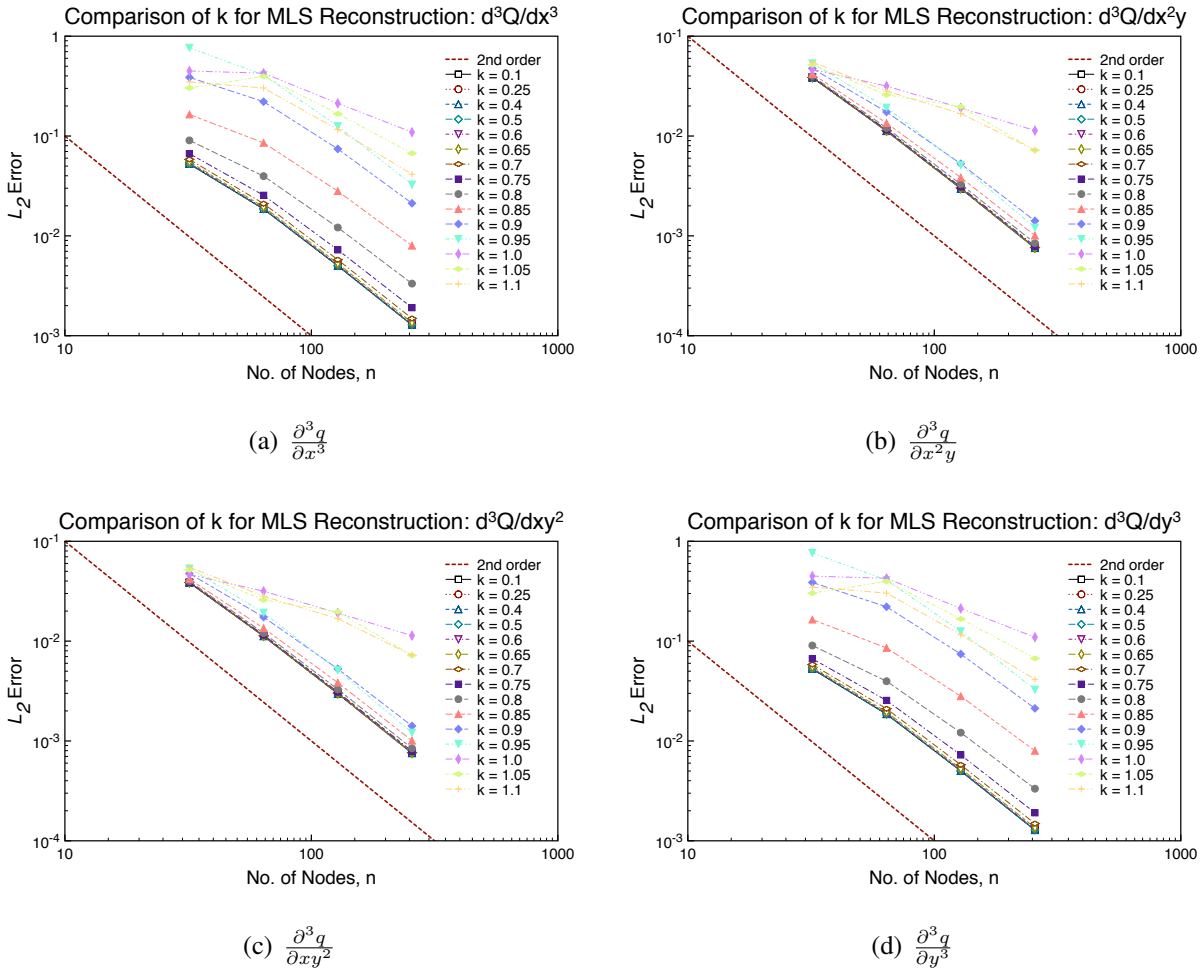
(b) $\frac{\partial^2 q}{\partial xy}$



(c) $\frac{\partial^2 q}{\partial y^2}$

Figure 6.118: Error Norms for Second Derivatives on Stretched Grids: 4^{th}-Order MLS with Isotropic Weights Varying k.

The results in Figure 6.118 show that the highest error observed occurs when using $k = [0.95 - 1.1]$. The error for the third-derivatives with varying k is shown in Figure 6.119.



(a) $\frac{\partial^3 q}{\partial x^3}$ (b) $\frac{\partial^3 q}{\partial x^2 y}$

(c) $\frac{\partial^3 q}{\partial x y^2}$ (d) $\frac{\partial^3 q}{\partial y^3}$

Figure 6.119: Error Norms for Third Derivatives on Stretched Grids: 4^{th}-Order MLS with Isotropic Weights Varying k.

Figure 6.119 shows a large spread in the error, with the general trend being increasing error as k increases. The highest error observed occurs when for $k \in [0.9 - 1.1]$. For the intermediate derivatives, $\frac{\partial^3 q}{\partial x^2 y}$ and $\frac{\partial^3 q}{\partial x y^2}$, the largest error is observed for $k \in [1.0 - 1.1]$, with the same clusters as the other derivatives. Generally, the error increases with increasing k for the third-derivatives, with the lowest error for $k = [0.1 - 0.65]$. The overall accuracy of the reconstruction for each k is presented in Table 6.120.

Table 6.122: Accuracy on Stretched Grids: 4^{th}-Order MLS with Isotropic Weights Varying k.

	Function									
k	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial xy^2}$	$\frac{\partial^3 q}{\partial y^3}$
0.1	3.88	3.90	3.90	1.90	1.95	1.90	1.93	1.95	1.95	1.93
0.25	3.88	3.90	3.90	1.90	1.95	1.90	1.93	1.95	1.95	1.93
0.4	3.88	3.90	3.90	1.90	1.95	1.90	1.93	1.95	1.95	1.93
0.5	3.88	3.90	3.90	1.90	1.95	1.90	1.93	1.95	1.95	1.93
0.6	3.89	3.90	3.90	1.89	1.95	1.89	1.93	1.95	1.95	1.93
0.65	3.89	3.90	3.90	1.89	1.95	1.89	1.92	1.95	1.95	1.92
0.7	3.89	3.91	3.91	1.88	1.95	1.88	1.91	1.94	1.94	1.91
0.75	3.90	3.91	3.91	1.87	1.95	1.87	1.87	1.94	1.94	1.87
0.8	3.90	3.92	3.92	1.86	1.94	1.86	1.79	1.92	1.92	1.79
0.85	3.91	3.94	3.94	1.84	1.94	1.84	1.71	1.87	1.87	1.71
0.9	3.93	3.87	3.87	1.83	1.93	1.83	1.69	1.81	1.81	1.69
0.95	3.95	3.71	3.71	1.82	1.91	1.82	1.82	2.00	2.00	1.82
1.0	3.96	3.57	3.57	1.63	1.93	1.63	0.98	0.74	0.74	0.98
1.05	3.96	3.40	3.40	1.83	1.95	1.83	1.28	0.92	0.92	1.28
1.1	3.96	3.67	3.67	2.01	1.92	2.01	1.43	0.98	0.98	1.43

From Table 6.122, the accuracy for the function increases as k increases, with maximum at $k = 1.1$, but on average the order of accuracy is roughly fourth-order. For the first derivatives, the accuracy increases with increasing k up to $k = 0.85$, after which the accuracy decreases. Overall, the accuracy for the first-derivatives is approximately fourth-order. For the second-derivatives, the accuracy is approximately second-order, the order decreasing as k increases. For the third-derivatives, the accuracy decreases as k is increased, with the minimum at $k = 1.0$. The average error for the third-derivatives is second-order for $k < 1.0$. Table 6.123 shows the condition num-

bers for each k.

Table 6.123: Average Condition Number of $\mathbf{M}$ on Stretched Grids: 4^{th}-Order MLS with Isotropic Weights Varying k.

	Mesh Size			
k	32	64	128	256
0.1	$9.88E5$	$1.02E6$	$9.69E5$	$8.96E5$
0.25	$6.39E4$	$6.62E4$	$6.19E4$	$5.77E4$
0.4	$1.58E4$	$1.63E4$	$1.52E4$	$1.42E4$
0.5	8196	8456	7885	7355
0.6	4783	4926	4591	4281
0.65	3773	3883	3617	3373
0.7	3027	3112	2898	2702
0.75	2462	2529	2355	2196
0.8	2028	2082	1938	1806
0.85	1690	1734	1612	1500
0.9	1423	1458	1353	1258
0.95	1213	1237	1146	1064
1.0	1044	1059	978.6	909.3
1.05	903.4	912.9	843.5	784.6
1.1	786.8	792.9	732.6	681.4

The condition numbers from Table 6.123 show that for increasing k, the condition number decreases. In conjunction with data in Table 6.122, it is determined that choosing $k \in [0.6 - 0.8]$ produces the best overall results for the reconstruction.

6.5.3.2 *Anisotropic Weights*

This subsection presents the results of varying the scaling parameter k when using anisotropic weighting.

6.5.3.2.1 Second-Order The error for the reconstruction of the field using a second-order MLS reconstruction is shown in Figure 6.120.

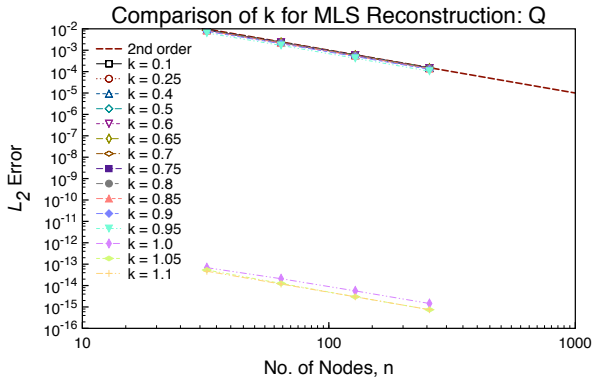
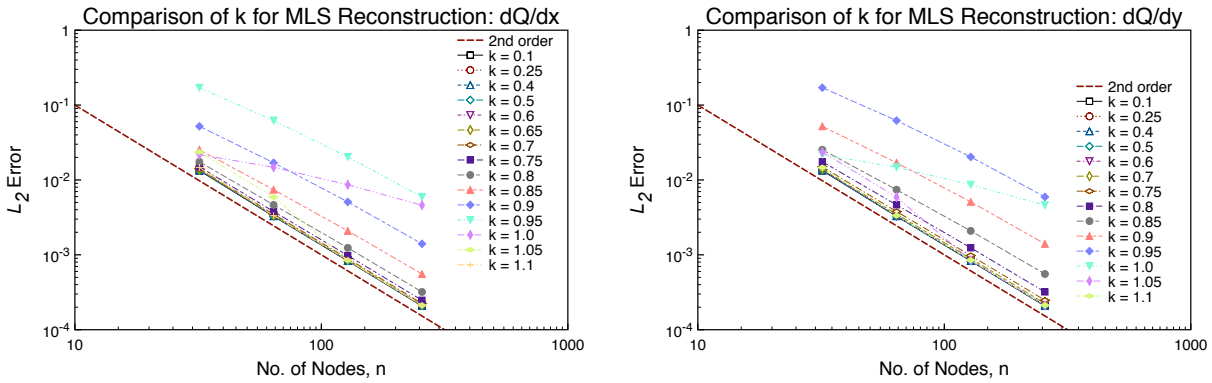


Figure 6.120: Error Norms for State on Stretched Grids: 2^{nd}-Order MLS with Anisotropic Weights Varying k.

From Figure 6.120, there is not much variation for the error of q for $k \in [0.1 - 0.95]$, though the least error occurs when using $k = 0.95$, which pushes the boundary of $\Omega_{\mathbf{x}_I}$ close to the outlying nodes in the stencil. The lowest overall error of q is for $k \in [1.0 - 1.1]$, showing similar results to the structured results shown in Figure 6.84. These values of k perform best, since they effectively exclude the outlying nodes from the reconstruction and use only the central node $\mathbf{x}_I$. Therefore, when choosing the scaling parameter k, some care has to be made to ensure the central node is included in the stencil, otherwise, $k \in [1.0 - 1.1]$ will give incorrect answers. The first derivatives are shown in Figure 6.121.



(a) $\frac{\partial q}{\partial x}$ (b) $\frac{\partial q}{\partial y}$

Figure 6.121: Error Norms for First Derivatives on Stretched Grids: 2^{nd}-Order MLS with Anisotropic Weights Varying k.

Examining both derivatives, there is a broad variation with respect to k. The lowest error appears to be for $k \in [1.05 - 1.1]$, with the larger errors occurring for $k \in [0.9 - 1.0]$. Table 6.124 shows the computed order of accuracy for the varying k.

Table 6.124: Accuracy on Stretched Grids: 2^{nd}-Order MLS with Anisotropic Weights Varying k.

	Function		
k	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
0.1	1.99	1.99	1.99
0.25	1.99	1.99	1.99
0.4	1.99	1.99	1.99
0.5	1.99	1.99	1.99
0.6	1.99	1.99	1.99
0.65	1.99	1.99	1.99
0.7	1.99	1.98	1.98
0.75	1.99	1.97	1.97
0.8	1.99	1.93	1.93
0.85	1.99	1.87	1.87
0.9	1.99	1.80	1.80
0.95	2.00	1.69	1.69
1.0	1.92	0.84	0.84
1.05	2.04	2.40	2.40
1.1	1.99	1.99	1.99

From Table 6.124, the accuracy of q is mostly constant until $k = 0.95$, after which there is high variability in the accuracy occurs. For the derivatives, the error increases with increasing k, with the exception of $k = 1.05$, which has the highest order of any k. The approximate accuracy for the function is second-order, and the accuracy for the first derivatives is also second-order. As with the structured case, the error is lower for the function due to clipping the extreme nodes in the stencil $\Omega_{\mathbf{x}_I}$ for when $k \in [1.0 - 1.1]$. Before determining which k might be best, the condition number of $\mathbf{M}$ should be checked. Table 6.125 shows the condition numbers for each k.

Table 6.125: Average Condition Number of $\mathbf{M}$ on Stretched Grids: 2^{nd}-Order MLS with Anisotropic Weights Varying k.

	Mesh Size			
k	32	64	128	256
0.1	58.29	53.71	51.33	50.11
0.25	23.48	21.58	20.61	20.11
0.4	14.77	13.56	12.93	12.61
0.5	11.87	10.88	10.37	10.11
0.6	9.93	9.09	8.66	8.44
0.65	9.18	8.40	8.00	7.80
0.7	8.53	7.81	7.44	7.26
0.75	7.97	7.29	6.96	6.70
0.8	7.48	6.86	6.57	6.43
0.85	7.19	6.61	6.31	6.16
0.9	7.21	6.53	6.20	6.04
0.95	7.86	6.92	6.47	6.28
1.0	$9.59E5$	$4.16E5$	$1.85E5$	$8.20E4$
1.05	$9.04E5$	$5.89E5$	$7.70E5$	$2.13E6$
1.1	$9.50E5$	$8.24E5$	$2.08E6$	$2.03E6$

The condition numbers from Table 6.125 show that while the order of accuracy may be 'best' for $k \in [1.0 - 1.1]$, by far those k have the poorest condition number. Generally, the condition number decreases with increasing k for $k < 1.0$. Based upon this data, it would appear better to use $k < 1.0$ for reconstructing the function and its derivatives.

6.5.3.2.2 Third-Order The error for the reconstruction of the field using a third-order MLS reconstruction is shown in Figure 6.122.

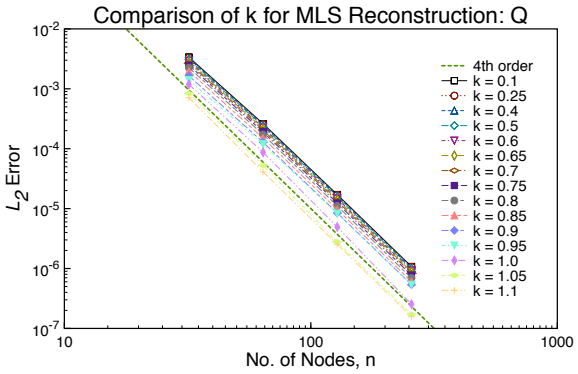
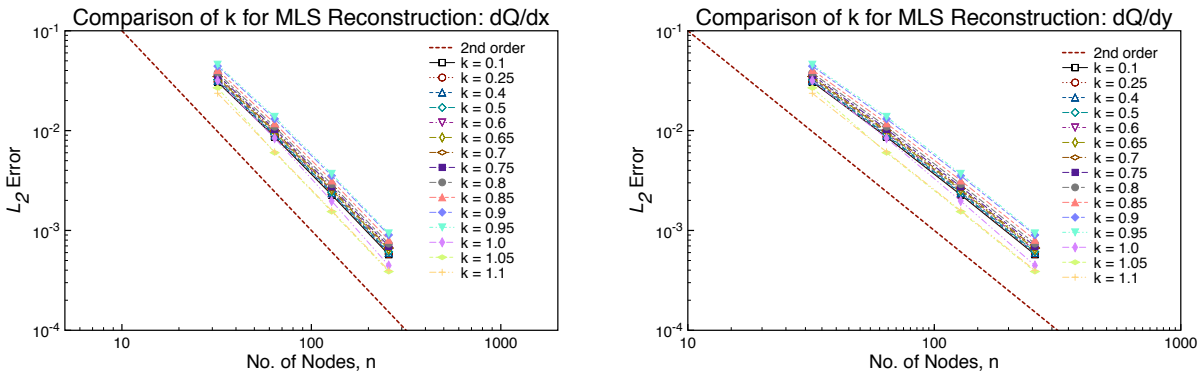


Figure 6.122: Error Norms for State on Stretched Grids: 3^{rd}-Order MLS with Anisotropic Weights Varying k.

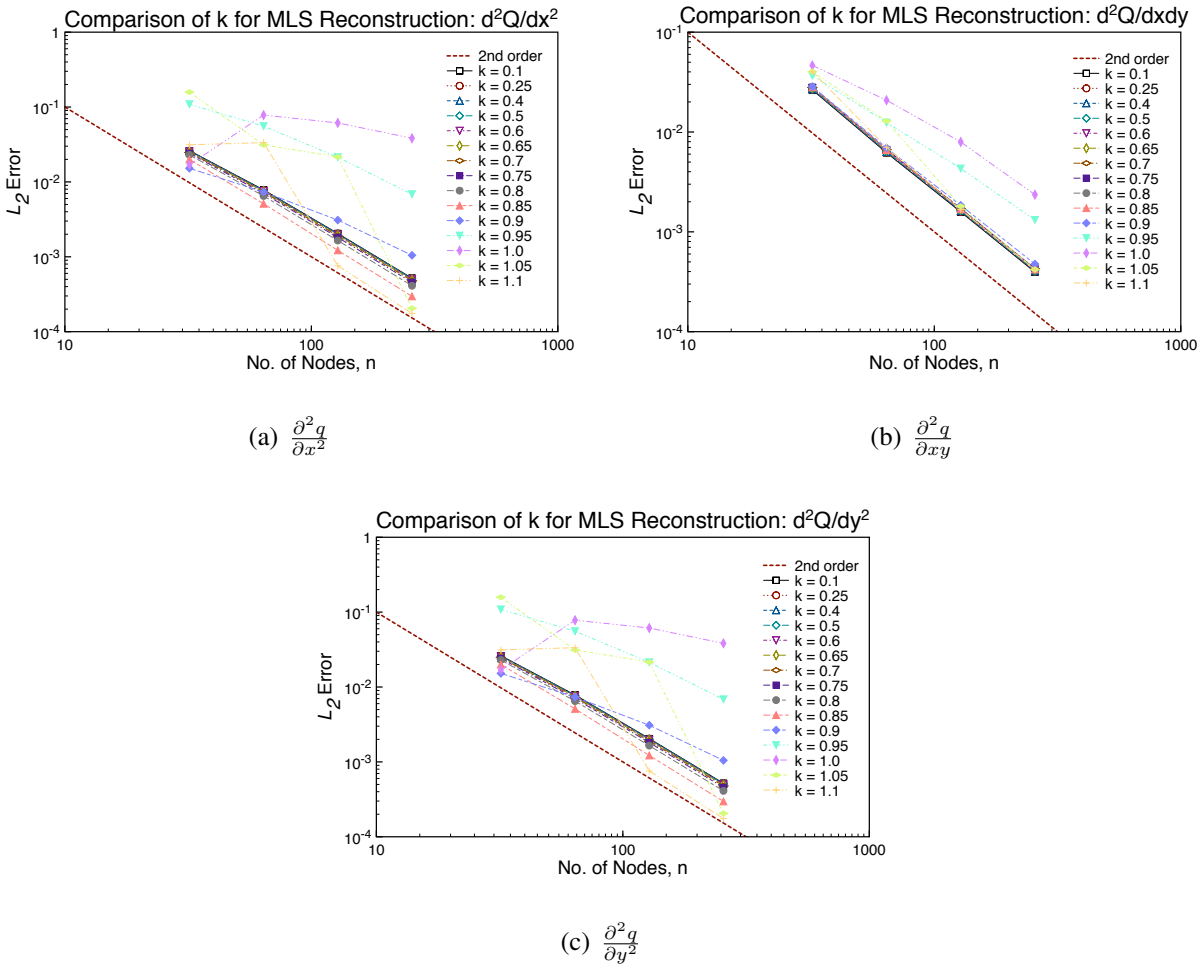
As with the second-order approximation, there is only slight spread in the error as a function of k for the reconstruction of the function. As k increases, the error drops in the reconstruction, with the lowest error for $k = 1.1$. For $k \in [1.0 - 1.1]$, the error is lowest since the 'furthest' stencil nodes are effectively excluded from the approximation. Otherwise, there is no major difference in the error for the reconstruction based on k for the function. The error for the reconstruction of the first derivatives is shown in Figure 6.123



(a) $\frac{\partial q}{\partial x}$ (b) $\frac{\partial q}{\partial y}$

Figure 6.123: Error Norms for First Derivatives on Stretched Grids: 3^{rd}-Order MLS with Anisotropic Weights Varying k.

Again, the lowest error is observed for $k = [1.0 - 1.1]$. However, unlike the second-order reconstruction, the error increases with increasing k for $k = [0.1 - 0.95]$, though there is no significant difference in the error for $k < 0.85$. The error in the second-derivatives for varying k is shown in Figure 6.124



(a) $\frac{\partial^2 q}{\partial x^2}$

(b) $\frac{\partial^2 q}{\partial xy}$

(c) $\frac{\partial^2 q}{\partial y^2}$

Figure 6.124: Error Norms for Second Derivatives on Stretched Grids: 3^{rd}-Order MLS with Anisotropic Weights Varying k.

Unlike with the first-derivatives, the highest error observed occurs when using $k = [0.95 - 1.1]$. The overall accuracy of the reconstruction for each k is presented in Table 6.126.

Table 6.126: Accuracy on Stretched Grids: 3^{rd}-Order MLS with Anisotropic Weights Varying k.

	Function					
k	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
0.1	3.96	1.96	1.96	1.96	1.99	1.96
0.25	3.96	1.96	1.96	1.96	1.99	1.96
0.4	3.96	1.96	1.96	1.96	1.98	1.96
0.5	3.96	1.95	1.95	1.96	1.98	1.96
0.6	3.97	1.95	1.95	1.96	1.98	1.96
0.65	3.97	1.95	1.95	1.96	1.98	1.96
0.7	3.97	1.95	1.95	1.97	1.98	1.97
0.75	3.97	1.95	1.95	1.97	1.98	1.97
0.8	3.98	1.94	1.94	1.99	1.99	1.99
0.85	3.98	1.94	1.94	2.05	1.99	2.05
0.9	3.95	1.93	1.93	1.41	1.93	1.41
0.95	3.90	1.94	1.94	1.51	1.62	1.51
1.0	4.21	2.12	2.12	0.51	1.57	0.51
1.05	4.13	1.98	1.98	3.62	2.47	3.62
1.1	4.01	1.97	1.97	3.79	2.04	3.79

From Table 6.126, the overall accuracy of the function is increases with increasing k, with the average accuracy being roughly fourth-order. For the first derivatives, all methods achieve second-order, so all k would produce similar accuracy as the mesh is refined. The second derivatives show decreasing accuracy as k is increased, with the accuracy becoming questionable for $k < 0.95$. Before determining which k might be best, the condition number of $\mathbf{M}$ should be checked. Table 6.127 shows the condition numbers for each k.

Table 6.127: Average Condition Number of M on Stretched Grids: 3^{rd}-Order MLS with Anisotropic Weights.

	Mesh Size			
k	32	64	128	256
0.1	7252	6554	6019	5702
0.25	1186	1070	982.9	931.2
0.4	481.5	433.8	398.4	377.8
0.5	319.1	287.3	264.1	250.8
0.6	232.2	208.7	192.2	182.9
0.65	203.8	182.9	168.7	160.9
0.7	182.6	163.7	151.1	144.3
0.75	167.7	149.8	138.1	132.1
0.8	158.7	140.2	129.2	124.0
0.85	155.7	135.3	124.4	120.1
0.9	160.5	135.5	124.5	121.3
0.95	174.5	141.1	130.1	129.3
1.0	181.4	142.6	134.8	138.1
1.05	165.7	134.2	133.8	135.6
1.1	152.4	128.9	131.7	124.6

The condition numbers from Table 6.127 show that for increasing k, the condition number decreases. In conjunction with data in Table 6.126, it is determined that choosing $k \in [0.6 - 0.8]$ produces the best overall results for the reconstruction.

6.5.3.2.3 Fourth-Order The error for the reconstruction of the field using a fourth-order MLS reconstruction is shown in Figure 6.125.

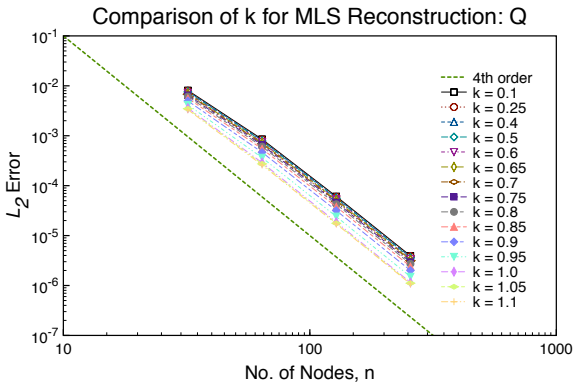
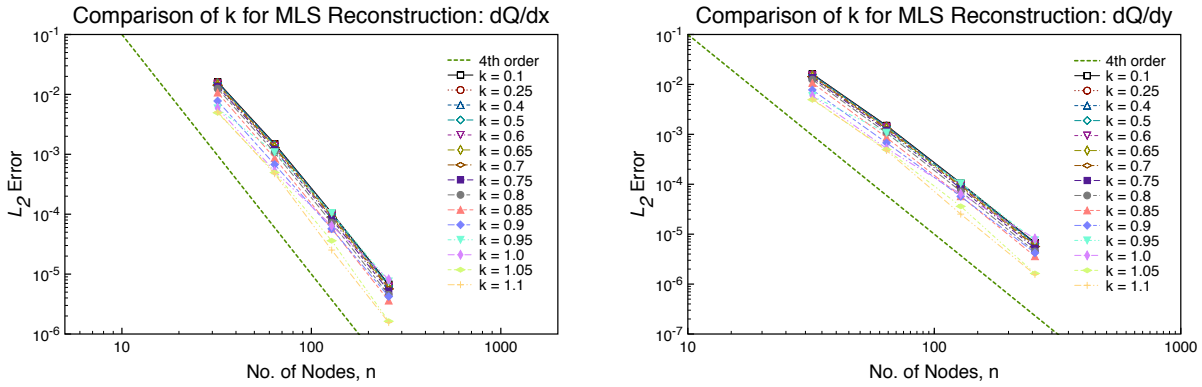


Figure 6.125: Error Norms for State on Stretched Grids: 4^{th}-Order MLS with Anisotropic Weights Varying k.

As with the third-order case, the fourth-order reconstruction has lower error as k is increased, with lowest error observed with $k = 1.1$. Otherwise, there is no major difference in the error for the reconstruction based on k for the function. The error for the reconstruction of the first derivatives is shown in Figure 6.126

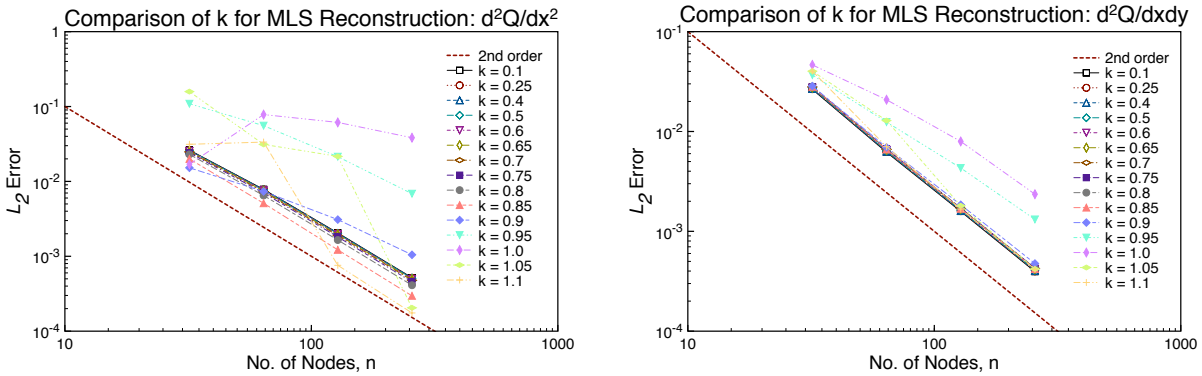


(a) $\frac{\partial q}{\partial x}$ (b) $\frac{\partial q}{\partial y}$

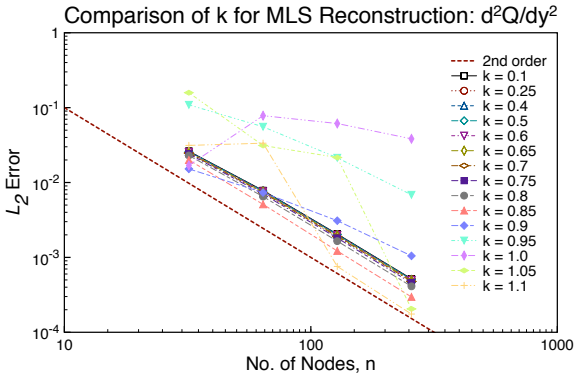
Figure 6.126: Error Norms for First Derivatives on Stretched Grids: 4^{th}-Order MLS with Anisotropic Weights Varying k.

The lowest error is observed for $k = [1.0 - 1.1]$. The error also decreases as k increases, though

there is little variation in the overall error for $k \in [0.1 - 0.85]$. The error in the second-derivatives for varying k is shown in Figure 6.127



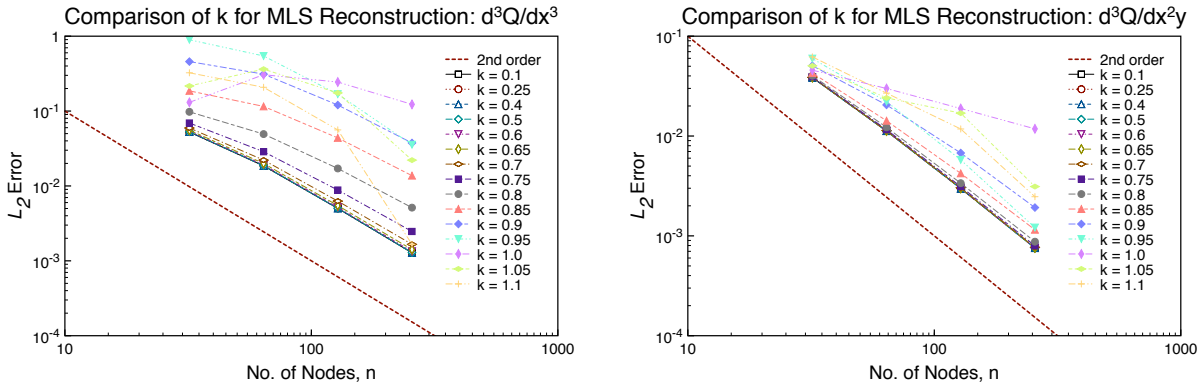
(a) $\frac{\partial^2 q}{\partial x^2}$ (b) $\frac{\partial^2 q}{\partial xy}$



(c) $\frac{\partial^2 q}{\partial y^2}$

Figure 6.127: Error Norms for Second Derivatives on Stretched Grids: 4^{th}-Order MLS with Anisotropic Weights Varying k.

The results in Figure 6.127 show that the highest error observed occurs when using $k = [0.95 - 1.1]$, with the lowest error occurring for $k = 0.85$. The error for the third-derivatives with varying k is shown in Figure 6.128.



(a) $\frac{\partial^3 q}{\partial x^3}$ (b) $\frac{\partial^3 q}{\partial x^2 y}$

(c) $\frac{\partial^3 q}{\partial x y^2}$ (d) $\frac{\partial^3 q}{\partial y^3}$

Figure 6.128: Error Norms for Third Derivatives on Stretched Grids: 4^{th}-Order MLS with Anisotropic Weights Varying k.

Figure 6.128 shows a large spread in the error, with the general trend being increasing error as k increases. The highest error observed occurs when using $k \in [0.85 - 1.1]$. The overall accuracy of the reconstruction for each k is presented in Table 6.126.

Table 6.128: Accuracy on Stretched Grids: 4^{th}-Order MLS with Anisotropic Weights Varying k.

	Function									
k	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial xy^2}$	$\frac{\partial^3 q}{\partial y^3}$
0.1	3.88	3.90	3.90	1.90	1.95	1.90	1.93	1.95	1.95	1.93
0.25	3.88	3.90	3.90	1.90	1.95	1.90	1.93	1.95	1.95	1.93
0.4	3.88	3.90	3.90	1.90	1.95	1.90	1.93	1.95	1.95	1.93
0.5	3.88	3.90	3.90	1.89	1.95	1.89	1.93	1.95	1.95	1.93
0.6	3.89	3.90	3.90	1.89	1.95	1.89	1.92	1.95	1.95	1.92
0.65	3.89	3.91	3.91	1.88	1.95	1.88	1.91	1.94	1.94	1.91
0.7	3.89	3.91	3.91	1.87	1.95	1.87	1.87	1.94	1.94	1.87
0.75	3.90	3.92	3.92	1.86	1.95	1.86	1.77	1.93	1.93	1.77
0.8	3.90	3.94	3.94	1.85	1.94	1.85	1.63	1.90	1.90	1.63
0.85	3.91	3.95	3.95	1.83	1.94	1.83	1.54	1.81	1.81	1.54
0.9	3.93	3.66	3.66	1.81	1.93	1.81	1.53	1.71	1.71	1.53
0.95	3.96	3.58	3.58	1.79	1.92	1.79	1.97	2.09	2.09	1.97
1.0	3.98	3.00	3.00	2.02	1.93	2.02	0.66	0.67	0.67	0.66
1.05	3.97	4.12	4.12	2.24	1.92	2.24	2.02	1.48	1.48	2.02
1.1	3.96	4.13	4.13	1.93	1.89	1.93	3.48	1.72	1.72	3.48

From Table 6.128, the accuracy for the function increases as k increases, with maximum at $k = 1.1$, but on average the order of accuracy is roughly fourth-order. For the first derivatives, the accuracy increases with increasing k up to $k = 0.9$, then decreases, though the maximum accuracy is for $k = [1.05 - 1.1]$. Overall, the accuracy for the first-derivatives is approximately fourth-order. For the second-derivatives, the accuracy is approximately second-order, with the accuracy decreasing for increasing k. For the third-derivatives, the accuracy decreases as k is increased, with the minimum at $k = 1.1$, with a secondary maximum accuracy occurring for $k = 0.8$. The

average error for the third-derivatives is second-order. Table 6.129 shows the condition numbers for each k.

Table 6.129: Average Condition Number of M on Stretched Grids: 4^{th}-Order MLS with Anisotropic Weights Varying k.

	Mesh Size			
k	32	64	128	256
0.1	$9.89E5$	$1.02E6$	$9.61E5$	$8.98E5$
0.25	$6.45E4$	$6.69E4$	$6.25E4$	$5.83E4$
0.4	$1.62E4$	$1.68E4$	$1.57E4$	$1.46E4$
0.5	8557	8854	7885	7355
0.6	5132	5306	4959	4643
0.65	4127	4266	3988	3737
0.7	3395	3506	3278	3077
0.75	2859	2949	2758	2594
0.8	2478	2549	2382	2244
0.85	2238	2285	2125	2007
0.9	2168	2162	1989	1883
0.95	2401	2227	1992	1895
1.0	2794	2291	2000	1952
1.05	2426	2014	1833	1886
1.1	2125	1801	1725	1739

The condition numbers from Table 6.129 show that for increasing k, the condition number decreases. In conjunction with data in Table 6.128, it is determined that choosing $k \in [0.6 - 0.8]$ produces the best overall results for the reconstruction.

6.5.3.3 *Affine MLS*

This subsection presents the results of varying the scaling parameter k with Affine MLS reconstruction on stretched grids using anisotropic weights.

6.5.3.3.1 Second-Order The error for the reconstruction of the field using a second-order MLS reconstruction is shown in Figure 6.129.

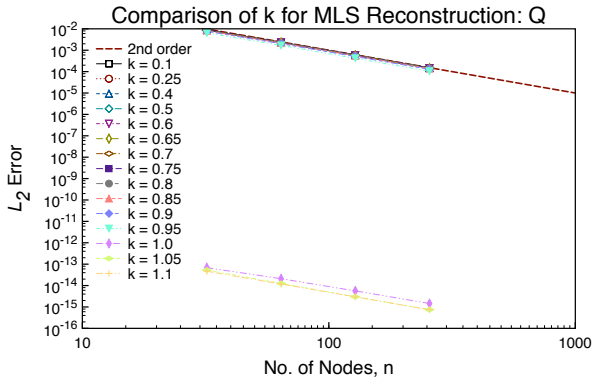
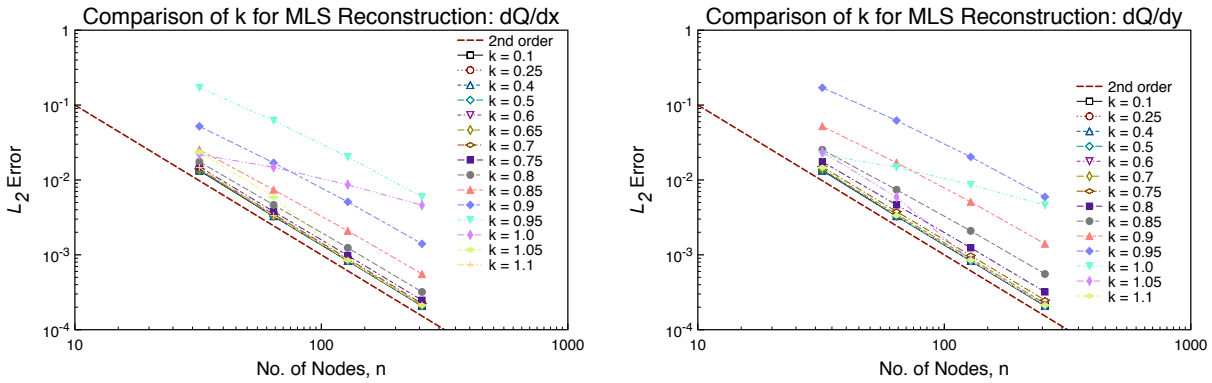


Figure 6.129: Error Norms for State on Stretched Grids: 2^{nd}-Order Affine MLS with Anisotropic Weights Varying k.

From Figure 6.129,there is not much variation for the error of q for $k \in [0.1 - 0.95]$, though the least error occurs when using $k = 0.95$, which pushes the boundary of $\Omega_{\mathbf{x}_I}$ close to the outlying nodes in the stencil. The lowest overall error of q is for $k \in [1.0 - 1.1]$, showing similar results to the structured results shown in Figure 6.84. These values of k perform best, since they effectively exclude the outlying nodes from the reconstruction and use only the central node $\mathbf{x}_I$. Therefore, when choosing the scaling parameter k, some care has to be made to ensure the central node is included in the stencil, otherwise, $k \in [1.0 - 1.1]$ will give incorrect answers. The first derivatives are shown in Figure 6.130.



(a) $\frac{\partial q}{\partial x}$ (b) $\frac{\partial q}{\partial y}$

Figure 6.130: Error Norms for First Derivatives on Stretched Grids: 2^{nd}-Order Affine MLS with Anisotropic Weights Varying k.

Examining both derivatives, there is a broad variation with respect to k. The lowest error appears to be for $k \in [1.05 - 1.1]$, with the larger errors occurring for $k \in [0.9 - 1.0]$. Table 6.130 shows the computed order of accuracy for the varying k.

Table 6.130: Accuracy on Stretched Grids: 2^{nd}-Order Affine MLS with Anisotropic Weights Varying k.

	Function		
k	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
0.1	1.99	1.99	1.99
0.25	1.99	1.99	1.99
0.4	1.99	1.99	1.99
0.5	1.99	1.99	1.99
0.6	1.99	1.99	1.99
0.65	1.99	1.99	1.99
0.7	1.99	1.98	1.98
0.75	1.99	1.97	1.97
0.8	1.99	1.93	1.93
0.85	1.99	1.87	1.87
0.9	1.99	1.80	1.80
0.95	2.00	1.69	1.69
1.0	1.92	0.84	0.84
1.05	2.04	2.40	2.40
1.1	1.99	1.99	1.99

From Table 6.130, the accuracy of q is mostly constant until $k = 0.95$, after which there is high variability in the accuracy occurs. For the derivatives, the error increases with increasing k, with the exception of $k = 1.05$, which has the highest order of any k. The approximate accuracy for the function is second-order, and the accuracy for the first derivatives is also second-order. As with the structured case, the error is lower for the function due to clipping the extreme nodes in the stencil $\Omega_{\mathbf{x}_I}$ for when $k \in [1.0 - 1.1]$. Before determining which k might be best, the condition number of

$\mathbf{M}$ should be checked. Table 6.131 shows the condition numbers for each k.

Table 6.131: Average Condition Number of $\mathbf{M}$ on Stretched Grids: 2^{nd}-Order Affine MLS with Anisotropic Weights Varying k.

	Mesh Size			
k	32	64	128	256
0.1	17.46	16.64	16.23	16.23
0.25	7.06	6.70	6.52	6.44
0.4	4.46	4.22	4.10	4.04
0.5	3.77	3.92	4.03	4.10
0.6	3.01	2.83	2.74	2.70
0.65	2.78	2.62	2.54	2.50
0.7	2.58	2.43	2.36	2.32
0.75	2.40	2.27	2.20	2.17
0.8	2.24	2.12	2.08	2.06
0.85	2.14	2.06	2.02	1.98
0.9	2.19	2.09	2.02	1.97
0.95	2.50	2.34	2.21	2.12
1.0	$2.50E5$	$9.59E4$	$4.30E4$	$1.90E4$
1.05	$3.01E5$	$3.65E5$	$5.76E5$	$6.81E5$
1.1	$4.61E5$	$5.67E5$	$6.58E5$	$6.50E5$

The condition numbers from Table 6.131 show that while the order of accuracy may be 'best' for $k \in [1.0 - 1.1]$, by far those k have the poorest condition number. Generally, the condition number decreases with increasing k for $k < 1.0$. Based upon this data, it would appear better to use $k < 1.0$ for reconstructing the function and its derivatives.

6.5.3.3.2 Third-Order The error for the reconstruction of the field using a third-order Affine MLS reconstruction is shown in Figure 6.131.

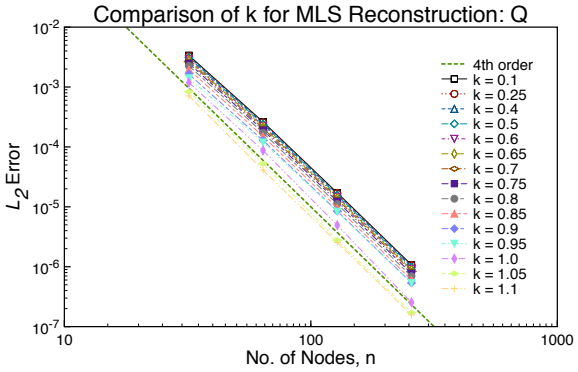
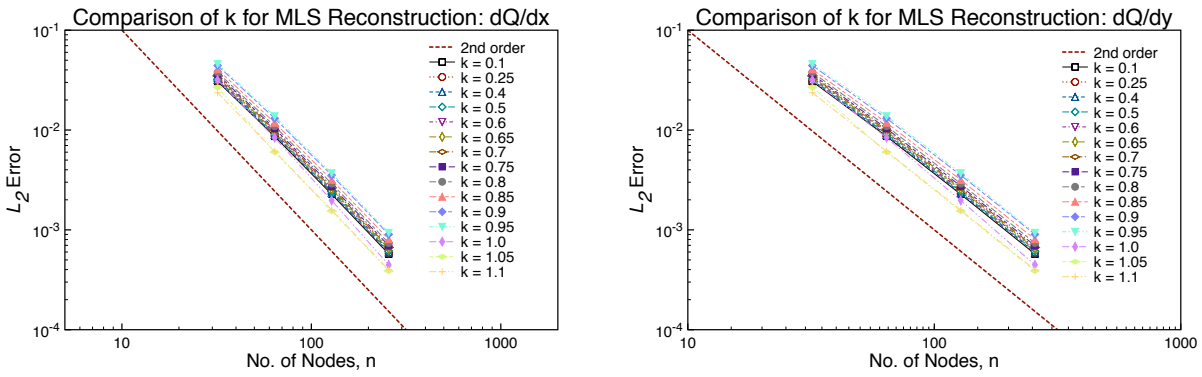


Figure 6.131: Error Norms for State on Stretched Grids: 3^{rd}-Order Affine MLS with Anisotropic Weights Varying k.

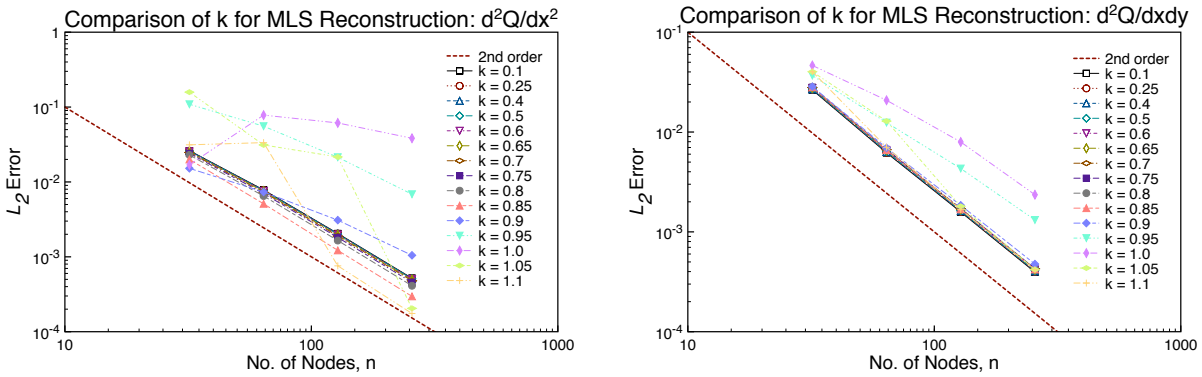
As with the second-order approximation, there is only slight spread in the error as a function of k for the reconstruction of the function. As k increases, the error drops in the reconstruction, with the lowest error for $k = 1.1$. For $k \in [1.0 - 1.1]$, the error is lowest since the 'furthest' stencil nodes are effectively excluded from the approximation. Otherwise, there is no major difference in the error for the reconstruction based on k for the function. The error for the reconstruction of the first derivatives is shown in Figure 6.132



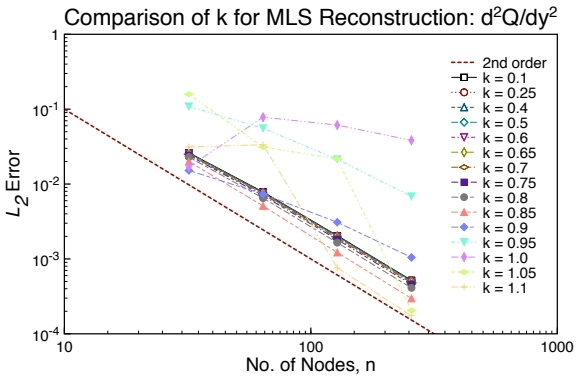
(a) $\frac{\partial q}{\partial x}$ (b) $\frac{\partial q}{\partial y}$

Figure 6.132: Error Norms for First Derivatives on Stretched Grids: 3^{rd}-Order Affine MLS with Anisotropic Weights Varying k.

Again, the lowest error is observed for $k = [1.0 - 1.1]$. However, unlike the second-order reconstruction, the error increases with increasing k for $k = [0.1 - 0.95]$, though there is no significant difference in the error for $k < 0.85$. The error in the second-derivatives for varying k is shown in Figure 6.133



(a) $\frac{\partial^2 q}{\partial x^2}$ (b) $\frac{\partial^2 q}{\partial xy}$



(c) $\frac{\partial^2 q}{\partial y^2}$

Figure 6.133: Error Norms for Second Derivatives on Stretched Grids: 3rd-Order Affine MLS with Anisotropic Weights Varying k.

Unlike with the first-derivatives, the highest error observed occurs when using $k = [0.95 - 1.1]$. The overall accuracy of the reconstruction for each k is presented in Table 6.132.

Table 6.132: Accuracy on Stretched Grids: 3^{rd}-Order Affine MLS with Anisotropic Weights Varying k.

	Function					
k	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
0.1	3.96	1.96	1.96	1.96	1.99	1.96
0.25	3.96	1.96	1.96	1.96	1.99	1.96
0.4	3.96	1.96	1.96	1.96	1.98	1.96
0.5	3.96	1.95	1.95	1.96	1.98	1.96
0.6	3.97	1.95	1.95	1.96	1.98	1.96
0.65	3.97	1.95	1.95	1.96	1.98	1.96
0.7	3.97	1.95	1.95	1.97	1.98	1.97
0.75	3.97	1.95	1.95	1.97	1.98	1.97
0.8	3.98	1.94	1.94	1.99	1.99	1.99
0.85	3.98	1.94	1.94	2.05	1.99	2.05
0.9	3.95	1.93	1.93	1.41	1.93	1.41
0.95	3.90	1.94	1.94	1.51	1.62	1.51
1.0	4.21	2.12	2.12	0.51	1.57	0.51
1.05	4.13	1.98	1.98	3.62	2.47	3.62
1.1	4.01	1.97	1.97	3.79	2.04	3.79

From Table 6.132, the overall accuracy of the function is increases with increasing k, with the average accuracy being roughly fourth-order. For the first derivatives, all methods achieve second-order, so all k would produce similar accuracy as the mesh is refined. The second derivatives show decreasing accuracy as k is increased, with the accuracy becoming questionable for $k < 0.95$. Before determining which k might be best, the condition number of $\mathbf{M}$ should be checked. Table 6.133 shows the condition numbers for each k.

Table 6.133: Average Condition Number of $\mathbf{M}$ on Stretched Grids: 3^{rd}-Order Affine MLS with Anisotropic Weights Varying k.

	Mesh Size			
k	32	64	128	256
0.1	995.0	850.8	785.6	754.7
0.25	163.8	139.8	129.1	123.9
0.4	66.46	56.86	52.55	50.52
0.5	43.69	37.50	34.73	33.44
0.6	31.08	26.08	24.91	24.02
0.65	26.75	23.13	21.54	20.81
0.7	23.26	20.24	18.91	18.24
0.75	20.45	17.96	16.75	16.14
0.8	18.30	16.07	14.95	14.37
0.85	16.53	14.48	13.42	12.87
0.9	15.05	13.13	12.00	11.05
0.95	13.86	12.00	11.05	10.78
1.0	13.28	11.40	10.81	10.84
1.05	12.39	10.70	10.41	10.26
1.1	11.62	10.16	9.85	9.64

The condition numbers from Table 6.133 show that for increasing k, the condition number decreases. In conjunction with data in Table 6.132, it is determined that choosing $k \in [0.6 - 0.8]$ produces the best overall results for the reconstruction.

6.5.3.3.3 Fourth-Order The error for the reconstruction of the field using a fourth-order Affine MLS reconstruction is shown in Figure 6.134.

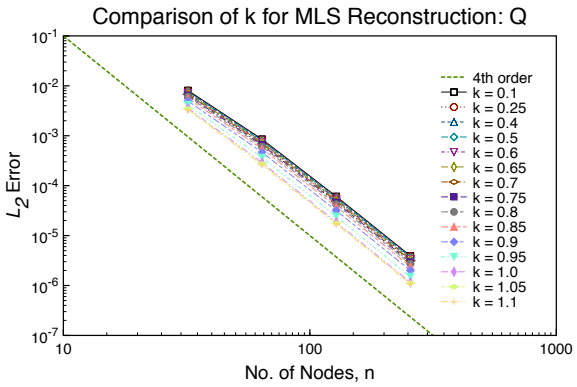
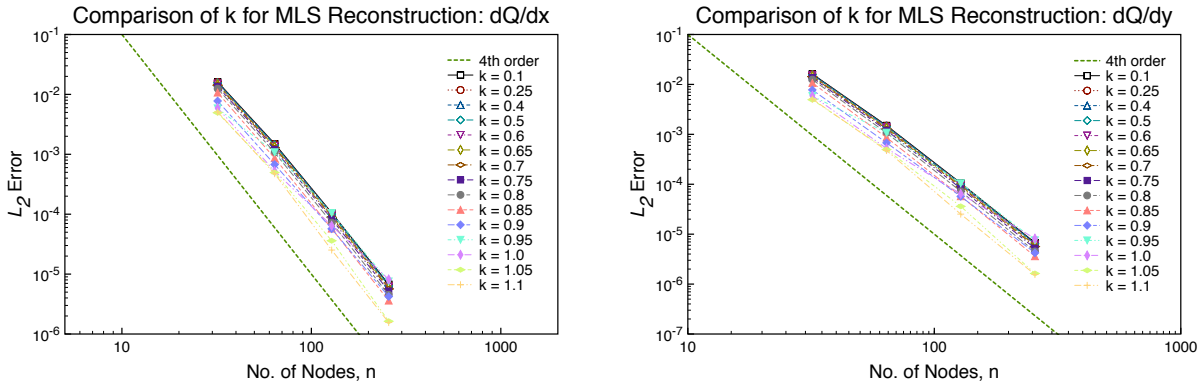


Figure 6.134: Error Norms for State on Stretched Grids: 4^{th}-Order Affine MLS with Anisotropic Weights Varying k.

As with the third-order case, the fourth-order reconstruction has lower error as k is increased, with lowest error observed with $k = 1.1$. Otherwise, there is no major difference in the error for the reconstruction based on k for the function. The error for the reconstruction of the first derivatives is shown in Figure 6.135

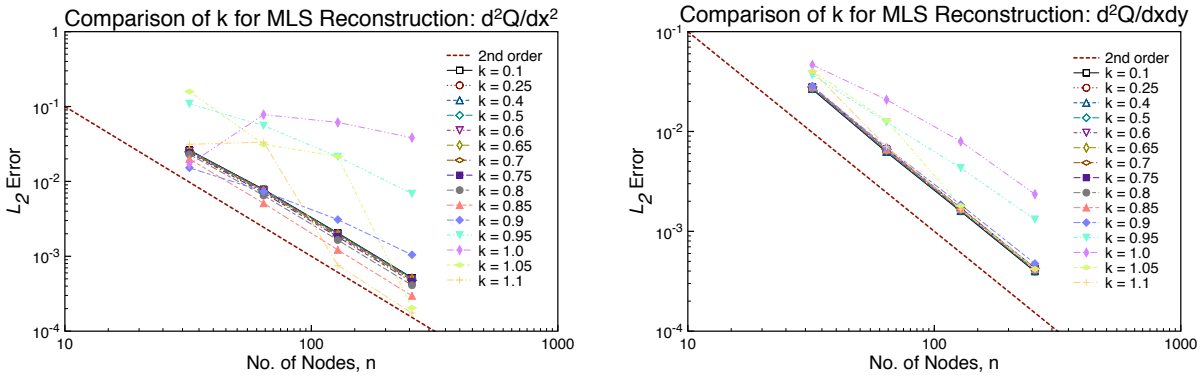


(a) $\frac{\partial q}{\partial x}$ (b) $\frac{\partial q}{\partial y}$

Figure 6.135: Error Norms for First Derivatives on Stretched Grids: 4^{th}-Order Affine MLS with Anisotropic Weights Varying k.

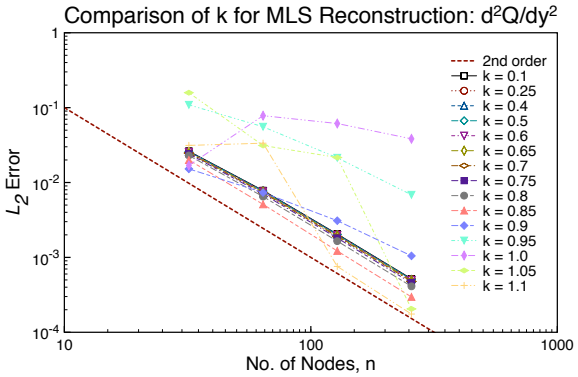
The lowest error is observed for $k = [1.0 - 1.1]$. The error also decreases as k increases, though

there is little variation in the overall error for $k \in [0.1 - 0.85]$. The error in the second-derivatives for varying k is shown in Figure 6.136



(a) $\frac{\partial^2 q}{\partial x^2}$

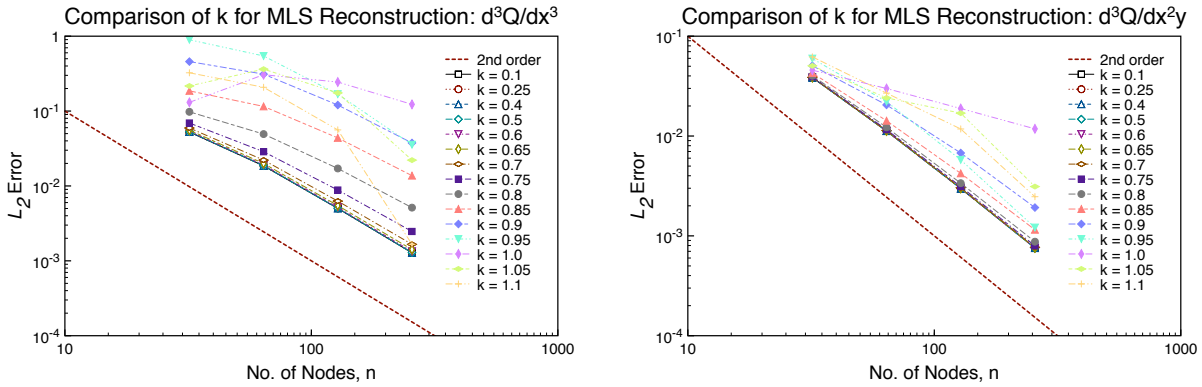
(b) $\frac{\partial^2 q}{\partial xy}$



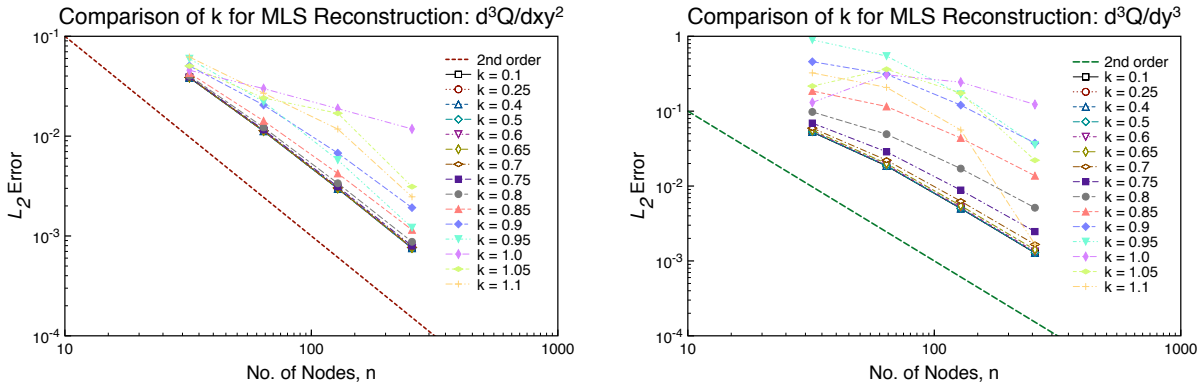
(c) $\frac{\partial^2 q}{\partial y^2}$

Figure 6.136: Error Norms for Second Derivatives on Stretched Grids: 4^{th}-Order Affine MLS with Anisotropic Weights Varying k.

The results in Figure 6.127 show that the highest error observed occurs when using $k = [0.95 - 1.1]$, with the lowest error occurring for $k = 0.85$. The error for the third-derivatives with varying k is shown in Figure 6.137.



(a) $\frac{\partial^3 q}{\partial x^3}$ (b) $\frac{\partial^3 q}{\partial x^2 y}$



(c) $\frac{\partial^3 q}{\partial x y^2}$ (d) $\frac{\partial^3 q}{\partial y^3}$

Figure 6.137: Error Norms for Third Derivatives on Stretched Grids: 4^{th}-Order Affine MLS with Anisotropic Weights Varying k.

Figure 6.137 shows a large spread in the error, with the general trend being increasing error as k increases. The highest error observed occurs when using $k \in [0.85 - 1.1]$. The overall accuracy of the reconstruction for each k is presented in Table 6.132.

Table 6.134: Accuracy on Stretched Grids: 4^{th}-Order Affine MLS with Anisotropic Weights Varying k.

	Function									
k	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial xy^2}$	$\frac{\partial^3 q}{\partial y^3}$
0.1	3.88	3.90	3.90	1.90	1.95	1.90	1.93	1.95	1.95	1.93
0.25	3.88	3.90	3.90	1.90	1.95	1.90	1.93	1.95	1.95	1.93
0.4	3.88	3.90	3.90	1.90	1.95	1.90	1.93	1.95	1.95	1.93
0.5	3.88	3.90	3.90	1.89	1.95	1.89	1.93	1.95	1.95	1.93
0.6	3.89	3.90	3.90	1.89	1.95	1.89	1.92	1.95	1.95	1.92
0.65	3.89	3.91	3.91	1.88	1.95	1.88	1.91	1.94	1.94	1.91
0.7	3.89	3.91	3.91	1.87	1.95	1.87	1.87	1.94	1.94	1.87
0.75	3.90	3.92	3.92	1.86	1.95	1.86	1.77	1.93	1.93	1.77
0.8	3.90	3.94	3.94	1.85	1.94	1.85	1.63	1.90	1.90	1.63
0.85	3.91	3.95	3.95	1.83	1.94	1.83	1.54	1.81	1.81	1.54
0.9	3.93	3.66	3.66	1.81	1.93	1.81	1.53	1.71	1.71	1.53
0.95	3.96	3.58	3.58	1.79	1.92	1.79	1.97	2.09	2.09	1.97
1.0	3.98	3.00	3.00	2.02	1.93	2.02	0.66	0.67	0.67	0.66
1.05	3.97	4.12	4.12	2.24	1.92	2.24	2.02	1.48	1.48	2.02
1.1	3.96	4.13	4.13	1.93	1.89	1.93	3.48	1.72	1.72	3.48

From Table 6.134, the accuracy for the function increases as k increases, with maximum at $k = 1.1$, but on average the order of accuracy is roughly fourth-order. For the first derivatives, the accuracy increases with increasing k up to $k = 0.9$, then decreases, though the maximum accuracy is for $k = [1.05 - 1.1]$. Overall, the accuracy for the first-derivatives is approximately fourth-order. For the second-derivatives, the accuracy is approximately second-order, with the accuracy decreasing for increasing k. For the third-derivatives, the accuracy decreases as k is increased,

with the minimum at $k = 1.1$, with a secondary maximum accuracy occurring for $k = 0.8$. The average error for the third-derivatives is second-order. Table 6.135 shows the condition numbers for each k.

Table 6.135: Average Condition Number of $\mathbf{M}$ on Stretched Grids: 4^{th}-Order Affine MLS with Anisotropic Weights Varying k.

	Mesh Size			
k	32	64	128	256
0.1	$4.52E4$	$3.22E4$	$2.69E4$	$2.44E4$
0.25	2979	2115	1763	1605
0.4	759.8	537.4	448.3	408.1
0.5	399.4	284.0	237.1	216.1
0.6	238.3	169.8	142.1	129.7
0.65	190.1	135.7	113.7	103.9
0.7	154.3	110.3	92.58	84.68
0.75	126.9	90.95	76.51	70.17
0.8	105.7	76.01	64.33	58.95
0.85	89.04	64.67	54.71	50.10
0.9	76.04	55.73	47.13	43.15
0.95	66.38	48.73	41.21	37.62
1.0	60.37	43.21	36.38	33.20
1.05	53.42	38.47	32.37	29.69
1.1	47.76	34.69	29.26	27.16

The condition numbers from Table 6.135 show that for increasing k, the condition number decreases. In conjunction with data in Table 6.134, it is determined that choosing $k \in [0.6 - 0.8]$

produces the best overall results for the reconstruction.

6.6 Comparison of Orthogonal Polynomials using Affine MLS

This section presents the results obtained by varying the polynomial basis used for the Affine MLS reconstruction. Each of the grids are examined using the Gaussian function. The bases used in the study are:

- Legendre
- Hermite
- Chebyshev of the First Kind
- Chebyshev of the Second Kind
- Zernike
- Gegenbauer

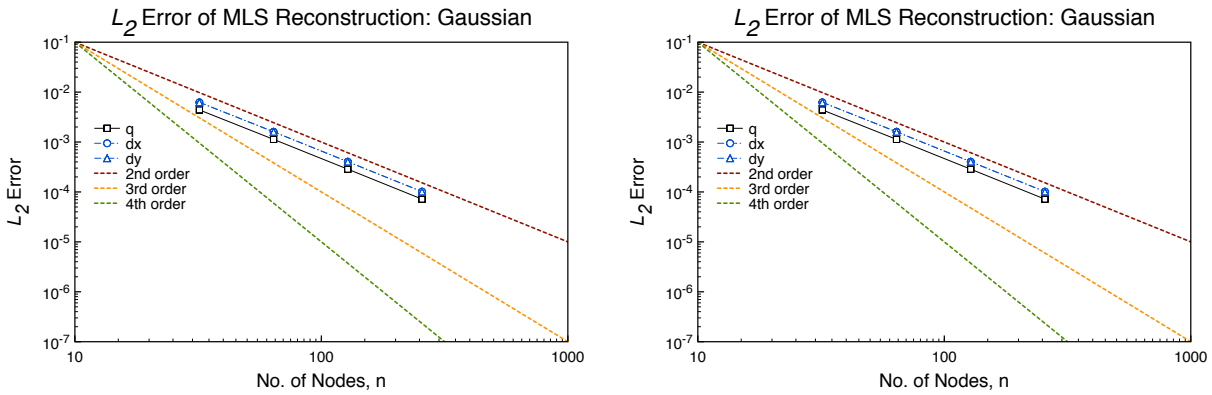
and their forms can be found in Abramowitz & Stegun [1], in addition to the references given in Section 5.1.3.3. The scaling parameter is set to $k = 0.8$. The inversion method of choice for this study is the Pivoting QR method, since it always produced the minimum condition number for $\mathbf{M}$ for each of the grids, as shown in Section 6.2.

6.6.1 Structured Grids

This section presents the orthogonal basis studies on the structured grid.

6.6.1.1 Second-Order

For second-order reconstructions, only the Chebyshev of the Second Kind and Gegenbauer polynomials differ from the basic monomial polynomial. The remaining bases will not have a different reconstruction from the monomial. Figure 6.138 shows the order of accuracy results for both the Chebyshev of the Second Kind and Gegenbauer polynomials.



(a) Chebyshev, 2nd Kind

(b) Gegenbauer

Figure 6.138: Error Norms on Structured Grids: 2nd-Order Affine MLS with Isotropic Weights Using Orthogonal Bases.

From Figure 6.138, there is no noticeable difference between the two bases and the monomial polynomial in Figure 6.18. Table 6.136 shows the computed order of accuracy for each basis. As

Table 6.136: Accuracy on Structured Grids: 2nd-Order Affine MLS with Isotropic Weights Using Orthogonal Bases.

Basis	Function		
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
Monomial	1.98	1.98	1.98
Chebyshev, 2nd Kind	1.98	1.98	1.98
Gegenbauer	1.98	1.98	1.98

expected, changing the basis does not affect the order of accuracy of the method. The function and the first-derivative both are second-order accurate. It is also important to look at the average condition number of the moment matrix $\mathbf{M}$. Table 6.137 shows the average condition number for each level of refinement of the various bases, including the general MLS reconstruction with both weight regimes. A few observations can be made from Table 6.137. First, as the mesh is refined, the

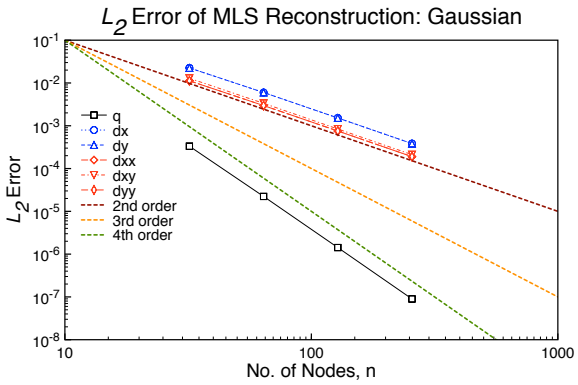
Table 6.137: Average Condition Number of $\mathbf{M}$ on Structured Grids: 2^{nd}-Order MLS with Different Bases & Weights.

Basis	Mesh Size			
	32	64	128	256
Monomial, Isotropic	2.04	2.04	2.04	2.04
Monomial, Anisotropic	2.04	2.04	2.04	2.04
Monomial, Affine	2.04	2.04	2.04	2.04
Chebyshev, 2^{nd} Kind	1.02	1.02	1.02	1.02
Gegenbauer	1.97	1.97	1.97	1.97

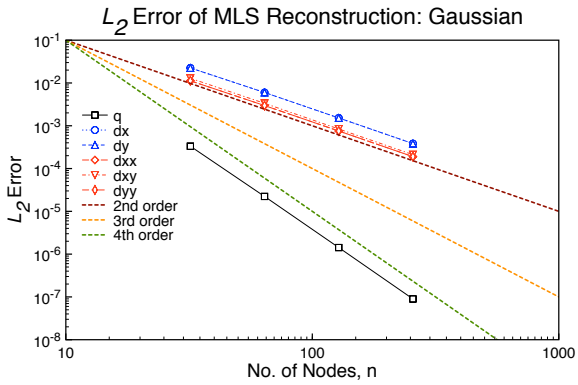
condition number of moment matrix $\mathbf{M}$ tends to decrease, The worst condition system is on using the monomial basis, which was to be expected. The best conditioned system for the structured grid is Affine MLS case using the Chebyshev of the Second Kind basis, with the Gegenbauer basis being the second best. Overall, though, the condition number of the moment matrix $\mathbf{M}$ is very good, with the largest condition number only just over. Therefore, any of these bases could be used to produce accurate reconstructions.

6.6.1.2 Third-Order

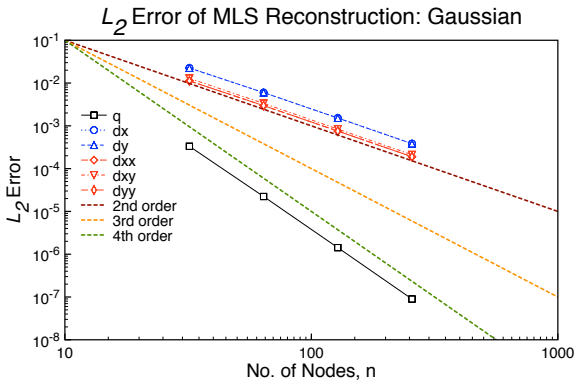
For third-order reconstructions, all of the bases differ from the general monomial polynomial. Figure 6.139 shows the order of accuracy results for the orthogonal bases.



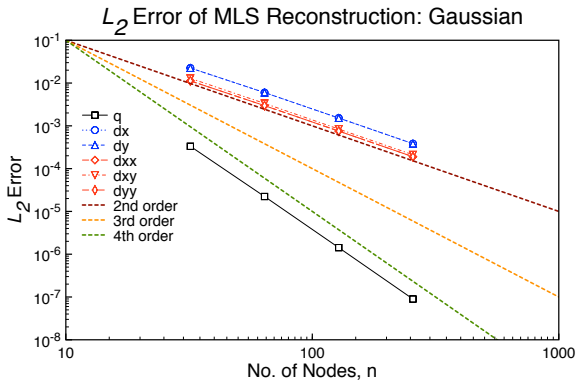
(a) Legendre



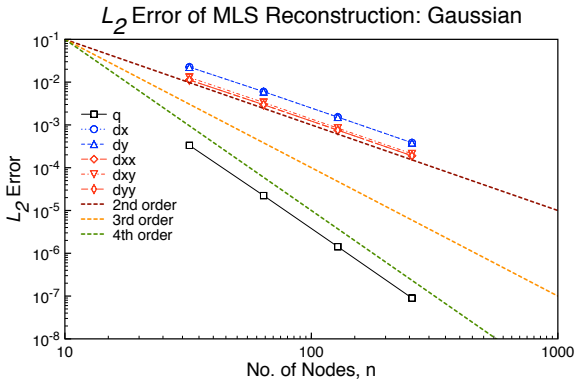
(b) Hermite



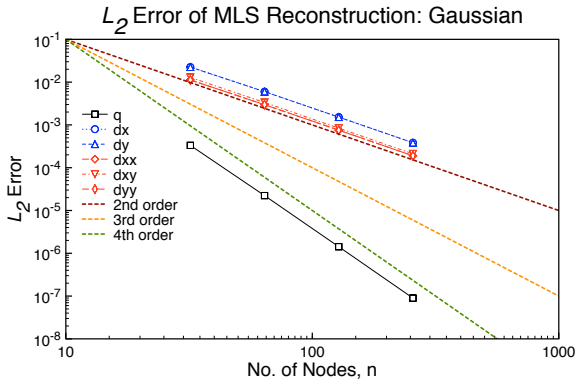
(c) Chebyshev, 1st Kind



(d) Chebyshev, 2nd Kind



(e) Zernike



(f) Gegenbauer

Figure 6.139: Error Norms on Structured Grids: 3rd-Order Affine MLS with Isotropic Weights Using Orthogonal Bases.

As with the second-order reconstruction, there is not a discernible difference between any of the bases and the monomial polynomial in Figure 6.19. Table 6.138 shows the computed order of accuracy for each basis. As expected, changing the basis does not affect the order of accuracy of

Table 6.138: Accuracy on Structured Grids: 3^{rd}-Order Affine MLS with Isotropic Weights Using Orthogonal Bases.

Basis	Function					
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
Monomial	3.95	1.96	1.96	1.96	1.98	1.96
Legendre	3.95	1.96	1.96	1.96	1.98	1.96
Hermite	3.95	1.96	1.96	1.96	1.98	1.96
Chebyshev, 1^{st} Kind	3.95	1.96	1.96	1.96	1.98	1.96
Chebyshev, 2^{nd} Kind	3.95	1.96	1.96	1.96	1.98	1.96
Zernike	3.95	1.96	1.96	1.96	1.98	1.96
Gegenbauer	3.95	1.96	1.96	1.96	1.98	1.96

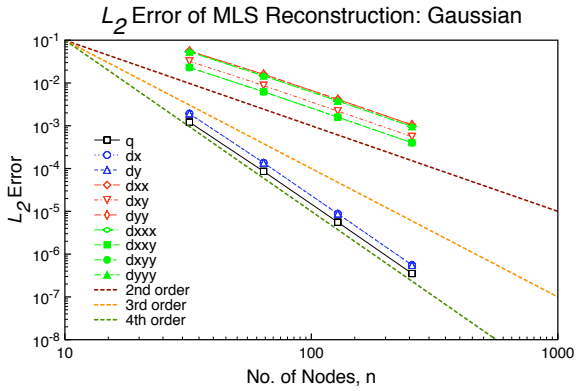
the method. The function is determined to be fourth-order accurate, while the first- and second-derivatives are just below second-order. As with the second-order, the condition number of the system should be inspected to see how the basis change affects the moment matrix $\mathbf{M}$. Table 6.139 shows the average condition number for each level of refinement of the various bases, including the general MLS reconstruction with both weight regimes. Table 6.139 shows some interesting results. As with the second-order reconstruction, the condition number of $\mathbf{M}$ tends to decrease as the mesh is refined. The monomial basis has the median condition number of the system, with some of the orthogonal bases performing worse: Legendre, Hermite, and Chebyshev of the First Kind. The remaining bases have a much better conditioned system, with the 'best' basis being the Gegenbauer, with a condition number just below three. The 'best' bases are in the class of spherical bases, so it makes sense that these (the Chebyshev of the Second Kind, Zernike, and Gegenbauer) perform best for this test. However, all of the bases produce accurate reconstructions, and even using the monomial basis is more than acceptable.

Table 6.139: Average Condition Number of M on Structured Grids: 3^{rd}-Order MLS with Different Bases & Weights.

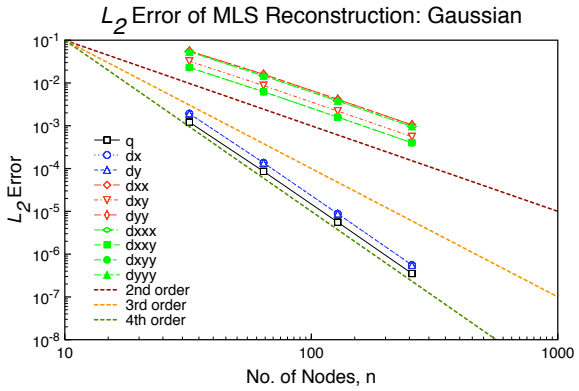
Basis	Mesh Size			
	32	64	128	256
Monomial, Isotropic	13.82	13.80	13.79	13.79
Monomial, Anisotropic	13.82	13.80	13.79	13.79
Monomial, Affine	13.82	13.80	13.79	13.79
Legendre	18.04	18.01	18.01	18.00
Hermite	40.72	40.66	40.64	40.64
Chebyshev, 1^{st} Kind	27.74	27.70	27.69	27.69
Chebyshev, 2^{nd} Kind	6.03	6.01	6.00	6.00
Zernike	4.29	4.28	4.27	4.27
Gegenbauer	2.86	2.85	2.85	2.85

6.6.1.3 Fourth-Order

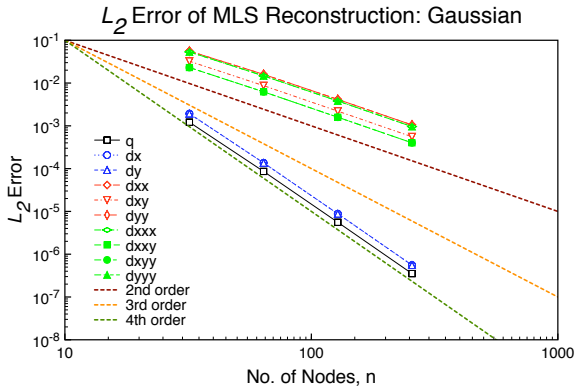
For third-order reconstructions, all of the bases differ from the general monomial polynomial. Figure 6.140 shows the order of accuracy results for the orthogonal bases.



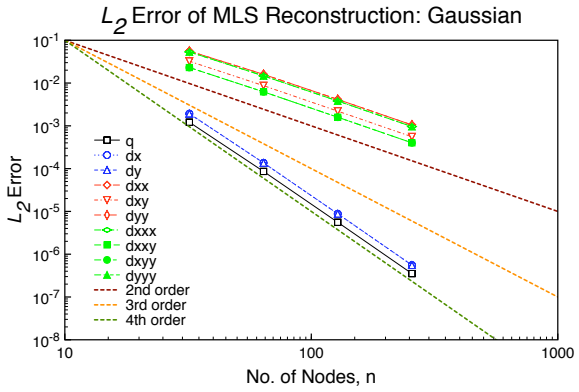
(a) Legendre



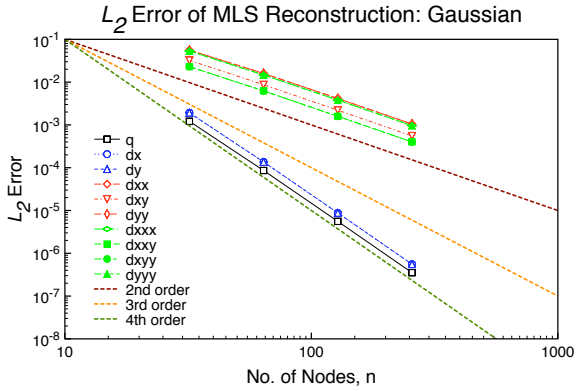
(b) Hermite



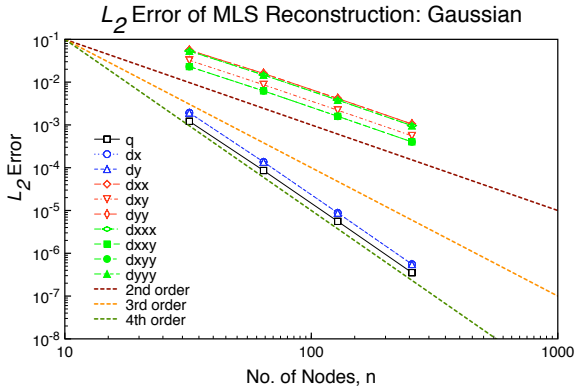
(c) Chebyshev, 1st Kind



(d) Chebyshev, 2nd Kind



(e) Zernike



(f) Gegenbauer

Figure 6.140: Error Norms on Structured Grids: 4th-Order Affine MLS with Isotropic Weights Using Orthogonal Bases.

As with the second-order reconstruction, there is not a discernible difference between any of the bases and the monomial polynomial in Figure 6.20. Table 6.140 shows the computed order of accuracy for each basis. As expected, changing the basis does not affect the order of accuracy of

Table 6.140: Accuracy on Structured Grids: 4^{th}-Order Affine MLS with Isotropic Weights Using Orthogonal Bases.

Basis	Function									
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial xy^2}$	$\frac{\partial^3 q}{\partial y^3}$
Monomial	3.92	3.92	3.92	1.91	1.95	1.91	1.93	1.95	1.95	1.93
Legendre	3.92	3.92	3.92	1.91	1.95	1.91	1.93	1.95	1.95	1.93
Hermite	3.92	3.92	3.92	1.91	1.95	1.91	1.93	1.95	1.95	1.93
Chebyshev, 1^{st} Kind	3.92	3.92	3.92	1.91	1.95	1.91	1.93	1.95	1.95	1.93
Chebyshev, 2^{nd} Kind	3.92	3.92	3.92	1.91	1.95	1.91	1.93	1.95	1.95	1.93
Zernike	3.92	3.92	3.92	1.91	1.95	1.91	1.93	1.95	1.95	1.93
Gegenbauer	3.92	3.92	3.92	1.91	1.95	1.91	1.93	1.95	1.95	1.93

the method. The function and first-derivatives are determined to be fourth-order accurate, while the second- and third-derivatives are just below second-order. As with the second-order, the condition number of the system should be inspected to see how the basis change affects the moment matrix $\mathbf{M}$. Table 6.141 shows the average condition number for each level of refinement of the various bases, including the general isotropic and anisotropic weighed MLS. Table 6.141 shows that the best performing bases are the 'spherical' bases. As with the third-order case, the condition number of $\mathbf{M}$ tends to decrease as the mesh is refined. The monomial basis has the median condition number of the system, with some of the orthogonal bases performing worse: Hermite and Chebyshev of the First Kind. The remaining bases have a much better conditioned system, with the 'best' basis being the Gegenbauer, with a condition number just above five. Unlike the third-order reconstruction, the second-best basis is the Chebyshev of the Second Kind. The worst basis by far is the Hermite basis. Overall, any basis besides the Hermite would give accurate results, though it would be best to use one of the 'spherical' bases when performing the reconstruction.

Table 6.141: Average Condition Number of $\mathbf{M}$ on Structured Grids: 4^{th}-Order MLS with Different Bases & Weights.

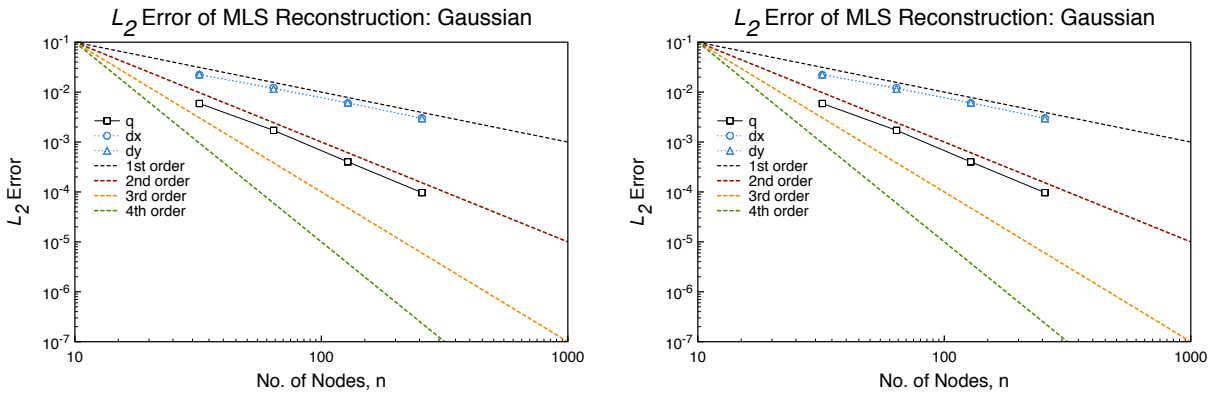
Basis	Mesh Size			
	32	64	128	256
Monomial, Isotropic	54.17	53.95	53.90	53.88
Monomial, Anisotropic	54.17	53.95	53.90	53.88
Monomial, Affine	54.17	53.95	53.90	53.88
Legendre	48.35	48.13	48.07	48.05
Hermite	219.0	218.0	217.7	217.6
Chebyshev, 1^{st} Kind	70.49	70.10	70.00	69.98
Chebyshev, 2^{nd} Kind	14.97	14.86	14.83	14.83
Zernike	20.71	20.50	20.45	20.43
Gegenbauer	5.29	5.23	5.21	5.21

6.6.2 Unstructured Grids

This subsection presents the results obtained by varying the polynomial basis used for the Affine MLS reconstruction on unstructured grids.

6.6.2.1 Second-Order

For second-order reconstructions, only the Chebyshev of the Second Kind and Gegenbauer polynomials differ from the basic monomial polynomial. The remaining bases will not have a different reconstruction from the monomial. Figure 6.141 shows the order of accuracy results for both the Chebyshev of the Second Kind and Gegenbauer polynomials.



(a) Chebyshev, 2nd Kind

(b) Gegenbauer

Figure 6.141: Error Norms on Unstructured Grids: 2nd-Order Affine MLS with Anisotropic Weights Using Orthogonal Bases.

From Figure 6.141, there is no noticeable difference between the two bases and the monomial polynomial in Figure 6.27. Table 6.142 shows the computed order of accuracy for each basis. As

Table 6.142: Accuracy on Unstructured Grids: 2nd-Order Affine MLS with Anisotropic Weights Using Orthogonal Bases.

Basis	Function		
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
Monomial	2.00	0.94	0.92
Chebyshev, 2nd Kind	2.00	0.94	0.92
Gegenbauer	2.00	0.94	0.92

expected, changing the basis does not affect the order of accuracy of the method. The function is second-order accurate and the first-derivative both are first-order accurate. It is also important to look at the average condition number of the moment matrix $\mathbf{M}$. Table 6.143 shows the average condition number for each level of refinement of the various bases, including the general MLS reconstruction with both weight regimes. A few observations can be made from Table 6.137.

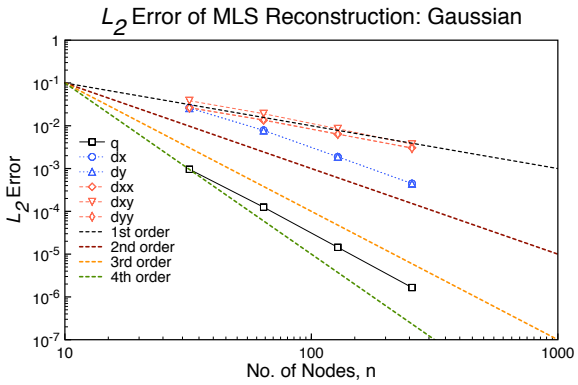
Table 6.143: Average Condition Number of $\mathbf{M}$ on Unstructured Grids: 2^{nd}-Order MLS with Different Bases & Weights.

Basis	Mesh Size			
	32	64	128	256
Monomial, Isotropic	2.34	2.35	2.36	2.37
Monomial, Anisotropic	2.42	2.43	2.46	2.47
Monomial, Affine	2.24	2.25	2.26	2.27
Chebyshev, 2^{nd} Kind	1.19	1.20	1.22	1.22
Gegenbauer	2.13	2.15	2.16	2.16

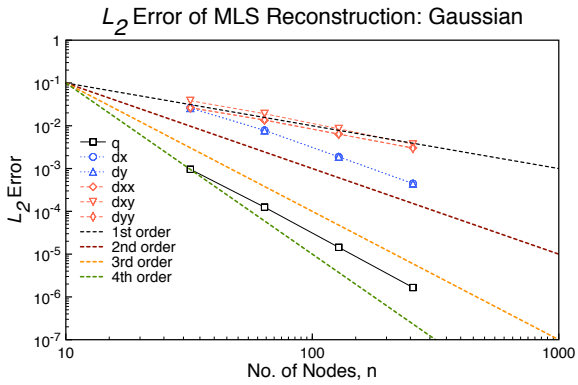
First, as the mesh is refined, the condition number of moment matrix $\mathbf{M}$ tends to increase. This is probably due to the change in stencil size. The worst condition system is on using the monomial basis with anisotropic weighting. The best conditioned system for the structured grid is Affine MLS case using the Chebyshev of the Second Kind basis, with the Gegenbauer basis being the second best. Overall, though, the condition number of the moment matrix $\mathbf{M}$ is very good, with the largest condition number only just over two. Therefore, any of these bases could be used to produce accurate reconstructions.

6.6.2.2 Third-Order

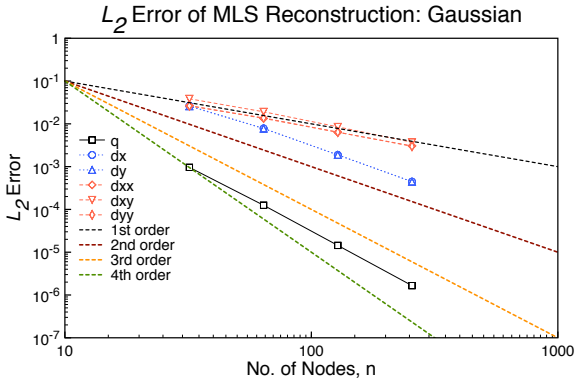
For third-order reconstructions, all of the bases differ from the general monomial polynomial. Figure 6.142 shows the order of accuracy results for the orthogonal bases.



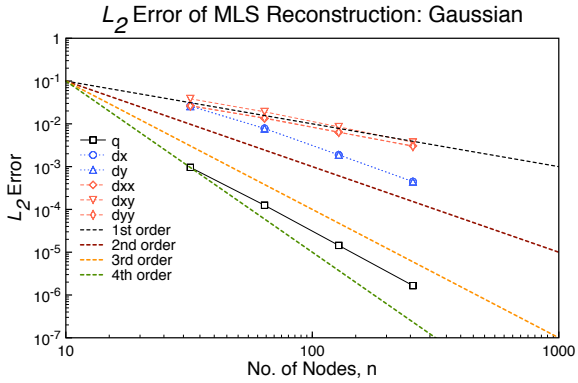
(a) Legendre



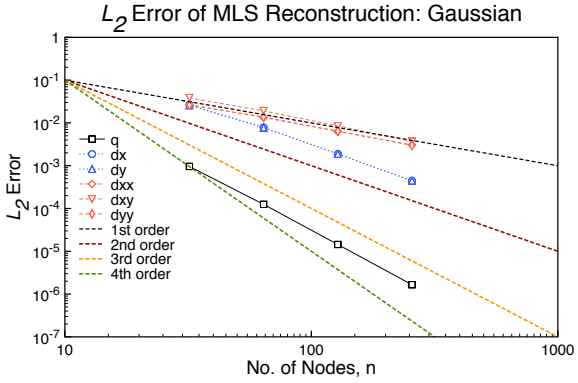
(b) Hermite



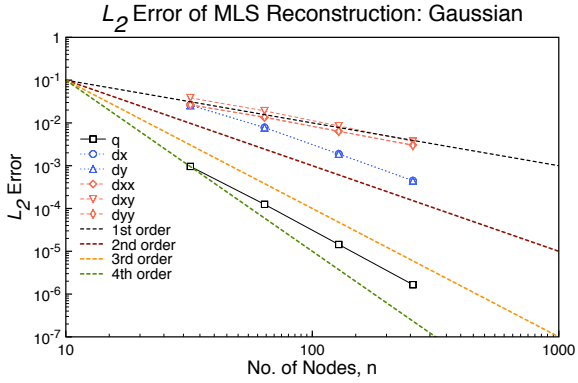
(c) Chebyshev, 1st Kind



(d) Chebyshev, 2nd Kind



(e) Zernike



(f) Gegenbauer

Figure 6.142: Error Norms on Unstructured Grids: 3rd-Order Affine MLS with Anisotropic Weights Using Orthogonal Bases.

As with the second-order reconstruction, there is not a discernible difference between any of the bases and the monomial polynomial in Figure 6.28. Table 6.144 shows the computed order of accuracy for each basis. As expected, changing the basis does not affect the order of accuracy of

Table 6.144: Accuracy on Unstructured Grids: 3^{rd}-Order Affine MLS with Anisotropic Weights Using Orthogonal Bases.

Basis	Function					
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
Monomial	3.05	2.00	1.99	1.00	1.11	1.03
Legendre	3.05	2.00	1.99	1.00	1.11	1.03
Hermite	3.05	2.00	1.99	1.00	1.11	1.03
Chebyshev, 1^{st} Kind	3.05	2.00	1.99	1.00	1.11	1.03
Chebyshev, 2^{nd} Kind	3.05	2.00	1.99	1.00	1.11	1.03
Zernike	3.05	2.00	1.99	1.00	1.11	1.03
Gegenbauer	3.05	2.00	1.99	1.00	1.11	1.03

the method. The function is determined to be just over third-order accurate, the first-derivatives are second-order accurate, and the second-derivatives are just above first-order. As with the second-order, the condition number of the system should be inspected to see how the basis change affects the moment matrix $\mathbf{M}$. Table 6.145 shows the average condition number for each level of refinement of the various bases including the general MLS reconstruction with both weight regimes. Table 6.145 shows some interesting results. As with the second-order reconstruction, the condition number of $\mathbf{M}$ tends to decrease as the mesh is refined, with the exception of the Zernike basis. The monomial basis has the median condition number of the system, with the best conditioned of the monomial systems being the Affine MLS. As with the structured cases, some of the orthogonal bases perform worse than the monomial basis: Legendre, Hermite, and Chebyshev of the First Kind. The worst basis is the Hermite, with a condition number near forty. The remaining bases have a much better conditioned system, with the 'best' basis being the Gegenbauer, with a condition number just below three. The 'best' bases are in the class of spherical bases, so it makes sense

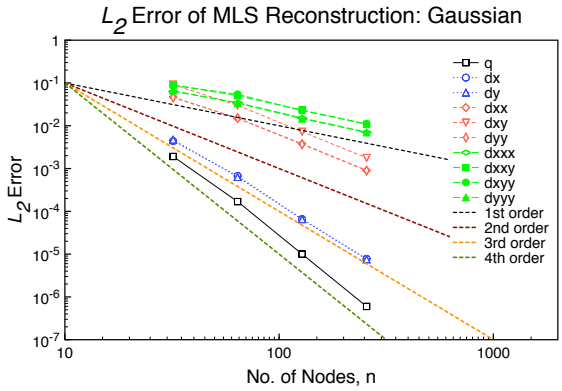
Table 6.145: Average Condition Number of M on Unstructured Grids: 3^{rd}-Order MLS with Different Bases & Weights.

Basis	Mesh Size			
	32	64	128	256
Monomial, Isotropic	13.27	12.82	12.69	12.62
Monomial, Anisotropic	14.13	13.75	13.71	13.68
Monomial, Affine	12.40	11.90	11.71	11.60
Legendre	15.72	14.97	14.70	14.53
Hermite	42.34	41.64	41.72	41.64
Chebyshev, 1^{st} Kind	24.02	22.88	22.48	22.23
Chebyshev, 2^{nd} Kind	6.26	6.10	6.08	6.05
Zernike	4.71	4.73	4.75	4.76
Gegenbauer	2.91	2.86	2.84	2.84

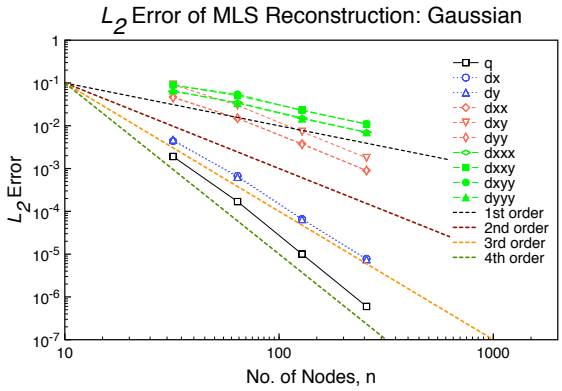
that these (the Chebyshev of the Second Kind, Zernike, and Gegenbauer) perform best for this test. However, all of the bases produce accurate reconstructions, and even using the monomial basis is more than acceptable.

6.6.2.3 Fourth-Order

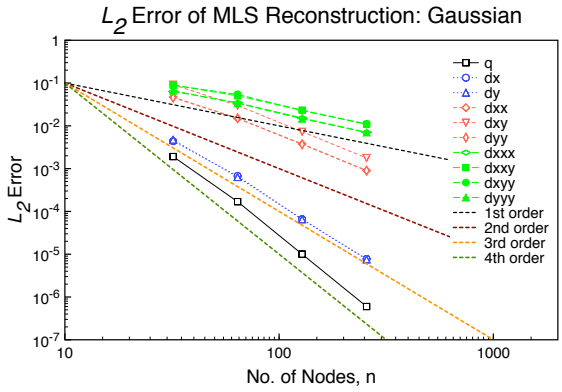
For third-order reconstructions, all of the bases differ from the general monomial polynomial. Figure 6.143 shows the order of accuracy results for the orthogonal bases.



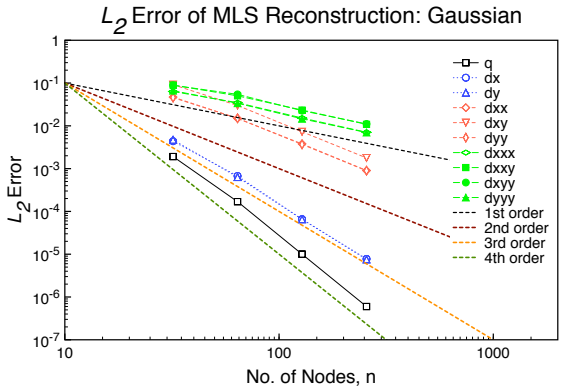
(a) Legendre



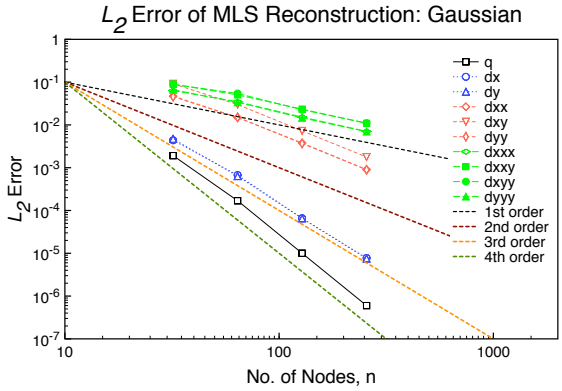
(b) Hermite



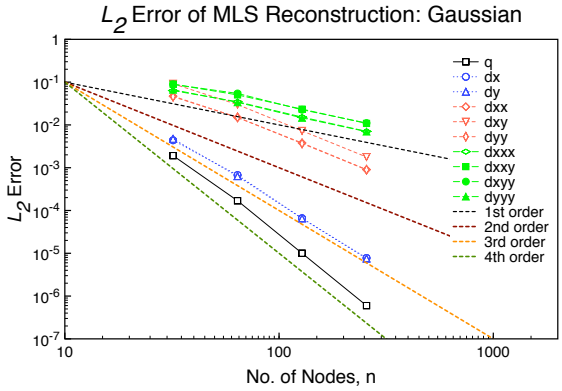
(c) Chebyshev, 1st Kind



(d) Chebyshev, 2nd Kind



(e) Zernike



(f) Gegenbauer

Figure 6.143: Error Norms on Unstructured Grids: 4th-Order Affine MLS with Anisotropic Weights Using Orthogonal Bases.

As with the second-order reconstruction, there is not a discernible difference between any of the bases and the monomial polynomial in Figure 6.29. Table 6.146 shows the computed order of accuracy for each basis. As expected, changing the basis does not affect the order of accuracy of

Table 6.146: Accuracy on Unstructured Grids: 4^{th}-Order Affine MLS with Anisotropic Weights Using Orthogonal Bases.

Basis	Function									
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial xy^2}$	$\frac{\partial^3 q}{\partial y^3}$
Monomial	4.00	3.15	3.13	1.95	1.97	1.96	1.08	1.04	1.08	1.07
Legendre	4.00	3.15	3.13	1.95	1.97	1.96	1.08	1.04	1.08	1.07
Hermite	4.00	3.15	3.13	1.95	1.97	1.96	1.08	1.04	1.08	1.07
Chebyshev, 1^{st} Kind	4.00	3.15	3.13	1.95	1.97	1.96	1.08	1.04	1.08	1.07
Chebyshev, 2^{nd} Kind	4.00	3.15	3.13	1.95	1.97	1.96	1.08	1.04	1.08	1.07
Zernike	4.00	3.15	3.13	1.95	1.97	1.96	1.08	1.04	1.08	1.07
Gegenbauer	4.00	3.15	3.13	1.95	1.97	1.96	1.08	1.04	1.08	1.07

the method. The function is determined to be fourth-order accurate, the first-derivatives are over third-order accurate, the second-derivatives are just below second-order, and the third-derivatives are roughly first-order accurate. The condition number of the moment matrix $\mathbf{M}$ is inspected. Table 6.147 shows the average condition number for each level of refinement of the various bases, including the general isotropic and anisotropic weighed MLS. Table 6.147 shows that the best performing bases are the 'spherical' bases. As with the third-order case, the condition number of $\mathbf{M}$ tends to decrease as the mesh is refined, though with the spherical bases the condition number increases slightly for the finest mesh. This is most likely due to a change in the stencil size. The monomial basis has the median condition number of the system, with some of the orthogonal bases performing worse: Hermite and Chebyshev of the First Kind. On those methods using the monomial basis, the Affine MLS performs the best. The remaining bases have a much better conditioned system, with the 'best' basis being the Gegenbauer, with a condition number just above five. Unlike the third-order reconstruction, the second-best basis is the Chebyshev of the Second

Table 6.147: Average Condition Number of M on Unstructured Grids: 4^{th}-Order MLS with Different Bases & Weights.

Basis	Mesh Size			
	32	64	128	256
Monomial, Isotropic	51.13	48.05	47.19	46.75
Monomial, Anisotropic	55.52	51.56	51.11	50.81
Monomial, Affine	45.44	42.27	41.03	40.21
Legendre	39.95	36.80	35.66	34.85
Hermite	216.0	207.5	205.3	203.2
Chebyshev, 1^{st} Kind	63.27	58.81	57.39	56.34
Chebyshev, 2^{nd} Kind	13.82	12.77	12.45	14.83
Zernike	22.15	20.51	21.48	21.38
Gegenbauer	5.61	5.28	5.19	5.11

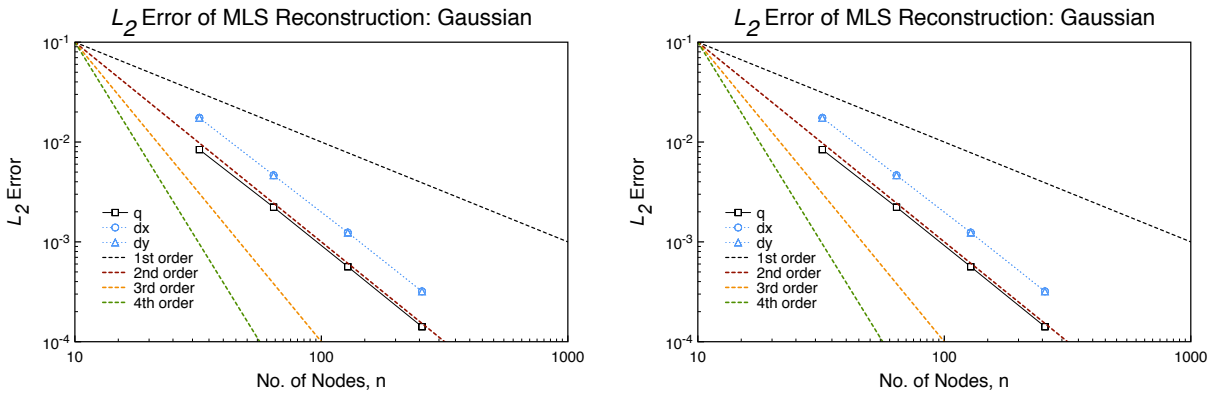
Kind. The worst basis by far is the Hermite basis. Overall, any basis besides the Hermite would give accurate results, though it would be best to use one of the 'spherical' bases when performing the reconstruction.

6.6.3 Stretched Grids

This subsection presents the results obtained by varying the polynomial basis used for the Affine MLS reconstruction on stretched grids.

6.6.3.1 Second-Order

For second-order reconstructions, only the Chebyshev of the Second Kind and Gegenbauer polynomials differ from the basic monomial polynomial. The remaining bases will not have a different reconstruction from the monomial. Figure 6.144 shows the order of accuracy results for both the Chebyshev of the Second Kind and Gegenbauer polynomials.



(a) Chebyshev, 2nd Kind

(b) Gegenbauer

Figure 6.144: Error Norms on Stretched Grids: 2nd-Order Affine MLS with Anisotropic Weights Using Orthogonal Bases.

From Figure 6.144, there is no noticeable difference between the two bases and the monomial polynomial in Figure 6.27. Table 6.148 shows the computed order of accuracy for each basis. As

Table 6.148: Accuracy on Stretched Grids: 2nd-Order Affine MLS with Anisotropic Weights Using Orthogonal Bases.

Basis	Function		
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$
Monomial	1.99	1.93	1.93
Chebyshev, 2nd Kind	1.99	1.93	1.93
Gegenbauer	1.99	1.93	1.93

expected, changing the basis does not affect the order of accuracy of the method. The function is second-order accurate and the first-derivative both are first-order accurate. It is also important to look at the average condition number of the moment matrix $\mathbf{M}$. Table 6.149 shows the average condition number for each level of refinement of the various bases, including the general MLS reconstruction with both weight regimes. A few observations can be made from Table 6.137.

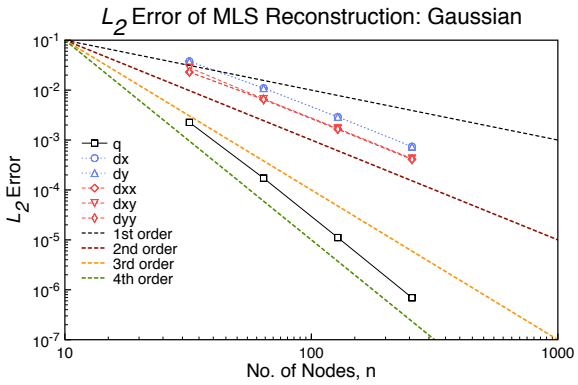
Table 6.149: Average Condition Number of $\mathbf{M}$ on Stretched Grids: 2^{nd}-Order MLS with Different Bases & Weights.

Basis	Mesh Size			
	32	64	128	256
Monomial, Isotropic	7.11	6.51	6.20	6.05
Monomial, Anisotropic	7.48	6.86	6.57	6.43
Monomial, Affine	2.24	2.12	2.08	2.06
Chebyshev, 2^{nd} Kind	1.13	1.06	1.04	1.03
Gegenbauer	1.87	1.91	1.94	1.95

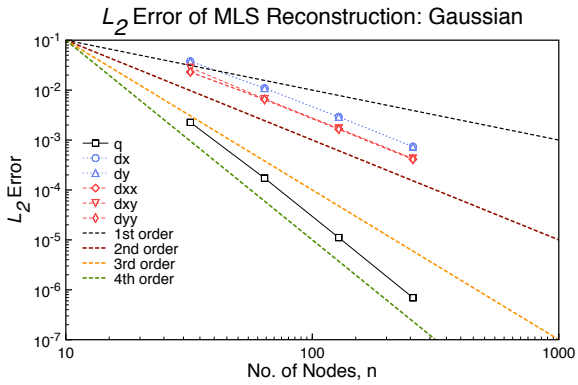
First, as the mesh is refined, the condition number of moment matrix $\mathbf{M}$ tends to increase. This is probably due to the change in stencil size. The worst condition system is on using the monomial basis with anisotropic weighting. The best conditioned system for the structured grid is Affine MLS case using the Chebyshev of the Second Kind basis, with the Gegenbauer basis being the second best. Overall, though, the condition number of the moment matrix $\mathbf{M}$ is very good, with the largest condition number only just over two. Therefore, any of these bases could be used to produce accurate reconstructions.

6.6.3.2 Third-Order

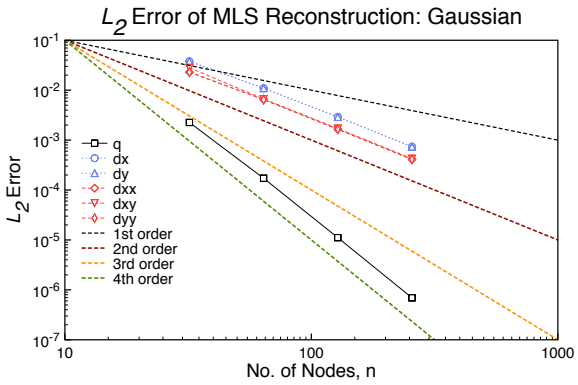
For third-order reconstructions, all of the bases differ from the general monomial polynomial. Figure 6.145 shows the order of accuracy results for the orthogonal bases.



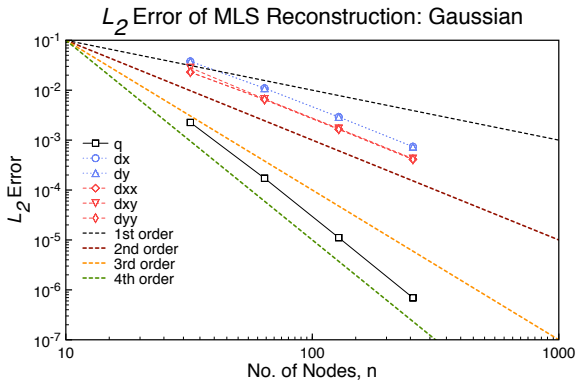
(a) Legendre



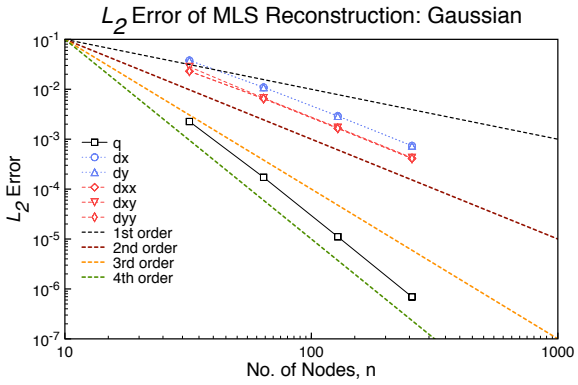
(b) Hermite



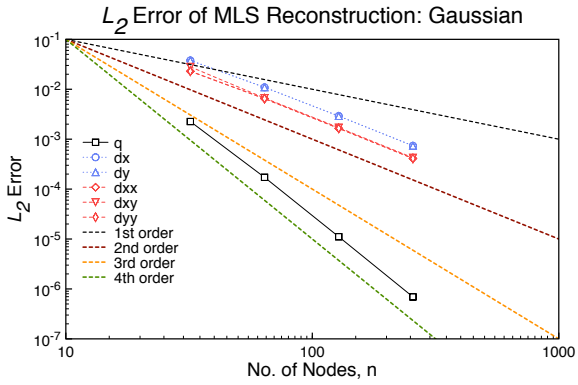
(c) Chebyshev, 1st Kind



(d) Chebyshev, 2nd Kind



(e) Zernike



(f) Gegenbauer

Figure 6.145: Error Norms on Stretched Grids: 3rd-Order Affine MLS with Anisotropic Weights Using Orthogonal Bases.

As with the second-order reconstruction, there is not a discernible difference between any of the bases and the monomial polynomial in Figure 6.28. Table 6.150 shows the computed order of accuracy for each basis. As expected, changing the basis does not affect the order of accuracy of

Table 6.150: Accuracy on Stretched Grids: 3^{rd}-Order Affine MLS with Anisotropic Weights Using Orthogonal Bases.

Basis	Function					
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$
Monomial	3.98	1.94	1.94	1.99	1.99	1.99
Legendre	3.98	1.94	1.94	1.99	1.99	1.99
Hermite	3.98	1.94	1.94	1.99	1.99	1.99
Chebyshev, 1^{st} Kind	3.98	1.94	1.94	1.99	1.99	1.99
Chebyshev, 2^{nd} Kind	3.98	1.94	1.94	1.99	1.99	1.99
Zernike	3.98	1.94	1.94	1.99	1.99	1.99
Gegenbauer	3.98	1.94	1.94	1.99	1.99	1.99

the method. The function is determined to be just over third-order accurate, the first-derivatives are second-order accurate, and the second-derivatives are just above first-order. As with the second-order, the condition number of the system should be inspected to see how the basis change affects the moment matrix $\mathbf{M}$. Table 6.151 shows the average condition number for each level of refinement of the various bases including the general MLS reconstruction with both weight regimes. Table 6.151 shows some interesting results. As with the second-order reconstruction, the condition number of $\mathbf{M}$ tends to decrease as the mesh is refined, with the exception of the Zernike basis. The monomial basis has the median condition number of the system, with the best conditioned of the monomial systems being the Affine MLS. As with the structured cases, some of the orthogonal bases perform worse than the monomial basis: Legendre, Hermite, and Chebyshev of the First Kind. The worst basis is the Hermite, with a condition number near forty. The remaining bases have a much better conditioned system, with the 'best' basis being the Gegenbauer, with a condition number just below three. The 'best' bases are in the class of spherical bases, so it makes sense

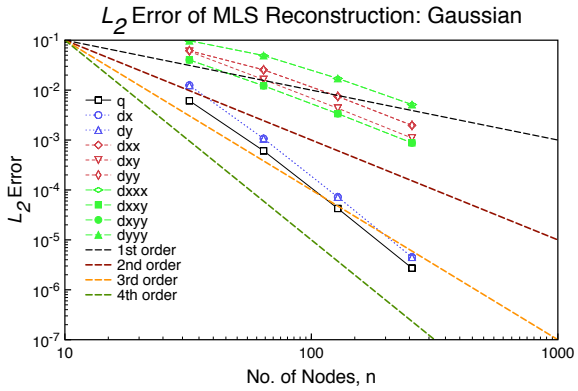
Table 6.151: Average Condition Number of $\mathbf{M}$ on Stretched Grids: 3^{rd}-Order MLS with Different Bases & Weights.

Basis	Mesh Size			
	32	64	128	256
Monomial, Isotropic	118.0	116.2	97.26	92.00
Monomial, Anisotropic	158.7	140.2	129.2	124.0
Monomial, Affine	18.30	16.07	14.95	14.37
Legendre	26.05	21.78	19.82	18.89
Hermite	53.35	46.26	43.10	41.75
Chebyshev, 1^{st} Kind	39.72	33.29	30.37	29.00
Chebyshev, 2^{nd} Kind	9.10	7.39	6.62	6.29
Zernike	6.75	5.40	4.79	4.51
Gegenbauer	3.63	3.09	2.97	2.90

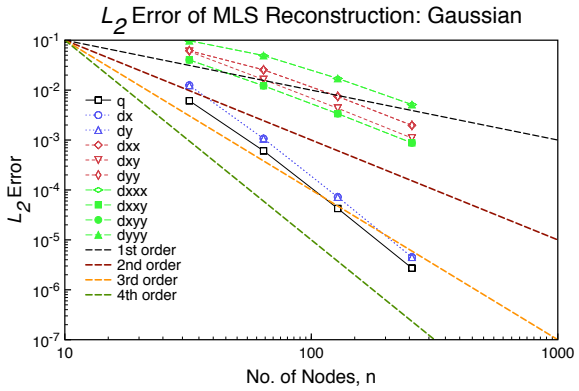
that these (the Chebyshev of the Second Kind, Zernike, and Gegenbauer) perform best for this test. However, all of the bases produce accurate reconstructions, and even using the monomial basis is more than acceptable.

6.6.3.3 Fourth-Order

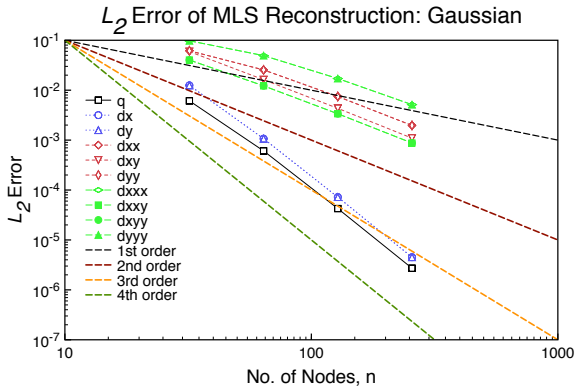
For third-order reconstructions, all of the bases differ from the general monomial polynomial. Figure 6.146 shows the order of accuracy results for the orthogonal bases.



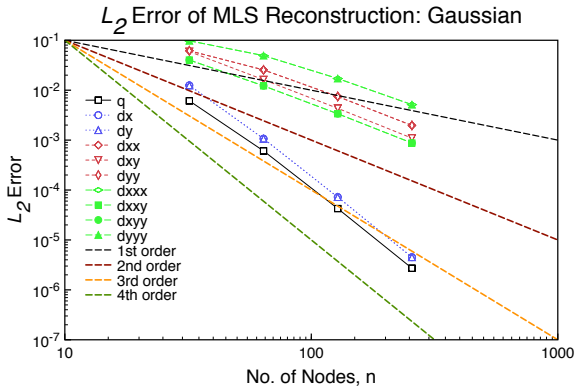
(a) Legendre



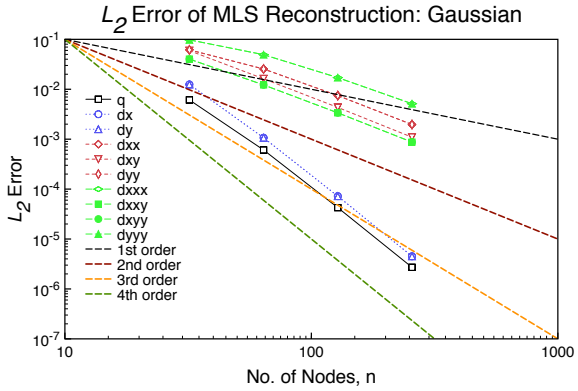
(b) Hermite



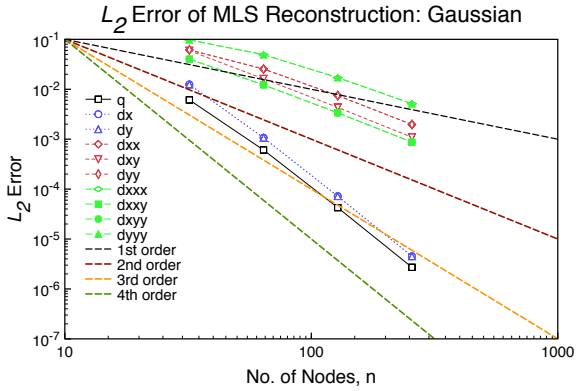
(c) Chebyshev, 1st Kind



(d) Chebyshev, 2nd Kind



(e) Zernike



(f) Gegenbauer

Figure 6.146: Error Norms on Stretched Grids: 4th-Order Affine MLS with Anisotropic Weights Using Orthogonal Bases.

As with the second-order reconstruction, there is not a discernible difference between any of the bases and the monomial polynomial in Figure 6.29. Table 6.152 shows the computed order of accuracy for each basis. As expected, changing the basis does not affect the order of accuracy of

Table 6.152: Accuracy on Stretched Grids: 4^{th}-Order Affine MLS with Anisotropic Weights Using Orthogonal Bases.

Basis	Function									
	q	$\frac{\partial q}{\partial x}$	$\frac{\partial q}{\partial y}$	$\frac{\partial^2 q}{\partial x^2}$	$\frac{\partial^2 q}{\partial xy}$	$\frac{\partial^2 q}{\partial y^2}$	$\frac{\partial^3 q}{\partial x^3}$	$\frac{\partial^3 q}{\partial x^2 y}$	$\frac{\partial^3 q}{\partial xy^2}$	$\frac{\partial^3 q}{\partial y^3}$
Monomial	3.90	3.94	3.94	1.85	1.94	1.85	1.63	1.90	1.90	1.63
Legendre	3.90	3.94	3.94	1.85	1.94	1.85	1.63	1.90	1.90	1.63
Hermite	3.90	3.94	3.94	1.85	1.94	1.85	1.63	1.90	1.90	1.63
Chebyshev, 1^{st} Kind	3.90	3.94	3.94	1.85	1.94	1.85	1.63	1.90	1.90	1.63
Chebyshev, 2^{nd} Kind	3.90	3.94	3.94	1.85	1.94	1.85	1.63	1.90	1.90	1.63
Zernike	3.90	3.94	3.94	1.85	1.94	1.85	1.63	1.90	1.90	1.63
Gegenbauer	3.90	3.94	3.94	1.85	1.94	1.85	1.63	1.90	1.90	1.63

the method. The function is determined to be fourth-order accurate, the first-derivatives are over third-order accurate, the second-derivatives are just below second-order, and the third-derivatives are roughly first-order accurate. The condition number of the moment matrix $\mathbf{M}$ is inspected. Table 6.153 shows the average condition number for each level of refinement of the various bases, including the general isotropic and anisotropic weighed MLS. Table 6.153 shows that the best performing bases are the 'spherical' bases. As with the third-order case, the condition number of $\mathbf{M}$ tends to decrease as the mesh is refined, though with the spherical bases the condition number increases slightly for the finest mesh. This is most likely due to a change in the stencil size. The monomial basis has the median condition number of the system, with some of the orthogonal bases performing worse: Hermite and Chebyshev of the First Kind. On those methods using the monomial basis, the Affine MLS performs the best. The remaining bases have a much better conditioned system, with the 'best' basis being the Gegenbauer, with a condition number just above five. Unlike the third-order reconstruction, the second-best basis is the Chebyshev of the Second

Table 6.153: Average Condition Number of $\mathbf{M}$ on Stretched Grids: 4^{th}-Order MLS with Different Bases & Weights.

Basis	Mesh Size			
	32	64	128	256
Monomial, Isotropic	2028	2082	1938	1806
Monomial, Anisotropic	2478	2549	2382	2244
Monomial, Affine	105.7	76.01	64.33	58.95
Legendre	106.5	73.00	59.69	53.64
Hermite	441.5	316.4	264.3	240.0
Chebyshev, 1^{st} Kind	154.5	106.0	86.58	77.88
Chebyshev, 2^{nd} Kind	37.22	24.04	19.05	16.84
Zernike	61.84	35.61	25.88	22.43
Gegenbauer	18.29	10.80	7.48	6.03

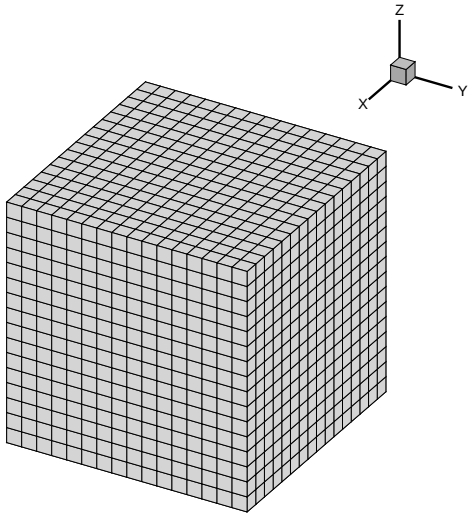
Kind. The worst basis by far is the Hermite basis. Overall, any basis besides the Hermite would give accurate results, though it would be best to use one of the 'spherical' bases when performing the reconstruction.

7. HIGHER-ORDER RESULTS

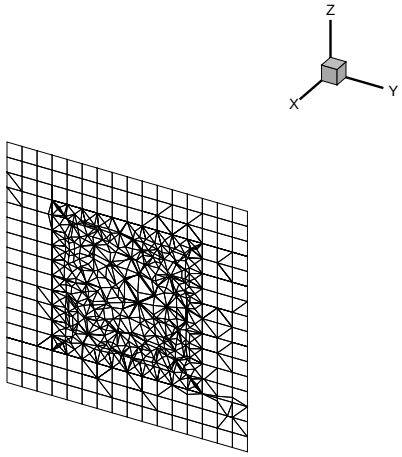
This chapter presents results of the implementation the higher-order unstructured CFD method. First, a study on the condition of the moment matrix $\mathbf{M}$ needed for the MLS basis vector $\mathbf{\Psi}$ in the MLS reconstruction (5.16) is presented for various grids. Next, a presentation of the new ghost boundary conditions are presented. Finally, results for the Taylor–Green Vortex case for second- through fourth-order are presented for coarse meshes. The chapter ends with a summary of the results.

7.1 Condition of Moment Matrix M for 3D Grids

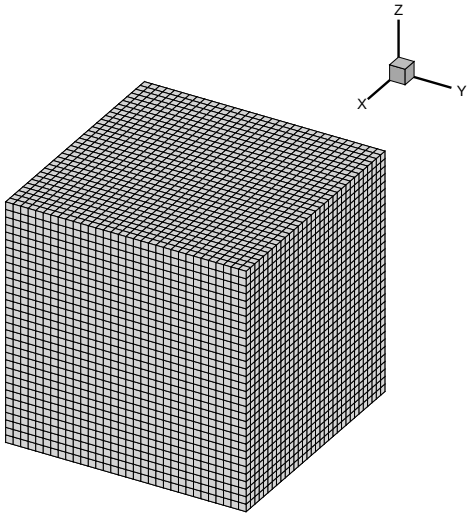
This section presents results studying the condition number of the moment matrix $\mathbf{M}$ on various grids used later in this chapter for boundary conditions. For each set of grids presented, the scaling parameter k is set to 0.8, since that proved to be a reasonable choice from Chapter 6. Additionally, each method of matrix inversion is presented using the different weighting schemes. For the Affine MLS, the method is also computed using the Chebyshev of the Second Kind and Gegenbauer bases, which performed best in the two-dimensional cases. First, the condition number is accessed on the cube domains used for the Taylor-Green Vortex cases. Two levels of refinement are used in the TGV cases, a 17^3 and 33^3 $[-\pi, \pi]^3$ cube, both with structured and unstructured grids. For the unstructured meshes, the periodic boundaries are padded with hexagons to ensure periodicity through the boundary. Figure 7.1 shows the structured grids and $x = 0$ slices for the unstructured grids.



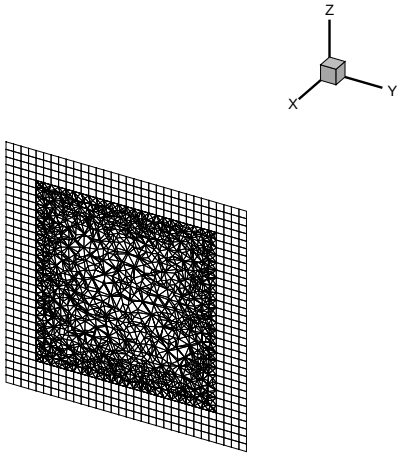
(a) 17^2, Structured



(b) 17^2, Unstructured Slice



(c) 33^2, Structured



(d) 33^2, Unstructured Slice

Figure 7.1: Meshes Used for Taylor–Green Vortex Cases.

Results from the second-order study are presented in Table 7.1 for the general SPD inverse, QR method, and Pivoting QR (PQR) method.

Table 7.1: Average & Maximum Condition Number of $\mathbf{M}$ on TGV Grids: 2^{nd}-Order MLS.

Method	Weight	Mesh Type					
		Structured		Unstructured			
		17^3	33^3	17^3		33^3	
		avg		avg	max	avg	max
SPD	Iso.	924	924	544	1049	453	1049
SPD	Ani.	924	924	638	2273	566	3330
SPD	Affine	924	924	589	1531	506	1519
QR	Iso.	30	30	20	40	18	40
QR	Ani.	30	30	24	54	22	68
QR	Affine	30	30	22	53	20	48
PQR	Iso.	30	30	20	41	17	41
PQR	Ani.	30	30	23	45	21	47
PQR	Affine	30	30	22	41	20	41
PQR	Affine, Ch2	15	15	11	22	10	23
PQR	Affine, Geg	7	7	6	14	5	14

For the structured grids, the condition number does not change with the weight for each of the methods. This follows from the two-dimensional studies, since the shape of the stencil $\Omega_{\mathbf{x}_I}$ is the same for each, and as the stencil is scaled, the 'scaled' stencils are effectively the same. Obviously, the SPD method produces the poorest condition systems, on the order of thirty times worse. It is note worthy that on structured grids it is beneficial to use the orthogonal bases, as the Gegenbauer basis Affine MLS using Pivoting QR has a condition number four times lower than the regular Affine MLS using Pivoting QR. Additionally, for each of the inversion methods, the isotropically weighted stencils are slightly better for the unstructured grids but not enough to make a significant difference. Overall, as was seen with the two-dimensional cases, the Pivoting QR based Affine

MLS using the Gegenbauer basis is best. Table 7.2 shows the third-order MLS reconstruction condition numbers for $\mathbf{M}$.

Table 7.2: Average & Maximum Condition Number of $\mathbf{M}$ on TGV Grids: 3^{rd}-Order MLS.

Method	Weight	Mesh Type					
		Structured		Unstructured			
		17^3	33^3	17^3		33^3	
		avg		avg	max	avg	max
SPD	Iso.	1032	1032	1092	2648	18392	$2.05E7$
SPD	Ani.	1032	1032	1827	12678	$1.36E5$	$1.59E8$
SPD	Affine	1032	1032	1102	3528	1191	$1.53E5$
QR	Iso.	33	33	35	61	39	5404
QR	Ani.	33	33	43	142	56	12667
QR	Affine	33	33	35	60	35	409
PQR	Iso.	33	33	34	57	37	4405
PQR	Ani.	33	33	41	103	52	12667
PQR	Affine	33	33	34	57	34	396
PQR	Affine, Ch2	39	39	46	76	33	357
PQR	Affine, Geg	20	20	22	37	23	303

It is readily apparent from Table 7.2 that for unstructured grids, using the general matrix inverse is worthless, with maximum condition numbers exceeding 10^8 for the non-Affine MLS. The Affine MLS does a reasonable job on average for the general inverse, but the maximum condition number is 10^5, which is still rather large. For the decomposition methods, the Affine MLS has a lower maximum condition number approximately ten times lower than the isotropically weighted methods. Additionally, using the Gegenbauer basis reduces the average condition number about

1.5 times the general monomial basis. Both unstructured grids, the QR and Pivoting QR inversion methods do not show a significant difference in the average condition numbers; however, the maximum condition number is lower for the Pivoting QR. It is important to point out and discuss the anisotropic weighted MLS results with the third-order case. The condition numbers for these weighted methods are all the worst-performing of the methods, and this is most likely due a mismatch of the weights with the polynomials. This is precisely why the Affine MLS method was implemented, such that the condition of the moment matrix $\mathbf{M}$ for ellipsoidal distributions could be restored. In fact, the condition number using the Affine MLS is improved. Therefore, the Affine MLS should be the go-to method for computing gradients. Further improvement to the system, which is not bad to begin with, can be had by using the Gegenbauer basis. Table 7.3 shows the fourth-order MLS reconstruction condition numbers. Even more than the third-order results in

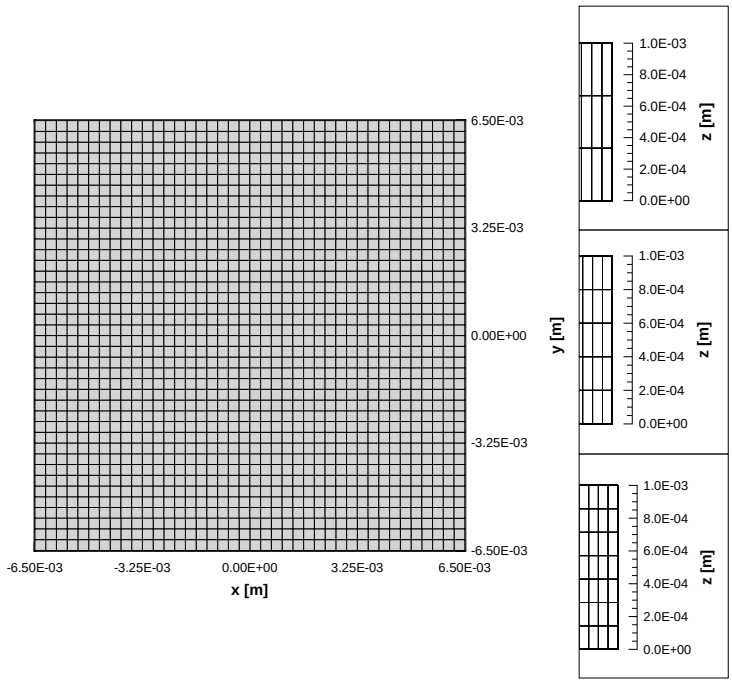
Table 7.3: Average & Maximum Condition Number of $\mathbf{M}$ on TGV Grids: 4^{th}-Order MLS.

Method	Weight	Mesh Type					
		Structured		Unstructured			
		17^3	33^3	17^3		33^3	
		avg		avg	max	avg	max
SPD	Iso.	40630	40630	$6.73E6$	$4.47E9$	$2.61E8$	$3.22E11$
SPD	Ani.	40630	40630	$1.79E7$	$1.67E10$	$6.26E8$	$1.22E12$
SPD	Affine	40630	40630	239160	$1.94E8$	$1.44E6$	$4.78E9$
QR	Iso.	211	211	502	89325	1707	$9.01E5$
QR	Ani.	211	211	756	$2.30E5$	2947	$2.81E6$
QR	Affine	211	211	282	15725	322	96704
PQR	Iso.	211	211	477	63522	1661	$5.47E5$
PQR	Ani.	211	211	628	113988	2307	$1.00E6$
PQR	Affine	211	211	244	10233	284	59811
PQR	Affine, Ch2	102	102	132	10380	148	33992
PQR	Affine, Geg	52	52	72	3924	79	16745

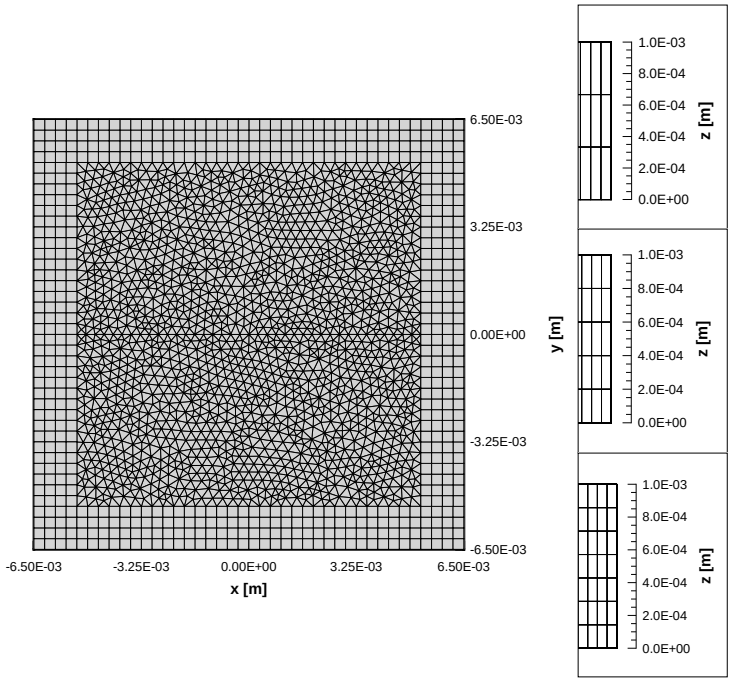
Table 7.2, it is abundantly clear that using a general matrix inverse is worthless on unstructured grids. Using the matrix decomposition methods dramatically reduces the condition number of the moment matrix $\mathbf{M}$ by as much as 848 times. Furthermore, the Pivoting QR method reduces the

maximum condition number over the QR method by about 1.5 times. Using the orthogonal bases, the condition number for the moment matrix $\mathbf{M}$ is reduced about 3.5 times over the general monomial Affine MLS, and by over 6.5 times that of the isotropic weighted MLS using Pivoting QR. The maximum condition number of the Affine MLS using Pivoting QR is six times smaller than the isotropic weighted, and the Gegenbauer basis Affine MLS has a maximum condition number an additional 2.6 times smaller. Reviewing all of the results from Tables 7.1-7.3, the Affine MLS reconstruction method results in a major improvement to the condition number of the moment matrix $\mathbf{M}$ over the standard method, and when combined with the Pivoting QR method, reduces the maximum condition number of the mesh by a significant number as well. Using an orthogonal basis, while not required, does represent an avenue for even further improvement of the condition number of the moment matrix $\mathbf{M}$, ensuring an accurate inverse and further cements confidence in the MLS reconstructed gradients.

Next, the grids used in the non-reflecting boundary condition are investigated. More on the case is given later, but the grids are 41^2 in the xy-plane for both the structured and unstructured grids. The grid lies on $[-0.0065, 0.0065]^2$ in the xy-plane. In the $z-$direction, there are 4, 6, and 8 nodes on $[0, 1E-3]$ for the second-, third-, and fourth-order cases. Figure shows the grids.



(a) Structured



(b) Unstructured

Figure 7.2: Meshes Used for Navier-Stokes Compatibility Boundary Conditions.

These grids have ghost nodes, so for each order and grid, the average and maximum condition

number for the ghost nodes is also reported. The results for the condition number evaluation for the second-order case in Table 7.4.

Table 7.4: Average & Maximum Condition Number of M on NSCBC Grids: 2^{nd}-Order MLS.

Method	Weight	Mesh Type						
		Structured			Unstructured			
		Main	Ghost		Main		Ghost	
		avg	avg	max	avg	max	avg	max
SPD	Iso.	924	3572	3697	885	1624	3572	3696
SPD	Ani.	924	3607	3692	1046	3305	3607	3692
SPD	Affine	924	3602	3692	950	1266	3602	3692
QR	Iso.	30	60	61	30	43	59	61
QR	Ani.	31	60	61	33	64	60	61
QR	Affine	30	60	61	31	41	60	61
PQR	Iso.	30	60	61	30	42	60	61
PQR	Ani.	31	60	61	33	54	60	61
PQR	Affine	30	60	61	31	37	60	61
PQR	Affine, Ch2	15	30	30	16	20	30	30
PQR	Affine, Geg	8	15	15	8	12	15	15

As with the previous case, the second-order results show that there is little difference between the QR and Pivoting QR results. Also, the orthogonal base results have a condition 2-4 times better than the base monomial Affine MLS. Additionally, there is little difference between the different weighing schemes, as seen in the previous results. The general method is thirty-times worse that the decomposition methods. The third-order results are presented in Table 7.5.

Table 7.5: Average & Maximum Condition Number of $\mathbf{M}$ on NSCBC Grids: 3^{rd}-Order MLS.

Method	Weight	Mesh Type							
		Structured				Unstructured			
		Main		Ghost		Main		Ghost	
		avg	max	avg	max	avg	max	avg	max
SPD	Iso.	1606	2495	$2.06E5$	$2.18E5$	1332	2965	$2.06E5$	$2.18E5$
SPD	Ani.	2670	2670	$1.99E5$	$2.10E5$	3094	8778	$1.99E5$	$2.10E5$
SPD	Affine	1034	1723	$1.99E5$	$2.11E5$	1024	2925	$1.99E5$	$2.11E5$
QR	Iso.	42	56	460	482	39	62	461	482
QR	Ani.	54	54	451	473	57	112	452	472
QR	Affine	34	47	48	436	34	68	452	474
PQR	Iso.	42	49	449	470	38	54	449	470
PQR	Ani.	54	54	440	461	58	104	440	461
PQR	Affine	33	41	368	435	33	49	440	461
PQR	Affine, Ch2	39	44	427	448	43	62	427	448
PQR	Affine, Geg	19	21	184	192	21	32	184	193

While the main mesh nodes have a respectable condition number for the general matrix inverse, the condition number of $\mathbf{M}$ for the ghost nodes are too large to be considered useful. As for the matrix decomposition methods, the Pivoting QR method is slightly better than the QR method. The Affine MLS is also slightly better than the isotropically weighed MLS, but not enough to make a difference. However, the Gegenbauer basis Affine MLS is about 1.7 times smaller than the general monomial Affine MLS. These trends are the same for both the structured and unstructured grids. Table 7.6 and 7.7 shows the fourth-order MLS reconstruction condition numbers for the structured and unstructured meshes.

Table 7.6: Average & Maximum Condition Number of M on NSCBC Grids: 4^{th}-Order MLS, Structured Mesh.

Method	Weight	Main		Ghost	
		avg	max	avg	max
SPD	Iso.	$1.80E6$	$2.86E6$	$1.99E6$	$2.16E6$
SPD	Ani.	$1.00E6$	$1.00E6$	$1.09E6$	$1.95E6$
SPD	Affine	40679	78494	$1.04E6$	$1.84E6$
QR	Iso.	442	598	1340	1409
QR	Ani.	993	998	1012	1356
QR	Affine	212	328	1003	1716
PQR	Iso.	442	530	1307	13811
PQR	Ani.	993	998	980	1321
PQR	Affine	212	290	947	1115
PQR	Affine, Ch2	104	162	695	877
PQR	Affine, Geg	53	85	379	438

Table 7.7: Average & Maximum Condition Number of $\mathbf{M}$ on NSCBC Grids: 4^{th}-Order MLS, Unstructured Mesh.

Method	Weight	Main		Ghost	
		avg	max	avg	max
SPD	Iso.	$1.72E5$	$5.01E5$	$1.99E6$	$2.16E6$
SPD	Ani.	$1.03E6$	$3.81E6$	$1.09E6$	$1.95E6$
SPD	Affine	40415	78494	$1.04E6$	$1.84E6$
QR	Iso.	412	771	1341	1409
QR	Ani.	1004	1985	2287	2562
QR	Affine	217	328	1002	1716
PQR	Iso.	481	738	1306	1381
PQR	Ani.	1004	1985	981	1321
PQR	Affine	208	290	269	1115
PQR	Affine, Ch2	105	163	695	877
PQR	Affine, Geg	56	85	379	438

With the fourth-order results, the general matrix inversion methods results should not be trusted with these condition numbers. Looking at the matrix decomposition methods, the Pivoting QR method is slightly better than the QR, as with the third-order results in Table 7.5. The Affine method bests the isotropic weighted method by having condition numbers half as large. Furthermore, the Gegenbauer basis Affine MLS condition number is four times smaller than the average condition number of the general monomial Affine MLS. Reviewing all of the results from Tables 7.4-7.7, the Affine MLS reconstruction method results in a major improvement to the condition number of the moment matrix $\mathbf{M}$ over the standard method, and when combined with the Pivoting QR method, reduces the maximum condition number of the mesh by a significant number as well. The isotropic weighted MLS results compare reasonably well to the Affine MLS results,

so if the costs associated with building the Affine MLS are too much, then the isotropic results using either QR method would be acceptable. As with the TGV grids, using an orthogonal basis is beneficial to the condition of $\mathbf{M}$.

Next the condition number of the moment matrix $\mathbf{M}$ on an inviscid grid used for a bump in channel. The grid investigated is a medium and fine grid for each order. The grid in the $x-y$ plane is $[x, y] = [33, 129]$ nodes for the medium grid , while the fine grid has $[x, y] = [65, 257]$ nodes in the $x-y$ plane. In the z-direction, the grids have 4, 6, or 8 nodes for second-, third-, or fourth-order. Figures 7.3 and 7.4 show the medium and fine meshes, respectively. .

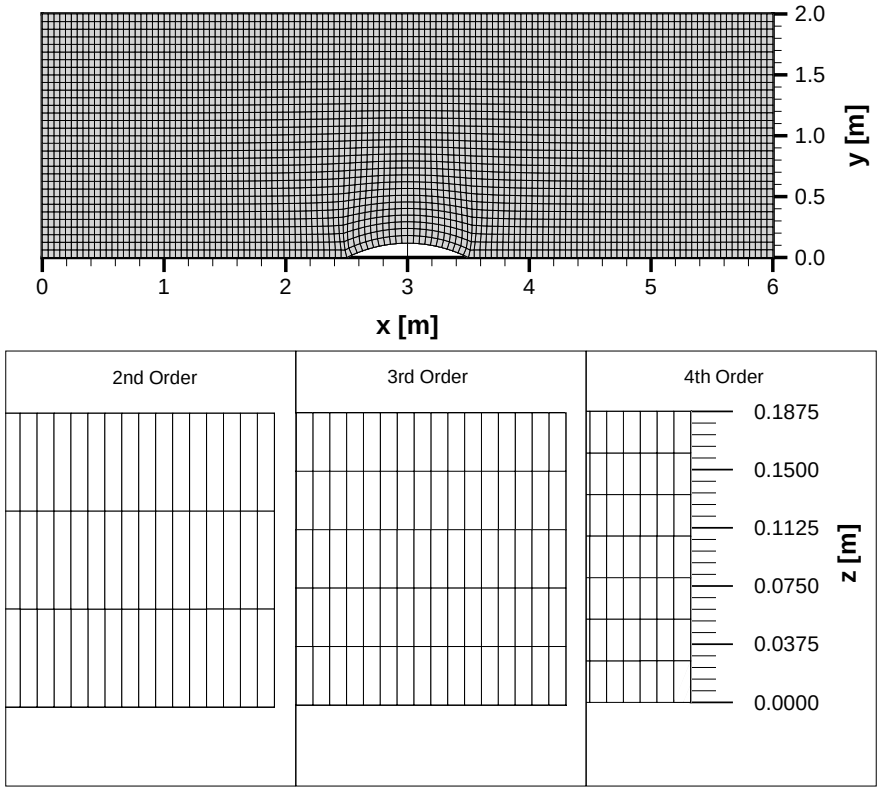


Figure 7.3: Medium Mesh for Bump-in-Channel Case.

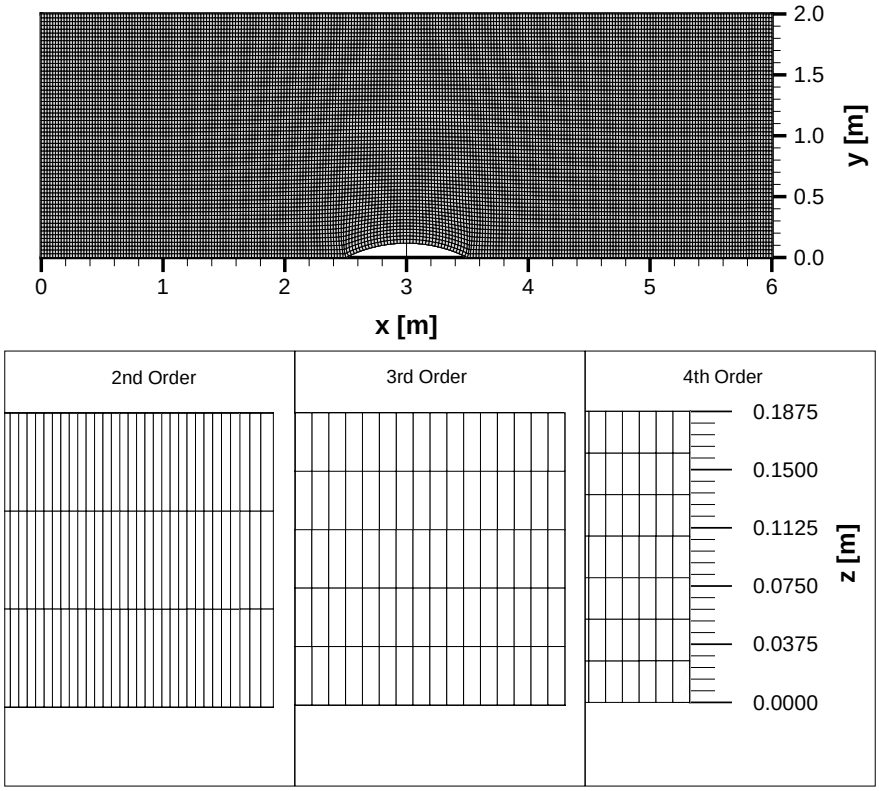


Figure 7.4: Fine Mesh for Bump-in-Channel Case.

Ghost node condition numbers are included as this is a bounded flow. Since these grids are structured, it would be expected that all of the weighting methods should have roughly the same performance. The condition number results for the second-order case 7.8.

Table 7.8: Average & Maximum Condition Number of $\mathbf{M}$ on Bump-in-Channel Grids: 2^{nd}-Order MLS.

Method	Weight	Mesh Type							
		Medium				Fine			
		Main		Ghost		Main		Ghost	
		avg	max	avg	max	avg	max	avg	max
SPD	Iso.	989	1456	3788	4627	2204	2493	6185	6277
SPD	Ani.	1635	4648	3668	4560	6502	7488	12942	13365
SPD	Affine	925	1241	3663	4564	925	1230	3670	3786
QR	Iso.	31	36	61	62	47	50	79	79
QR	Ani.	41	86	60	62	81	91	114	119
QR	Affine	30	37	60	61	30	34	61	62
PQR	Iso.	31	36	60	62	45	50	79	79
PQR	Ani.	40	63	60	62	80	84	113	116
PQR	Affine	30	33	60	61	30	32	61	62
PQR	Affine, Ch2	15	17	30	31	15	17	30	31
PQR	Affine, Geg	8	9	15	16	8	9	15	16

The second-order results in Table 7.8 show that all of the methods have acceptable condition numbers for $\mathbf{M}$. Both decomposition methods are effectively identical for both the medium and fine grids. The Affine MLS method produces the best condition numbers, with the Gegenbauer basis Affine MLS having average condition numbers about four times smaller than the general monomial basis Affine MLS. For the fine grid, the Affine MLS has average condition numbers 1.5 times smaller than the isotropic weighted MLS. The third-order results are presented in Table 7.9

Table 7.9: Average & Maximum Condition Number of $\mathbf{M}$ on Bump-in-Channel Grids: 3^{rd}-Order MLS.

Method	Weight	Mesh Type							
		Medium				Fine			
		Main		Ghost		Main		Ghost	
		avg	max	avg	max	avg	max	avg	max
SPD	Iso.	1611	3001	$2.27E5$	$3.26E5$	1438	2372	$2.09E5$	$2.22E5$
SPD	Ani.	4881	10038	$3.53E5$	$3.79E5$	3751	6189	$2.95E5$	$3.04E5$
SPD	Affine	1042	1723	$2.01E5$	$2.78E5$	1038	1723	$2.05E5$	$2.12E5$
QR	Iso.	42	63	489	590	40	57	469	490
QR	Ani.	73	110	608	711	64	93	552	567
QR	Affine	34	46	462	558	34	47	461	473
PQR	Iso.	41	61	461	490	40	53	454	474
PQR	Ani.	72	102	584	618	63	76	539	553
PQR	Affine	33	43	442	468	34	42	450	463
PQR	Affine, Ch2	39	47	429	456	40	49	437	448
PQR	Affine, Geg	20	23	185	196	20	24	188	193

As with the previous cases, the third-order results show that the general matrix inversion condition numbers begin becoming poor for the ghost nodes. The main mesh condition numbers are still reasonable, being less than $1E4$. The decomposition methods have condition numbers thirty times smaller than the general methods, and the Pivoting QR method is again slightly better than the QR method. The Affine MLS condition numbers for both the medium and fine grids are about 1.25 times smaller than the isotropic weighted MLS. The Gegenbauer basis Affine MLS is about 1.5 times smaller than the general Affine MLS. The ghost node condition numbers for the Gegenbauer basis Affine MLS are a about 2.5 times smaller than the general Affine MLS ghost node condition

numbers. The fourth-order condition numbers are presented in Table 7.10 and 7.11.

Table 7.10: Average & Maximum Condition Number of $\mathbf{M}$ on Bump-in-Channel Grids: 4^{th}-Order MLS, Medium Mesh.

Method	Weight	Main		Ghost	
		avg	max	avg	max
SPD	Iso.	$2.79E5$	$7.78E5$	$6.39E6$	$1.20E7$
SPD	Ani.	$1.16E6$	$3.20E6$	$6.01E6$	$1.47E7$
SPD	Affine	41127	78494	$5.72E6$	$1.19E7$
QR	Iso.	550	1011	2348	2976
QR	Ani.	1049	1869	2267	317
QR	Affine	213	328	2212	3387
PQR	Iso.	542	819	2321	2956
PQR	Ani.	1047	1756	2239	2819
PQR	Affine	212	290	2179	2682
PQR	Affine, Ch2	104	163	1459	1744
PQR	Affine, Geg	53	92	795	942

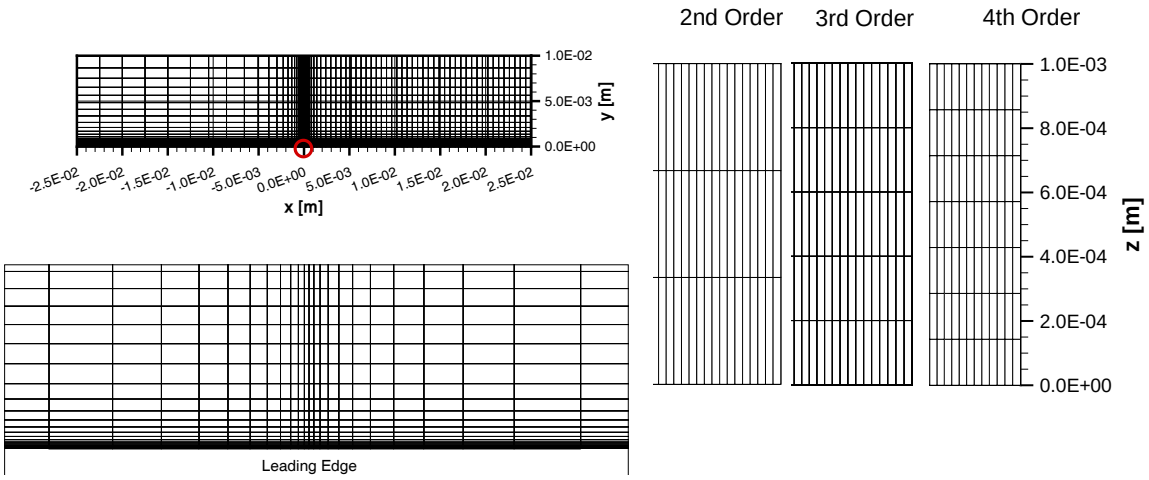
Table 7.11: Average & Maximum Condition Number of $\mathbf{M}$ on Bump-in-Channel Grids: 4^{th}-Order MLS, Fine Mesh.

Method	Weight	Main		Ghost	
		avg	max	avg	max
SPD	Iso.	72654	23519	$2.91E5$	$1.39E6$
SPD	Ani.	98095	$5.02E5$	$5.47E6$	$1.06E7$
SPD	Affine	40909	83442	96109	$5.88E5$
QR	Iso.	284	543	561	1132
QR	Ani.	351	845	2158	2590
QR	Affine	213	334	2144	2384
PQR	Iso.	278	450	2147	2272
PQR	Ani.	319	696	2138	2359
PQR	Affine	212	290	2123	2277
PQR	Affine, Ch2	103	175	1422	1461
PQR	Affine, Geg	53	95	776	798

As with the previous cases, the general method condition numbers are unacceptably large. The decomposition method condition numbers are at best almost 200 times smaller than the general decomposition methods. Both the QR and Pivoting QR methods produce similar condition numbers. The Affine MLS method has condition numbers half as large as the isotropically weighted methods. The Gegenbauer basis Affine MLS is again the best, with a condition number 4 times smaller than the monomial Affine MLS. The ghost node condition numbers for the Gegenbauer basis Affine MLS are about 2.7 times smaller than the monomial Affine MLS. Reviewing all of the results from Tables 7.8-7.11, the Affine MLS reconstruction method again has the best overall results, with the lowest condition numbers of $\mathbf{M}$. Using the orthogonal basis will result in the best results. As with the NSCBC grids, the isotropic weighted MLS results compare reasonably well to

the Affine MLS results, so if the costs associated with building the Affine MLS are too much, then the isotropic results using either QR method would be acceptable. As with the TGV grids, using an orthogonal basis is beneficial to the condition of M.

Finally, the grid used for a flat-plate case is investigated. This grid is structured, but unlike the previous cases, the grid is refined near the wall and leading edge. The grid for the flat-plate case is shown in Figure 7.5.



(a) Mesh, xy-plane

(b) Mesh, xz-plane

Figure 7.5: Meshes Used for Laminar Flat Plate Case. Leading Edge is Highlighted Area of Main Mesh.

The plate length is $x = 0.025$m. The $y+$ number near the leading-edge is approximately 1, and the cells are grown at a rate of 1.2 from the wall. Cells are equally spaced normal to the characteristic boundaries. In the $z-$direction, the grids have 4, 6, or 8 nodes for second-, third-, or fourth-order over $z = [0, 10^{-3}]$ m. Ghost node condition numbers are included as this is a bounded flow. Table 7.12 shows the computed condition numbers for the second-order flat plate grid.

Table 7.12: Average & Maximum Condition Number of $\mathbf{M}$ on Laminar Flat Plate Grids: 2^{nd}-Order MLS.

Method	Weight	Node			
		Main		Ghost	
		avg	max	avg	max
SPD	Iso.	$8.45E5$	$1.47E8$	$7.95E6$	$3.15E8$
SPD	Ani.	$9.20E7$	$3.73E10$	$2.71E9$	$7.22E10$
SPD	Affine	944	977	3661	3779
QR	Iso.	304	12166	1388	17747
QR	Ani.	2407	$1.94E5$	24134	$2.68E5$
QR	Affine	31	32	61	62
PQR	Iso.	304	12166	1388	17747
PQR	Ani.	2407	$1.94E5$	24134	$2.68E5$
PQR	Affine	31	32	61	62
PQR	Affine, Ch2	15	17	30	31
PQR	Affine, Geg	8	9	15	16

The general matrix inversion method produces poor results at the outset, unlike the previous cases. However, the Affine MLS general matrix inversion condition numbers are fairly reasonable. The anisotropic weighted methods have large maximum condition numbers. Even the isotropic weighed matrix decomposition maximum condition numbers are somewhat large. The Affine MLS built with the matrix decomposition methods are the best, with the condition numbers thirty times smaller than the general matrix inversion method. The Gegenbauer basis Affine MLS produces the best results, as with the previous cases. The third-order grid results are presented in Table 7.13.

Table 7.13: Average & Maximum Condition Number of $\mathbf{M}$ on Laminar Flat Plate Grids: 3^{rd}-Order MLS.

Method	Weight	Node			
		Main		Ghost	
		avg	max	avg	max
SPD	Iso.	$1.28E14$	$5.63E16$	$1.15E16$	$2.56E17$
SPD	Ani.	$7.92E14$	$4.36E17$	$1.31E16$	$2.77E18$
SPD	Affine	1263	1973	$7.38E5$	$1.08E6$
QR	Iso.	$1.77E6$	$3.24E8$	$5.69E6$	$5.49E8$
QR	Ani.	$3.55E6$	$7.14E8$	$2.31E7$	$1.79E9$
QR	Affine	39	48	966	1356
PQR	Iso.	$1.34E6$	$2.40E8$	$4.86E6$	$5.15E8$
PQR	Ani.	$3.25E6$	$6.73E8$	$2.08E7$	$1.70E8$
PQR	Affine	39	48	810	1003
PQR	Affine, Ch2	46	54	773	949
PQR	Affine, Geg	23	28	334	407

Using anything other than the Affine MLS method produces large condition numbers for $\mathbf{M}$, as shown in Table 7.13. Even when using the QR or Pivoting QR method, the usually reliable isotropically weighted MLS has condition numbers over 10^8, at which point the inverse is most likely useless. The condition number for the Affine MLS with QR or Pivoting QR are in the 30-40 range for the main nodes and less than 1000 for the ghost nodes. The Gegenbauer basis Affine MLS again produces the lowest condition numbers. Table 7.14 presented the fourth-order condition numbers of $\mathbf{M}$.

Table 7.14: Average & Maximum Condition Number of $\mathbf{M}$ on Laminar Flat Plate Grids: 4^{th}-Order MLS.

Method	Weight	Node			
		Main		Ghost	
		avg	max	avg	max
SPD	Iso.	$2.42E22$	$1.45E25$	$3.54E23$	$7.89E25$
SPD	Ani.	$2.61E23$	$1.46E26$	$4.16E24$	$5.72E26$
SPD	Affine	84320	$1.48E5$	$5.43E6$	$9.28E6$
QR	Iso.	$1.60E9$	$5.38E11$	$1.03E10$	$9.38E11$
QR	Ani.	$4.75E9$	$1.44E12$	$4.13E10$	$2.86E12$
QR	Affine	309	430	2163	2994
PQR	Iso.	$1.27E10$	$3.86E12$	$8.25E10$	$9.08E12$
PQR	Ani.	$4.15E9$	$1.24E13$	$3.73E11$	$2.46E13$
PQR	Affine	307	418	2111	2840
PQR	Affine, Ch2	161	260	1290	1679
PQR	Affine, Geg	92	157	764	966

The non-Affine MLS condition numbers are absolutely terrible, and in no way ever should these methods be used for these types of grids. On the other hand, the Affine MLS methods have condition numbers that are excellent, and are only 10 times larger than the third-order condition numbers in Table 7.13. Reviewing all of the results from Tables 7.12-7.14, the Affine MLS reconstruction method proves it is the only feasible method to reconstruct gradients on these types of grids. The condition numbers for any of the other methods are so large the inverses computed from them are worthless. Using the orthogonal basis generates lower condition numbers for $\mathbf{M}$.

Overall, the Affine MLS method consistently produces the lowest condition number on all the meshes presented. While the isotropically weighed MLS will sometimes produce comparable

condition numbers, the Affine MLS should be used since it's performance is mesh independent. Also, using an orthogonal basis produces a lower condition number for $\mathbf{M}$ without any additional cost. For the cases presented here, the QR and Pivoting QR method have similar condition numbers, though the Pivoting QR method always produces lower condition numbers, and should be used when building the MLS system regardless of the method. Finally, as was shown here, a general matrix inversion method should never be used, as it always produces the largest conditioned systems.

7.2 Boundary Conditions

This section presents results for the boundary conditions discussed in Section 5.4. First, results for the periodic boundary conditions presented in Section 7.2.1 are presented. Next, the implementation of Navier–Stokes Characteristic Boundary Conditions discussed in Sections 5.4.2.3-5.4.2.6. The section concludes with applications on an inviscid bump-in-channel flow and a laminar flat-plate.

7.2.1 Higher-Order Periodic Boundary Conditions

This section presents three cases highlighting the improved periodic boundary conditions of presented herein. Good periodic boundaries should be $C^{\text{order}+1}$ continuous and not distort the flow. The cases presented herein are:

- 2D TGV: 7.2.1.1
- Stationary Vortex: 7.2.1.2

As mentioned previously, these boundary conditions have been used and published in Freno et al. [53], Freno et al. [54] and Krath et al. [93]. The results presented herein will focus on the differences between the original and new periodic boundaries with the 2D TGV case. Additionally, the boundaries will be used to aid in showing the differing dissipation rates for a stationary vortex.

7.2.1.1 2D Taylor–Green Vortex

This section presents the results for a two-dimensional Taylor–Green Vortex. The 2D TGV flow is used to show the effectiveness of the new periodic boundaries, and compare the new paradigm to the old periodic boundary description. The case is a good test for the periodic boundaries as the classical Taylor–Green Vortex (TGV) problem described by the 4^{th} *International Workshop on High-Order CFD Methods*, requires $C^{\text{order}+1}$ continuous periodic boundaries for simulation. This is a two-dimensional version of the flow, described by:

$$u = V_0 \sin\frac{x}{L}\cos\frac{y}{L} \tag{7.1}$$

$$v = -V_0 \cos\frac{x}{L}\sin\frac{y}{L} \tag{7.2}$$

$$w = 0 \tag{7.3}$$

$$p = p_0 + \frac{\rho_0 V_0^2}{16}\left(\left(\cos\frac{2x}{L} + \cos\frac{2y}{L}\right)\right) \tag{7.4}$$

where V_0, p_0, and ρ_0 are the initial reference states and L is the length of the domain. The domain is a periodic box defined as $-\pi L \leq x, y \leq \pi L$, where L is the characteristic length, and $z = [0 - 0.001]$m. The initial flow conditions are given in Table 7.15.

Table 7.15: Initial Conditions for 2D TGV Case.

Quantity	Value	Units
Freestream Mach No., M_∞	0.10	-
Reynold No., Re	1600	-
Characteristic Length, L	1	m
Freestream Temperature, T_0	300	K
Dynamic Viscosity, μ	1.716×10^{-5}	kg/(m s)

The Mach number is defined as

$$M_0 = \frac{V_0}{c_0} \tag{7.5}$$

where the sound speed c_0 determined from the initial uniform temperature field T_0. The meshes used in this section are scaled versions of the NSCBC grids shown in Figure 7.2, but with all the boundaries being periodic. The flow is simulated using a $CFL = 0.25$. The initial pressure field with velocity vectors is shown in Figure 7.6.

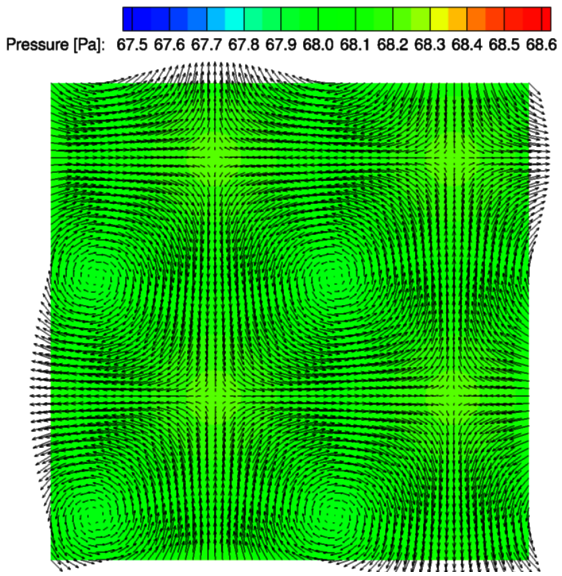
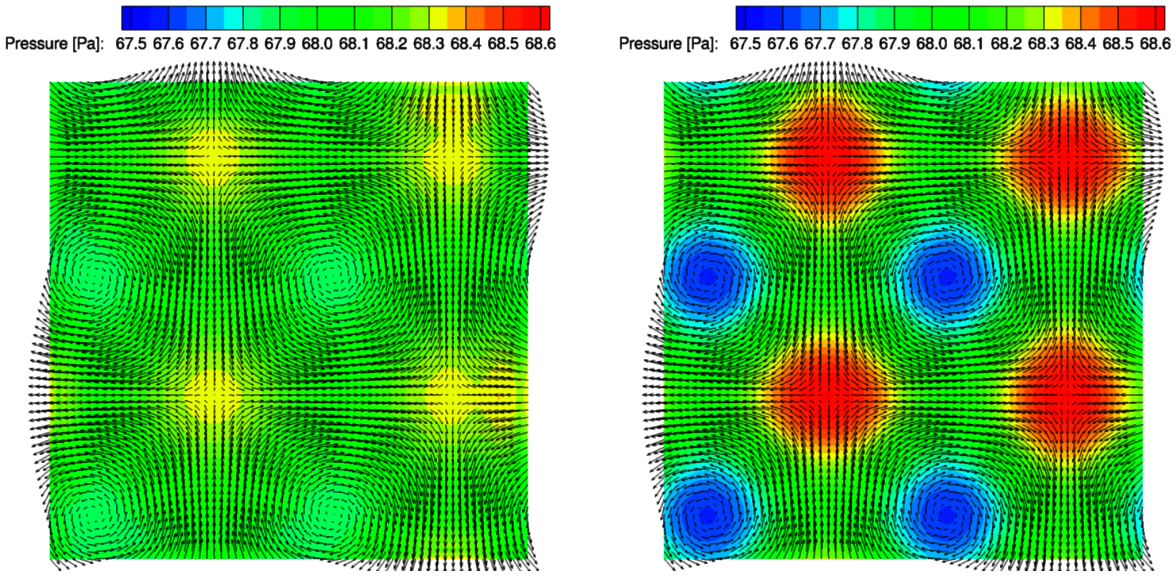


Figure 7.6: Initial Pressure Contour and Velocity Vectors for the 2D TGV Case.

Figure 7.6 highlights the vortical structures, showing the periodic nature of the flow. As this is only a case to highlight the improvements to the boundaries, and since the third- and fourth-order results do not show any differences from the improved second-order results, only second-order results for the old and new periodic boundaries are shown. Figure 7.7 shows the results after 10000 iterations.



(a) Old Boundaries

(b) New Boundaries

Figure 7.7: Comparison of Pressure Contours and Velocity Vectors for the 2D TGV Case with the Old and New Periodic Boundaries.

Figure 7.7 clearly shows the distortions caused by the old periodic boundaries in the vortexes. It is interesting that the velocity field appears unaffected, so for non-vortex dominated flows it is possible the old boundaries might fair well. However, the issues with the old boundaries also affect the time step. Since the volumes are approximately half (or a quarter for the corners) what they should be, the time step for an unsteady case is incorrect when using the old boundary. What causes the pressure rise or fall near the old periodic boundaries is primarily a due to the geometric issues discussed in Section 5.4.1, namely the edge-based area. This error alone destroys the accuracy of the fluxes at the boundaries, leading to the errors seen in Figure 7.7. The gradients not being $C^{\text{order}+1}$ continuous does contribute to the error as well, but not as significantly as the geometric problems. The new periodic boundaries, being $C^{\text{order}+1}$ continuous, represent a huge improvement to the flow field and to the solver as a whole, allowing for complex periodic flows to be simulated correctly.

7.2.1.2 Stationary Vortex

This section presents the results of a stationary vortex using the Moving Least-Squares methodology to compute gradients. The vortex, in the absence of a background flow, should only degrade due to numerical dissipation. Higher-order methods typically have much less numerical dissipation, and this case is designed to show the improvement due to the increased accuracy. The flow was initialized by the vortex equations described by Balsara & Shu [7]. The equations are:

$$\begin{aligned} r &= \sqrt{(x-x_c)^2+(y_c-y)^2} \\ \alpha &= e^{1-r^2} \\ \beta &= \frac{\beta}{2\pi\sqrt{\gamma}} e^{\frac{1}{2}(1-r^2)} \\ \Delta T &= -\alpha \frac{\beta^2(\gamma-1)}{8\gamma\pi^2} \\ T &= T_0 + \Delta T \\ u &= u_0 + \beta(y_c - y) \\ v &= v_0 + \beta(x - x_c) \\ \rho &= e^{\frac{\ln T}{\gamma-1}} \\ P &= \frac{e^{\gamma \ln \rho}}{\gamma} \end{aligned} \tag{7.6}$$

where β is the vortex strength, (x_c, y_c) is the location of the vortex center. The initial Mach contours are given in Figure 7.8.

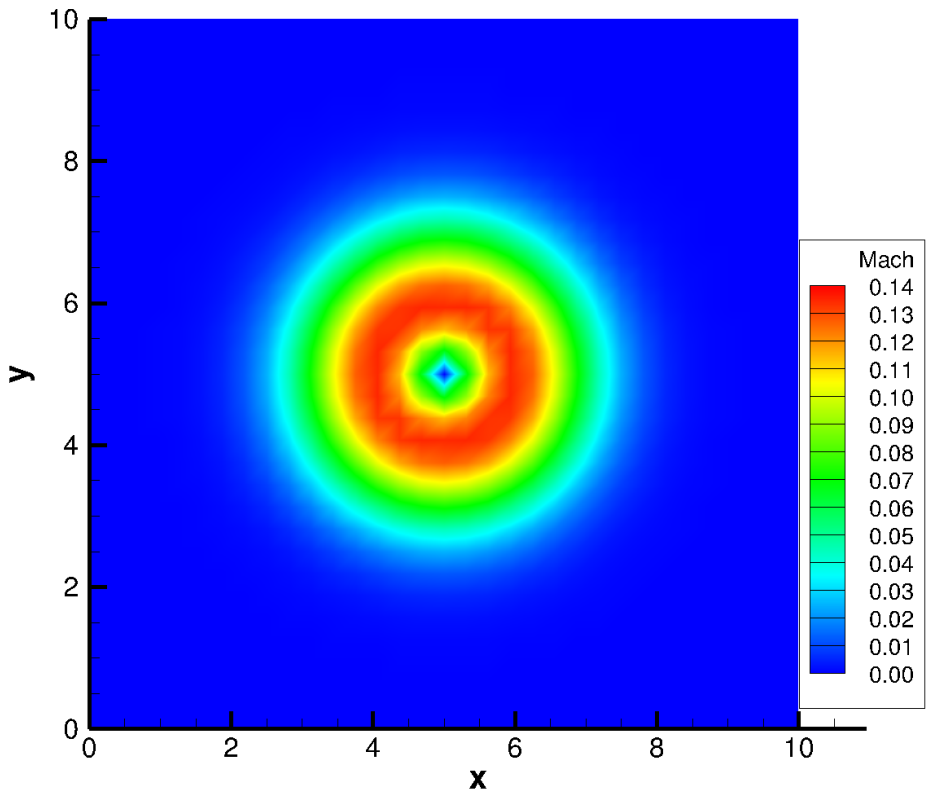
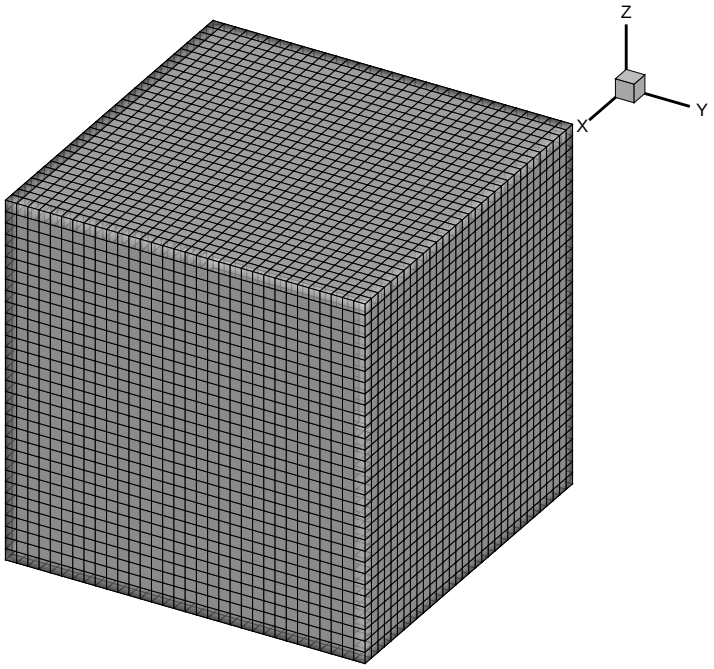
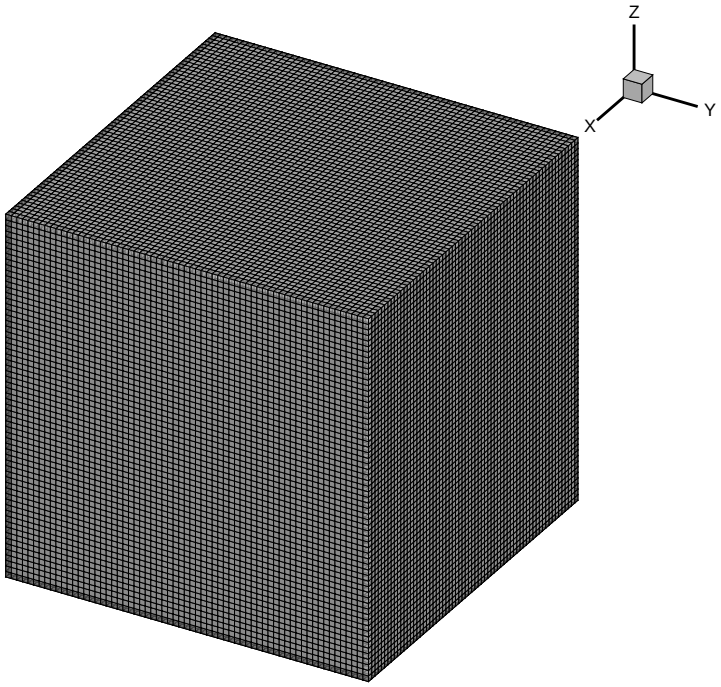


Figure 7.8: Initial Mach Contours of the Stationary Vortex Described by Balsara & Shu [7].

The case was simulated with $\beta = 1$, $T_0 = 1$, and $(u_0, v_0) = 0$. The case was run steady and inviscid for 1000 iterations. The case was simulated on two structured meshes, both of which were $[0 - 10]^3$ with nodal discretizations of 33^3 and 66^3 with all boundaries modeled as periodic. Figures 7.9(a) and 7.9(b) show both of the meshes used in this case.



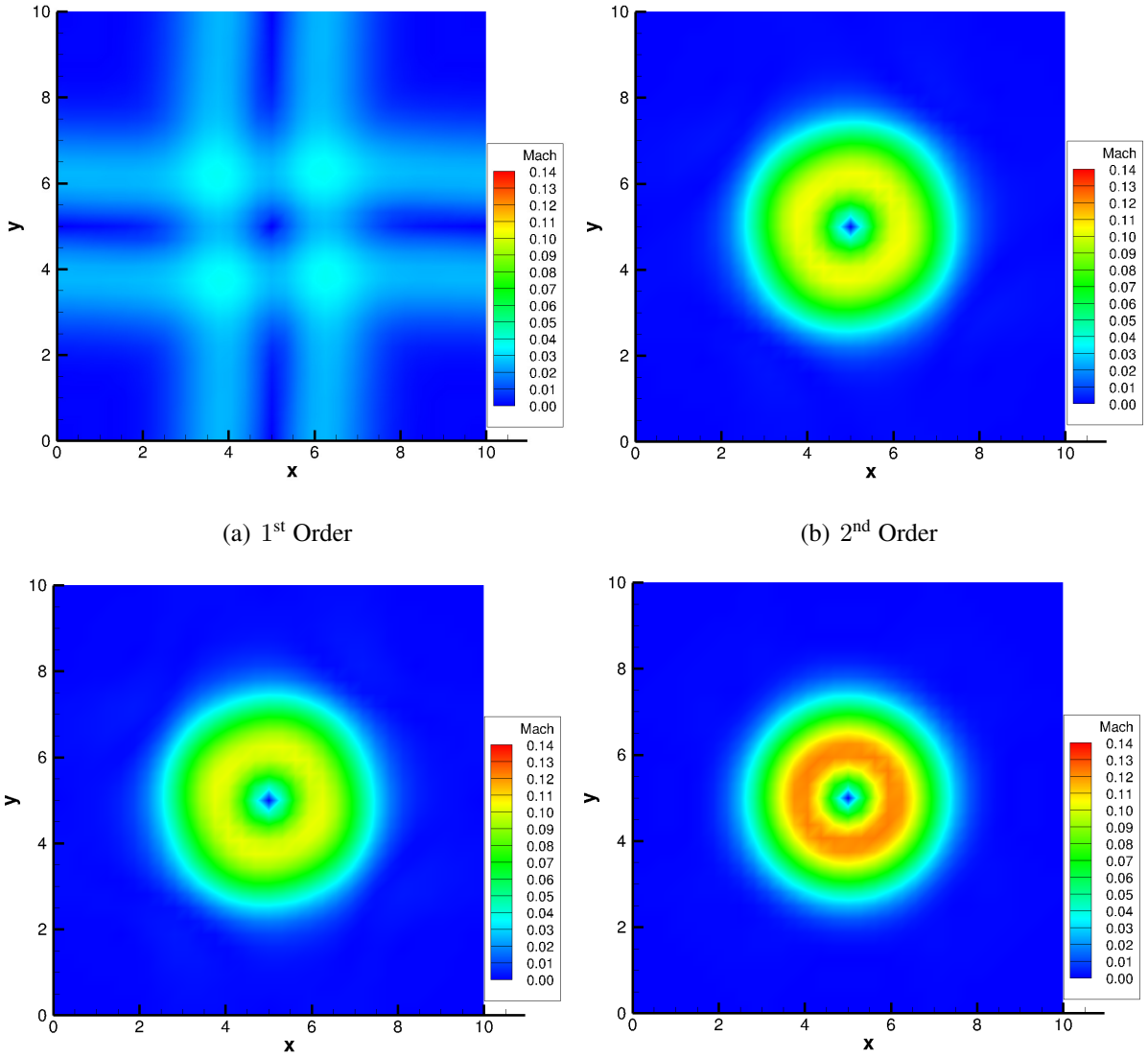
(a) 33^3 Mesh



(b) 66^3 Mesh

Figure 7.9: Meshes Used for Stationary Vortex Case.

Figures 7.10 show the resulting Mach contours after 1000 iterations on the 33^3 mesh.



(a) 1^{st} Order

(b) 2^{nd} Order

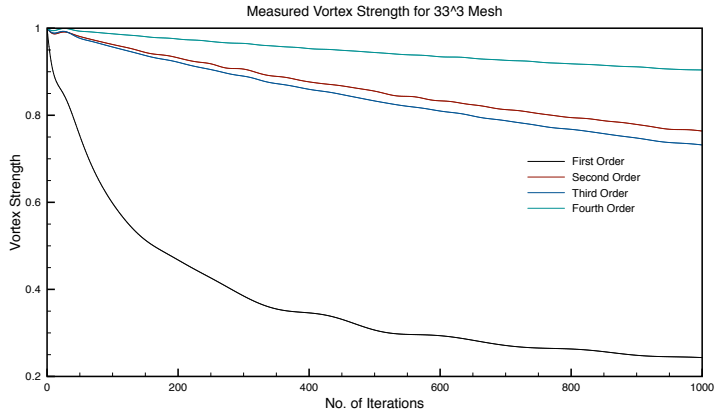
(c) 3^{rd} Order

(d) 4^{th} Order

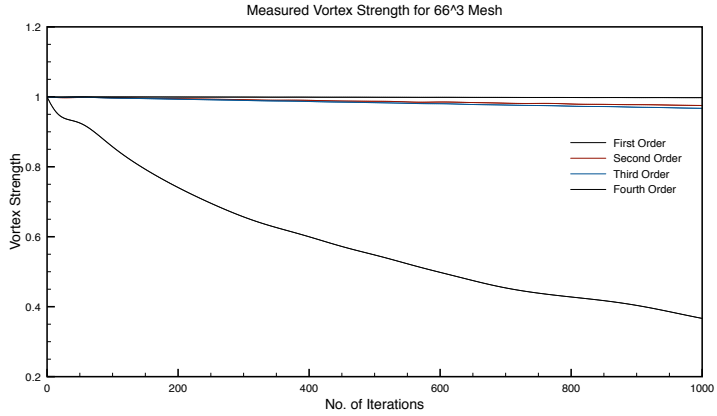
Figure 7.10: $1^{st} - 4^{th}$-Order Stationary Vortex Results.

Figure 7.10(d) clearly shows that the fourth-order simulation has the least numerical dissipation, while Figure 7.10(a) shows that the first-order simulation completely destroys the vortex. It is much more difficult to discern differences between the second and third-order results, but they are

certainly in between the first and fourth-order results. To gain a better perspective of the results, the vortex decay rates for both meshes are examined using using (7.6). The vortex strength β is computed as a function of the number of iterations, and these results are presented for each order and mesh in Figures 7.11(a) and 7.11(b).



(a) 33^3 Mesh



(b) 66^3 Mesh

Figure 7.11: Vortex Strength as a Function of Iterations for Stationary Vortex Case for Each Mesh.

As with the Mach contours, it is clear that the fourth-order code shows the least decay of the vortex, while the first shows the most for both meshes. Surprisingly, the second-order code shows

slightly less decay of the vortex than the third-order code. When further looking at the maximum Mach numbers of the resulting flow, summarized in Table 7.16, we see that for the 33^3 mesh, the second-order code has a higher maximum Mach number than the third-order code. However, as one would expect, the third-order code has a higher maximum Mach number than the second-order code on the 66^3 mesh. It is therefore a toss-up based on these results if the second or third-order code is better, and this warrants further study. It is fair to conclude that the fourth-order code works as advertised, producing the best results by far for the stationary vortex case. Further studies of the true accuracy of the code are necessary to confirm the overall accuracy of the code for each of the specified orders.

Table 7.16: Maximum Mach Number for the Stationary Vortex Case.

	Mesh Size	
Order	33^3	66^3
	Max Mach No.	
$Exact$	.1347	
1	.0365	.0592
2	.1038	.1318
3	.1023	.1323
4	.1226	.1345

7.2.2 Navier–Stokes Characteristic Boundary Conditions with Ghost Nodes

This section presents results for the Navier–Stokes Characteristic Boundary Condition. To test the NSCBC, a vortex is convected out of the domain at speed M_∞. This case was described by Granet et al. [70], specifically case C. The vortex is defined using the stream function:

$$\Psi(\mathbf{x}) = \beta e^{\frac{(x-x_c)^2+(y-y_c)^2}{2R^2}} \tag{7.7}$$

where (x_c, y_c) is the vortex center (defined here as $(0, 0)$), R is the radius of the vortex, and β is the vortex strength. The velocity field is defined as:

$$\begin{aligned} u &= \frac{\partial \Psi}{\partial y} \\ v &= -\frac{\partial \Psi}{\partial x} \\ U_{\text{max}} &= \frac{\beta}{R\sqrt{e}} \end{aligned} \tag{7.8}$$

The pressure is defined as:

$$p = p_\infty - \frac{\rho \beta^2}{2R^2} e^{\frac{(x-x_c)^2+(y-y_c)^2}{R^2}} \tag{7.9}$$

where p_∞ is the reference pressure. The grids used herein were shown in Figure 7.2. Unlike Granet et al. [70], the top and bottom boundaries are not periodic, but instead the far-field boundaries defined in Section 5.4.2.5, since the characteristic boundaries are coupled as described in Section 5.4.2.6. The boundaries in the xy-plane are defined as periodic boundaries. The flow parameters for this case are defined in Table 7.17

Table 7.17: Initial Conditions for NSCBC Case.

Quantity	Value	Units
Freestream Mach No., M_∞	0.28	-
Freestream Velocity, u_o	10	m/s
Pressure, p_∞	101325	Pa
Vortex Strength, β	0.11	m^2/s
Radius, R	0.005	m

For each order and grid, the axial velocity and the normalized pressure:

$$p^*(\mathbf{x}, t^*) = \frac{p(\mathbf{x}, t^*) - p_\infty}{p(0,0) - p_\infty} = -(p(\mathbf{x}, t^*) - p_\infty)\frac{2R^2}{\rho\Gamma^2} \tag{7.10}$$

are plotted at the outlet $x = \frac{L}{2}$ at the non-dimensional time

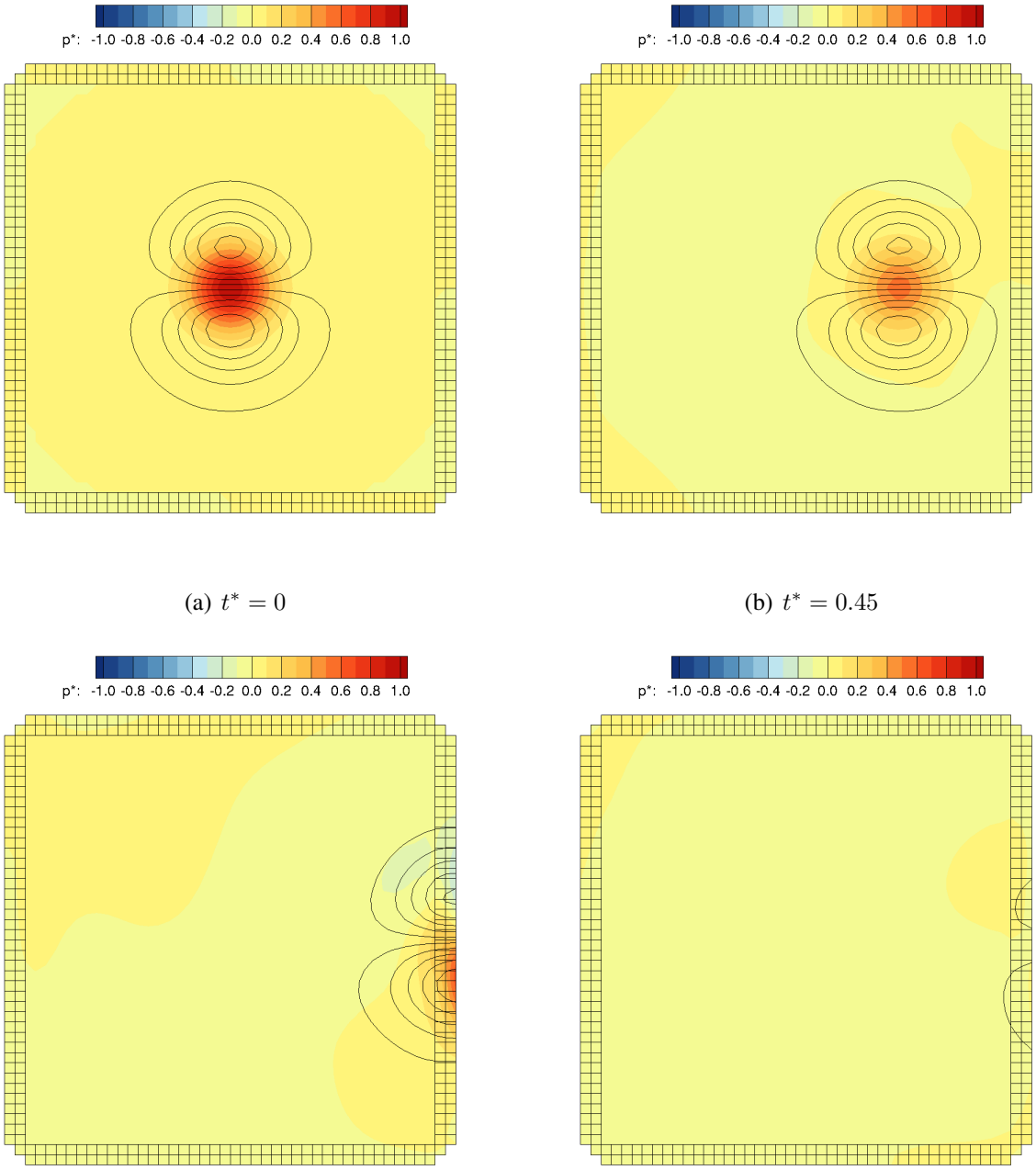
$$t^* = \frac{t}{\tau} = \frac{2u_o t}{L} \tag{7.11}$$

at $t^* = [1.25, 1.43, 1.5, 1.75]$. The normalized pressure contours and velocity iso-contours are also plotted at $t^* = [0, 0.45, 1.12, 1.43]$. Three cases are presented: the full NSCBC, the NSCBC without transverse terms, and the full NSCBC with fully-diffuse MLS derivatives. The results are presented for second- through fourth-orders. First-order results are not shown as the flow is too dissipative to correct translate the vortex out of the domain. When viewing the results, $t^* = 1.75$ is considered the time when the vortex has completely left the domain.

7.2.2.1 Full Derivatives with Transverse Terms

The first set of cases compute the full MLS derivatives described in Section 5.1.5.1 and the transverse terms in the NSCBC described in Sections 5.4.2.3- 5.4.2.6.

7.2.2.1.1 Second-Order Figure 7.12 shows the second-order structured mesh results.



(a) $t^* = 0$

(b) $t^* = 0.45$

(c) $t^* = 1.12$

(d) $t^* = 1.43$

Figure 7.12: Non-Dimensional Pressure Contours and Velocity Iso-Contours of Vortex Exiting Domain: 2^{nd}-Order Full NSCBC on the Structured Mesh.

As the vortex leaves the domain, the velocity iso-contours do not appear to be distorted at

all. The pressure field does appear to slightly amplify at the outlet and 'slug' downward in Figure 7.12(c) at $t^* = 1.12$ within the ghost domain. However, by $t^* = 1.43$ none of higher amplitude information has traveled upstream and has disappeared from the ghost domain. This slugging is effectively a numerical energy sink to keep reflections minimized. So while it may appear that the vortex is amplified outside of the domain, this really is of little concern since the NSCBC equations presented in Sections 5.4.2.3- 5.4.2.6 are designed this way. From here onward in the NSCBC, when an amplification is seen in the ghost region (or the so-called slugging effect), it is of no ill-concern but rather desired and expected. The pressure and velocity profiles at the outlet are shown in Figures 7.13 and 7.14.

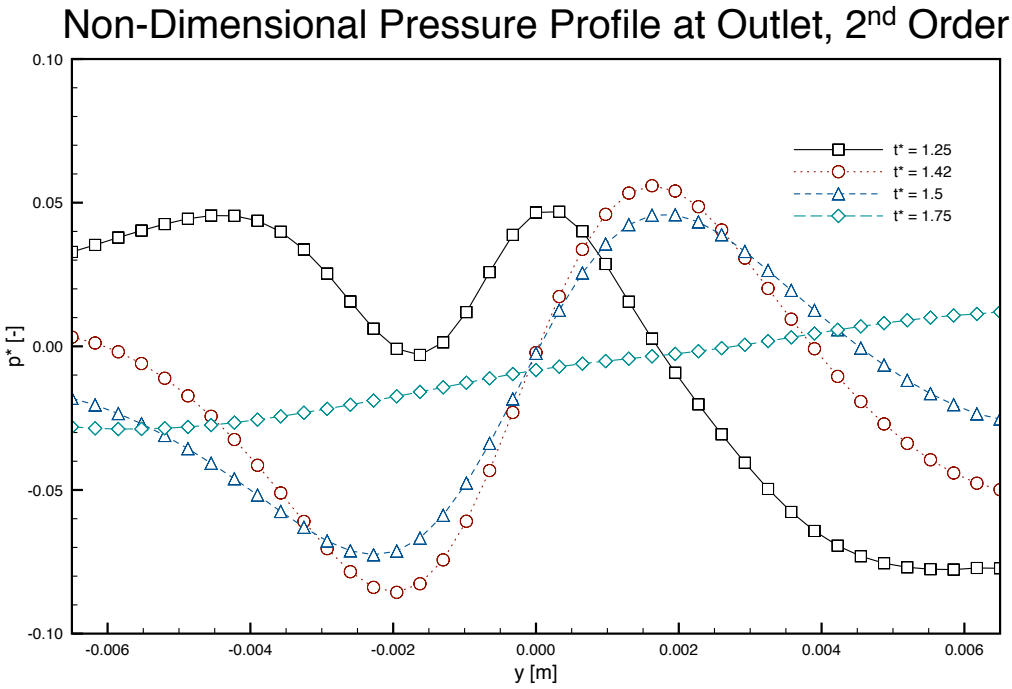


Figure 7.13: Non-Dimensional Pressure at Outlet: 2nd-Order Full NSCBC on the Structured Mesh.

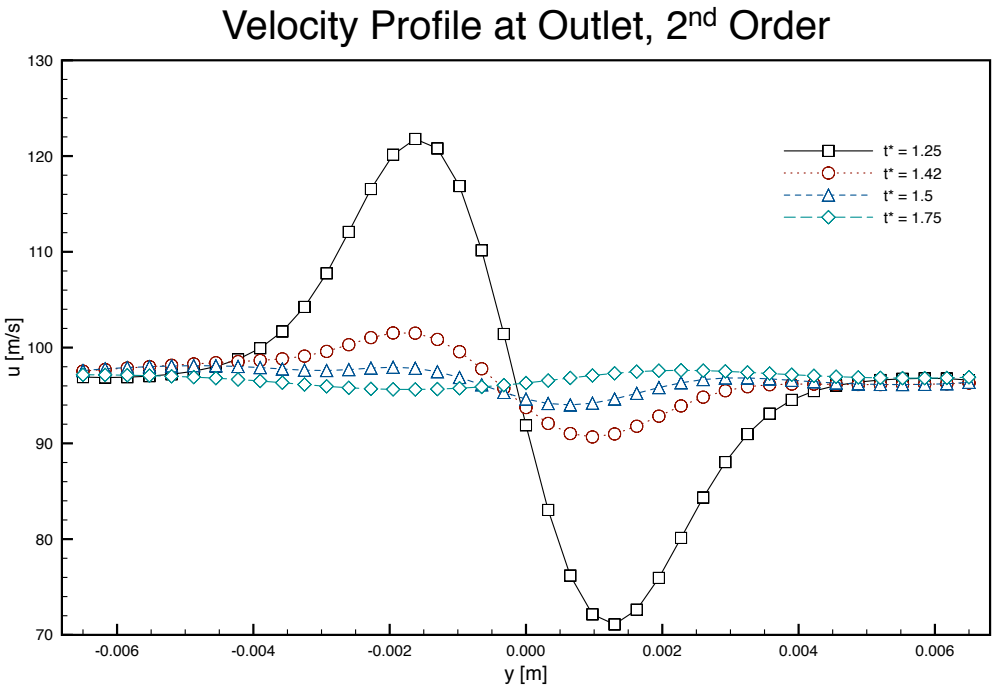
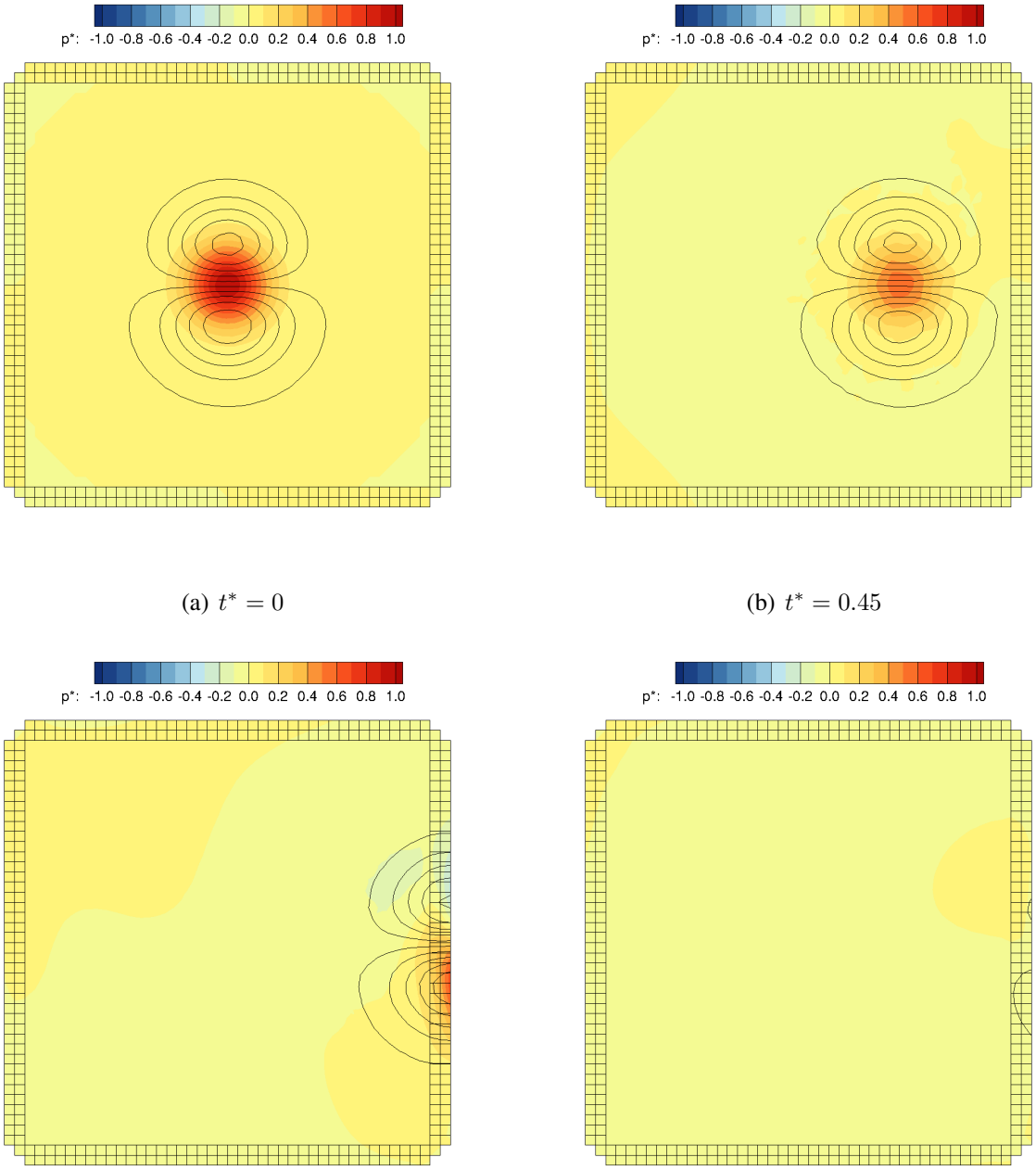


Figure 7.14: Non-Dimensional Velocity Profiles at Outlet: 2^{nd}-Order Full NSCBC on the Structured Mesh.

From Figure 7.13, the pressure at $t^* = 1.25$ is non-symmetric. However, the remaining non-dimensional pressure profiles are symmetric and decreasing with time to a 'flat' profile at $t^* = 1.75$. The velocity profiles in Figure 7.14 are symmetric and centered at approximately $y = 0$, as well as decreasing through the time period. Figure 7.15 shows the second-order unstructured mesh results.



(a) $t^* = 0$

(b) $t^* = 0.45$

(c) $t^* = 1.12$

(d) $t^* = 1.43$

Figure 7.15: Non-Dimensional Pressure Contours and Velocity Iso-Contours of Vortex Exiting Domain: 2^{nd}-Order Full NSCBC on the Unstructured Mesh.

The unstructured mesh results appear qualitatively similar to the second-order structured re-

sults, with the same 'slugging' at $t^* = 1.12$ as the vortex exits the domain as shown in Figure 7.15(c). The velocity iso-contours remain intact and virtually non-distorted as the vortex exits the domain. The pressure and velocity profiles at the outlet are shown in Figures 7.16 and 7.17.

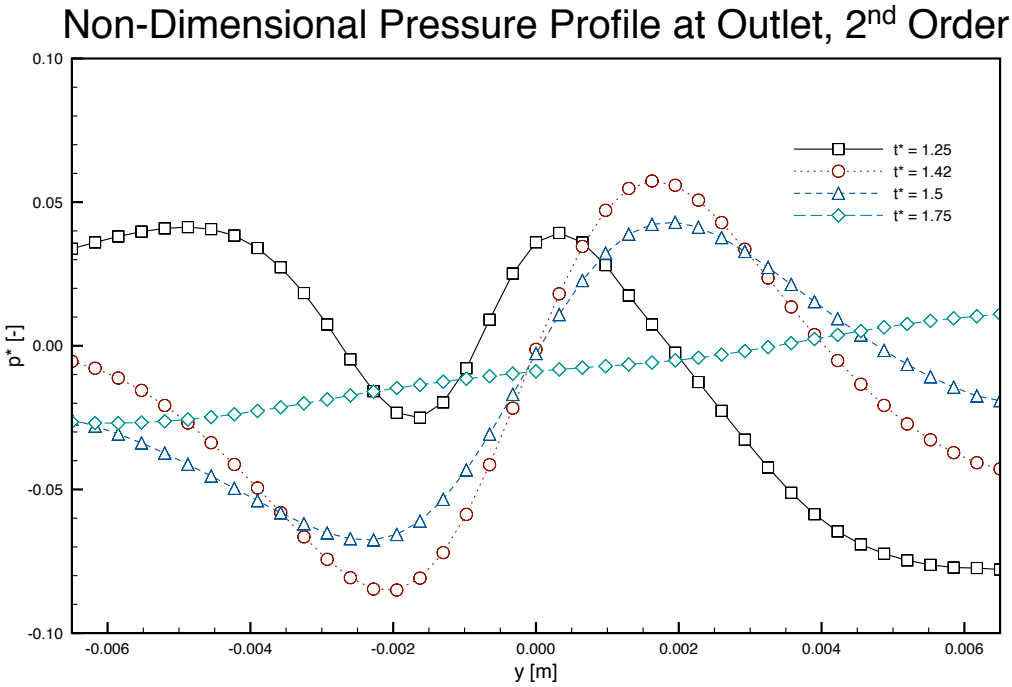


Figure 7.16: Non-Dimensional Pressure at Outlet: 2^{nd}-Order Full NSCBC on the Unstructured Mesh.

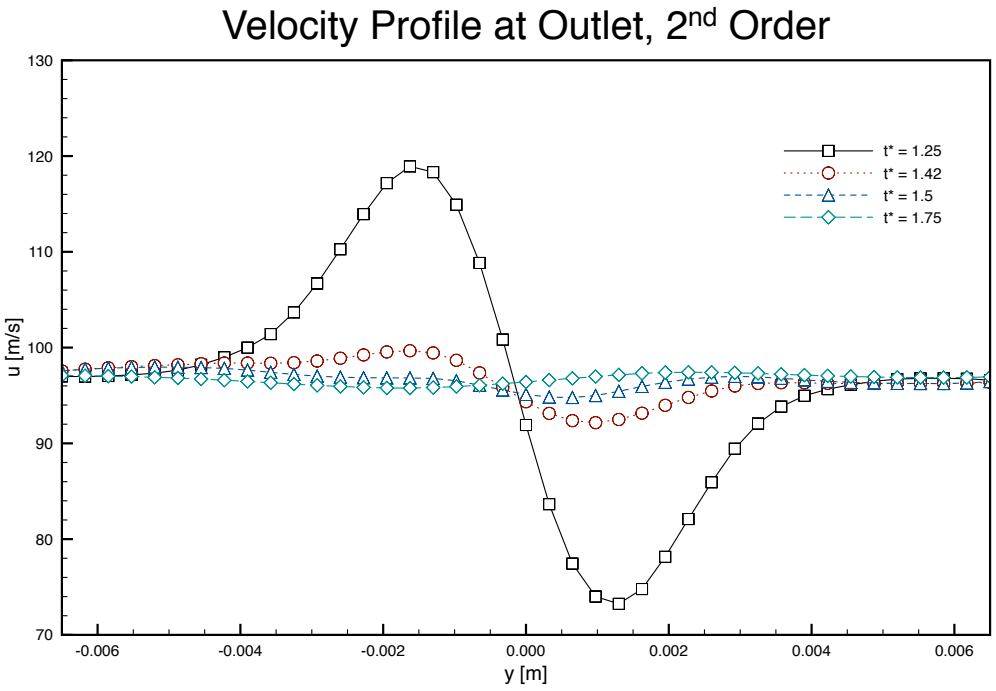
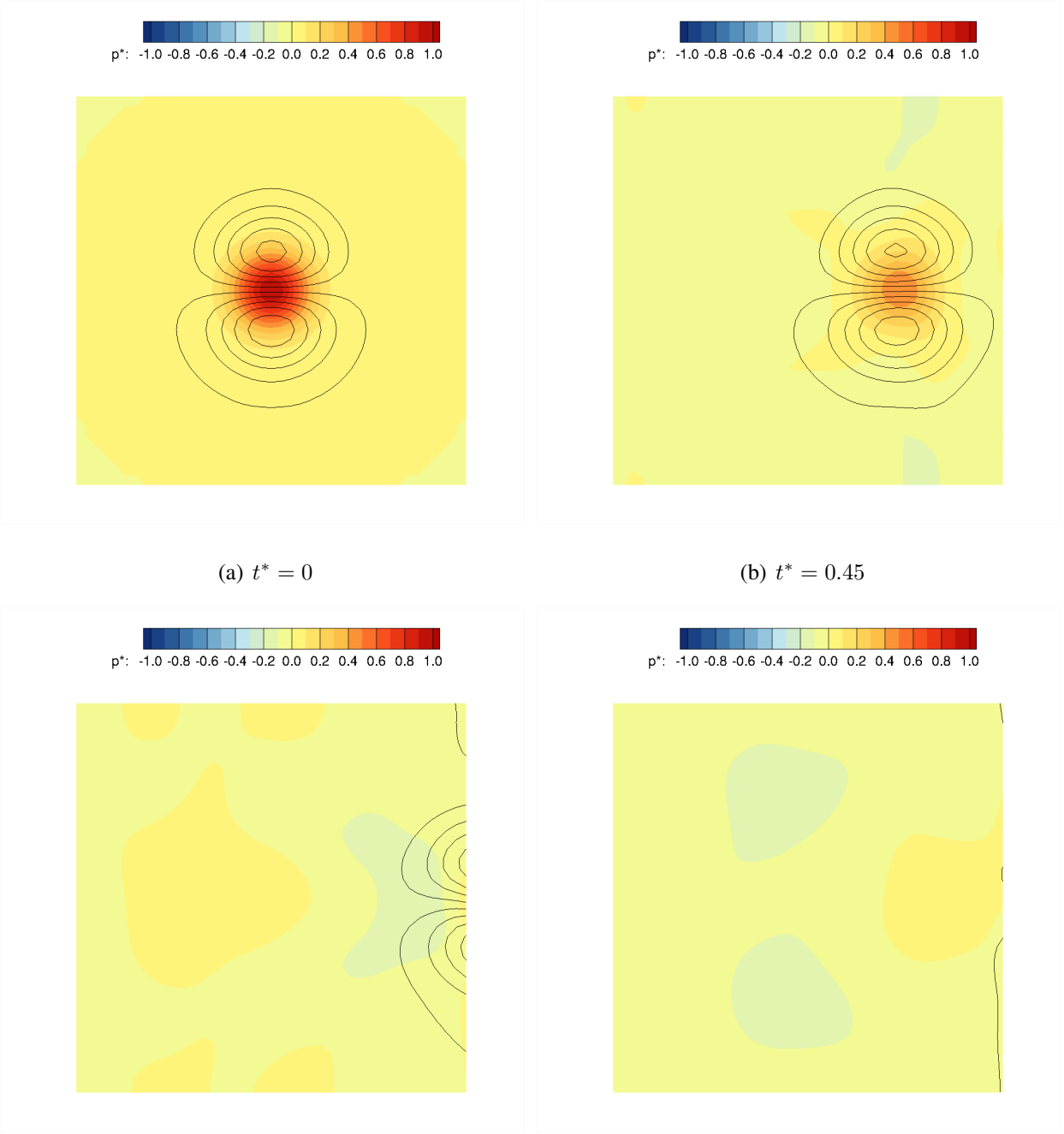


Figure 7.17: Non-Dimensional Velocity Profiles at Outlet: 2^{nd}-Order Full NSCBC on the Unstructured Mesh.

As with the structured mesh, Figure 7.16 shows that pressure at $t^* = 1.25$ is non-symmetric. However, the remaining non-dimensional pressure profiles are symmetric and decreasing with time to a 'flat' profile at $t^* = 1.75$. The velocity profiles in Figure 7.17 are symmetric and centered at approximately $y = 0$, as well as decreasing through the time period.

It is also illustrative to compare the old boundary conditions described in Chapter 4 to the NSCBC boundary conditions described in Sections 5.4.2.3-5.4.2.6 of Chapter 5. For results using the numerical methods of Chapter 4, the inlet and outlet boundary conditions are nonreflecting using Riemann invariants [12, Chapter 8], and the far-field boundary is effectively a symmetry boundary. The gradients are computed using the LSQR method [91]. Figure 7.18 shows the second-order structured mesh results with the boundary conditions of Chapter 4.



(a) $t^* = 0$

(b) $t^* = 0.45$

(c) $t^* = 1.12$

(d) $t^* = 1.43$

Figure 7.18: Non-Dimensional Pressure Contours and Velocity Iso-Contours of Vortex Exiting Domain: 2^{nd}-Order Old Boundary Conditions on the Structured Mesh.

As the vortex attempts to leave the domain, the vortex effectively is 'smashed' into the outlet,

reflecting back most of the information upstream rather than down. This is readily apparent in Figure 7.18(c), where the velocity contours are stretched out along the boundary with large pressure packets now present upstream of the outlet. The pressure and velocity profiles for each boundary condition are compared in Figures 7.19 and 7.20.

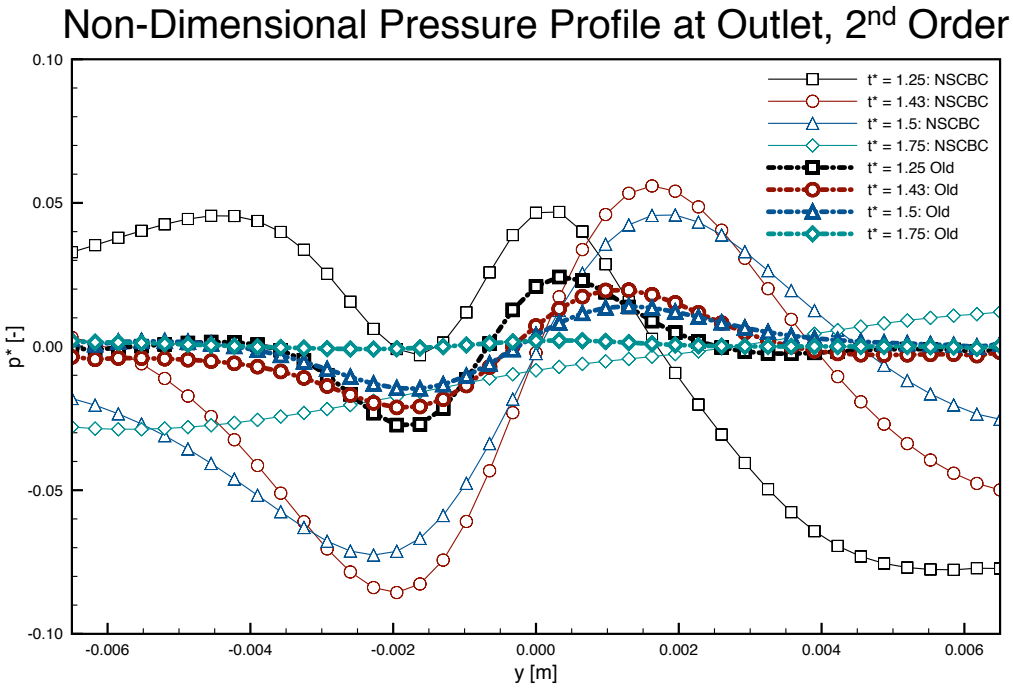


Figure 7.19: Non-Dimensional Pressure at Outlet: Comparison of 2nd-Order Boundary Conditions on Structured Mesh.

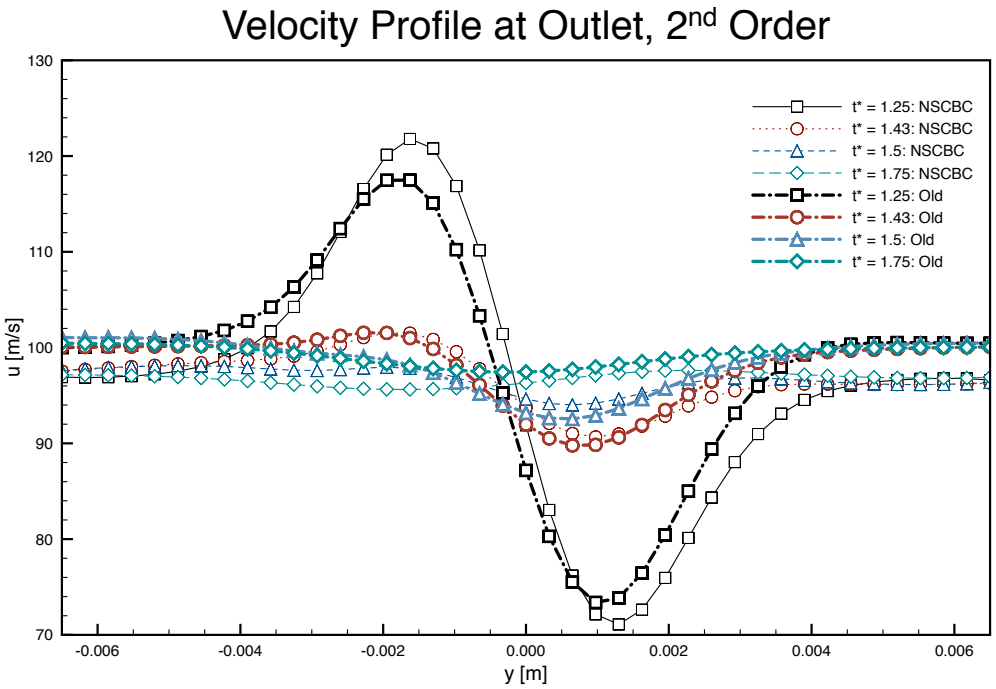
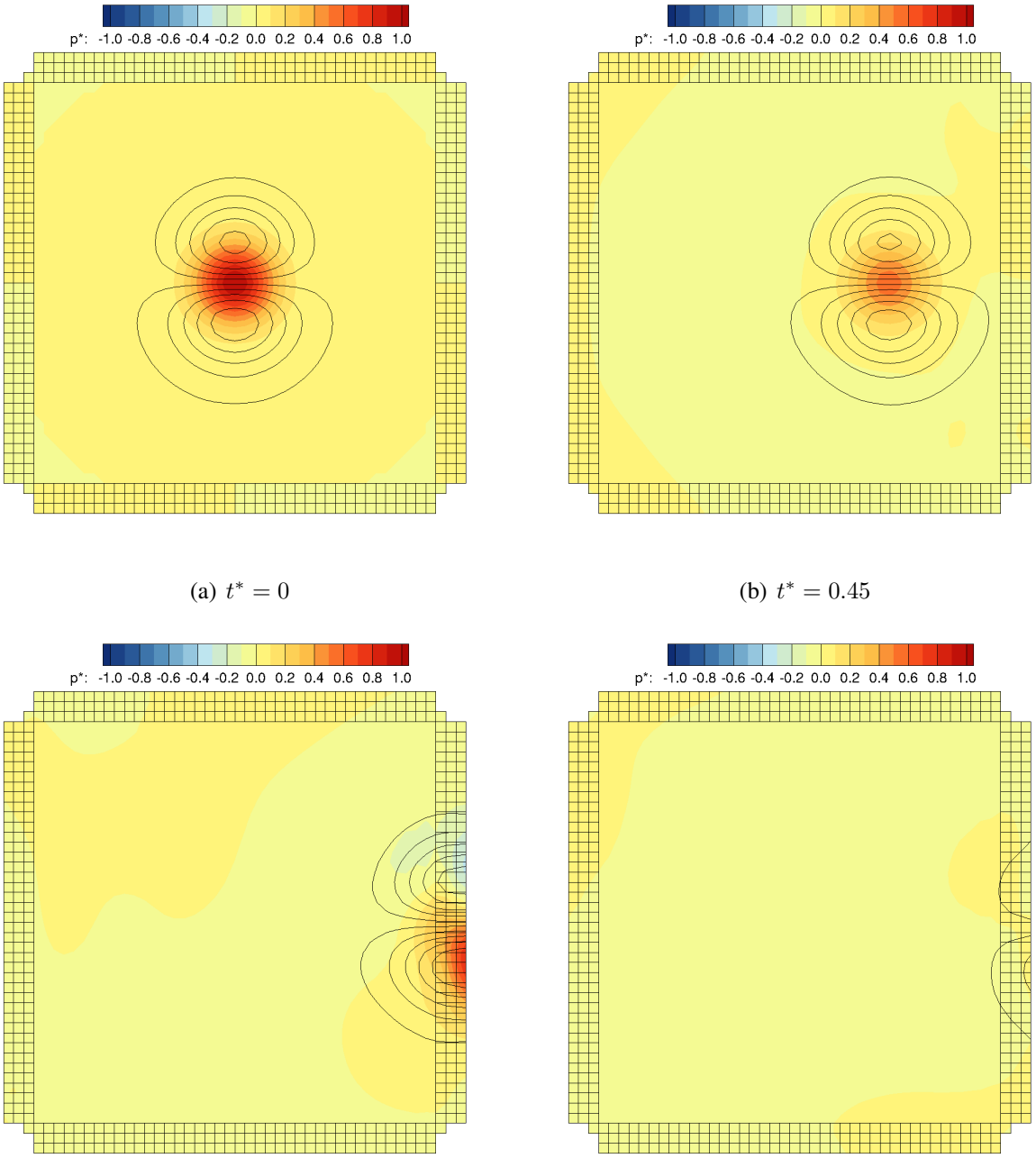


Figure 7.20: Non-Dimensional Velocity Profiles at Outlet: Comparison of 2^{nd}-Order Boundary Conditions on Structured Mesh.

If one where to examine Figure 7.19 only, the old boundaries would appear to be better than the NSCBC in terms of outlet pressure magnitude. However, recalling that the vortex has not fully left the domain until $t^* = 1.75$, the velocity profiles from the old boundaries observed in Figure 7.20 are clearly wrong, since the velocity is effectively the freestream. The pressures are low at the outlet with the old boundaries since the information is mostly reflected back upstream. Figure 7.12 clearly shows the effectiveness of using the NSCBC and improvement over the old boundary conditions.

7.2.2.1.2 Third-Order Figure 7.21 shows the third-order structured mesh results.



(a) $t^* = 0$

(b) $t^* = 0.45$

(c) $t^* = 1.15$

(d) $t^* = 1.43$

Figure 7.21: Non-Dimensional Pressure Contours and Velocity Iso-Contours of Vortex Exiting Domain: 3^{rd}-Order Full NSCBC on the Structured Mesh.

The structured third-order results appear to produce similar results to the second-order results

shown previously in Figure 7.21. The slugging in the pressure field is still apparent at $t^* = 1.12$, but has disappeared at $t^* = 1.43$. The velocity iso-contours remain fairly uniform as the vortex leaves the domain. The energy increase observed in the ghost region in Figure 7.21 is what facilitates the non-reflecting nature of the boundary conditions, acting as a energy sink. The pressure and velocity profiles at the outlet are shown in Figures 7.22 and 7.23.

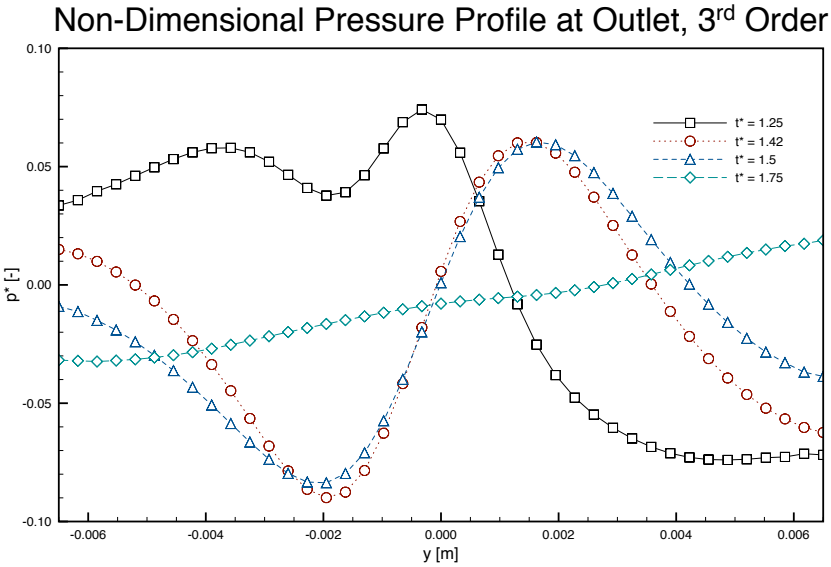


Figure 7.22: Non-Dimensional Pressure at Outlet: 3^{rd}-Order Full NSCBC on the Structured Mesh.

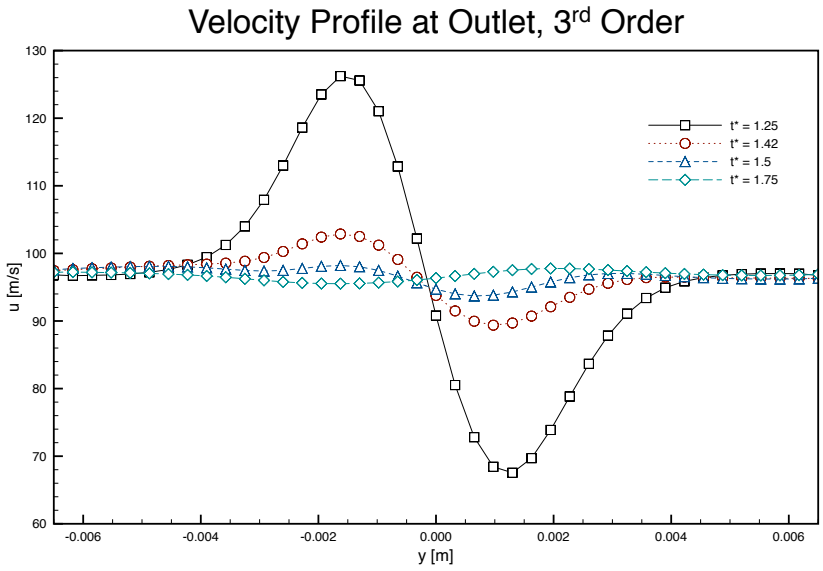
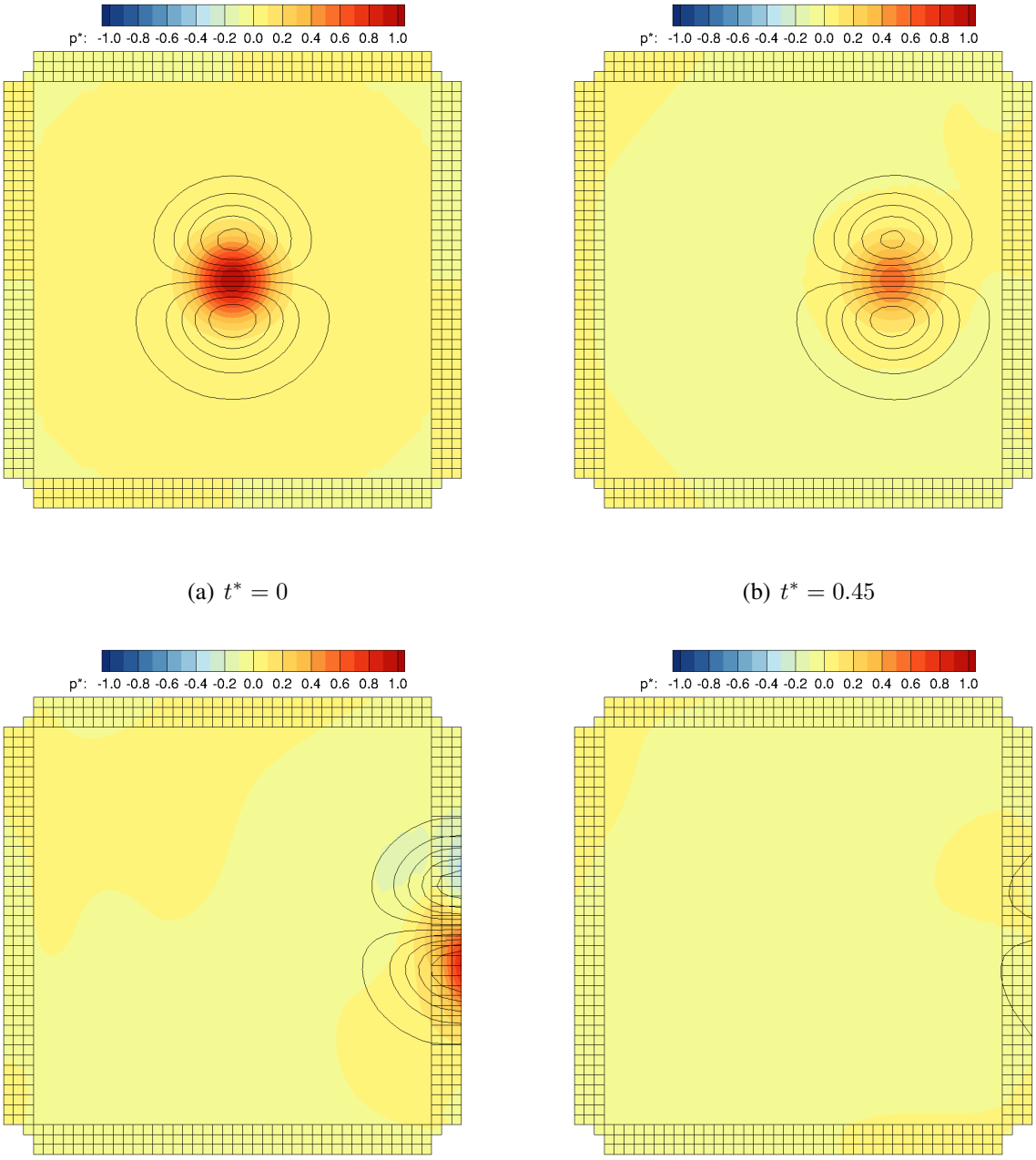


Figure 7.23: Non-Dimensional Velocity Profiles at Outlet: 3^{rd}-Order Full NSCBC on the Structured Mesh.

From Figure 7.22, the pressure at $t^* = 1.25$ is non-symmetric. The pressure profiles for the next two time periods are nearly identical, but the pressure profile at the end of the period is 'flat'. The velocity profiles in Figure 7.23 are symmetric and centered at approximately $y = 0$, as well as decreasing to nearly the freestream velocity at the end of the time period. Figure 7.24 shows the third-order unstructured mesh results.



(a) $t^* = 0$

(b) $t^* = 0.45$

(c) $t^* = 1.12$

(d) $t^* = 1.43$

Figure 7.24: Non-Dimensional Pressure Contours and Velocity Iso-Contours of Vortex Exiting Domain: 3^{rd}-Order Full NSCBC on the Unstructured Mesh.

The unstructured mesh results are, as with the previous cases, nearly identical to the struc-

tured results, with the pressure slugging at $t^* = 1.12$. The pressure and velocity profiles for the unstructured mesh are shown in Figures 7.25 and 7.26.

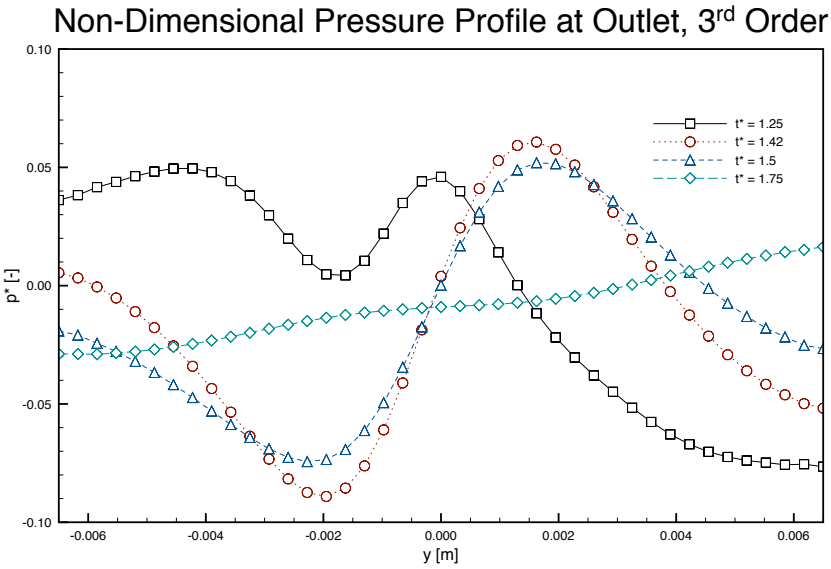


Figure 7.25: Non-Dimensional Pressure at Outlet: 3^{rd}-Order Full NSCBC on the Unstructured Mesh.

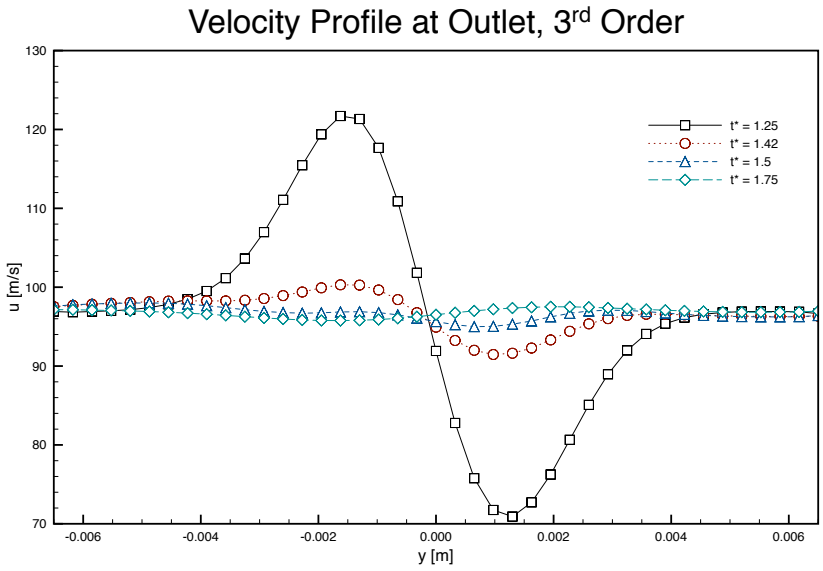
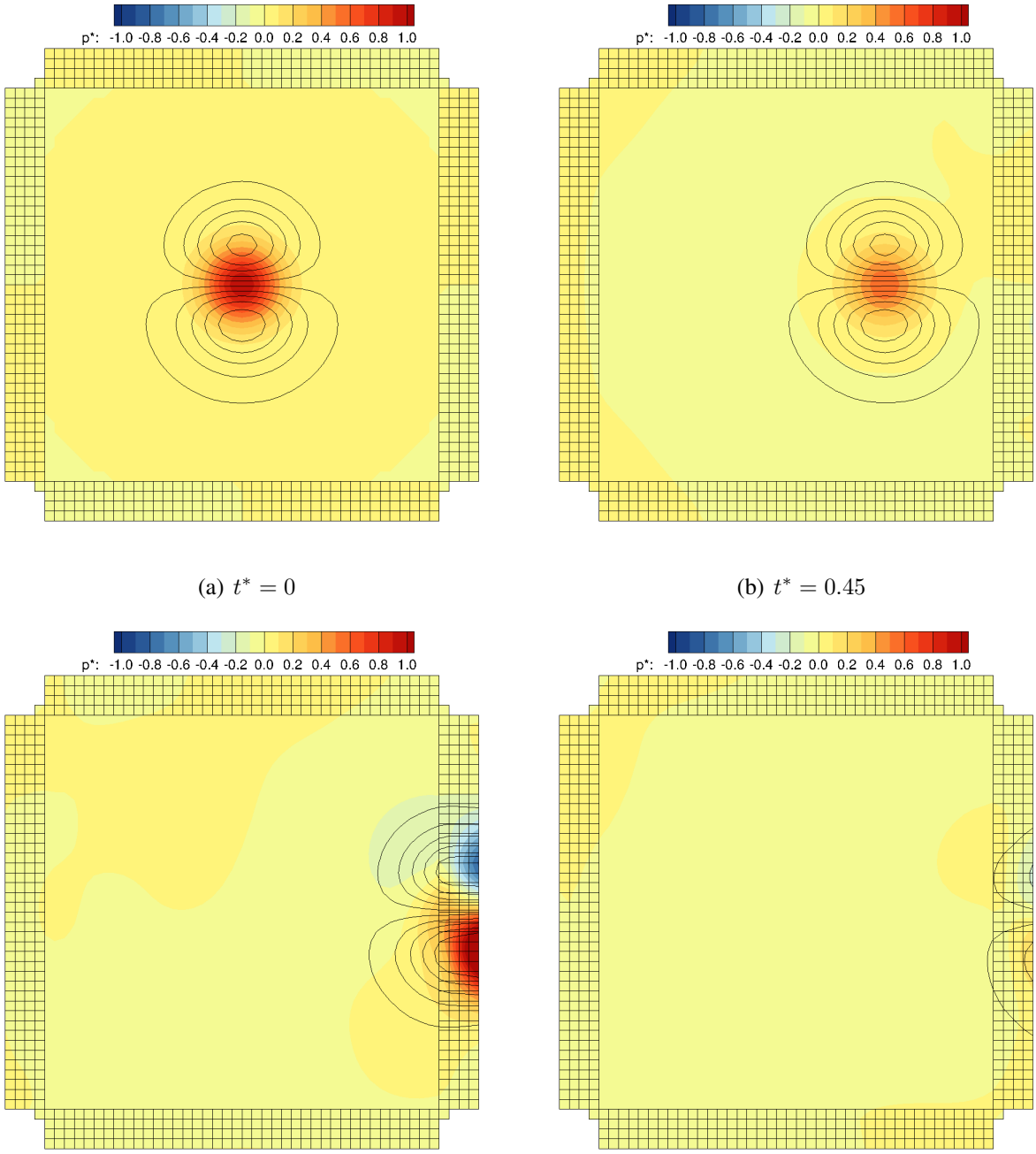


Figure 7.26: Non-Dimensional Velocity Profiles at Outlet: 3^{rd}-Order Full NSCBC on the Unstructured Mesh.

The unstructured profiles in Figures 7.25 and 7.26 have the same trends as those seen on the structured mesh results shown in Figures 7.22 and 7.23. Effectively, the results on the boundaries only differ because of the internal unstructured mesh, which will add additional dissipation to the simulation decreasing the strength of the vortex that actually reaches the boundary compared to the structured mesh.

7.2.2.1.3 Fourth-Order Figure 7.27 shows the fourth-order structured mesh results.



(a) $t^* = 0$

(b) $t^* = 0.45$

(c) $t^* = 1.12$

(d) $t^* = 1.43$

Figure 7.27: Non-Dimensional Pressure Contours and Velocity Iso-Contours of Vortex Exiting Domain: 4^{th}-Order Full NSCBC on the Structured Mesh.

For the fourth-order results, the non-dimensional pressure at $t^* = 1.12$ is in the ghost region is

large, approximately one. However, if these results are compared to second- and third-order results in Figures 7.12 and 7.21, the non-dimensional pressure just inside the domain is qualitatively similar for each case. As mentioned earlier, this increase in energy in the ghost region is what facilitates the non-reflecting nature of the boundary conditions, as this acts as an energy sink. Figures 7.28 and 7.29 show the pressure and velocity profiles at the outlet fourth-order structured mesh.

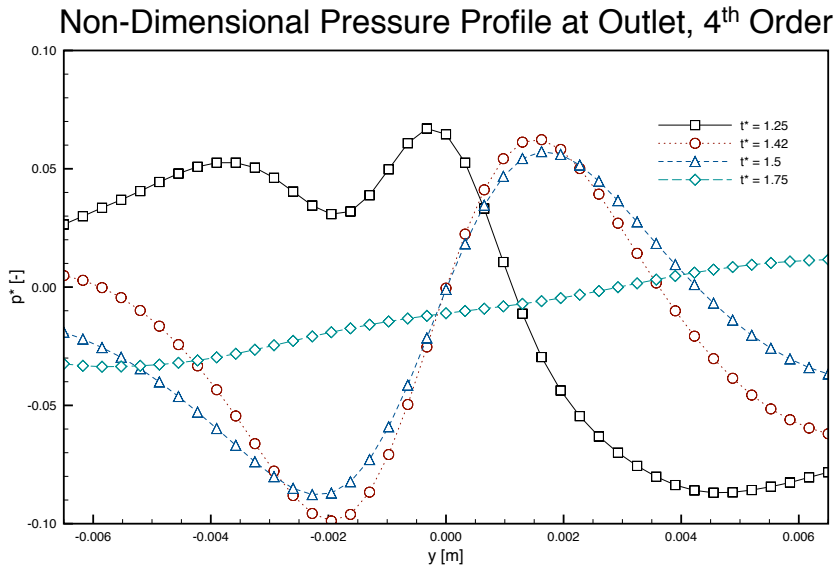


Figure 7.28: Non-Dimensional Pressure at Outlet: 4^{th}-Order Full NSCBC on the Structured Mesh.

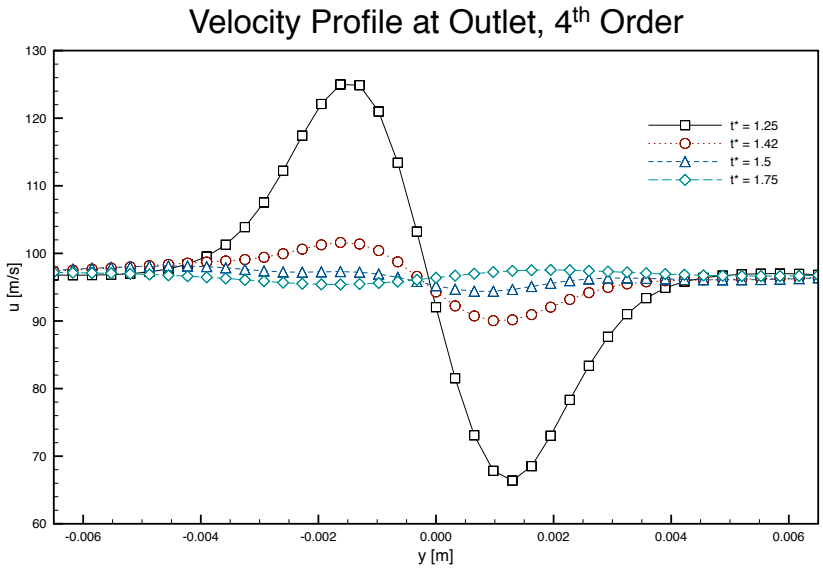
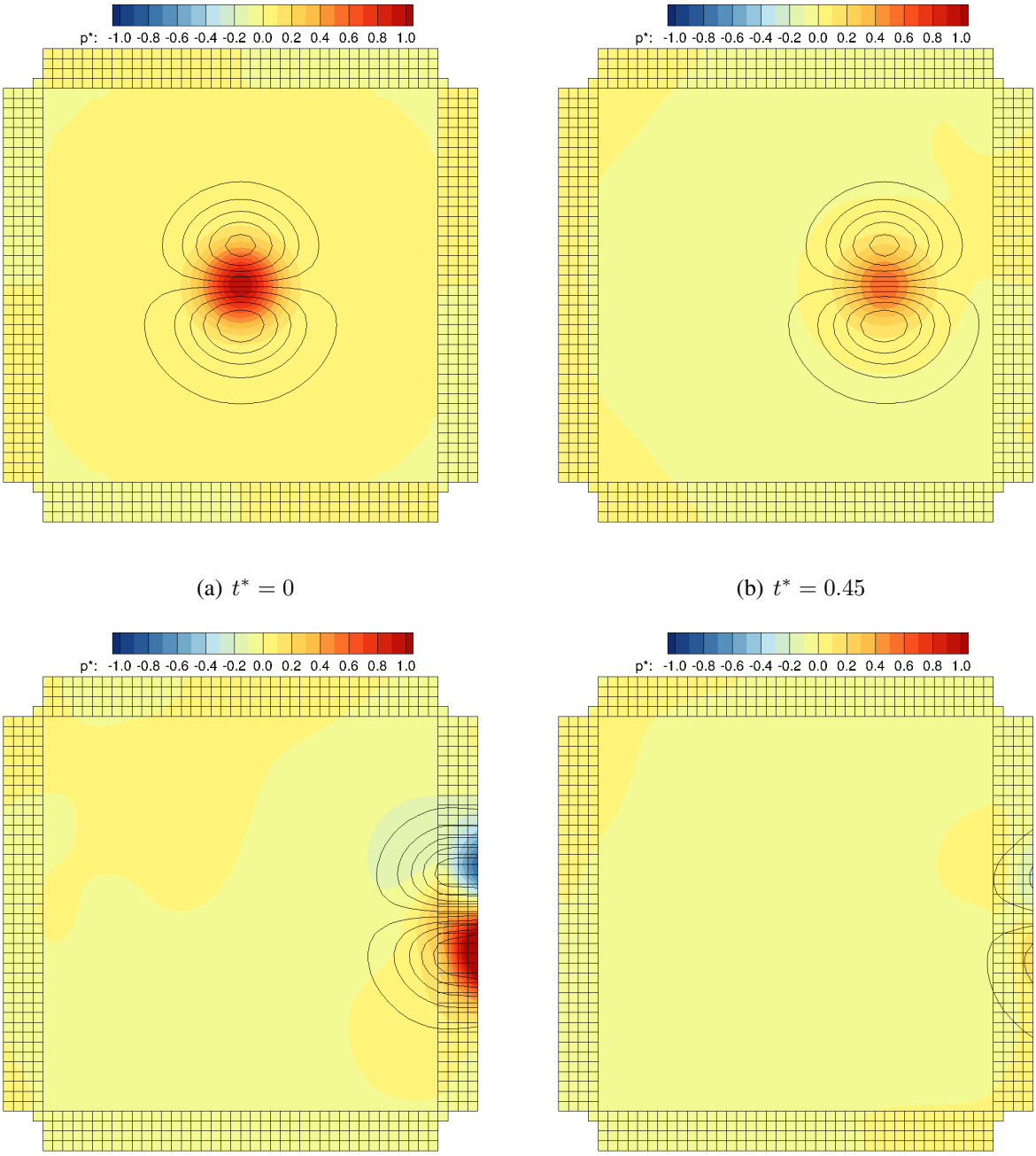


Figure 7.29: Non-Dimensional Velocity Profiles at Outlet: 4^{th}-Order Full NSCBC on the Structured Mesh.

Figures 7.28 and 7.29 show that the fourth-order results are similar to the second- and third-order results shown in Figures 7.13 and 7.14 and Figures 7.22 and 7.23. In addition to observing all of the same trends as the third-order profiles shown in Figure 7.22, the magnitude of the peak non-dimensional pressure is only about $p^* = 0.1$ at the outlet. Furthermore, as with all the previous cases, the outlet does not appear to send any large pressure waves upstream. The results verify that the boundary conditions are not pushing excessive energy upstream. Additionally, the boundary conditions are effective at maintaining the velocity profiles of the vortex as shown in Figure 7.29. Figure 7.30 shows the fourth-order unstructured mesh results.



(a) $t^* = 0$

(b) $t^* = 0.45$

(c) $t^* = 1.12$

(d) $t^* = 1.43$

Figure 7.30: Non-Dimensional Pressure Contours and Velocity Iso-Contours of Vortex Exiting Domain: 4^{th}-Order Full NSCBC on the Unstructured Mesh.

These results are again similar to the structured results as shown in Figure 7.27, where as men-

tioned earlier, the results only differ due to the added dissipation produced from the unstructured mesh. Figures 7.31 and 7.32 show the pressure and velocity profiles for the unstructured mesh.

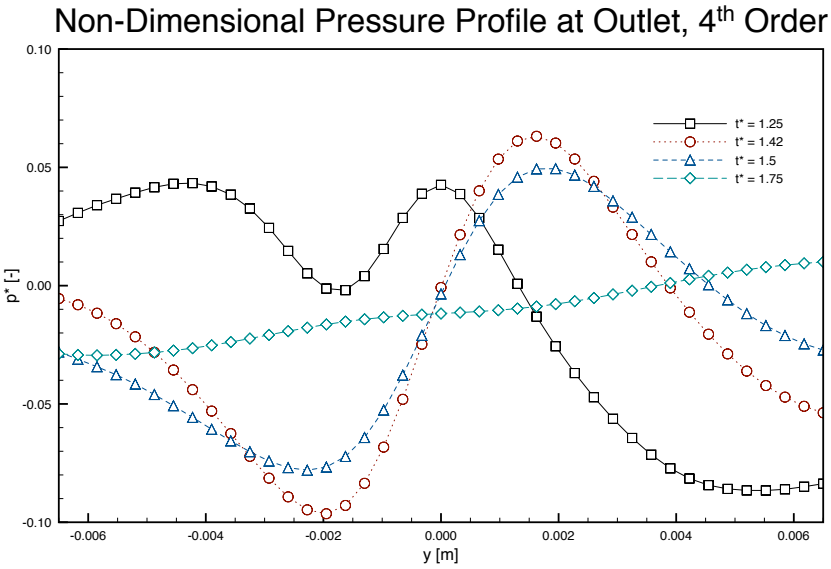


Figure 7.31: Non-Dimensional Pressure at Outlet: 4^{th}-Order Full NSCBC on the Unstructured Mesh.

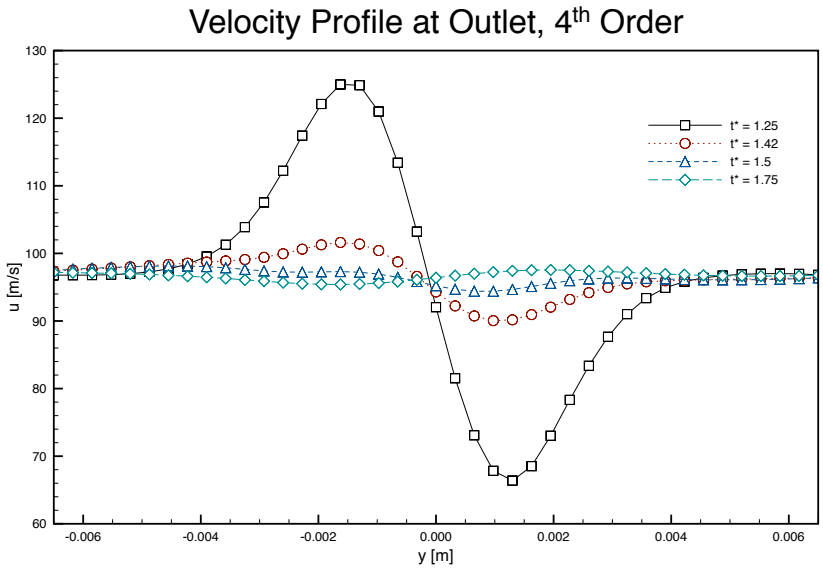


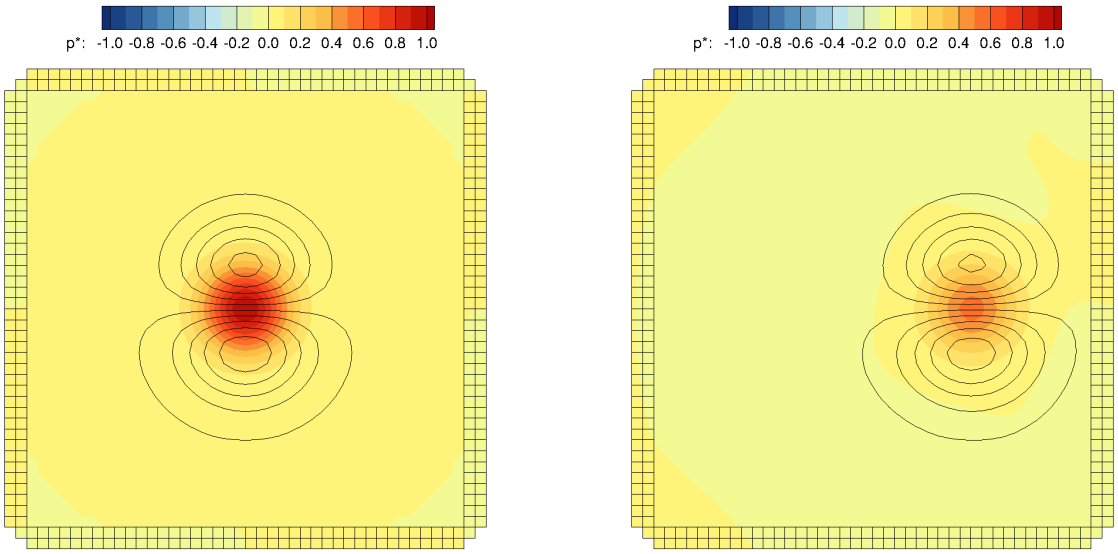
Figure 7.32: Non-Dimensional Velocity Profiles at Outlet: 4^{th}-Order Full NSCBC on the Unstructured Mesh.

The unstructured profiles in Figures 7.31 and 7.32 have the same trends as those seen on the structured mesh results shown in Figures 7.28 and 7.29. Again, the results on the boundaries only differ because of the internal unstructured mesh, which adds additional dissipation to the simulation decreasing the strength of the vortex that actually reaches the boundary as compared to the structured mesh. If there was any significant difference between the two results, it would then be due to the internal scheme. As demonstrated by this section, the NSCBC produce low reflections and allow for efficient information transfer out of the domain for any order, which is necessary for accurate simulations.

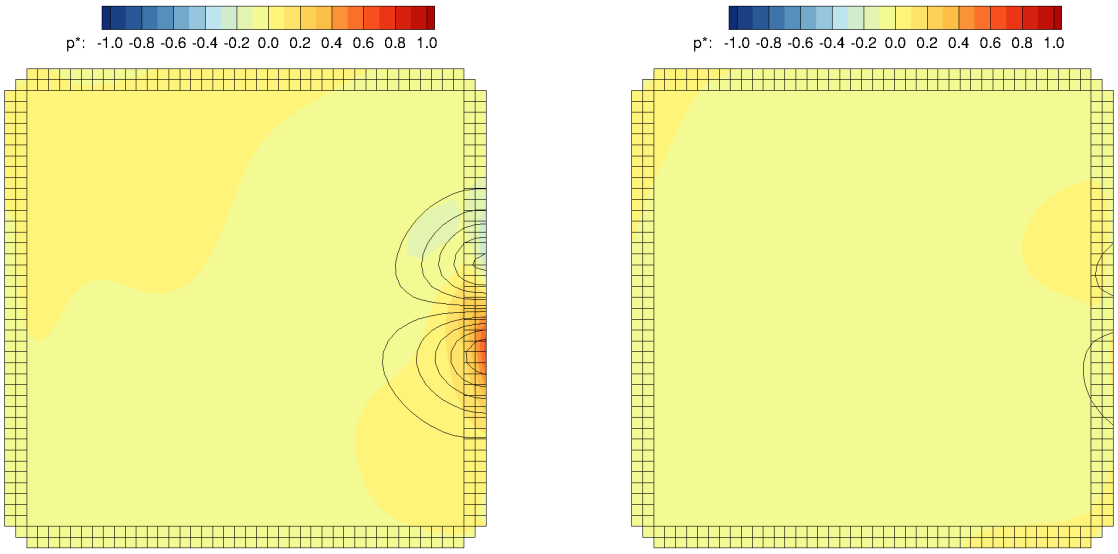
7.2.2.2 Fully-Diffuse Derivatives with Transverse Terms

These cases testing the NSCBC include the transverse terms in the NSCBC, but use the fully-diffuse MLS derivatives described in Section 5.1.5.3. Second- through fourth-order results are presented. Results are shown on the structured and unstructured meshes, but as discussed in the previous section, the results should be similar since the boundaries are effectively the same for each mesh.

7.2.2.2.1 Second-Order Figure 7.33 shows the second-order structured mesh results.



(a) $t^* = 0$ (b) $t^* = 0.45$



(c) $t^* = 1.12$ (d) $t^* = 1.43$

Figure 7.33: Non-Dimensional Pressure Contours and Velocity Iso-Contours of Vortex Exiting Domain: 2^{nd}-Order Full NSCBC on the Structured Mesh with Diffuse Derivatives.

The results presented in Figure 7.33 are very similar to those shown in Figure 7.12 for the full derivatives. The pressure and velocity profiles are shown in Figure 7.34 and 7.35.

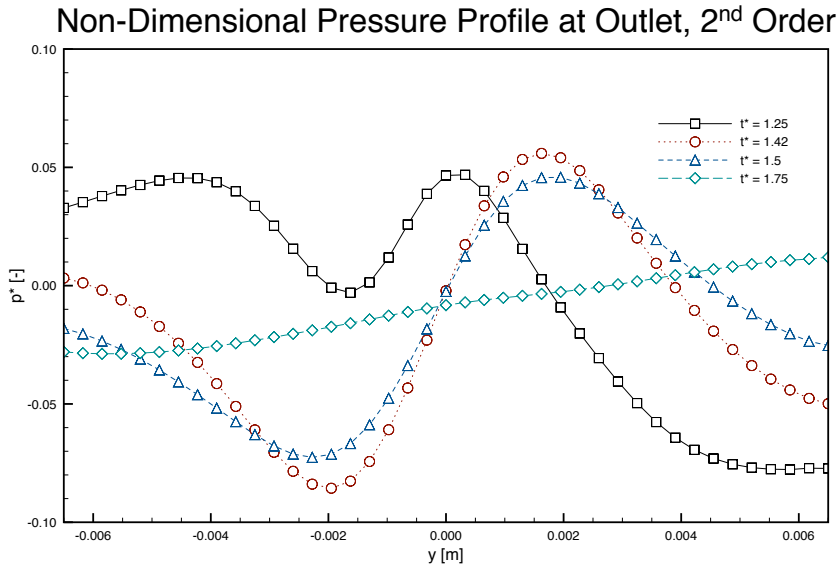


Figure 7.34: Non-Dimensional Pressure at Outlet: 2nd-Order Full NSCBC on the Structured Mesh with Diffuse Derivatives.

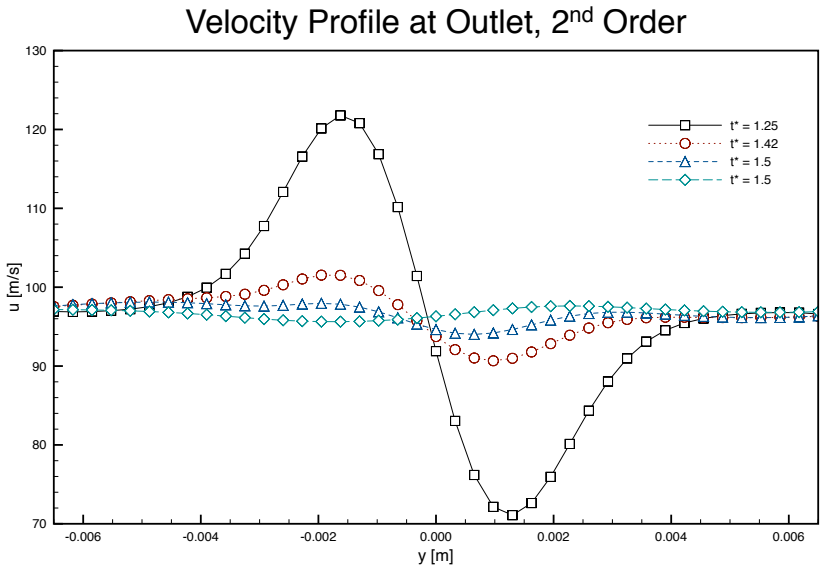
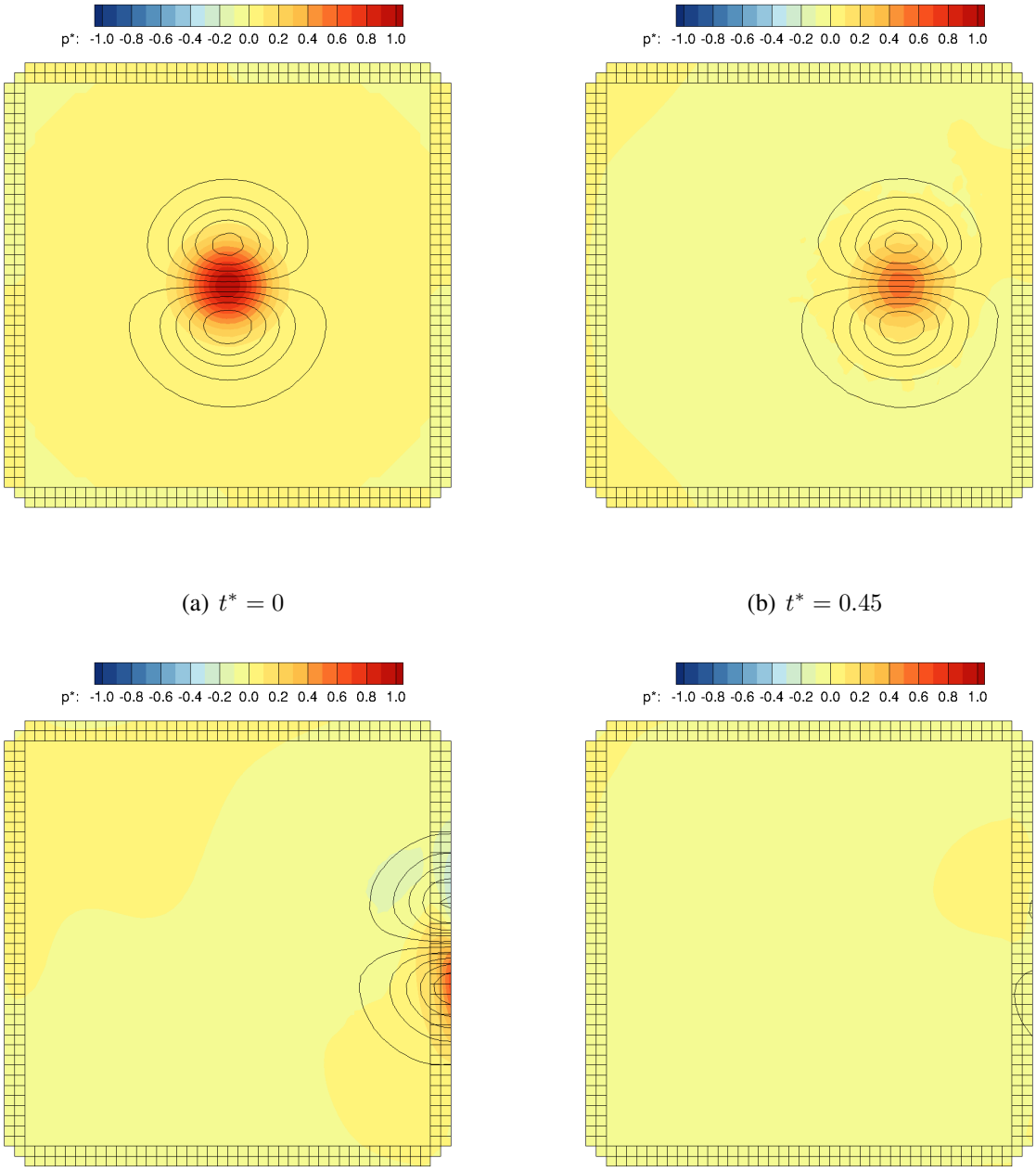


Figure 7.35: Non-Dimensional Velocity Profiles at Outlet: 2nd-Order Full NSCBC on the Structured Mesh with Diffuse Derivatives.

The profiles shown in Figures 7.34 and 7.35 are qualitatively similar to the results using the full derivatives, as shown in Figures 7.13 and 7.14. This shows that the use of diffuse derivatives does not affect the flow field, as expected based on the results in Chapter 6. Figure 7.15 shows the second-order unstructured mesh results.



(a) $t^* = 0$

(b) $t^* = 0.45$

(c) $t^* = 1.12$

(d) $t^* = 1.43$

Figure 7.36: Non-Dimensional Pressure Contours and Velocity Iso-Contours of Vortex Exiting Domain: 2^{nd}-Order Full NSCBC on the Unstructured Mesh with Diffuse Derivatives.

As with the structured meshes, the unstructured mesh results shown in Figure 7.36 are similar

to those for the full derivatives shown in Figure 7.15. The pressure and velocity profiles are shown in Figure 7.37 and 7.38.

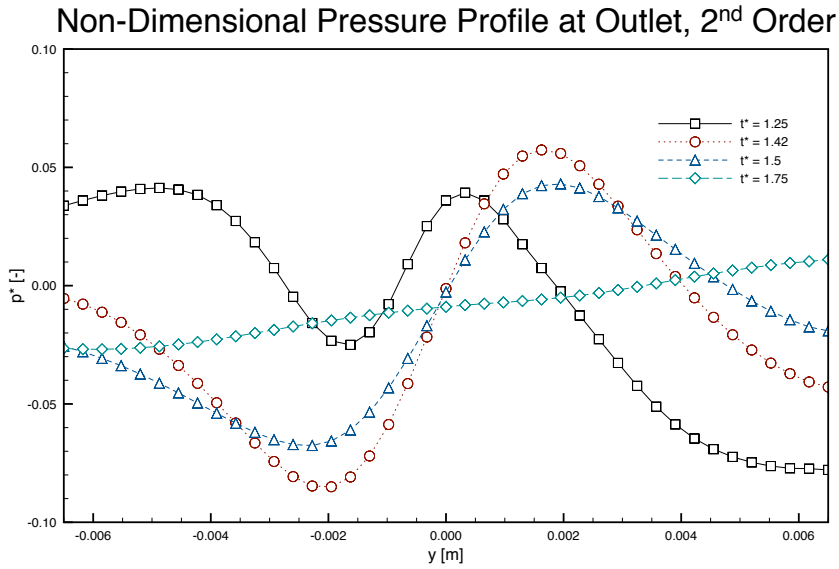


Figure 7.37: Non-Dimensional Pressure at Outlet: 2^{nd}-Order Full NSCBC on the Unstructured Mesh with Diffuse Derivatives.

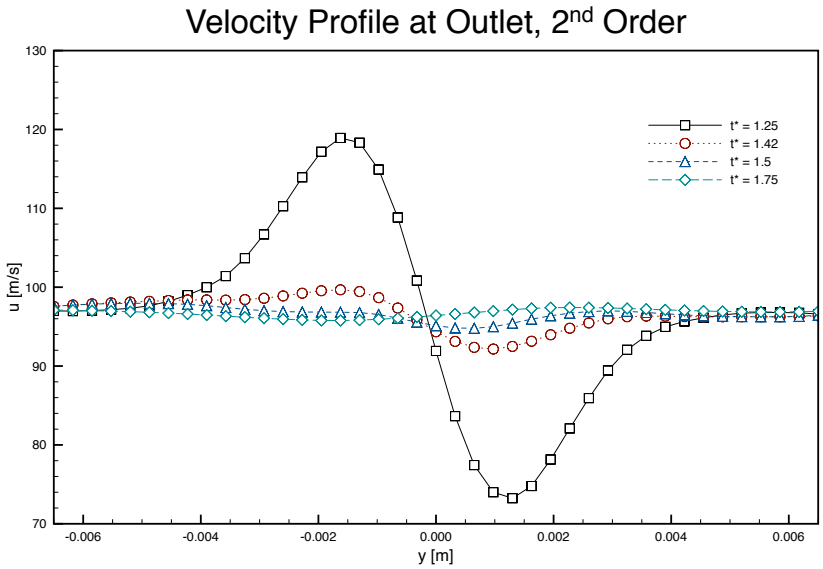
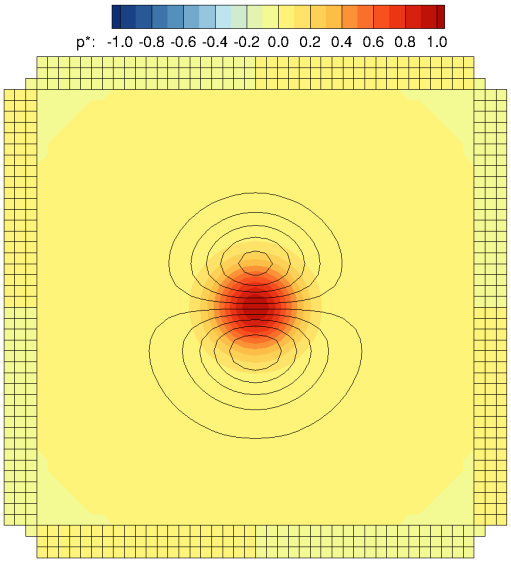


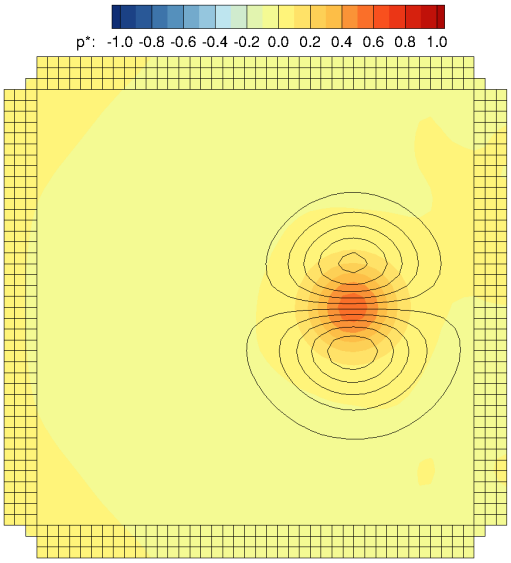
Figure 7.38: Non-Dimensional Velocity Profiles at Outlet: 2^{nd}-Order Full NSCBC on the Unstructured Mesh with Diffuse Derivatives.

The profiles shown in Figures 7.37 and 7.38 are qualitatively similar to structured mesh results shown in Figures 7.34 and 7.35. As with the full-derivative results in the previous section, the differences between the structured and unstructured mesh are a result of the added dissipation from the unstructured mesh.

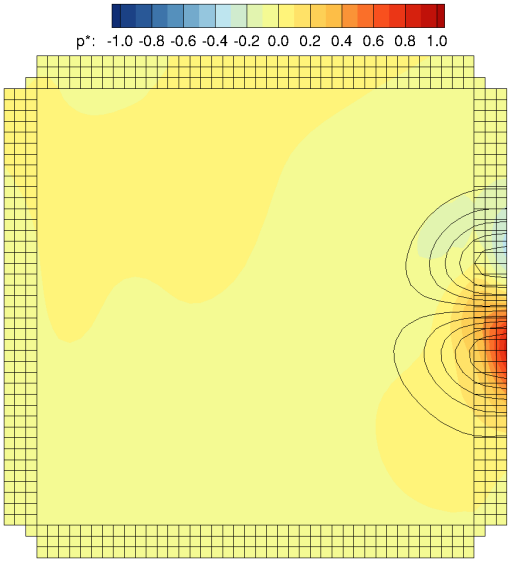
7.2.2.2.2 Third-Order Figure 7.39 shows the third-order structured mesh results.



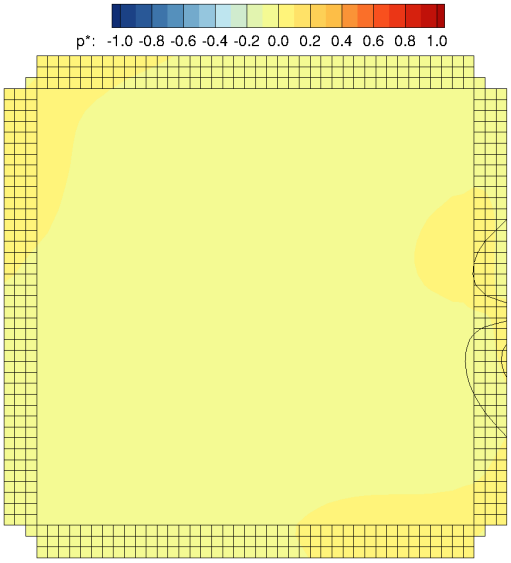
(a) $t^* = 0$



(b) $t^* = 0.45$



(c) $t^* = 1.15$



(d) $t^* = 1.43$

Figure 7.39: Non-Dimensional Pressure Contours and Velocity Iso-Contours of Vortex Exiting Domain: 3^{rd}-Order Full NSCBC on the Structured Mesh with Diffuse Derivatives.

The results presented in Figure 7.39 are qualitatively similar to those shown in Figure 7.21

for the full derivatives, as expected based on the results in Chapter 6. The pressure and velocity profiles are shown in Figures 7.40 and 7.41.

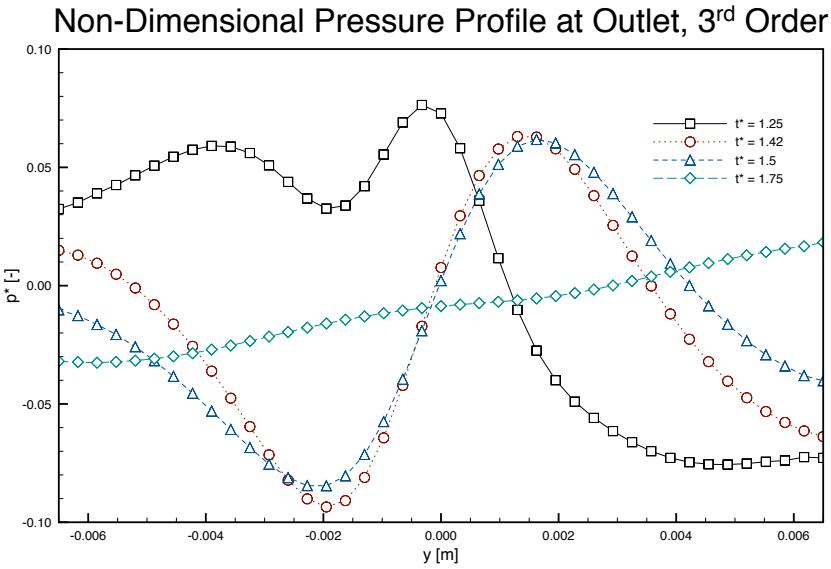


Figure 7.40: Non-Dimensional Pressure at Outlet: 3^{rd}-Order Full NSCBC on the Structured Mesh with Diffuse Derivatives.

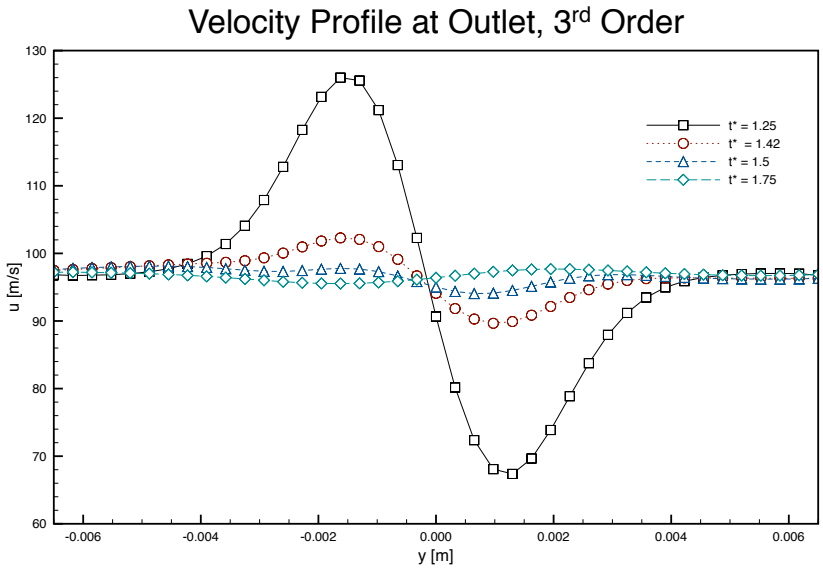
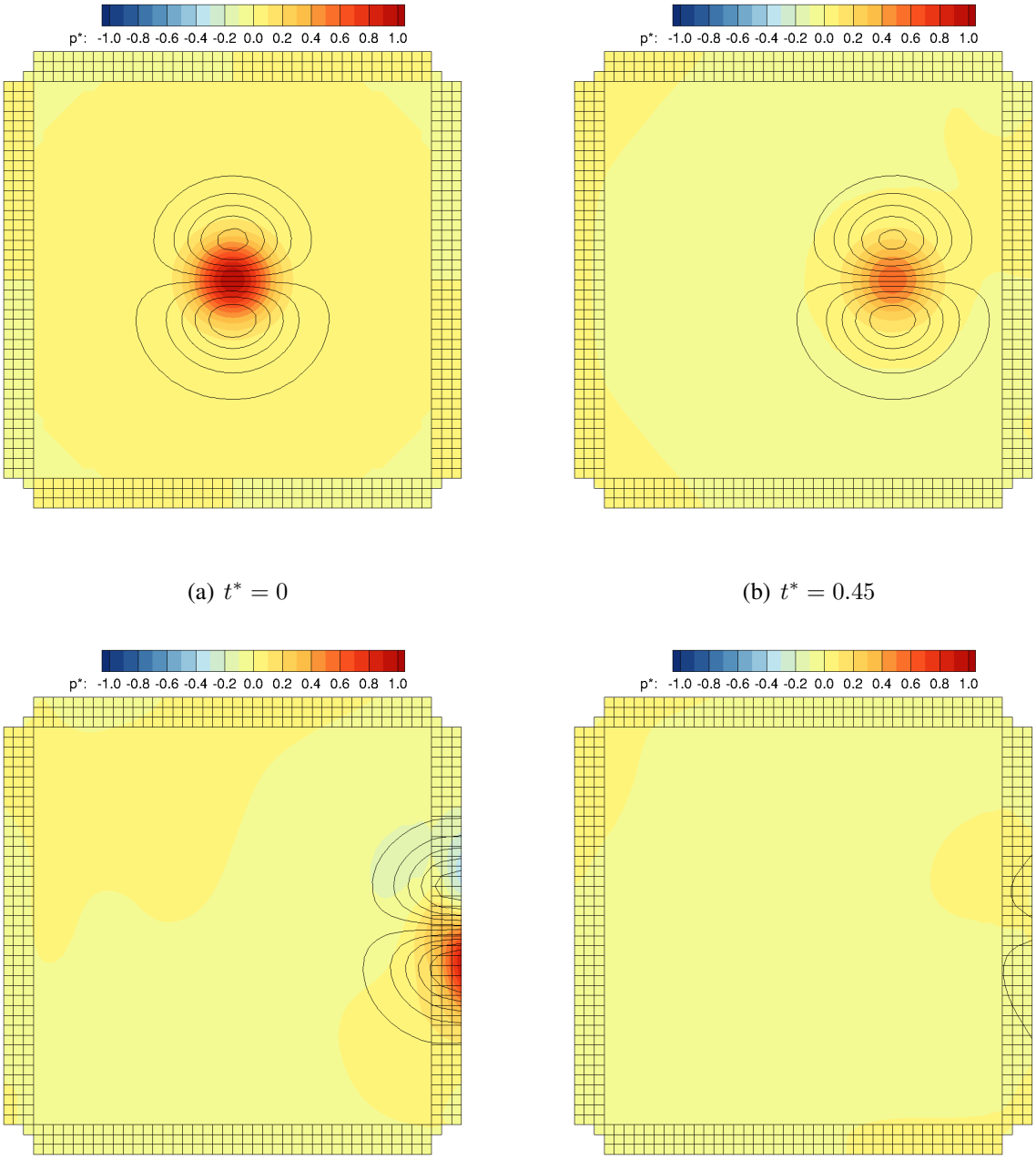


Figure 7.41: Non-Dimensional Velocity Profiles at Outlet: 3^{rd}-Order Full NSCBC on the Structured Mesh with Diffuse Derivatives.

The profiles shown in Figures 7.40 and 7.41 are qualitatively similar to the results using the full derivatives, as shown in Figures 7.22 and 7.23. This shows that the use of diffuse derivatives does not affect the flow field. Figure 7.42 shows the third-order unstructured mesh results.



(a) $t^* = 0$

(b) $t^* = 0.45$

(c) $t^* = 1.12$

(d) $t^* = 1.43$

Figure 7.42: Non-Dimensional Pressure Contours and Velocity Iso-Contours of Vortex Exiting Domain: 3^{rd}-Order Full NSCBC on the Unstructured Mesh with Diffuse Derivatives.

As with the structured meshes, the unstructured mesh results shown in Figure 7.42 are similar

to those for the full derivatives shown in Figure 7.24. The pressure and velocity profiles are shown in Figure 7.43 and 7.44.

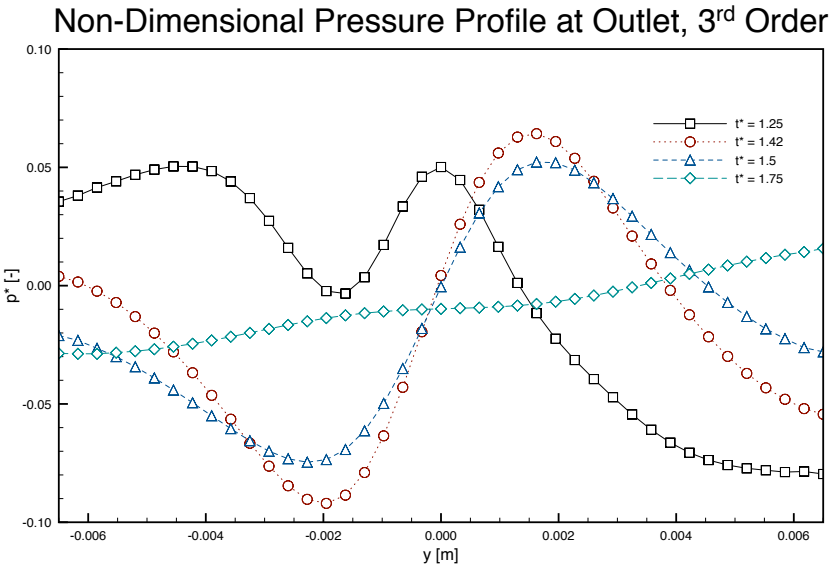


Figure 7.43: Non-Dimensional Pressure at Outlet: 3^{rd}-Order Full NSCBC on the Unstructured Mesh with Diffuse Derivatives.

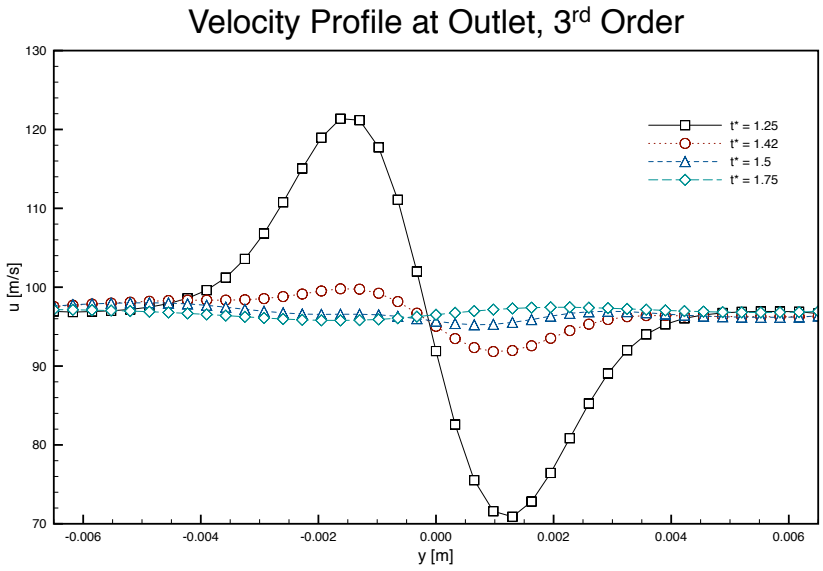
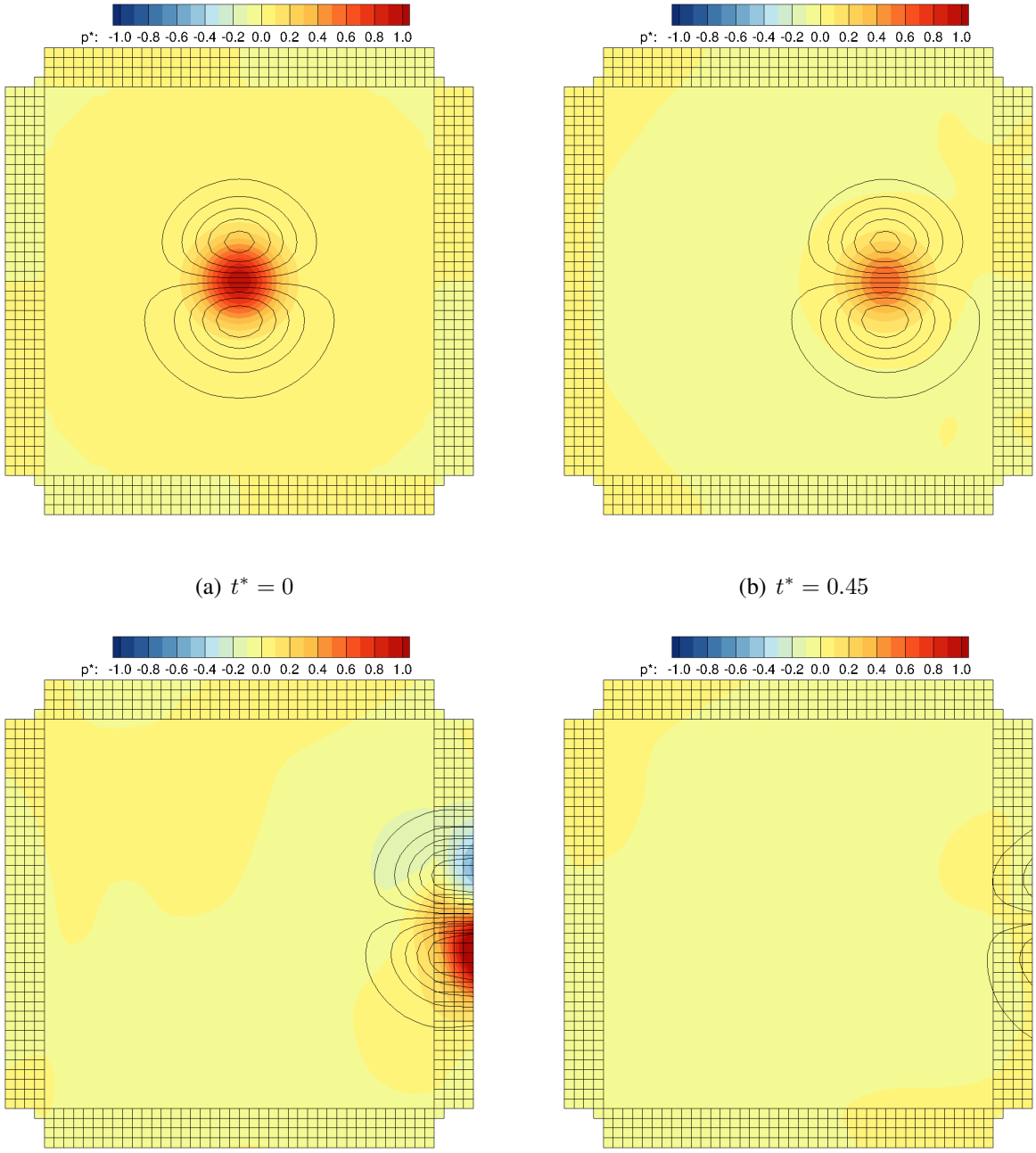


Figure 7.44: Non-Dimensional Velocity Profiles at Outlet: 3^{rd}-Order Full NSCBC on the Unstructured Mesh with Diffuse Derivatives.

The profiles shown in Figures 7.43 and 7.44 are qualitatively similar to structured mesh results shown in Figures 7.40 and 7.41. As with second-order results, any difference between the structured and unstructured mesh are a result of the added dissipation from the unstructured mesh.

7.2.2.2.3 Fourth-Order Figure 7.45 shows the fourth-order structured mesh results.



(a) $t^* = 0$

(b) $t^* = 0.45$

(c) $t^* = 1.12$

(d) $t^* = 1.43$

Figure 7.45: Non-Dimensional Pressure Contours and Velocity Iso-Contours of Vortex Exiting Domain: 4^{th}-Order Full NSCBC on the Structured Mesh with Diffuse Derivatives.

The results presented in Figure 7.45 are qualitatively similar to those shown in Figure 7.27

for the full derivatives, as expected based on the results in Chapter 6. The pressure and velocity profiles are shown in Figures 7.46 and 7.47.

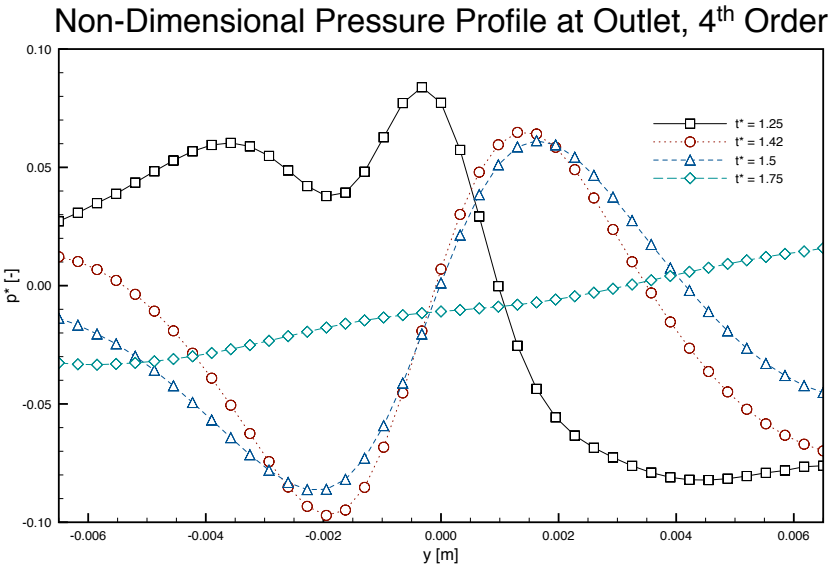


Figure 7.46: Non-Dimensional Pressure at Outlet: 4^{th}-Order Full NSCBC on the Structured Mesh with Diffuse Derivatives.

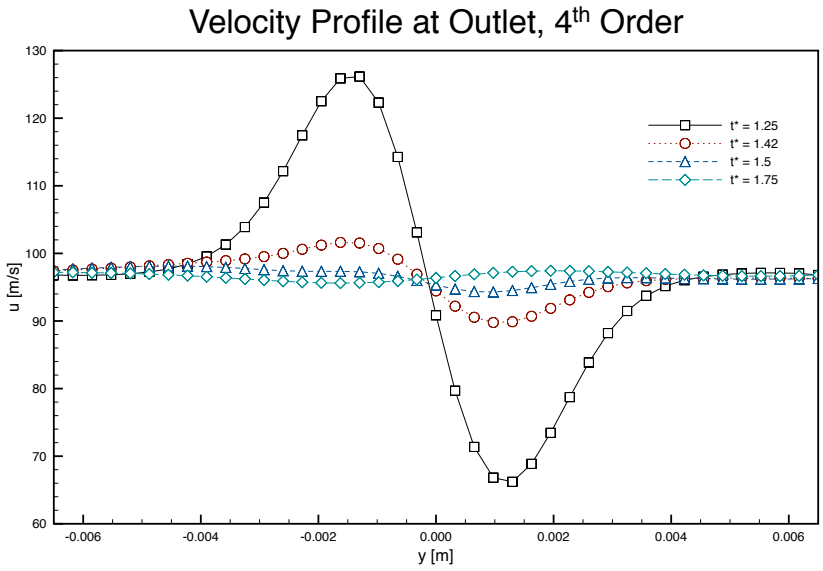
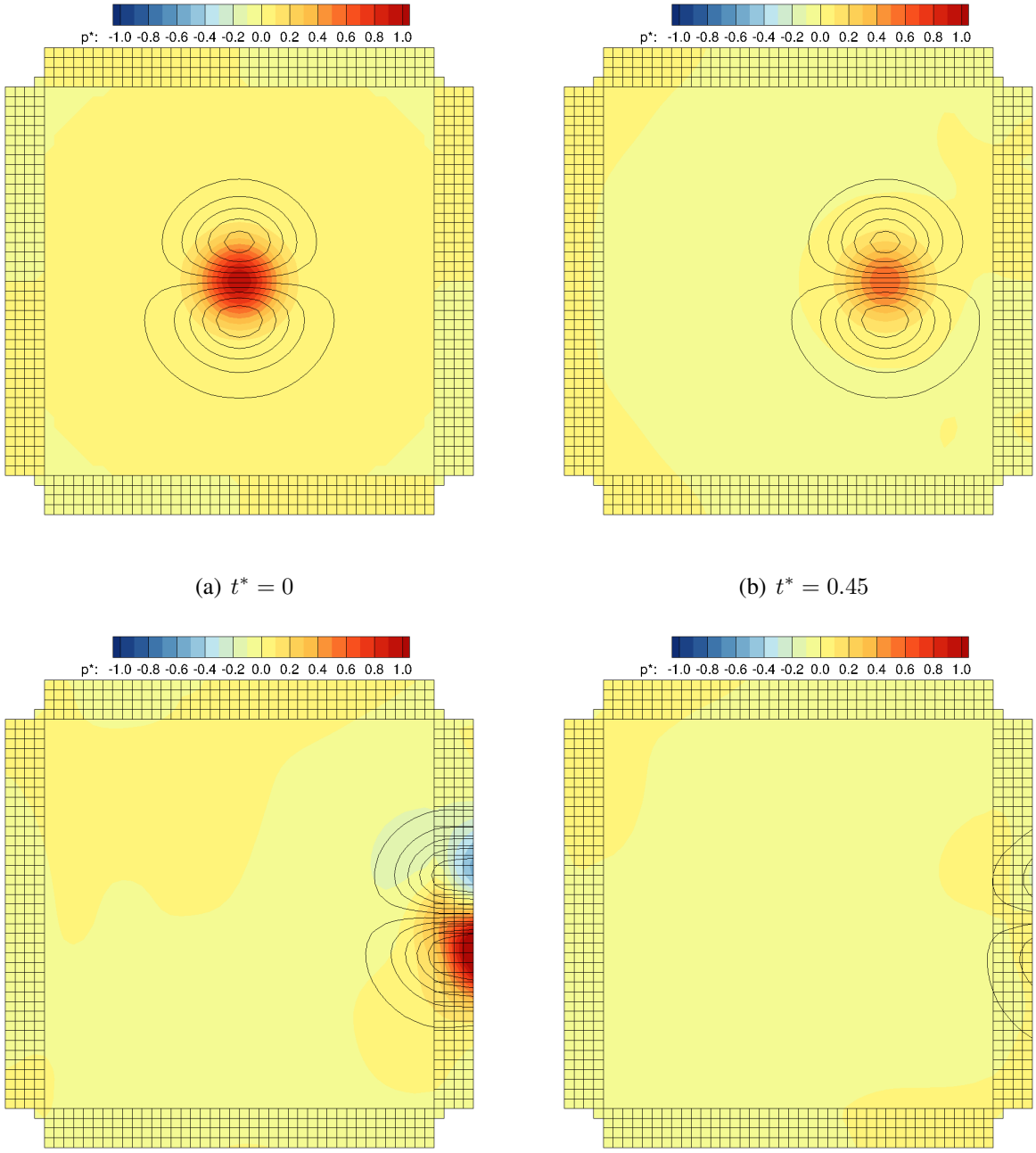


Figure 7.47: Non-Dimensional Velocity Profiles at Outlet: 4^{th}-Order Full NSCBC on the Structured Mesh with Diffuse Derivatives.

The profiles shown in Figures 7.46 and 7.47 are qualitatively similar to the results using the full derivatives, as shown in Figures 7.28 and 7.29. This shows that the use of diffuse derivatives does not affect the flow field. Figure 7.48 shows the fourth-order unstructured mesh results.



(a) $t^* = 0$

(b) $t^* = 0.45$

(c) $t^* = 1.12$

(d) $t^* = 1.43$

Figure 7.48: Non-Dimensional Pressure Contours and Velocity Iso-Contours of Vortex Exiting Domain: 4^{th}-Order Full NSCBC on the Unstructured Mesh with Diffuse Derivatives.

As with the structured meshes, the unstructured mesh results shown in Figure 7.48 are similar

to those for the full derivatives shown in Figure 7.30. The pressure and velocity profiles are shown in Figure 7.49 and 7.50.

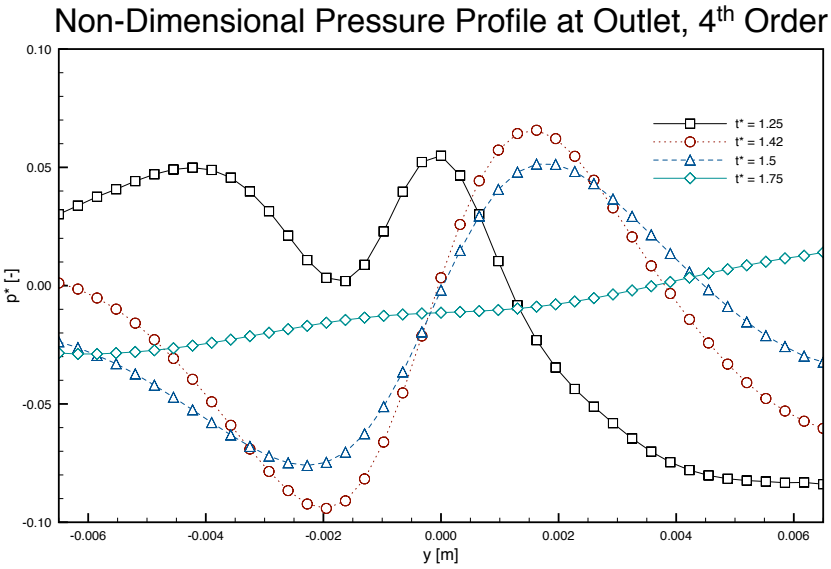


Figure 7.49: Non-Dimensional Pressure at Outlet: 4^{th}-Order Full NSCBC on the Unstructured Mesh with Diffuse Derivatives.

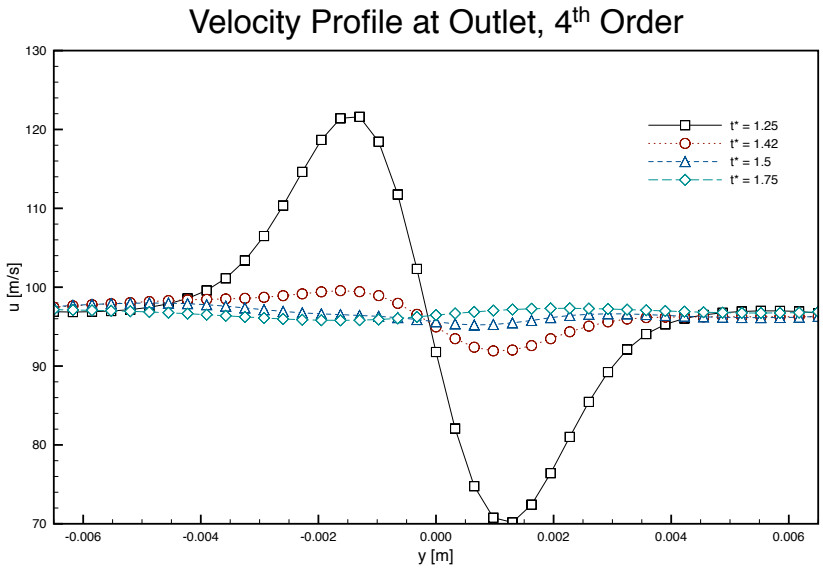


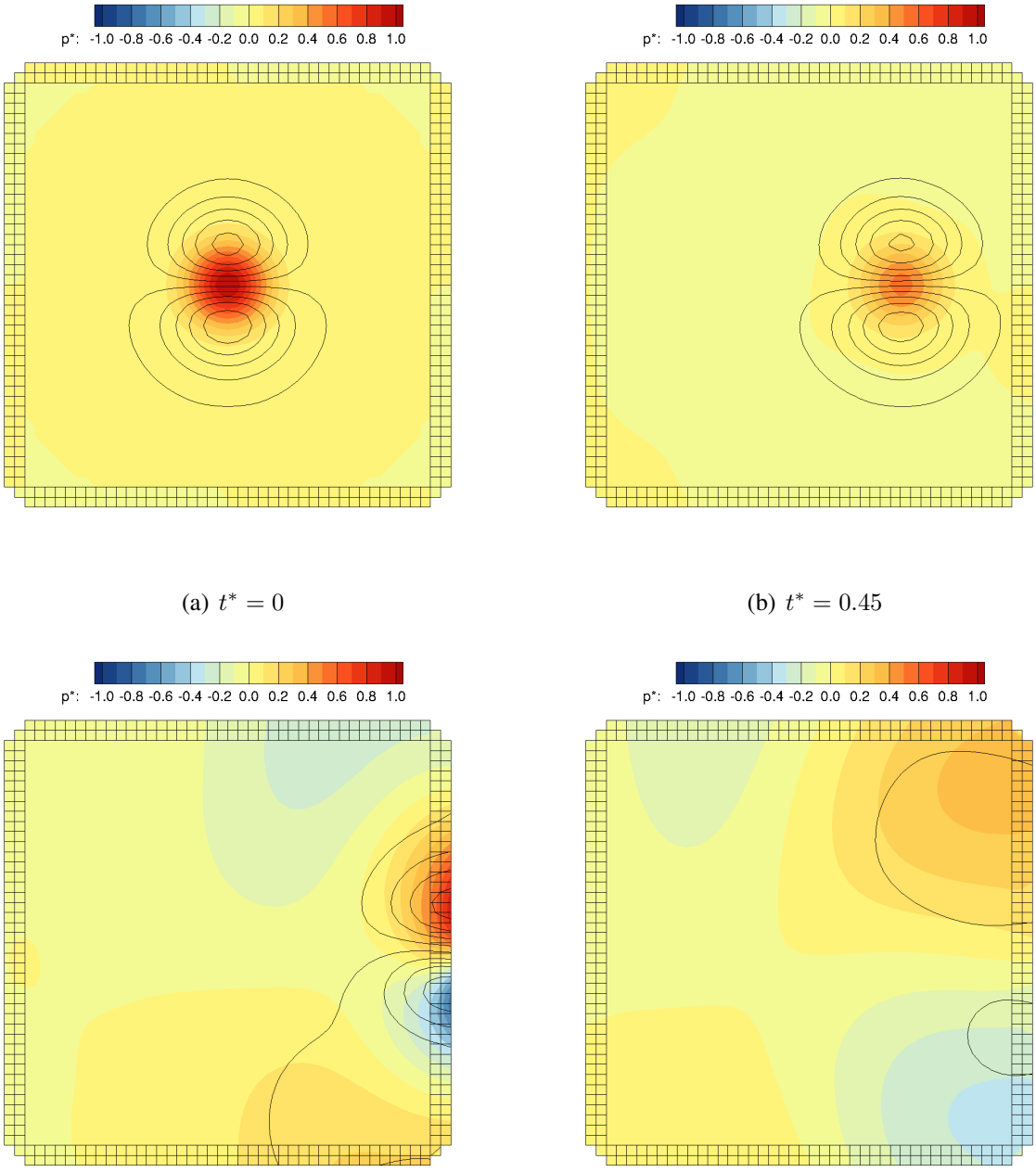
Figure 7.50: Non-Dimensional Velocity Profiles at Outlet: 4^{th}-Order Full NSCBC on the Unstructured Mesh with Diffuse Derivatives.

The profiles shown in Figures 7.49 and 7.50 are qualitatively similar to structured mesh results shown in Figures 7.46 and 7.47. As with second- and third-order results, any difference between the structured and unstructured mesh are a result of the added dissipation from the unstructured mesh.

7.2.2.3 Full Derivatives without Transverse Terms

This section presents the results of the vortex exiting the domain using the NSCBC conditions excluding the transverse terms. Second- through fourth-order results are presented, but unlike the previous two sections, only the structured mesh results are presented. Unstructured mesh results are omitted since the boundary conditions are the same for each, and therefore the results would be redundant.

7.2.2.3.1 Second-Order Figure 7.51 shows the second-order structured mesh results.



(a) $t^* = 0$

(b) $t^* = 0.45$

(c) $t^* = 1.12$

(d) $t^* = 1.43$

Figure 7.51: Non-Dimensional Pressure Contours and Velocity Iso-Contours of Vortex Exiting Domain: 2^{nd}-Order NSCBC without Transverse Terms on the Structured Mesh.

The second-order scheme has a more coherent vortex, which means it has more energy to send

upstream into the domain when not using the transverse terms, as seen in Figure 7.51. Additionally, at $t^* = 0.45$ the outlet boundary 'reaches out' to the vortex and drags in toward the outlet. The velocity iso-contours are also distorted at the outlet as the vortex passes through the domain. This can be clearly seen in the velocity and pressure profiles shown in Figures 7.52 and 7.53.

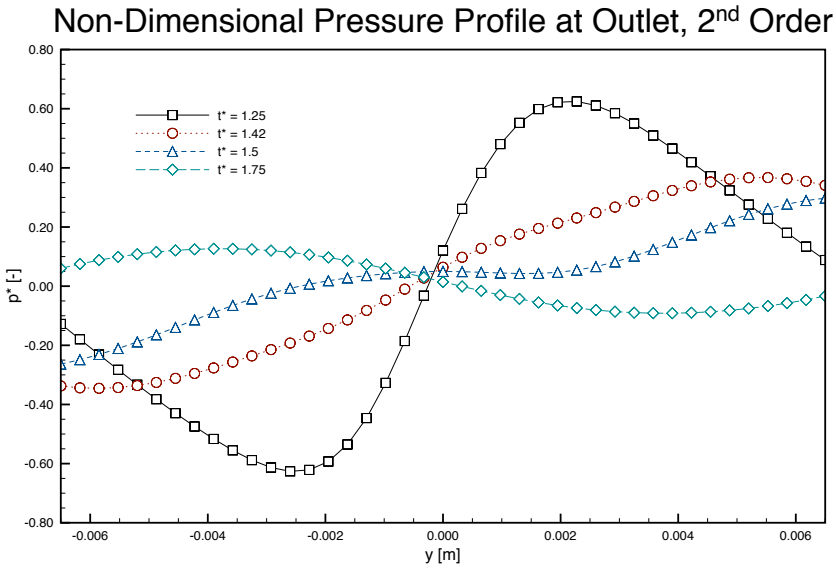


Figure 7.52: Non-Dimensional Pressure at Outlet: 2^{nd}-Order NSCBC without Transverse Terms on the Structured Mesh.

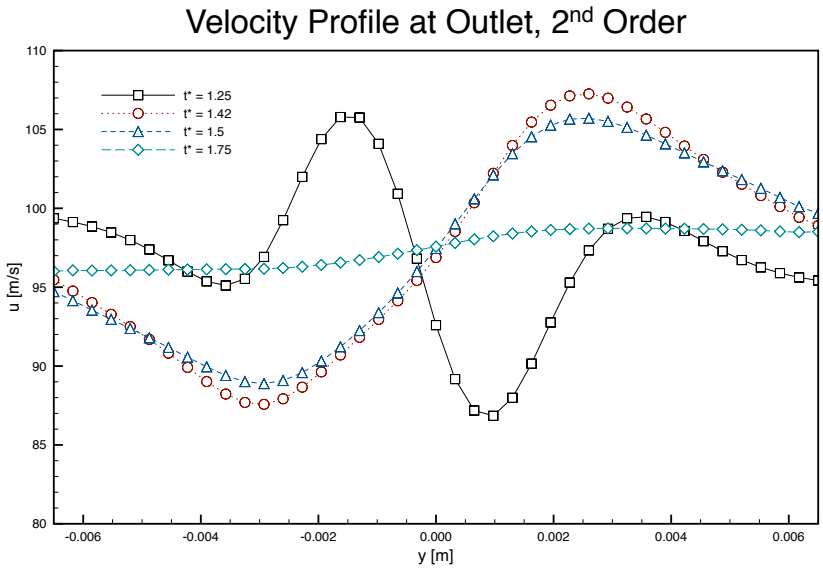
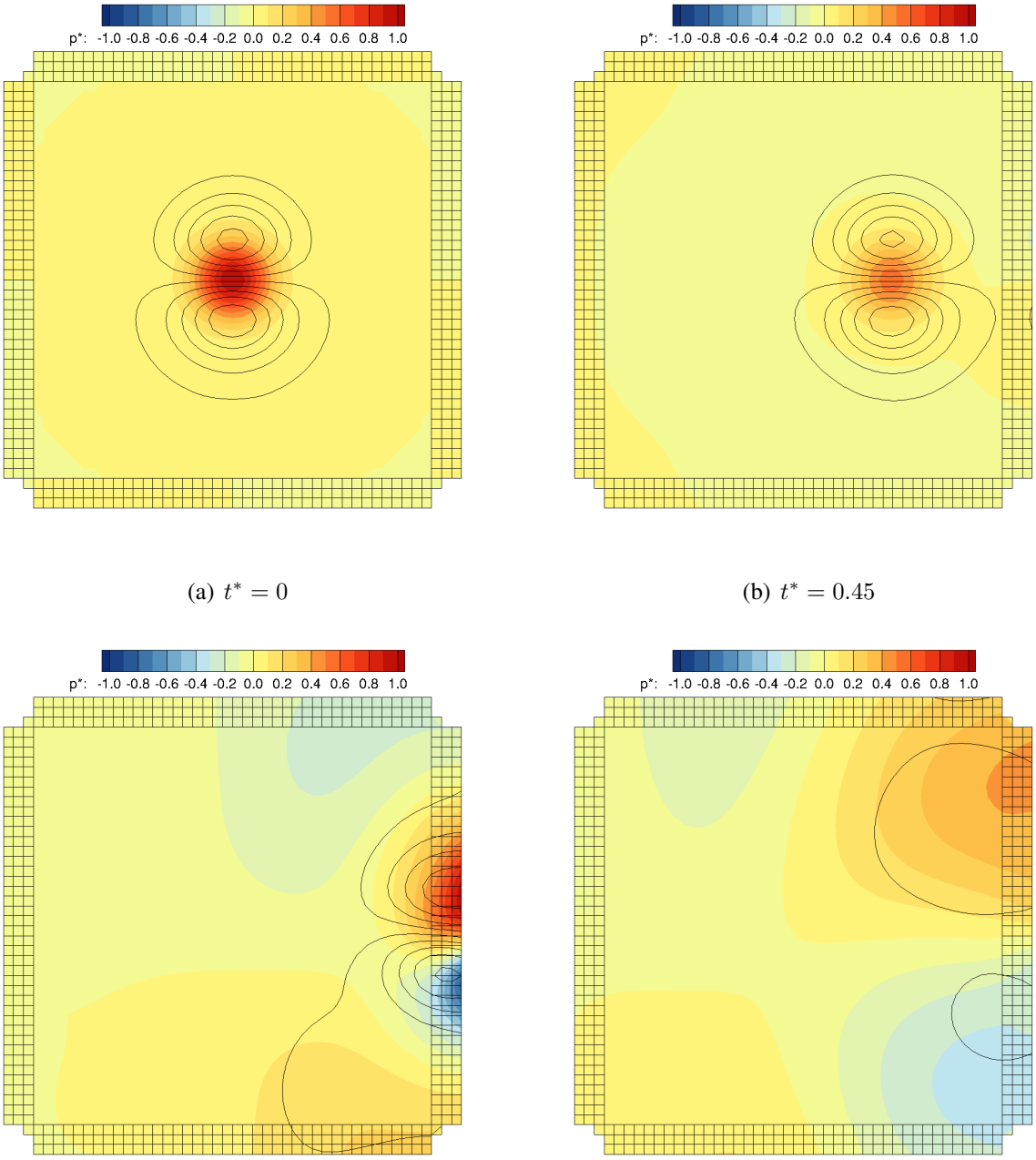


Figure 7.53: Non-Dimensional Velocity Profiles at Outlet: 2^{nd}-Order NSCBC without Transverse Terms on the Structured Mesh.

The maximum peak non-dimensional pressure is approximately $p^* = 0.6$ at $t^* = 1.25$, seen in Figure 7.52. Additionally, the pressure in the later time periods is largest near the edges of the domain. The velocity profile at $t^* = 1.25$ seen in Figure 7.53 also shows a double peak velocity which is unphysical. Neglecting the transverse terms is extremely detrimental to the flow, and the transverse terms must be included for vortex dominated flows.

7.2.2.3.2 Third-Order Figure 7.54 shows the third-order structured mesh results.



(a) $t^* = 0$

(b) $t^* = 0.45$

(c) $t^* = 1.15$

(d) $t^* = 1.43$

Figure 7.54: Non-Dimensional Pressure Contours and Velocity Iso-Contours of Vortex Exiting Domain: 3^{rd}-Order NSCBC without Transverse Terms on the Structured Mesh.

The third-order results without the transverse terms appears to generate a lobe of upstream

traveling vorticity. The upward traveling lobe also appears to have a larger peak non-dimensional pressure than the second-order results shown in Figure 7.51. The velocity and pressure profiles shown in Figures 7.55 and 7.56 collaborate this observation.

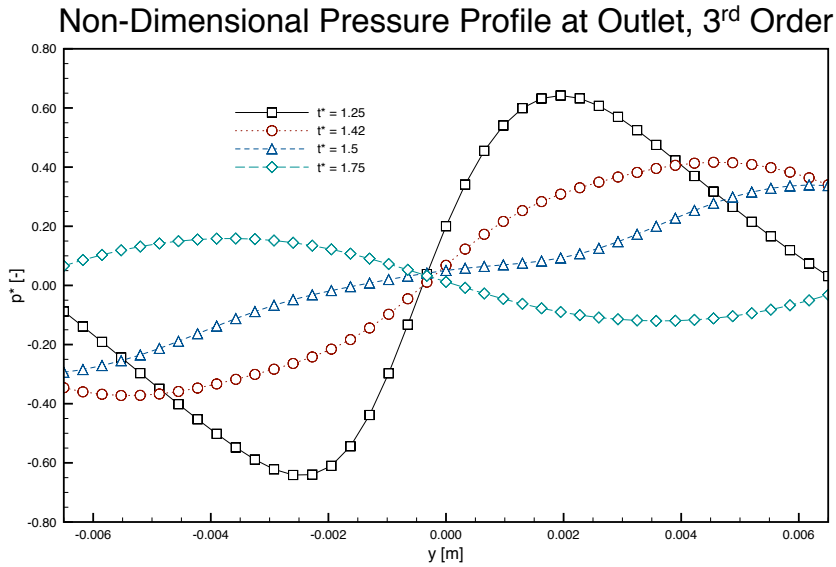


Figure 7.55: Non-Dimensional Pressure at Outlet: 3^{rd}-Order NSCBC without Transverse Terms on the Structured Mesh.

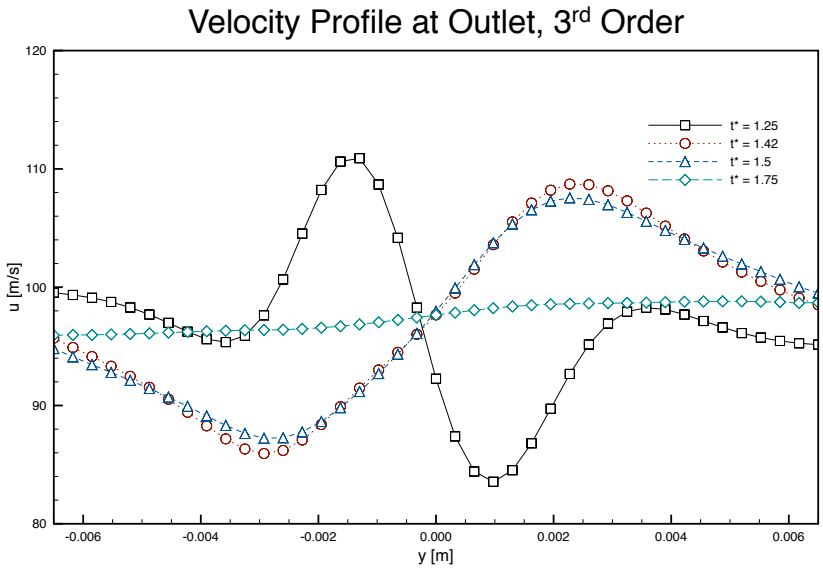
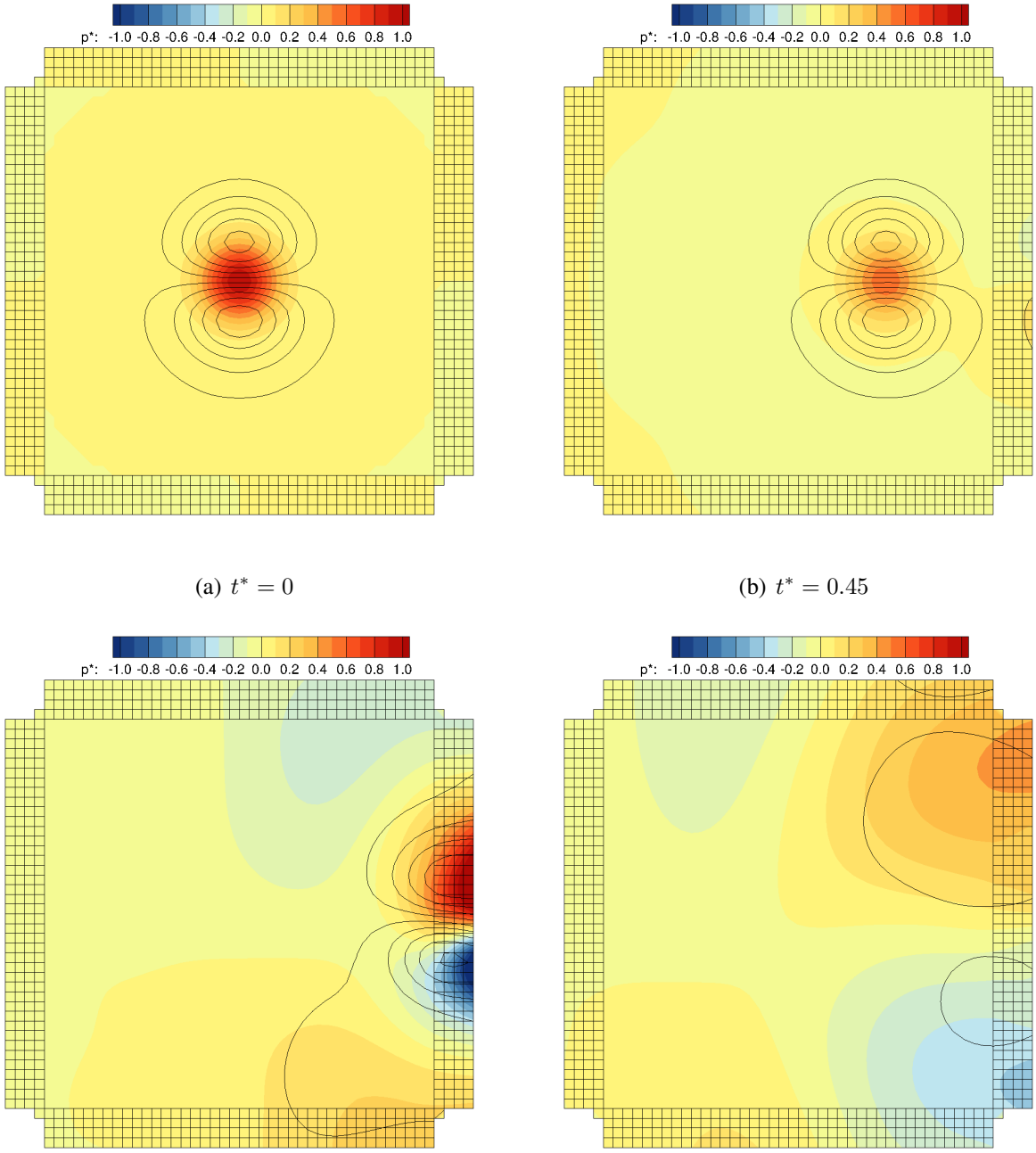


Figure 7.56: Non-Dimensional Velocity Profiles at Outlet: 3^{rd}-Order NSCBC without Transverse Terms on the Structured Mesh.

As with the second-order results of Figure 7.52, the maximum peak non-dimensional pressure is approximately $p^* = 0.6$ at $t^* = 1.25$ for the third-order results of Figure 7.55. Additionally, the pressure in the later time periods is largest near the edges of the domain, where the vorticity lobes have been forced toward the edges of the domain. The velocity profile at $t^* = 1.25$ seen in Figure 7.56 also shows a double peak velocity which is unphysical. The profiles are larger for the third-order case, so by neglecting the transverse terms the detrimental effect is worse with higher-order simulations.

7.2.2.3.3 Fourth-Order Figure 7.57 shows the fourth-order structured mesh results.



(a) $t^* = 0$

(b) $t^* = 0.45$

(c) $t^* = 1.12$

(d) $t^* = 1.43$

Figure 7.57: Non-Dimensional Pressure Contours and Velocity Iso-Contours of Vortex Exiting Domain: 4^{th}-Order NSCBC without Transverse Terms on the Structured Mesh.

As with the fourth-order NSCBC including the transverse terms results shown in Figure 7.27,

the non-dimensional pressure is largest for the fourth-order results. However, the highest-peak pressure for these results is much larger than the peak pressure for the fourth-order results which include transverse terms. A sizable upstream traveling lobe ($p^* \in [0.2 - 0.3]$) is generated by the outlet, and the upward traveling lobe is rather large. The velocity and pressure profiles shown in Figures 7.58 and 7.59 collaborate this observation.

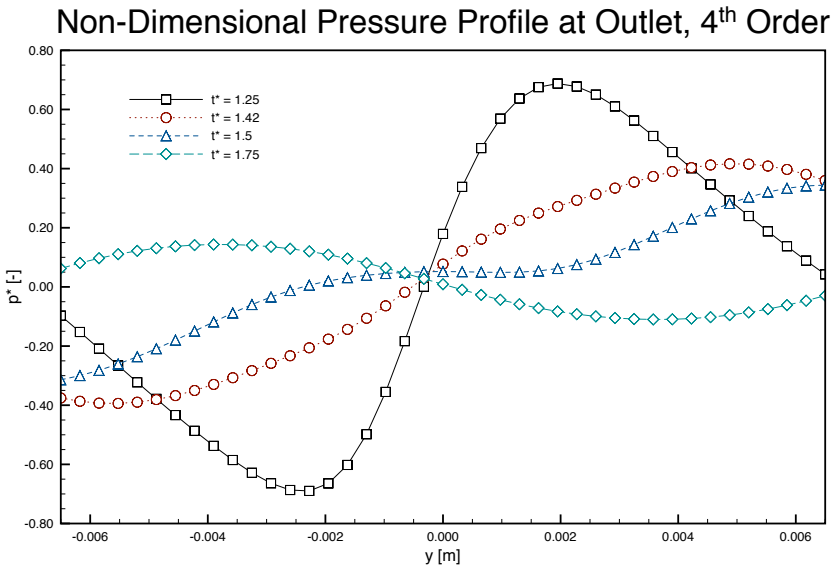


Figure 7.58: Non-Dimensional Pressure at Outlet: 4^{th}-Order NSCBC without Transverse Terms on the Structured Mesh.

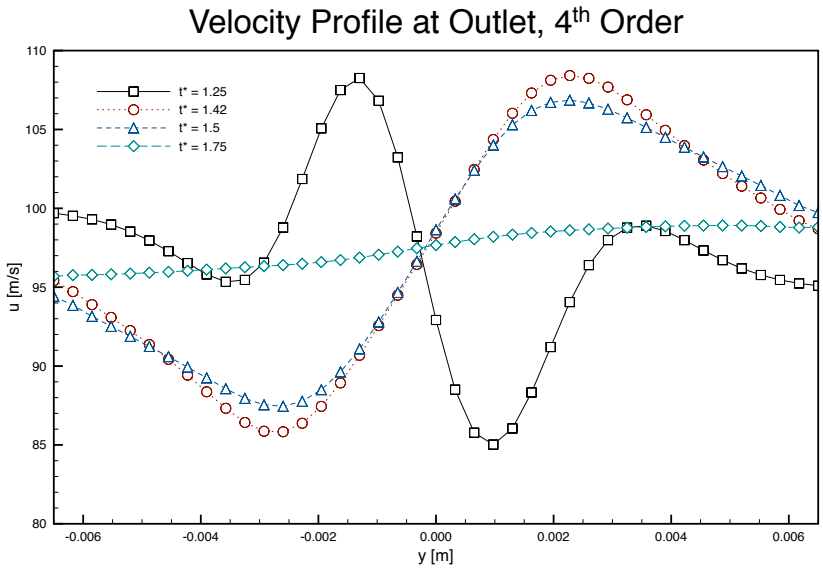


Figure 7.59: Non-Dimensional Velocity Profiles at Outlet: 4^{th}-Order NSCBC without Transverse Terms on the Structured Mesh.

Examining the pressure profiles in Figure 7.58, the peak pressure is $p^* \approx 0.7$. Additionally, the pressure in the later time periods is again largest near the edges of the domain, where the vorticity lobes have been forced toward the edges of the domain. The velocity profile at $t^* = 1.25$ seen in Figure 7.59 also shows an unphysical double peak velocity. The profiles are similar to the third-order case shown in Figure 7.56. Neglecting the transverse terms is again shown to be extremely detrimental to the flow for higher-order simulations.

7.2.2.4 Section Review

This section reviews the results of the NSCBC boundary condition tests using the vortex described by (7.8). Before diving into the results containing the transverse terms, the results presented in Section 7.2.2.3 for the NSCBC excluding the transverse terms should be mentioned. For vortex dominated flows, the transverse terms should always be included. As highlighted here, failing to include the transverse terms will produce large upstream traveling lobes of vorticity that will be detrimental to overall results. It should be mentioned that sometimes not including the transverse terms may be preferred, but this is generally on a case-by-case basis. The NSCBC without

transverse term results compare similarly to the results presented by Granet et al. [70] as BC1.

Comparing the results using the transverse terms, the full derivative MLS results presented in Section 7.2.2.1 are not different than those obtained for the fully-diffuse MLS derivatives in Section 7.2.2.2. This basically follows from the 2D MLS studies presented in Chapter 6. This proves that if situation arises, such as a moving-mesh situation, the fully-diffuse MLS derivatives will produce the same results as those obtained using the more expensive full MLS derivatives.

As far as the effectiveness of the NSCBC boundaries, the NSCBC presented here compare favorably with those discussed by Granet et al. [70]. If comparing the results of Granet et al. [70], which are third-order, to the third-order results herein, the peak pressures at the outlet appears to be similar to Granet et al. [70] at $t^* = 1.75$. For the earlier times, the peak pressure reported herein is difficult to compare to that of Granet et al. [70] conditions BC2 and BC3, since the scale is not discernible. For all of the results reported the 'slugging' is unique to the ghost node NSCBC. This only happens within the ghost domain, so while this may be happening with the results of Granet et al. [70], the ghosts visualize the exterior effect of the NSCBC. Interestingly, it does not affect the upstream travel of information.

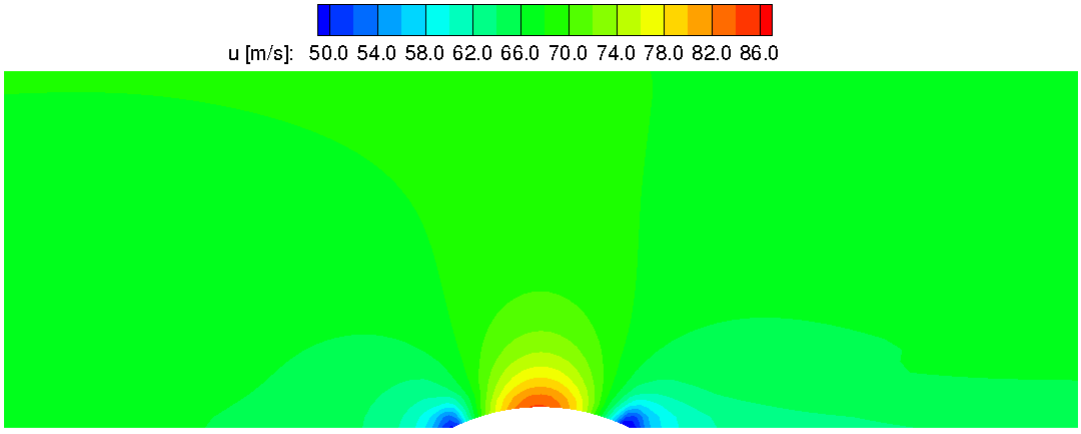
7.2.3 Bump in a Channel: Mach = 0.2

This section presents the results of subsonic flow over a bump in a channel with freestream Mach numbers $M = 0.2$. The grid is structured, with the grid near the wall orthogonal to the boundary. The grids used were shown in Figures 7.3 and 7.4. The flow parameters for this case are defined in Table 7.18

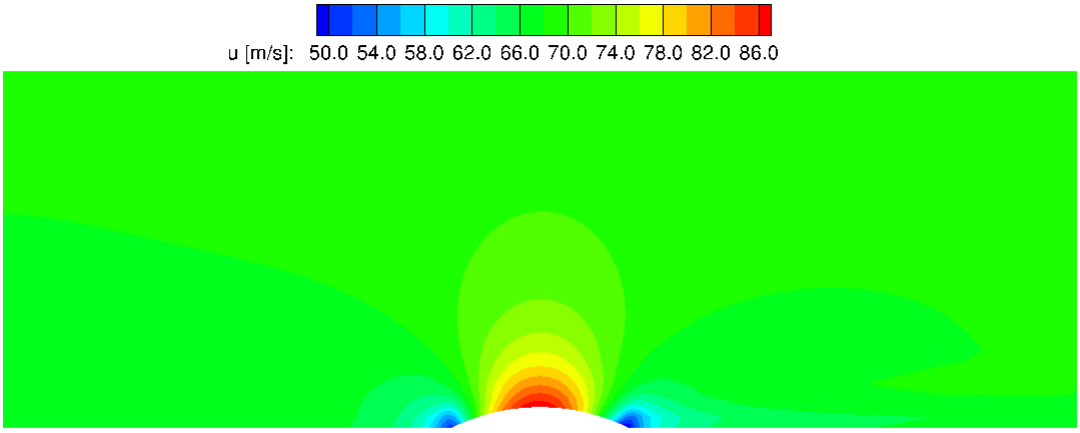
Table 7.18: Initial Conditions for Bump-in-Channel $M = 0.2$ Case.

Quantity	Value	Units
Freestream Mach No., M_∞	0.2	-
Temperature, T_∞	288.15	K
Pressure, p_∞	101325	Pa

The CFL for this case is 0.5, and the flows are run as close to convergence as possible. For results using the numerical methods of Chapter 4, herein referred to as standard, the inlet and outlet boundary conditions are nonreflecting using Riemann invariants [12, Chapter 8], and the far-field boundary is effectively a symmetry boundary. The LSQR method is used for the gradients for the standard boundary results [91]. For the results using the numerical method, including ghost nodes, of Chapter 5, the inlet, outlet, and far-field boundary conditions are NSCBC boundary conditions a described in Sections 5.4.2.3-5.4.2.6. Figures 7.60-7.64 show the contours for (ρ, u, v, p, M) of both the ghost and standard boundary conditions for the medium meshes.

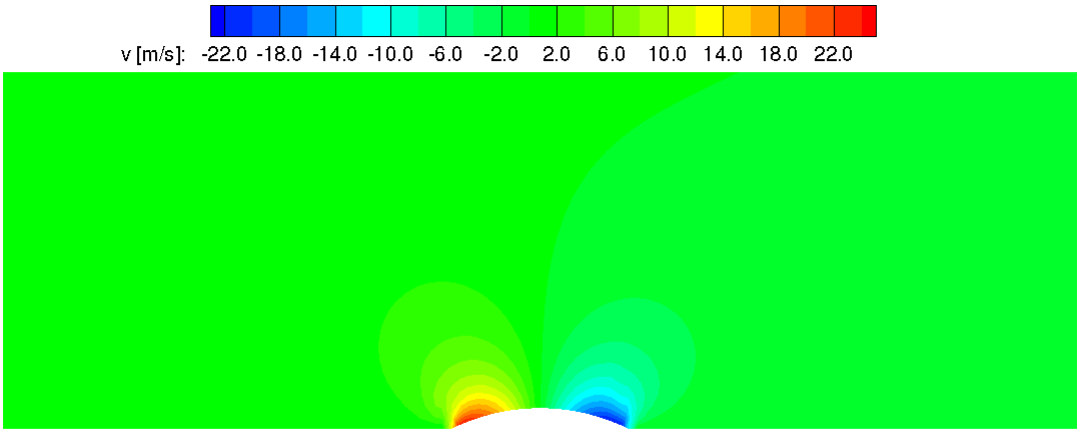


(a) Ghost

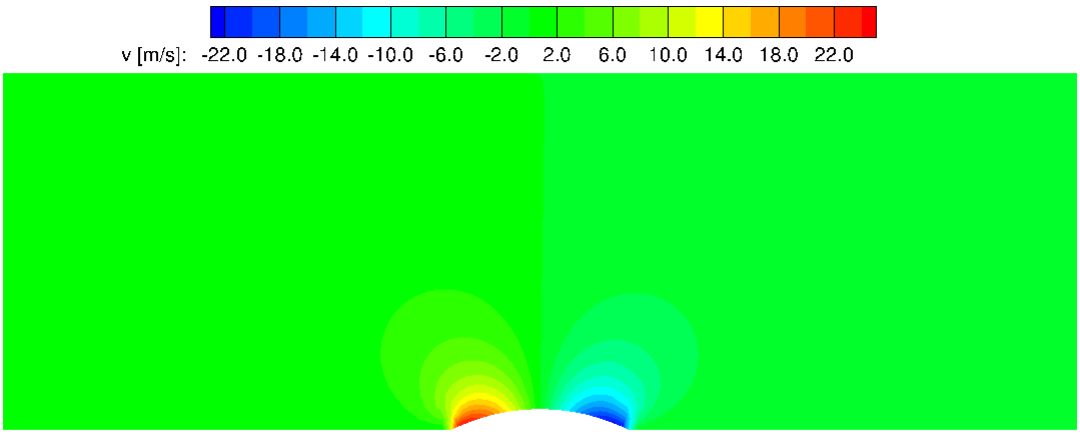


(b) Standard

Figure 7.60: $u-$Velocity Contours for Medium Mesh Bump-in-Channel Case: 2^{nd}-Order $M = 0.2$

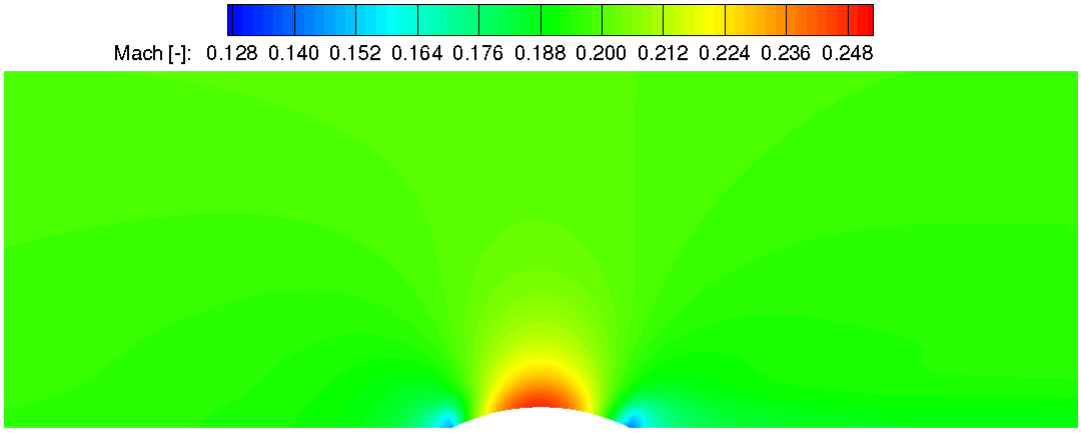


(a) Ghost

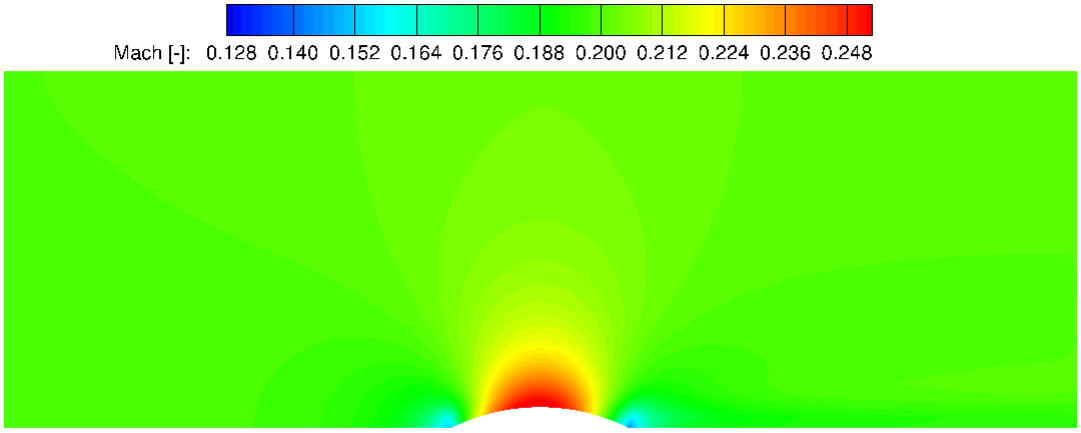


(b) Standard

Figure 7.61: $v-$Velocity Contours for Medium Mesh Bump-in-Channel Case: 2^{nd}-Order $M = 0.2$

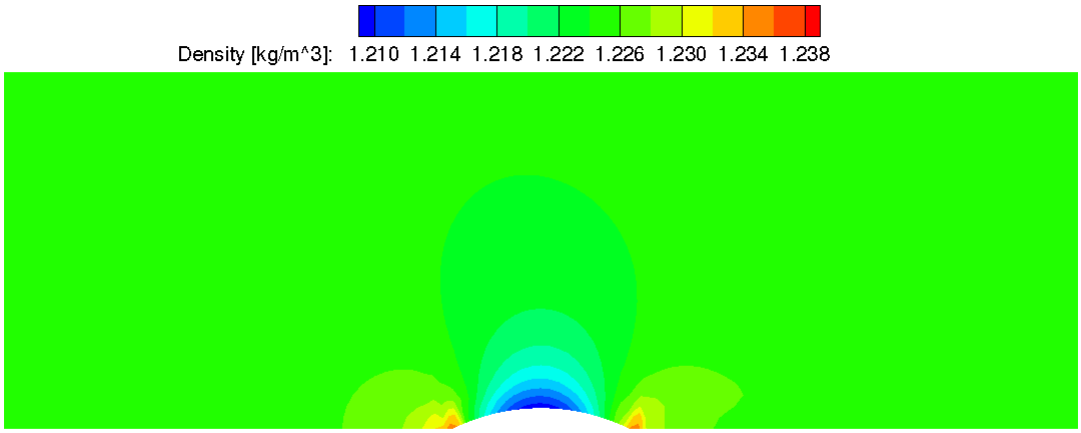


(a) Ghost

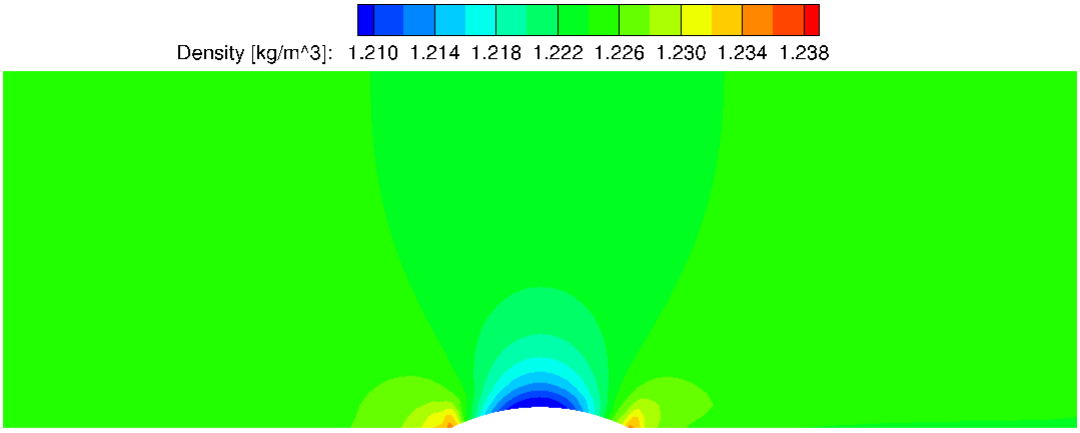


(b) Standard

Figure 7.62: Mach Contours for Medium Mesh Bump-in-Channel Case: 2^{nd}-Order $M = 0.2$

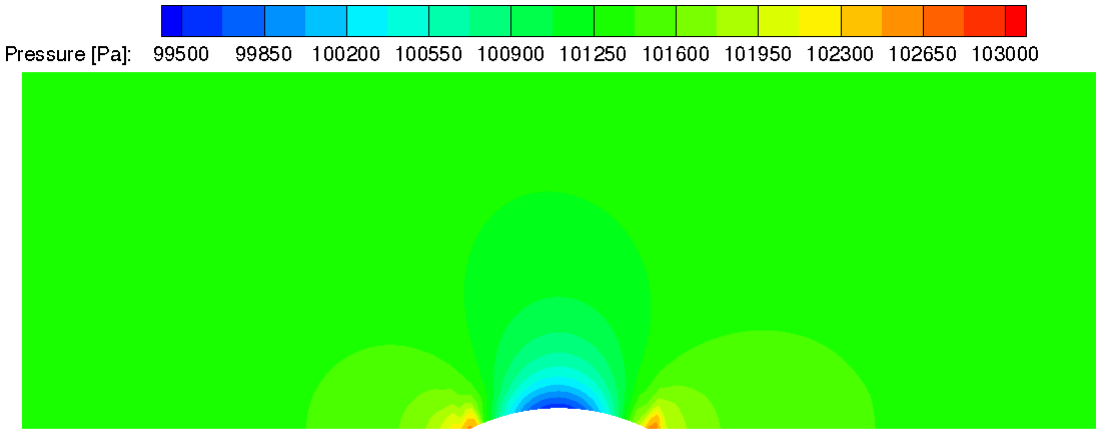


(a) Ghost

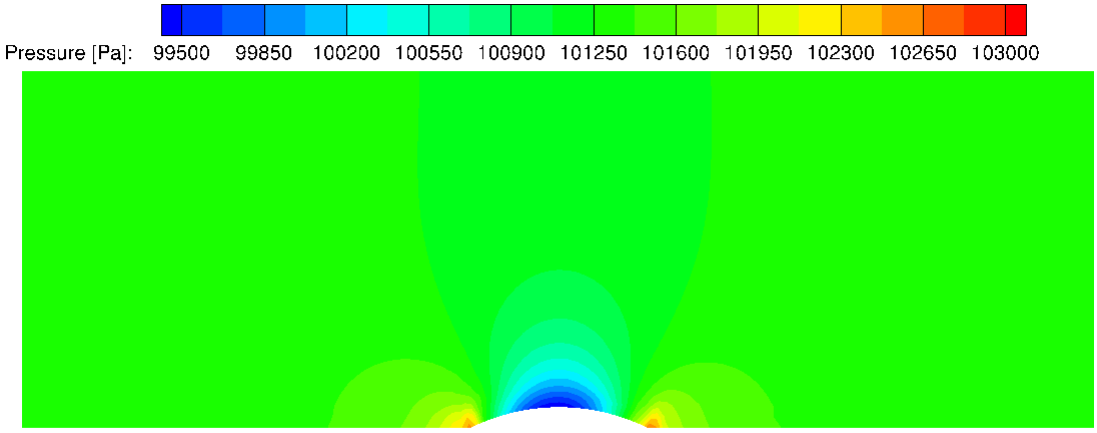


(b) Standard

Figure 7.63: Density Contours for Medium Mesh Bump-in-Channel Case: 2^{nd}-Order $M = 0.2$



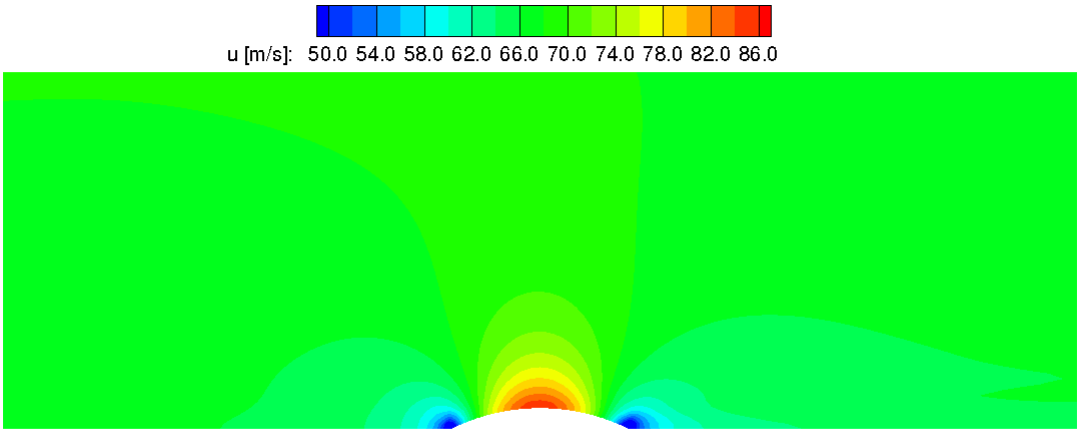
(a) Ghost



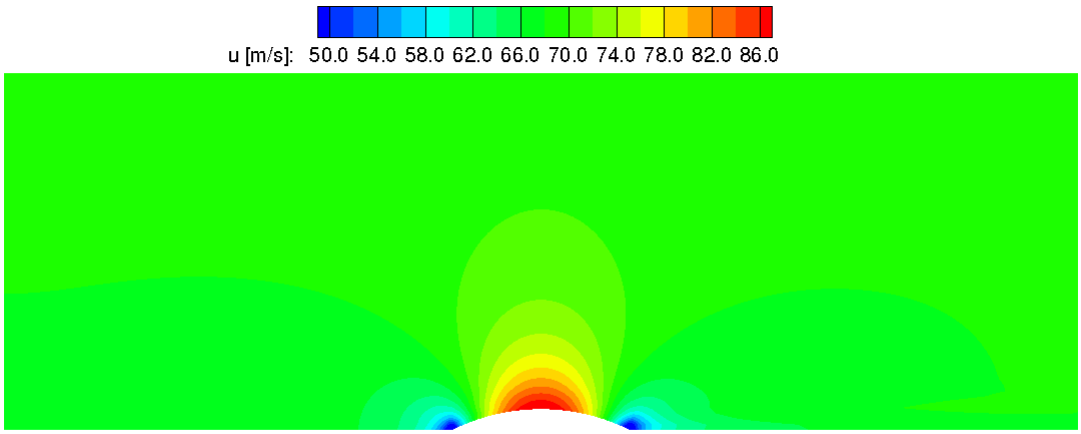
(b) Standard

Figure 7.64: Pressure Contours for Medium Mesh Bump-in-Channel Case: 2^{nd}-Order $M = 0.2$

On visual inspection, there appears to be only minor differences between the ghost results and those with the standard boundary conditions. Some noticeable differences include the density and pressure contours in Figures 7.63 and 7.64, where the standard far-field boundary tends to 'pull' the field up to the boundary while the ghost boundary is more uniform. The remaining boundaries seem similar for both cases, with the ghost boundary condition results not as strong as those produced from the standard boundaries. The finer grid results are presented in Figures 7.65-7.69 to see if results for the medium grids are similar to the finer grids.

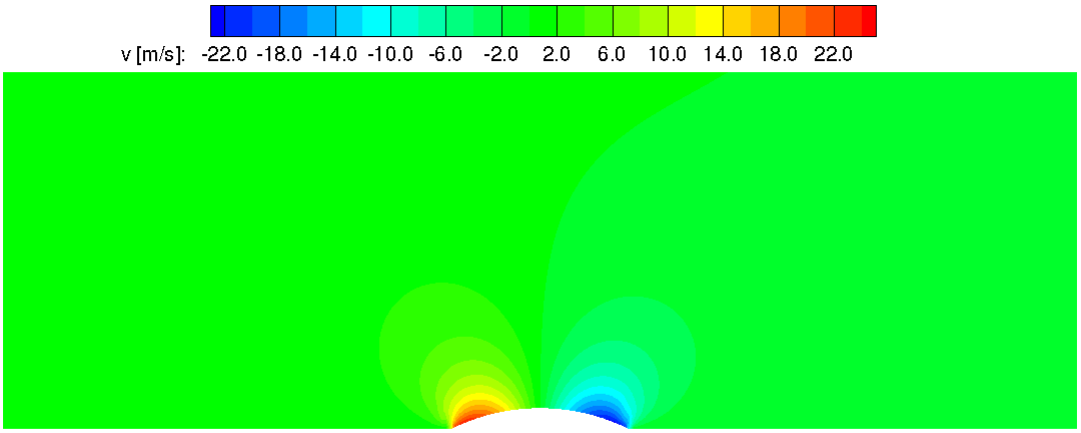


(a) Ghost

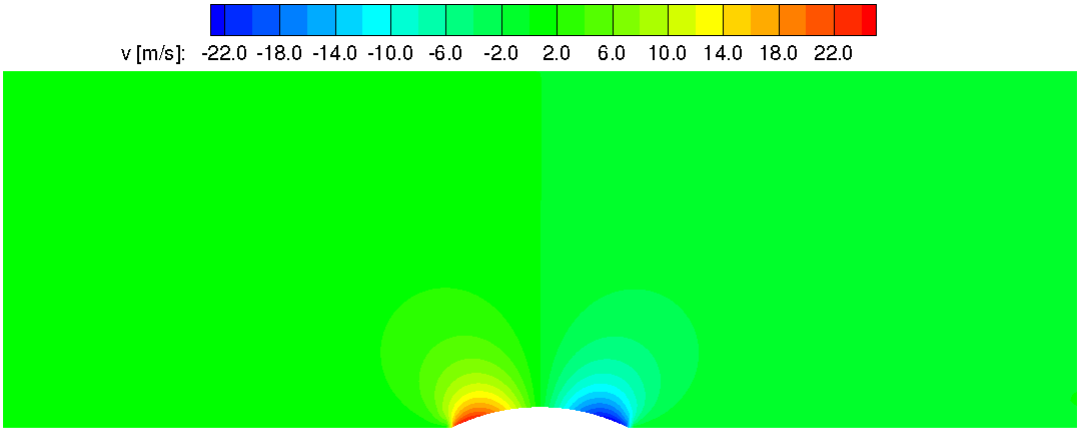


(b) Standard

Figure 7.65: $u-$Velocity Contours for Fine Mesh Bump-in-Channel Case: 2^{nd}-Order $M = 0.2$.

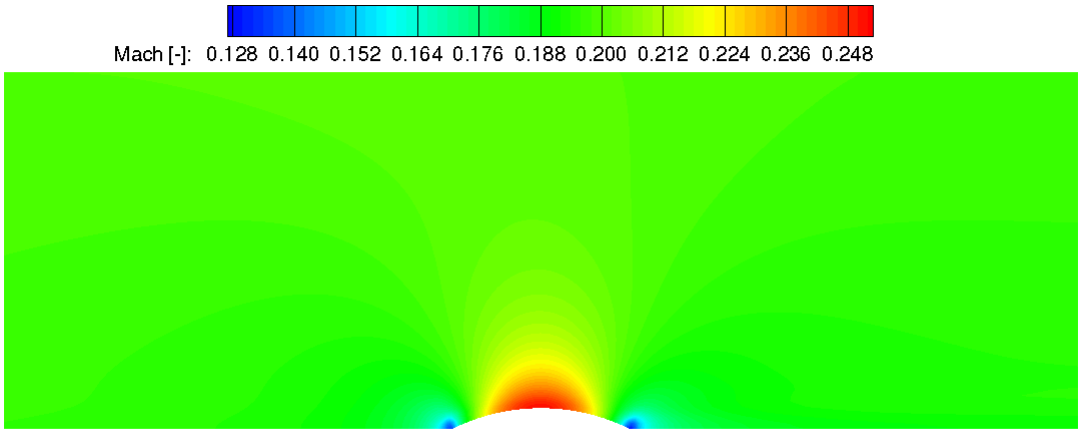


(a) Ghost

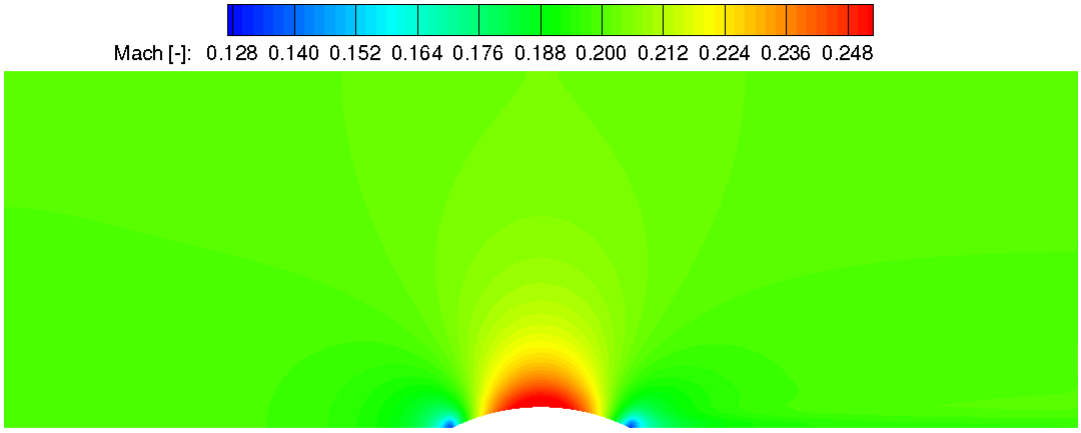


(b) Standard

Figure 7.66: $v-$Velocity Contours for Fine Mesh Bump-in-Channel Case: 2^{nd}-Order $M = 0.2$.

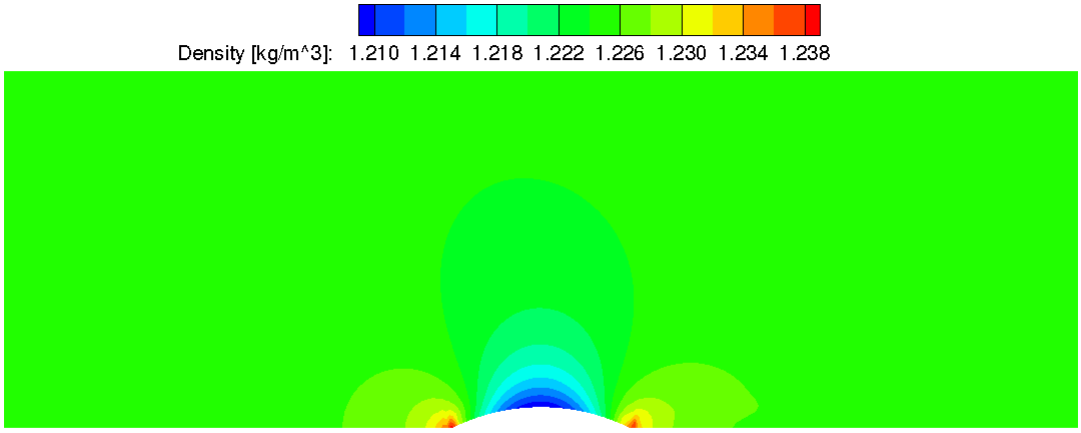


(a) Ghost

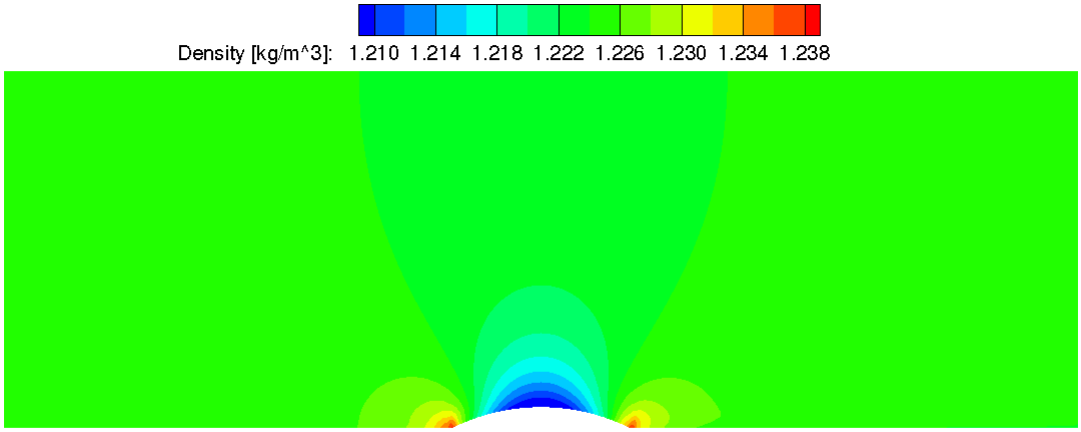


(b) Standard

Figure 7.67: Mach Contours for Fine Mesh Bump-in-Channel Case: 2^{nd}-Order $M = 0.2$.

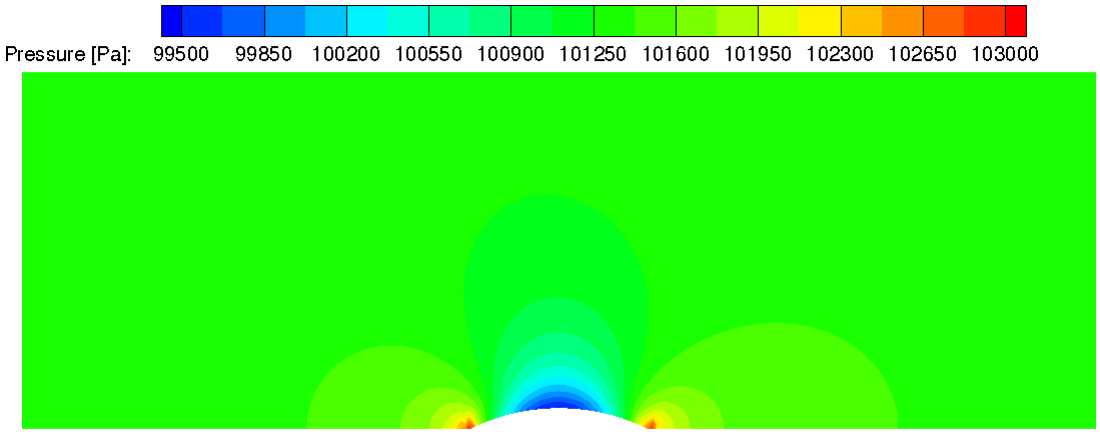


(a) Ghost

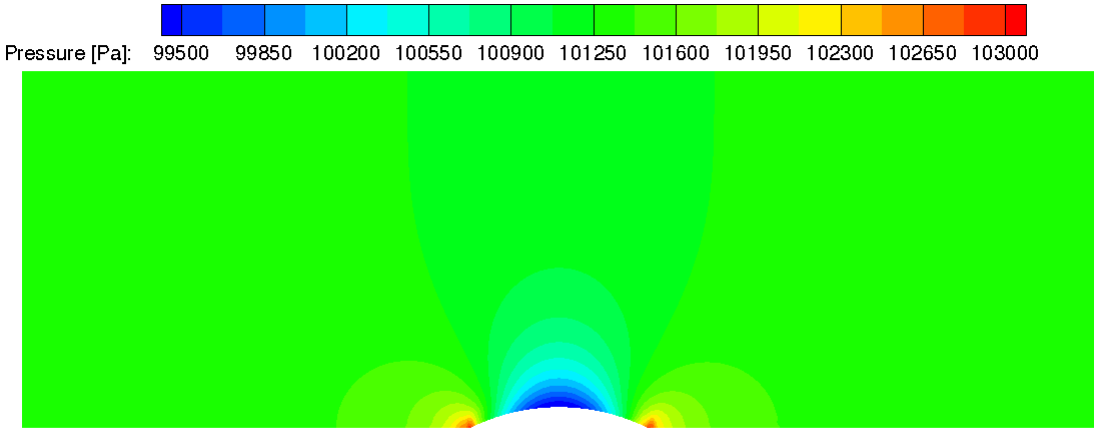


(b) Standard

Figure 7.68: Density Contours for Fine Mesh Bump-in-Channel Case: 2^{nd}-Order $M = 0.2$.



(a) Ghost



(b) Standard

Figure 7.69: Pressure Contours for Fine Mesh Bump-in-Channel Case: 2^{nd}-Order $M = 0.2$.

As with the medium grid results, the finer grid results do not appear significantly different. The far-field boundary again 'pulls' the information up toward to boundary in the standard case, and the ghost case has a more uniform field. The question should then be asked, is there any benefit to using ghost nodes? The residual plots for the medium grid are shown in Figure 7.70.

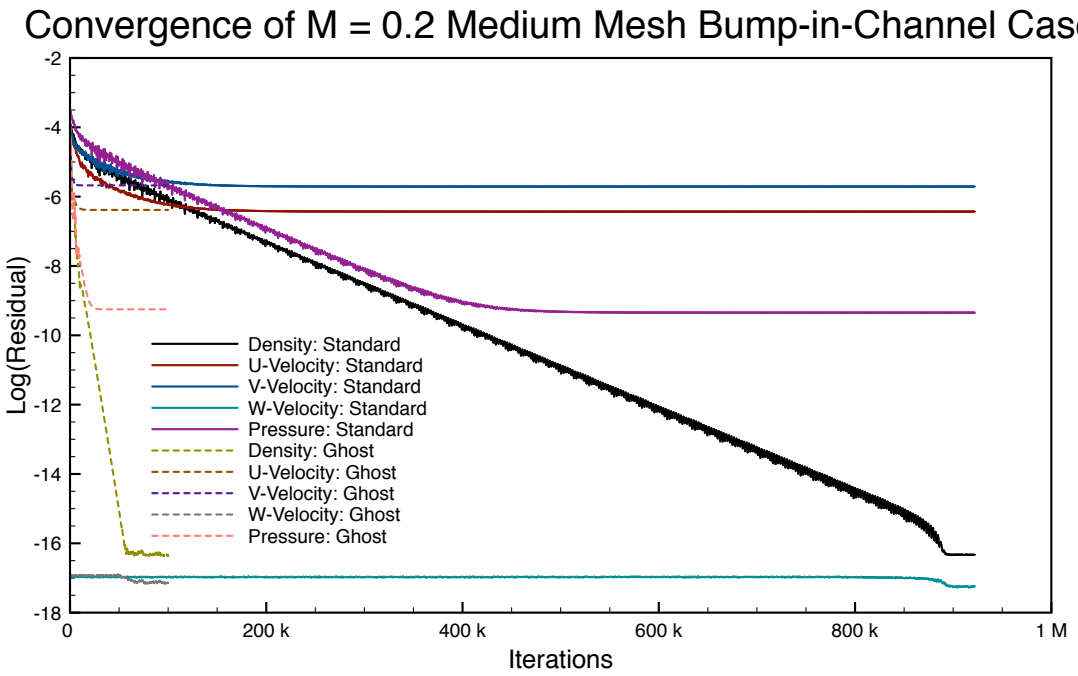


Figure 7.70: Residuals for Medium Mesh Bump-in-Channel Case: 2^{nd}-Order $M = 0.2$.

The ghost boundaries converge nearly ten times faster than the standard boundary condition. Figure 7.71 shows the residuals for the finer mesh.

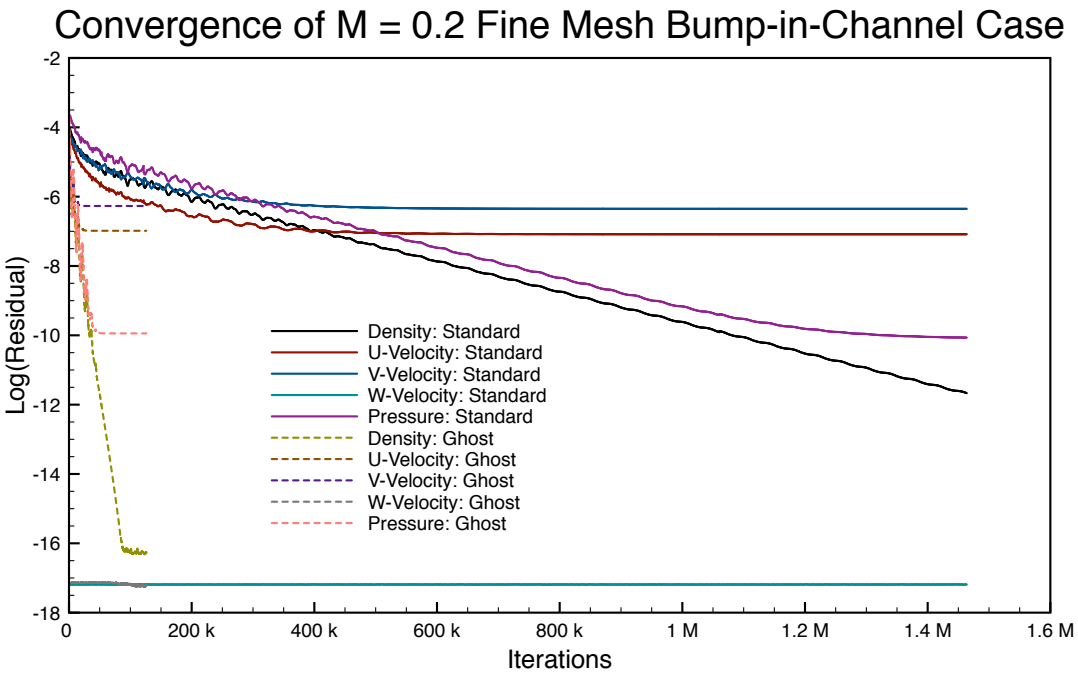


Figure 7.71: Residuals for Fine Mesh Bump-in-Channel Case: 2^{nd}-Order $M = 0.2$.

The ghost boundaries converge at approximately one-hundred thousand iterations, while even at nearly 1.5 million iterations, the standard boundaries have not converged. Why do the ghost node boundaries converge so much better than the standard boundaries? The obvious place to check first is the far-field boundaries. Figures 7.72 and 7.73 show the first 900 iterations of the simulation on the fine mesh.

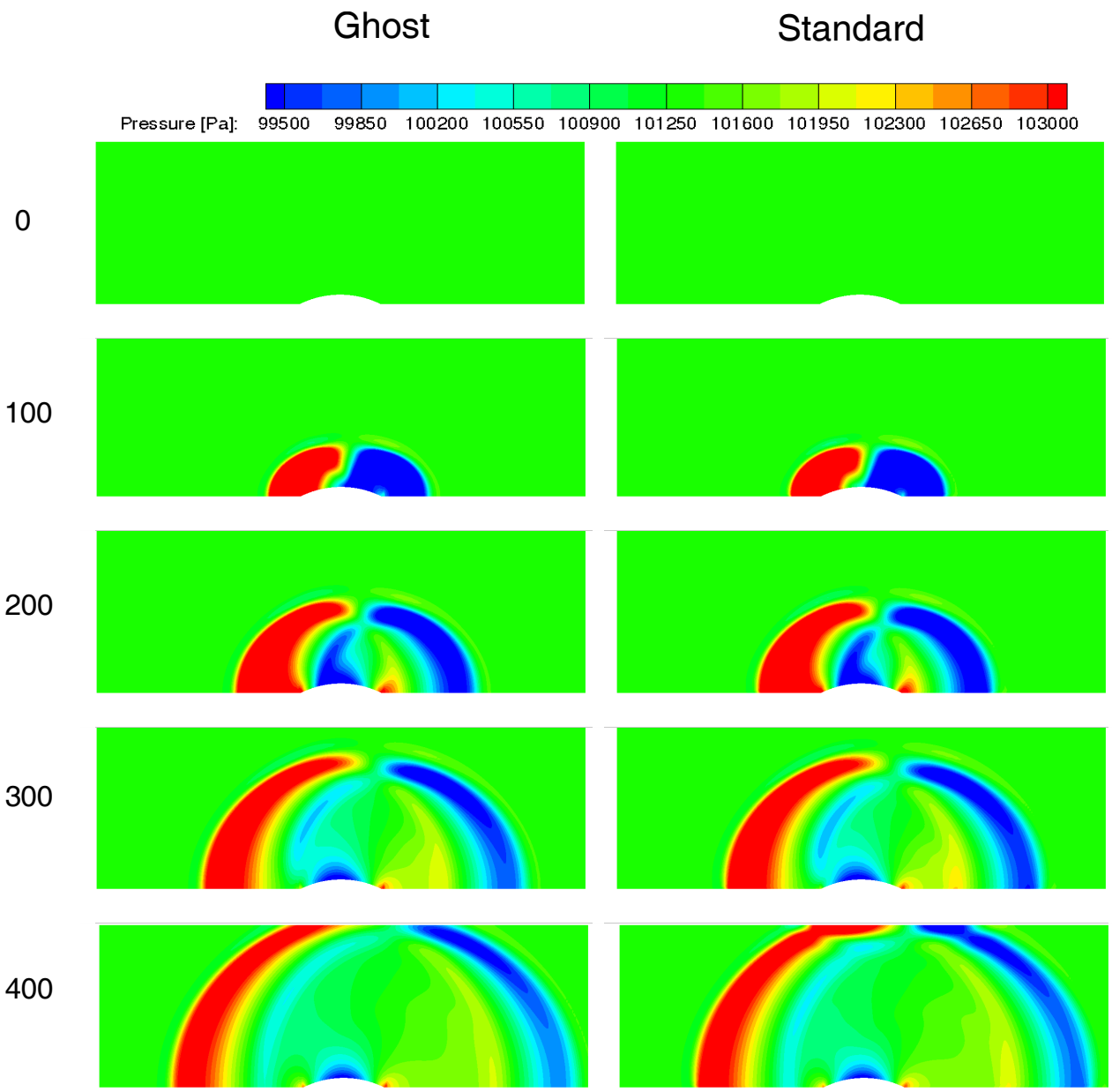


Figure 7.72: Comparison of Initial Startup for Bump-in-Channel $M = 0.2$ Case: Iterations 0-400.

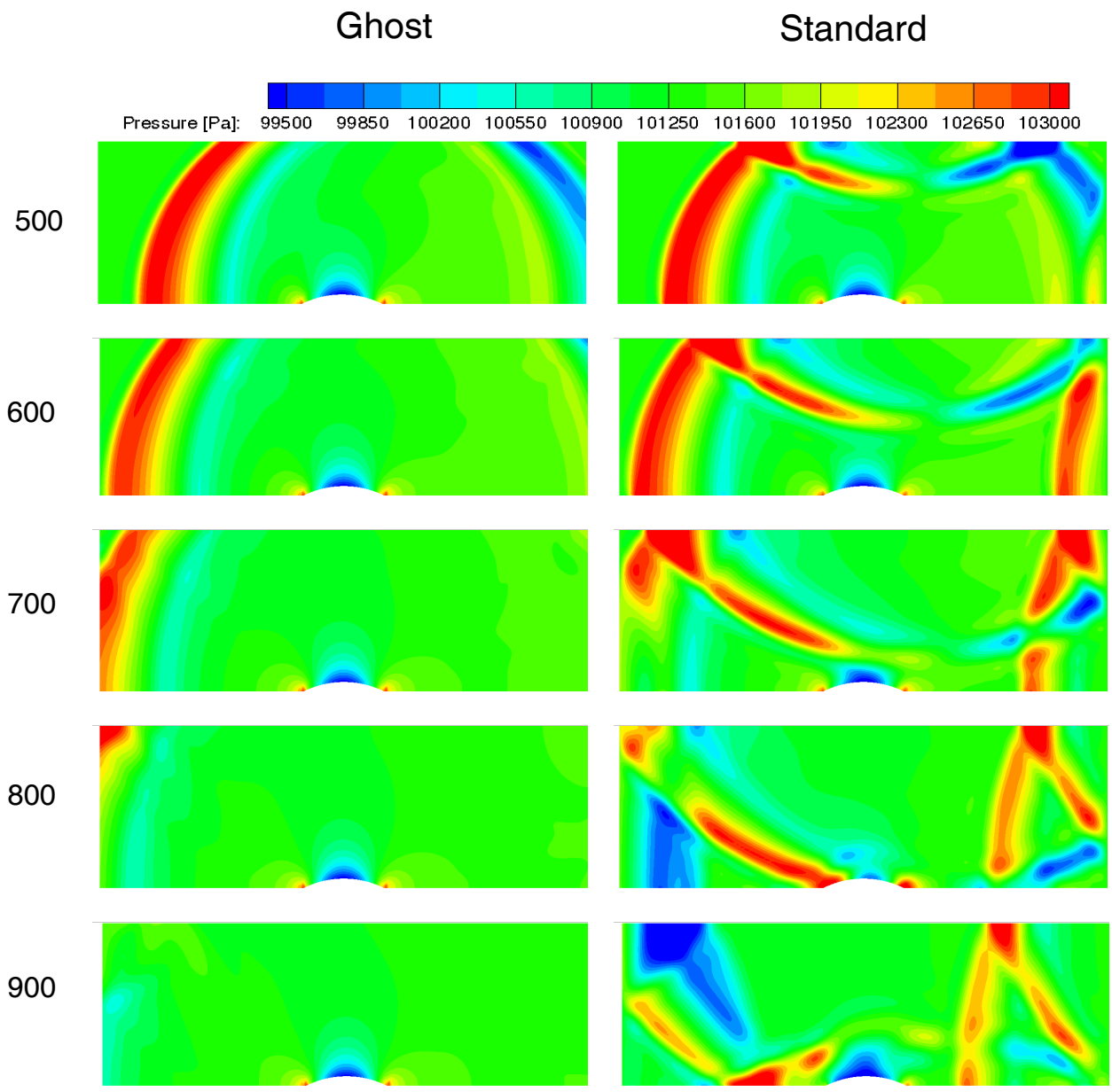


Figure 7.73: Comparison of Initial Startup for Bump-in-Channel $M = 0.2$ Case: Iterations 500-900.

The initial field is uniform, and by iteration 100 large pressure waves have formed and are moving away from the bump. The flow fields are similar up to iteration 300. At iteration 400, the pressure waves have reached the far-field boundary. The ghost boundary has allowed the waves to exit the boundary undisturbed. The standard boundaries, however, have generated a reflection that will propagate downward into the domain. Viewing Figure 7.73, the ghost boundaries have allowed the waves to travel relatively undisturbed for both iterations 500 and 600, such that the downstream traveling wave has left the domain effectively undisturbed. The standard boundaries at those times, however, have generated a large reflection at the outlet of the domain that now begins to travel upstream and will interact with the waves generated from the far-field boundary. For the remaining iterations (700-900) shown here, the standard boundary conditions bounce the pressure waves around the domain. The ghost boundaries, however, nearly eliminate the initially generated waves, though some are reflected back downstream from the inlet. This, though, is expected and required to ensure the freestream conditions are maintained.

With the standard boundaries are so reflective, the large numerical waves initially generated at the start of the simulation effectively are trapped in the domain and bounce around until they dampen out, if they ever really do. If these boundaries were fixed in the standard configuration, does the standard wall boundary out-perform the ghost wall boundary? Recall that at the wall boundary, the wall normal pressure gradient should be 0 (5.165), except on a curved wall, where the normal pressure gradient is a function of radius of curvature, as given by (5.167). Figure 7.74 shows a comparison of the wall normal pressure gradient for both the medium and fine cases for each boundary type.

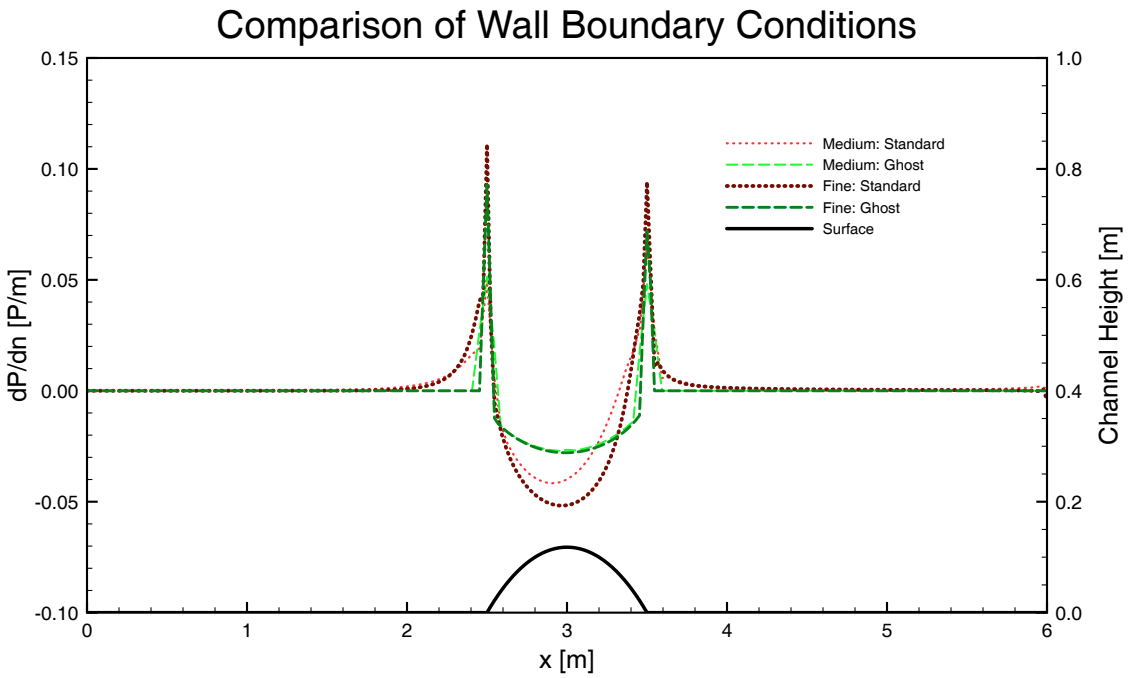


Figure 7.74: Comparison of Wall-Normal Pressure Gradient for Bump-in-Channel Case: $M = 0.2$.

First, the on bump surface, both ghost boundary wall pressure gradients are the same, since the boundary condition is prescribed by (5.167). The standard boundary wall normal pressures are not, nor are they centered on the bump, which for this low speed should not be the case. There appears to be a noticeable difference on the wall behind the bump as well. Figure 7.75 shows the pressure on the back half of the domain.

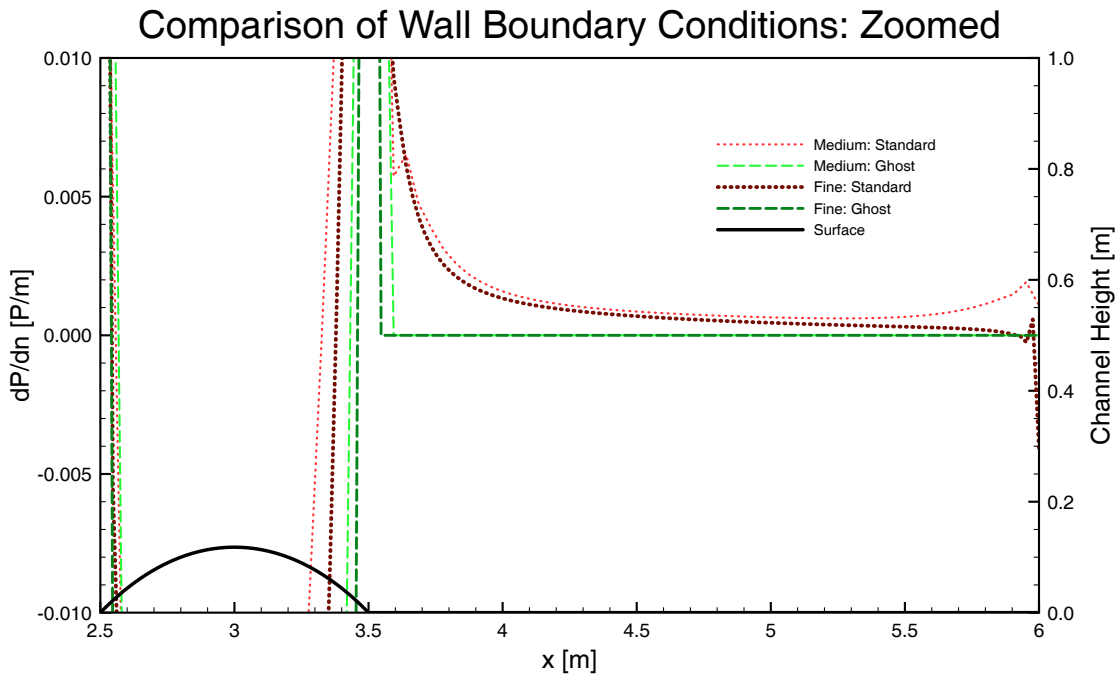


Figure 7.75: Comparison of Wall-Normal Pressure Gradient for Bump-in-Channel Case: $M = 0.2$, Trailing Edge of Domain.

The ghost node boundary has a zero-gradient boundary, since it is prescribed. The standard boundary condition never does for both the medium and fine case, which is incorrect. Near the outlet of the domain, the standard boundary condition has an increase in the wall normal pressure. This can be linked to the fact that the wall is not extended beyond the domain and that the outlet does not account for the wall, leading to a mismatch at the intersection of the boundaries. From this case, it is fairly clear the ghost node solutions is superior to the standard boundary solutions.

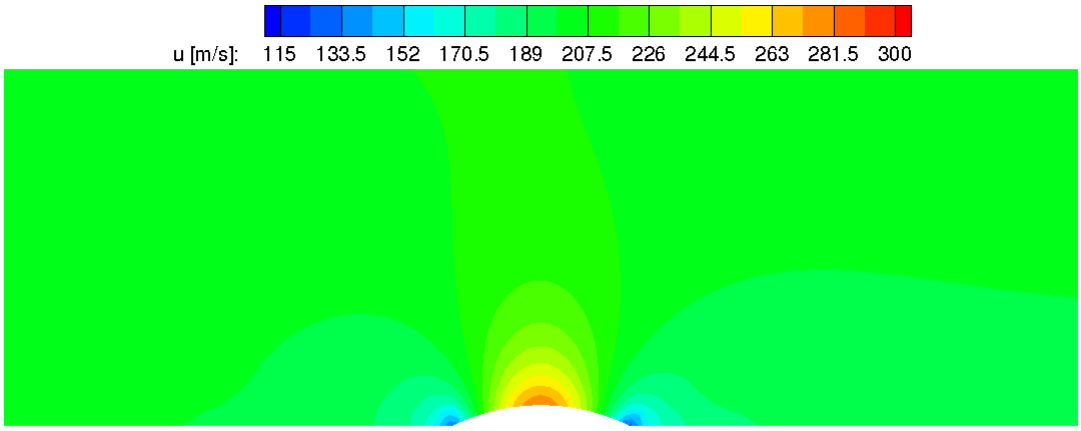
7.2.4 Bump in a Channel: Mach = 0.6

This section presents the results of subsonic flow over a bump in a channel with freestream Mach numbers $M = 0.6$. The grid is structured, with the grid near the wall orthogonal to the boundary. The grids used were shown in Figures 7.3 and 7.4. The flow parameters for this case are defined in Table 7.19

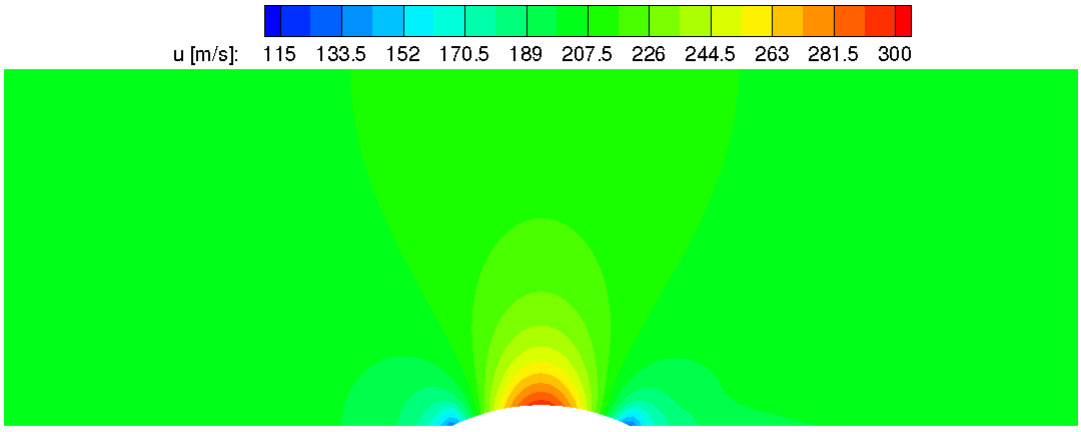
Table 7.19: Initial Conditions for Bump-in-Channel $M = 0.6$ Case.

Quantity	Value	Units
Freestream Mach No., M_∞	0.6	-
Temperature, T_∞	288.15	K
Pressure, p_∞	101325	Pa

The CFL for this case is 0.5, and the flows are run as close to convergence as possible. For results using the numerical methods of Chapter 4, herein referred to as standard, the inlet and outlet boundary conditions are nonreflecting using Riemann invariants [12, Chapter 8], and the far-field boundary is effectively a symmetry boundary. The LSQR method is used for the gradients for the standard boundary results [91]. For the results using the numerical method, including ghost nodes, of Chapter 5, the inlet, outlet, and far-field boundary conditions are NSCBC boundary conditions a described in Sections 5.4.2.3-5.4.2.6. Figures 7.76-7.80 show the contours for (ρ, u, v, p, M) of both the ghost and standard boundary conditions for the medium meshes.

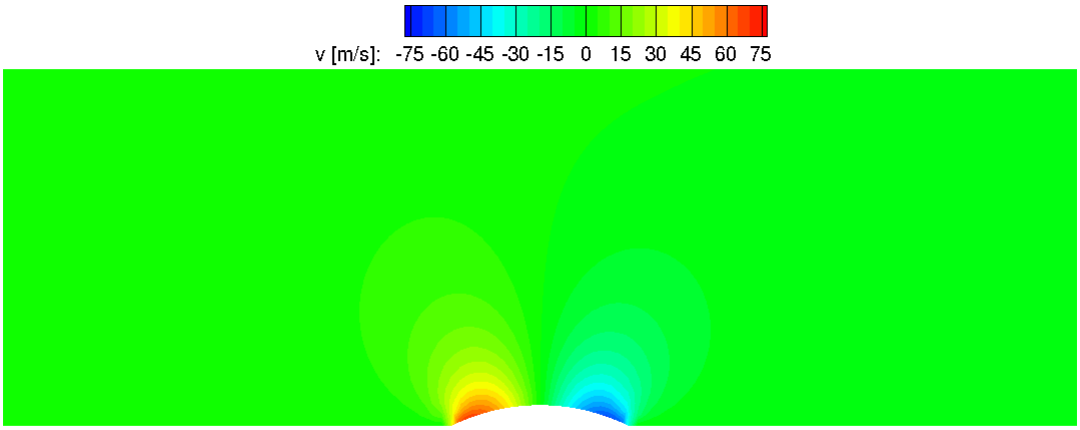


(a) Ghost

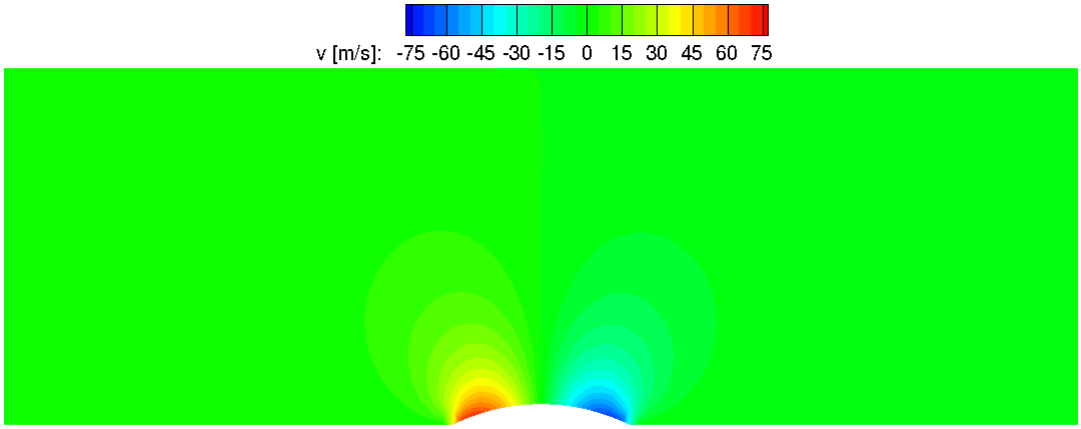


(b) Standard

Figure 7.76: $u-$Velocity Contours for Medium Mesh Bump-in-Channel Case: 2^{nd}-Order $M = 0.6$.

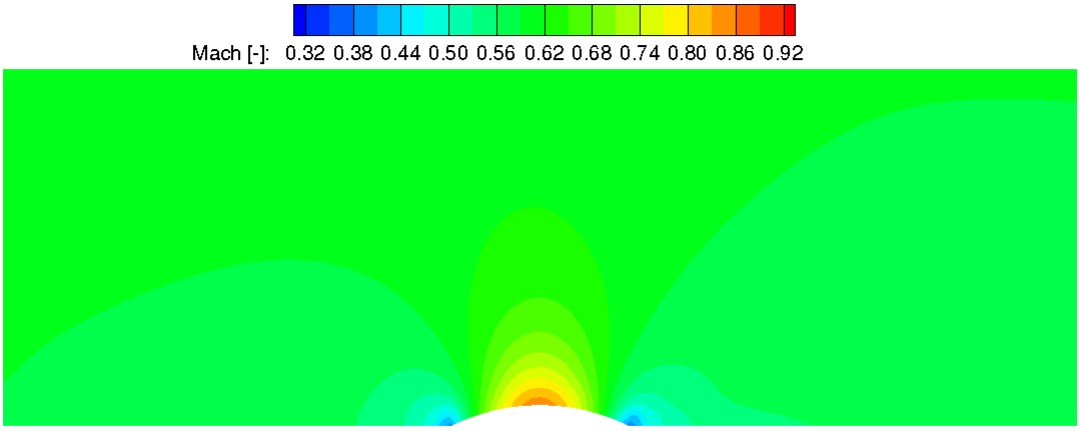


(a) Ghost

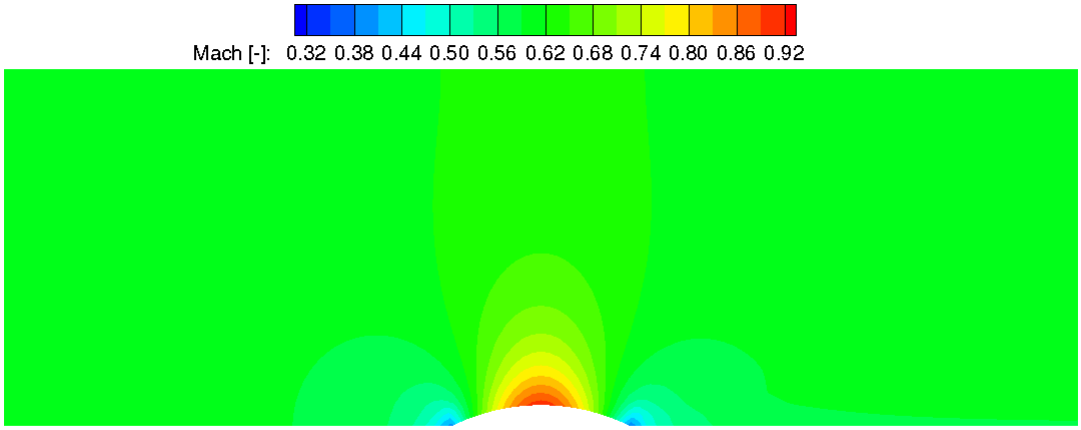


(b) Standard

Figure 7.77: $v-$Velocity Contours for Medium Mesh Bump-in-Channel Case: 2^{nd}-Order $M = 0.6$.

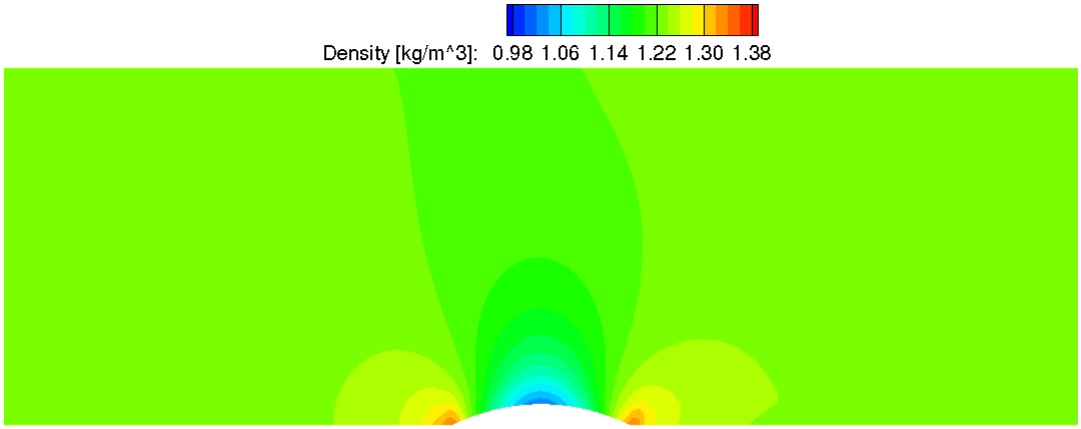


(a) Ghost

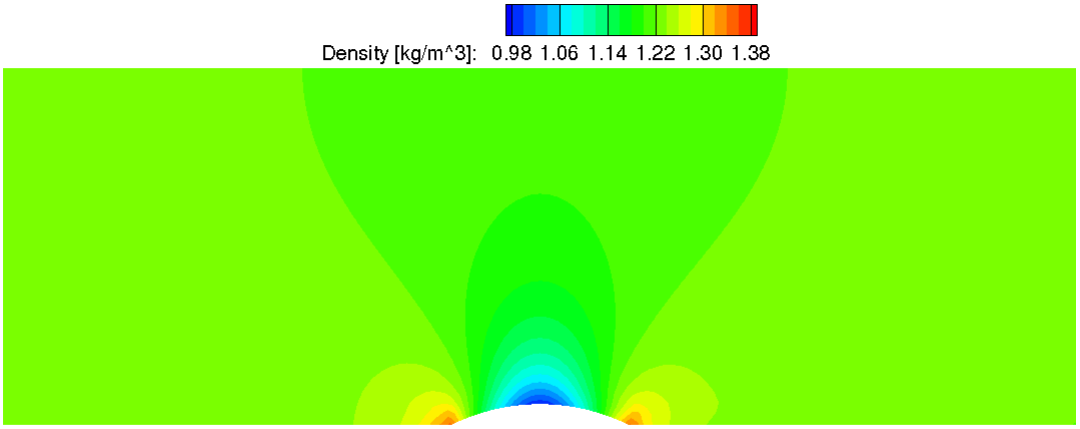


(b) Standard

Figure 7.78: Mach Contours for Medium Mesh Bump-in-Channel Case: 2^{nd}-Order $M = 0.6$.

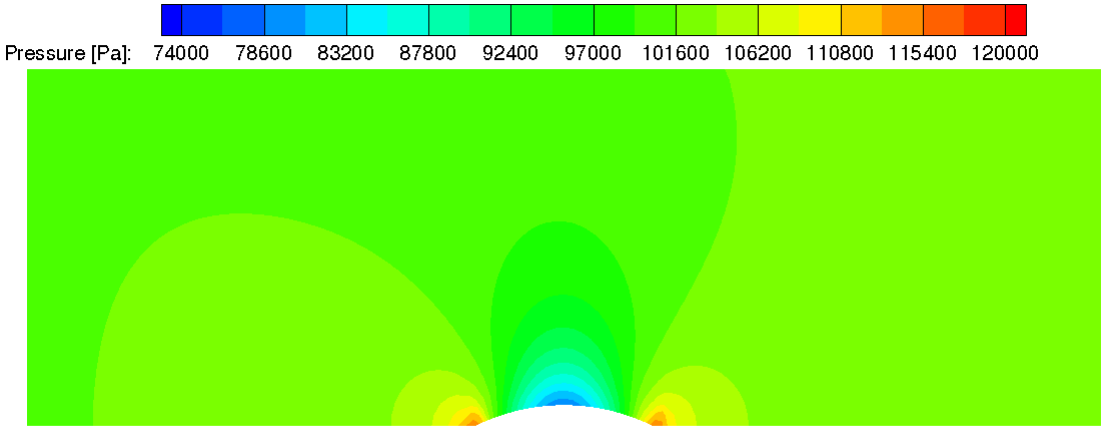


(a) Ghost

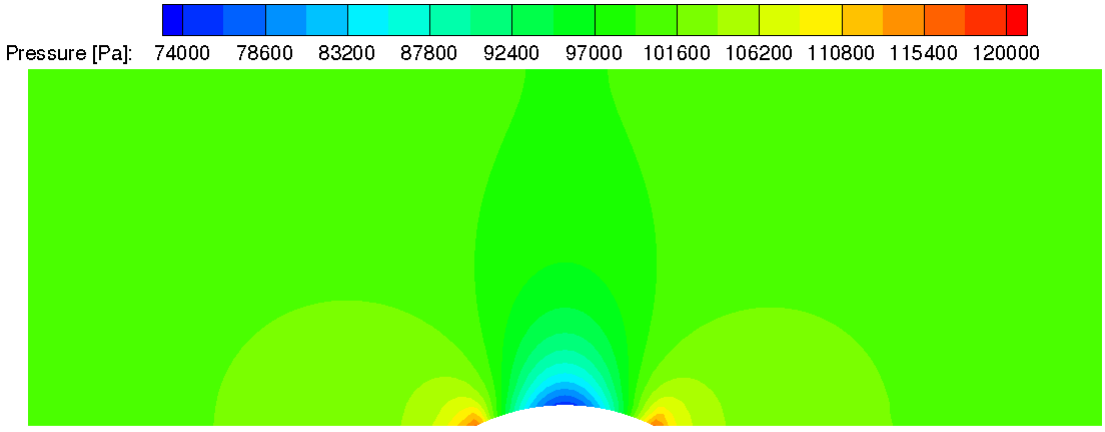


(b) Standard

Figure 7.79: Density Contours for Medium Mesh Bump-in-Channel Case: 2^{nd}-Order $M = 0.6$.



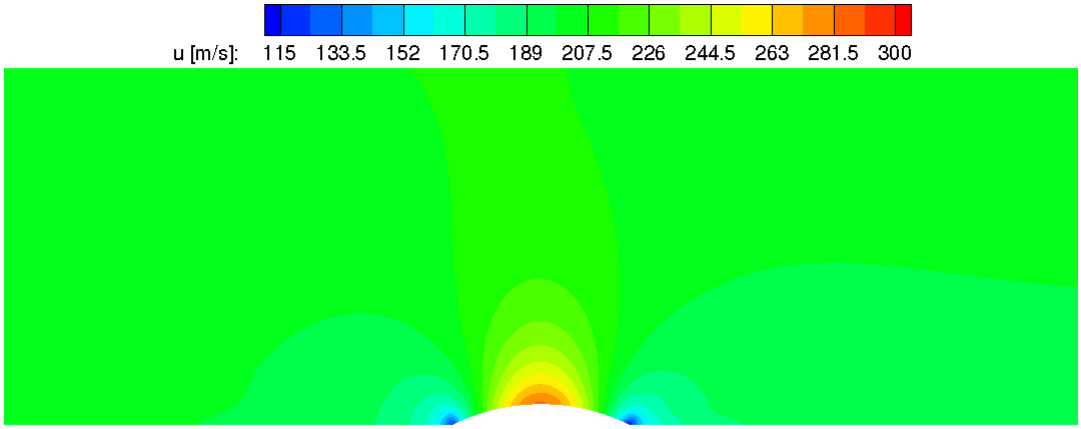
(a) Ghost



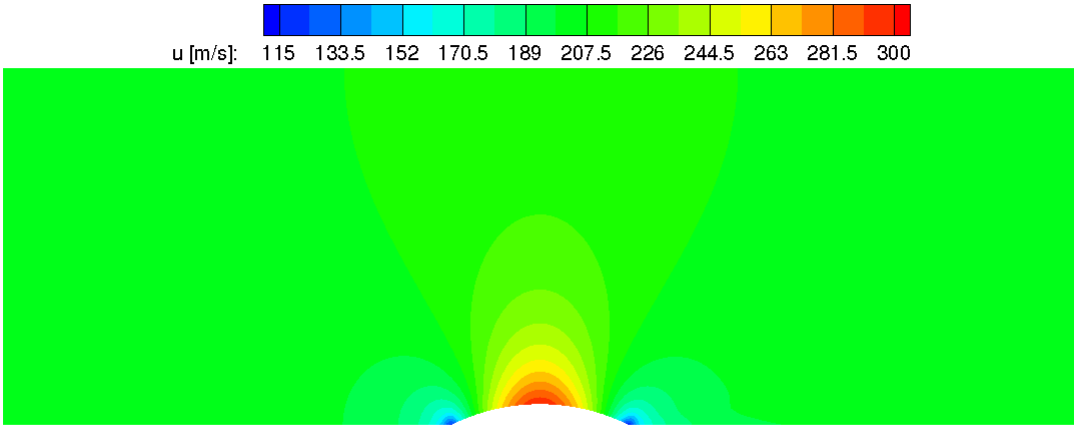
(b) Standard

Figure 7.80: Pressure Contours for Medium Mesh Bump-in-Channel Case: 2^{nd}-Order $M = 0.6$.

On visual inspection, there appears to be only minor differences between the ghost results and those with the standard boundary conditions. Some noticeable differences include the density and pressure contours in Figures 7.63 and 7.64, where the standard far-field boundary tends to 'pull' the field up to the boundary while the ghost boundary is more uniform. The remaining boundaries seem similar for both cases, with the ghost boundary condition results not as strong as those produced from the standard boundaries. The finer grid results are presented in Figures 7.81-7.85 to see if results for the medium grids are similar to the finer grids.

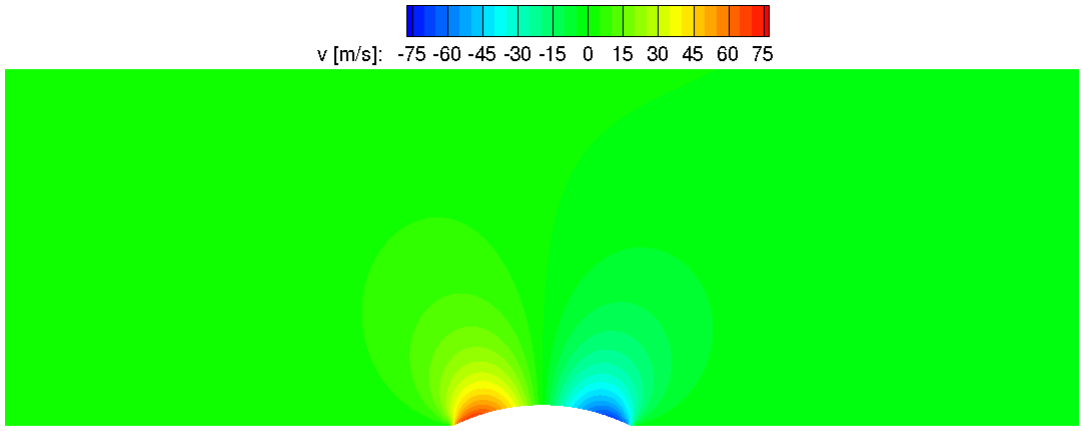


(a) Ghost

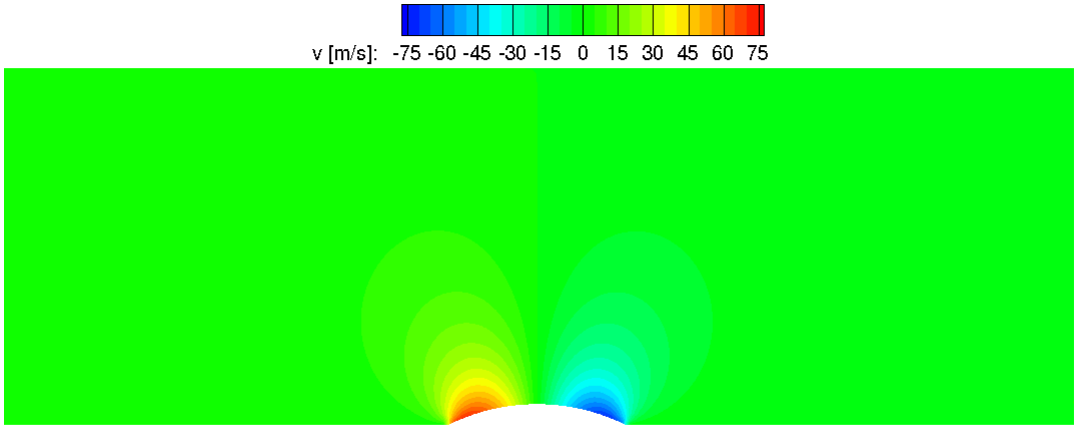


(b) Standard

Figure 7.81: $u-$Velocity Contours for Fine Mesh Bump-in-Channel Case: 2^{nd}-Order $M = 0.6$.

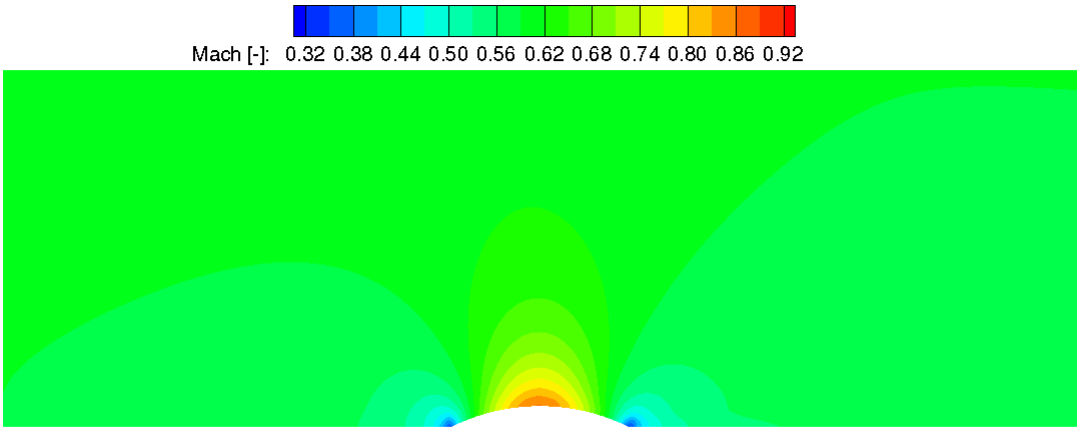


(a) Ghost

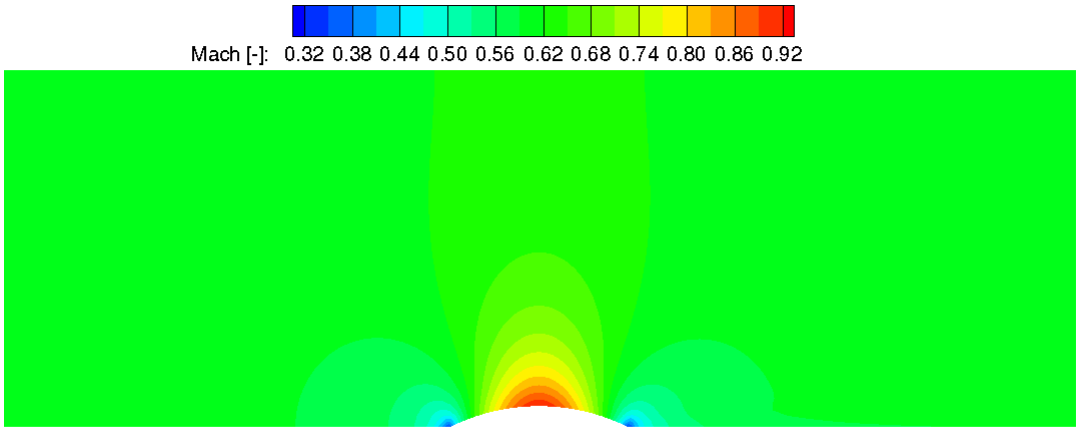


(b) Standard

Figure 7.82: $v-$Velocity Contours for Fine Mesh Bump-in-Channel Case: 2^{nd}-Order $M = 0.6$.

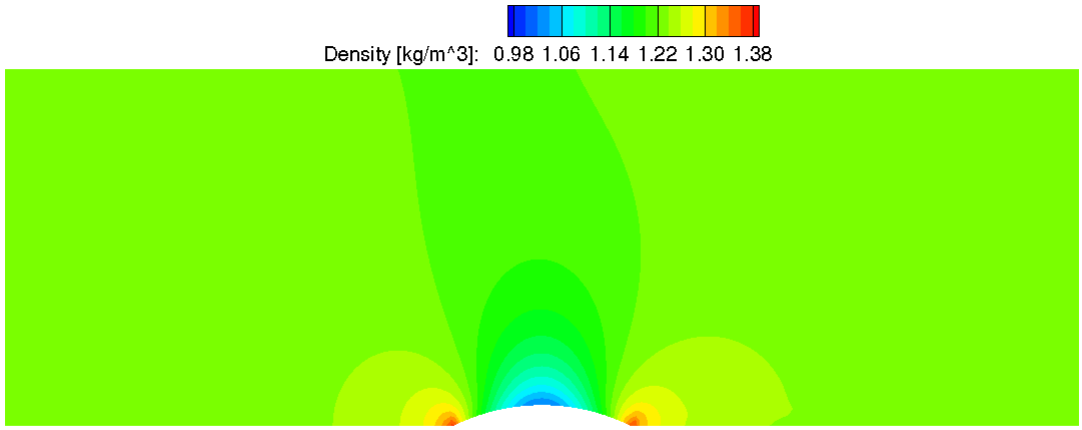


(a) Ghost

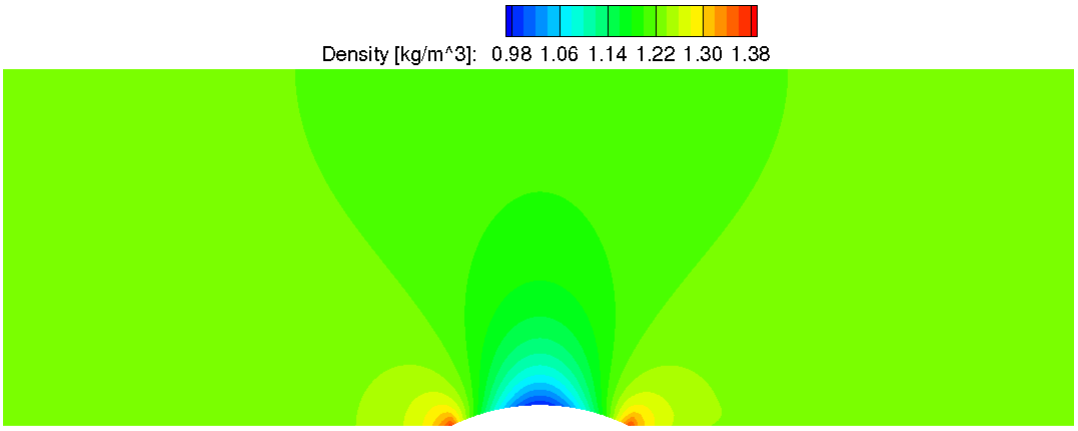


(b) Standard

Figure 7.83: Mach Contours for Fine Mesh Bump-in-Channel Case: 2^{nd}-Order $M = 0.6$.

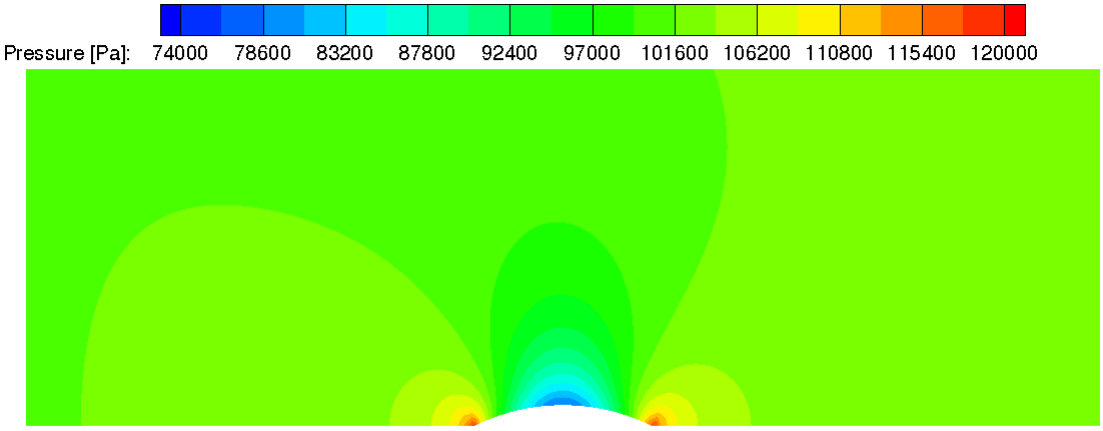


(a) Ghost

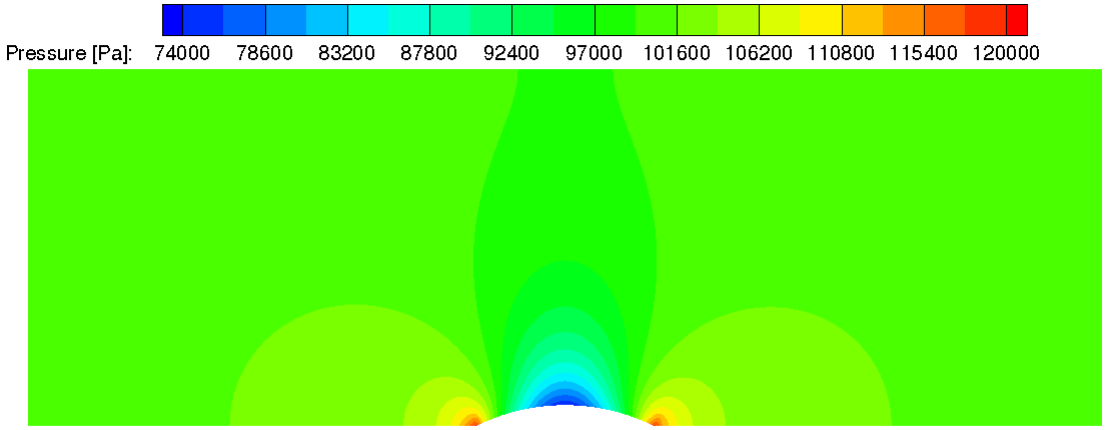


(b) Standard

Figure 7.84: Density Contours for Fine Mesh Bump-in-Channel Case: 2^{nd}-Order $M = 0.6$.



(a) Ghost



(b) Standard

Figure 7.85: Pressure Contours for Fine Mesh Bump-in-Channel Case: 2^{nd}-Order $M = 0.6$.

As with the medium grid results, the finer grid results do not appear significantly different. The far-field boundary again 'pulls' the information up toward to boundary in the standard case, and the ghost case has a more uniform field. The question should then be asked, is there any benefit to using ghost nodes? The residual plots for the medium grid are shown in Figure 7.86.

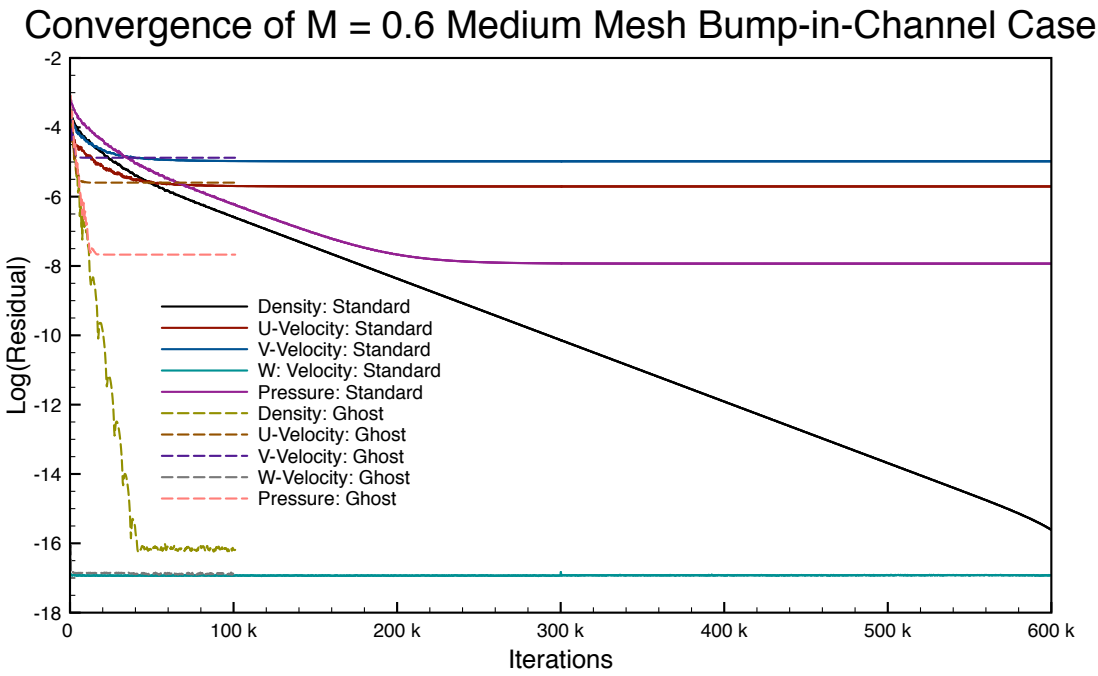


Figure 7.86: Residuals for Medium Mesh Bump-in-Channel Case: 2^{nd}-Order $M = 0.6$.

The ghost boundaries converge nearly ten times faster than the standard boundary condition. Figure 7.87 shows the residuals for the finer mesh.

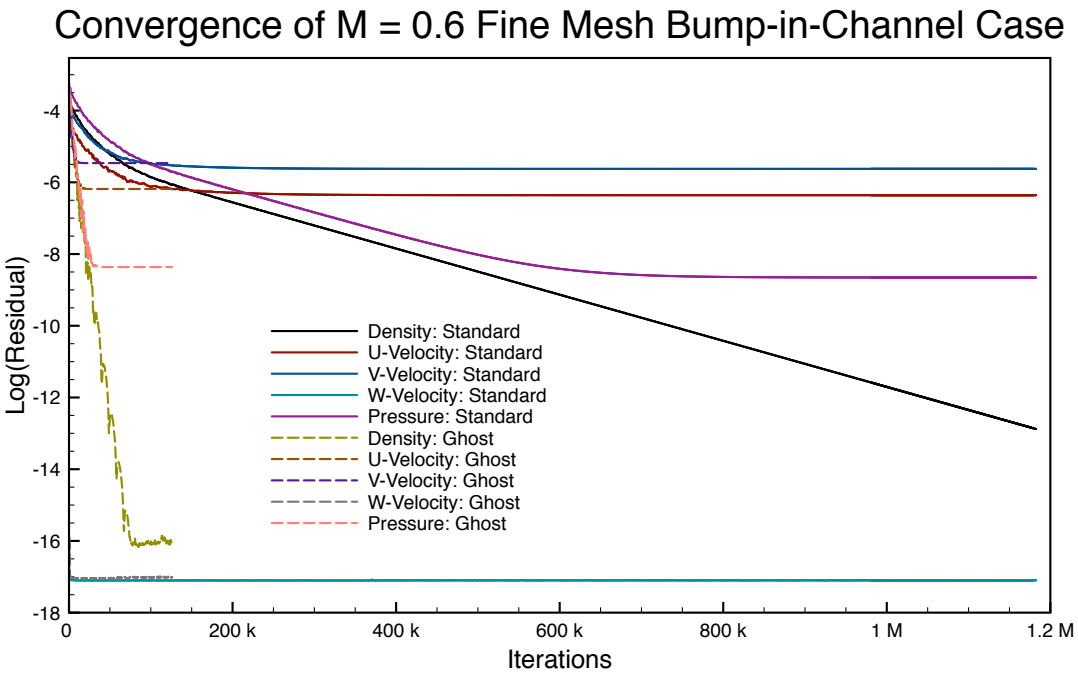


Figure 7.87: Residuals for Fine Mesh Bump-in-Channel Case: 2^{nd}-Order $M = 0.6$.

The ghost boundaries converge at approximately one-hundred thousand iterations, while even at nearly 1.5 million iterations, the standard boundaries have not converged. Why do the ghost node boundaries converge so much better than the standard boundaries? The obvious place to check first is the far-field boundaries. Figures 7.88 show the first 400 iterations on the fine mesh.

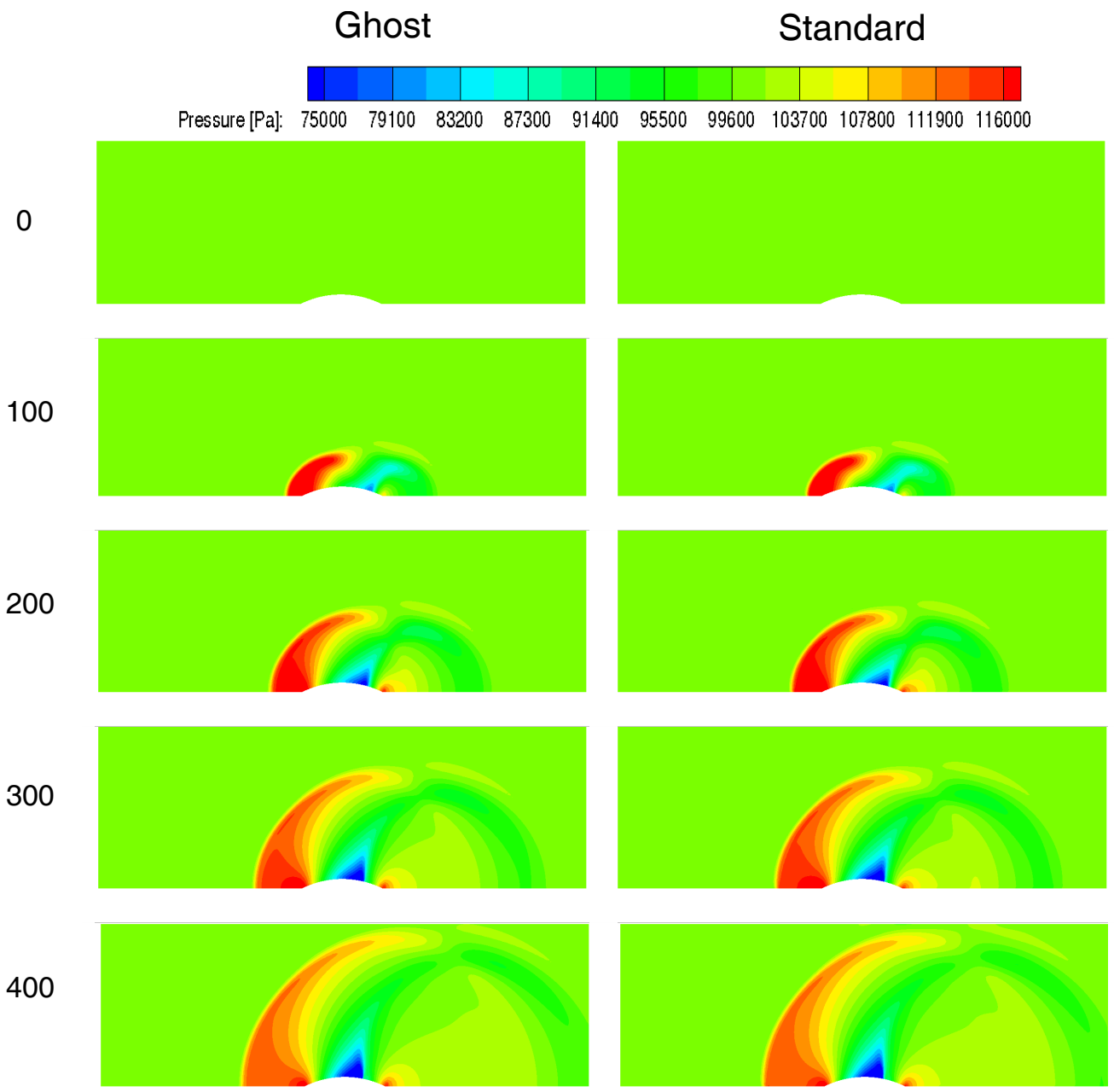


Figure 7.88: Comparison of Initial Startup for Bump-in-Channel $M = 0.6$ Case: Iterations 0-400.

For the first 400 iterations, there is no appreciable difference to the flows using the ghost or standard boundaries, with the initial pressure waves generated by iteration 100. Figure 7.89 shows iterations 500-900.

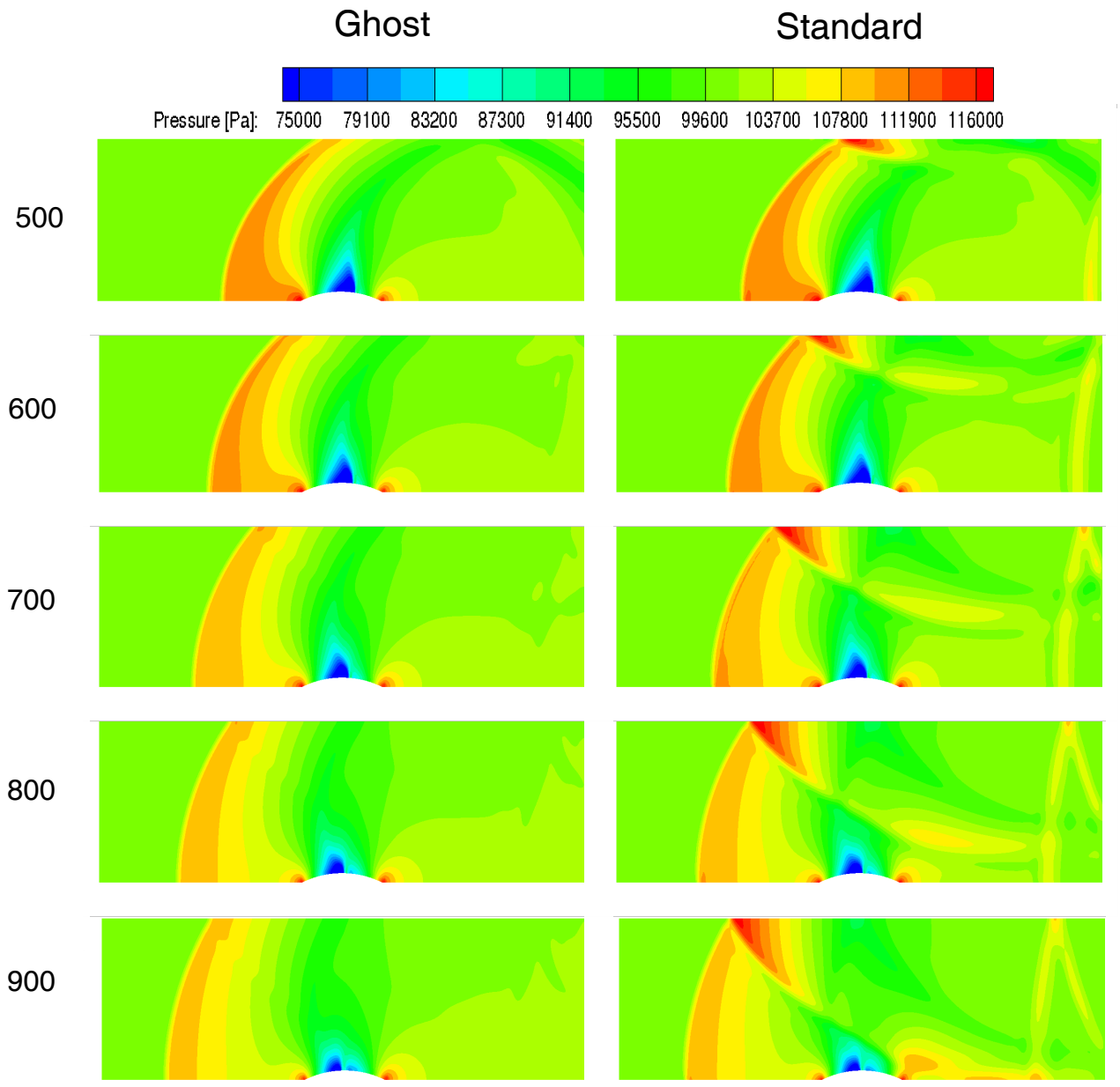


Figure 7.89: Comparison of Initial Startup for Bump-in-Channel $M = 0.6$ Case: Iterations 500-900.

At iteration 500, the waves have reached both the far-field and outlet boundary. The standard boundary has generated a downward traveling wave in addition to an upstream traveling wave, whereas the ghost boundary has allowed the wave to generally pass undisturbed. By iteration 900, the upstream traveling wave has not yet reached the inlet. The standard boundary has amplified

the upstream traveling waves, with the wave front nearly reaching the bump. Figure 7.90 shows iterations 1000-1400.

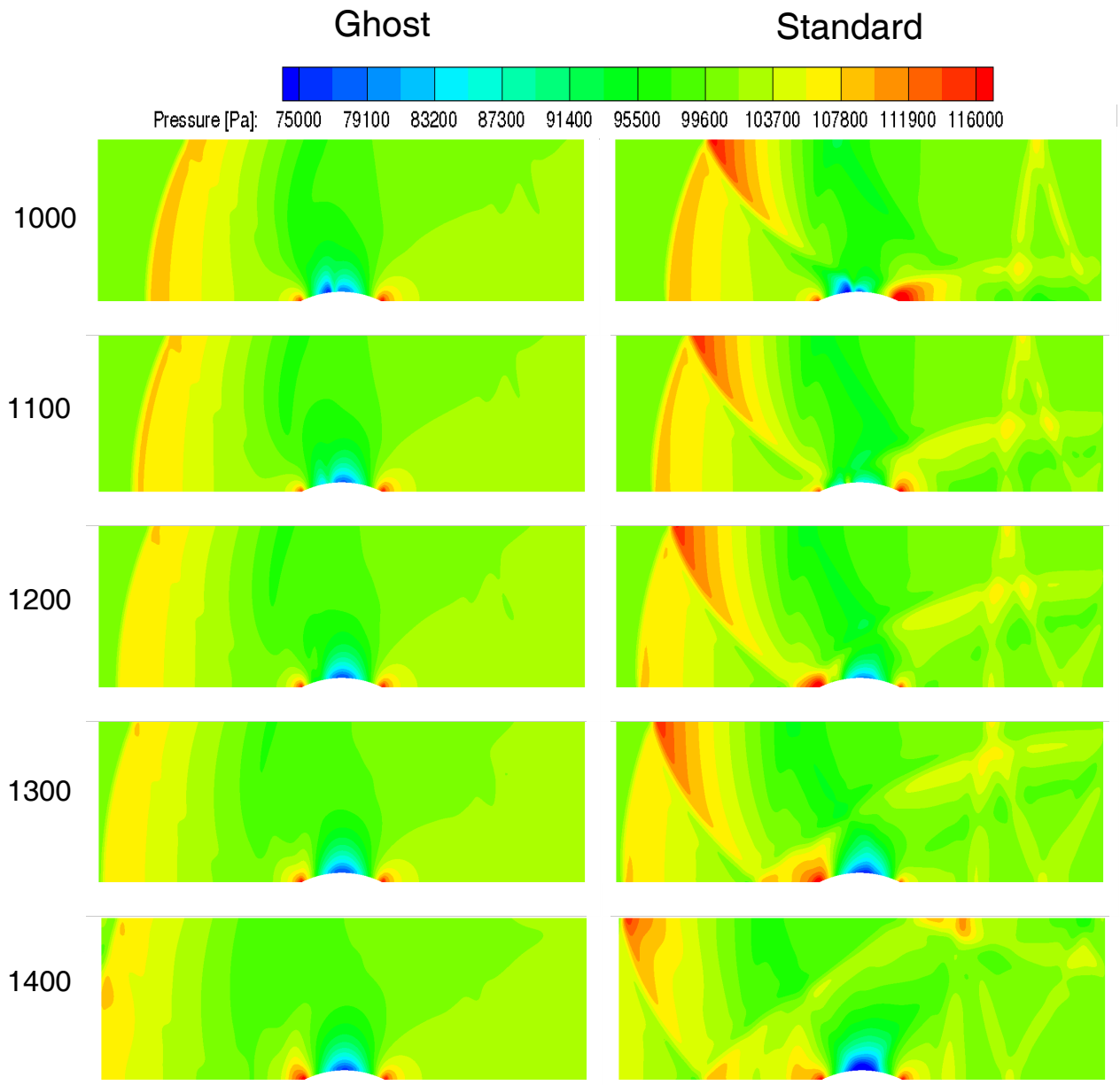


Figure 7.90: Comparison of Initial Startup for Bump-in-Channel $M = 0.6$ Case: Iterations 1000-1400.

The leading upstream traveling pressure wave finally reaches the inlet boundary at approxi-

mately iteration 1400. The ghost boundary produces some reflection at that point near the inlet, but since the inlet is not a pure non-reflecting boundary, this is expected. As for the standard boundary results, additional waves are generated and amplified as the various waves impact the boundaries and intersect one another. By iteration 1400, the upstream traveling wave generated by the far-field boundary at iteration 400 has finally reached the inlet. Figure 7.91 shows iterations 1400-1900. At iteration 1500, the ghost boundary generates an increase in pressure at the inlet/far-field interface, but that is washed out by iteration 1700 with only some information traveling downstream at the end of the period. As for the standard boundary conditions, the upstream wave begins leaving the inlet and is mostly gone by iteration 1800. However, the flow field is still very much plagued by pressure waves throughout the domain. Overall, as with the slower Mach = 0.2 flow, the ghost boundaries nearly eliminate the initially generated waves with some downstream reflection at the inlet. The standard boundaries generate significant noise, which corrupts the solution and causes convergence to take a much longer time.

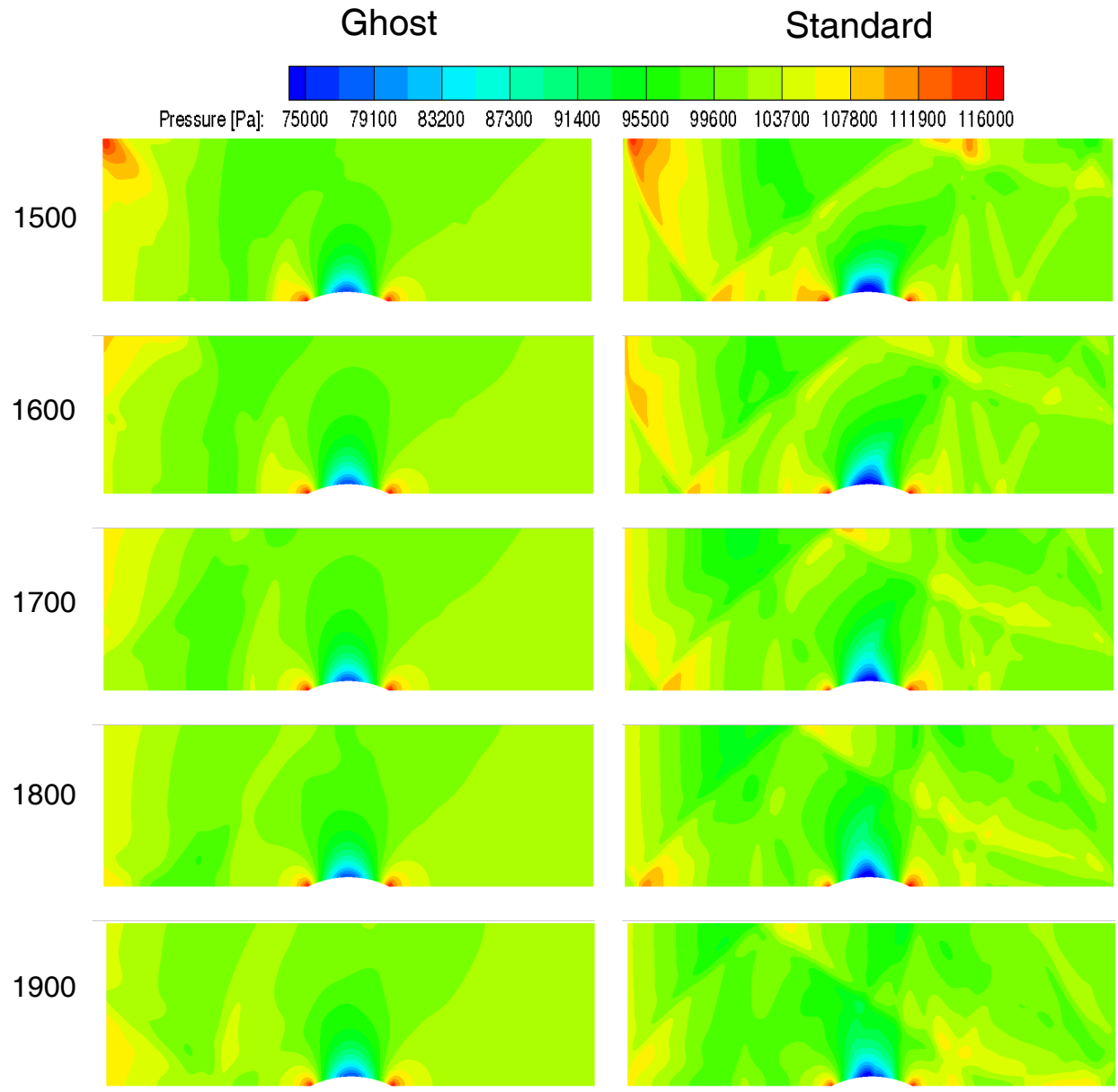


Figure 7.91: Comparison of Initial Startup for Bump-in-Channel $M = 0.6$ Case: Iterations 1400-1900.

The wall boundaries are inspected to see if the wall normal pressure gradient condition of (5.165) and (5.167) is met in Figure 7.92 for each mesh.

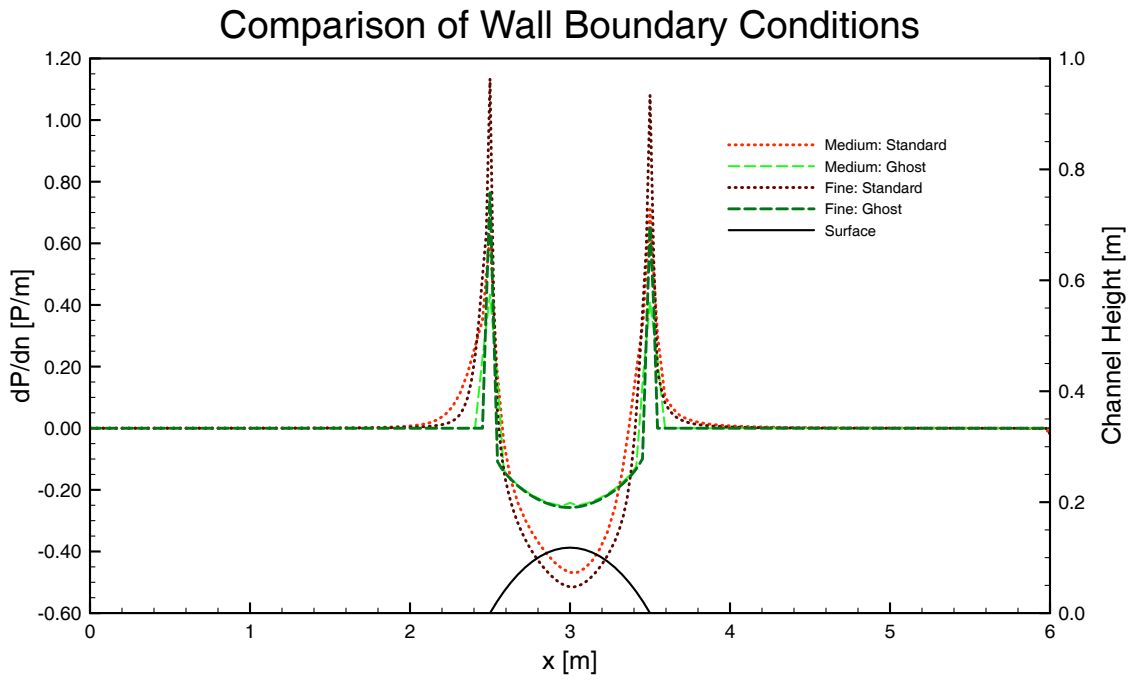


Figure 7.92: Comparison of Wall-Normal Pressure Gradient for Bump-in-Channel Case: $M = 0.6$.

As observed in Figure 7.74, the standard boundary condition has a non-zero pressure gradient on the bottom of the channel, with the ghost boundaries maintaining the prescribed conditions. Figure 7.93 shows the pressure on the back half of the domain.

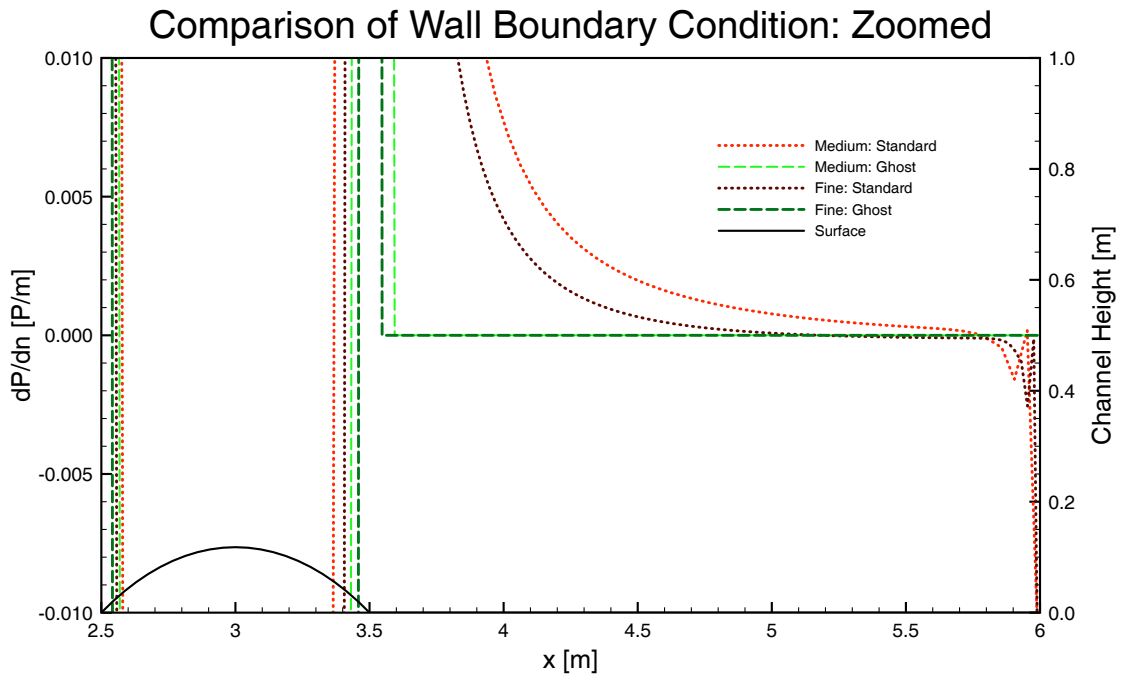


Figure 7.93: Comparison of Wall-Normal Pressure Gradient for Bump-in-Channel Case: $M = 0.6$ Trailing Edge of Domain.

The ghost node boundary has a zero-gradient boundary, since it is prescribed. The standard boundary condition has a rather large normal pressure gradient at the end of the channel. As with the Mach = 0.2 case, this error for the standard boundary condition is a result of the wall is not extended beyond the domain and that the outlet does not account for the wall, leading to a mismatch at the intersection of the boundaries. From this case, it is fairly clear the ghost node solutions is superior to the standard boundary solutions.

7.2.5 Laminar Flat Plate

This section presents results comparing the standard boundary conditions described in Chapter 4 to the ghost boundary conditions presented in Chapter 5 for adiabatic and isothermal wall boundaries for a second-order simulation. Both wall boundaries use the mesh shown in Figure 7.5. The flow parameters are given in Table 7.20.

Table 7.20: Initial Conditions for Laminar Flat Plate Case.

Quantity	Value	Units
Freestream Mach No., M_∞	0.3	-
Temperature, T_∞	288.15	K
Pressure, p_∞	101325	Pa
Isothermal Wall Temperature, T_{wall}	315	K

The flow is steady, and the $CFL = 0.8$. The Affine MLS method using $k = 0.8$ and a monomial basis is utilized for the ghost boundaries, and the LSQR method is used for the standard boundaries [91]. The grid used herein was shown in Figure 7.5, which has a $y+ \approx 1$ at the leading-edge of the plate and a cell-growth ratio of 1.2 normal to the plate. For results using the numerical methods of Chapter 4, the inlet and outlet boundary conditions are nonreflecting using Riemann invariants [12, Chapter 8], and the far-field boundary is effectively a symmetry boundary. For the results using the numerical method, including ghost nodes, of Chapter 5, the inlet, outlet, and far-field boundary conditions are NSCBC boundary conditions a described in Sections 5.4.2.3-5.4.2.6. The results for all of the methods are compared to the exact solution of Blasius for a compressible laminar flat plate [167, Chapter 7].

7.2.5.1 Adiabatic Laminar Flat Plate

This section presents the results comparing the standard and ghost boundary conditions for the adiabatic wall boundary condition. The flow fields are compared at the leading edge of the plate for the standard boundaries, ghost boundaries, and ghost boundaries using the viscous flux stencils described in Section 5.3.2 (VF = 3). The $u-$velocity is presented in Figure 7.94

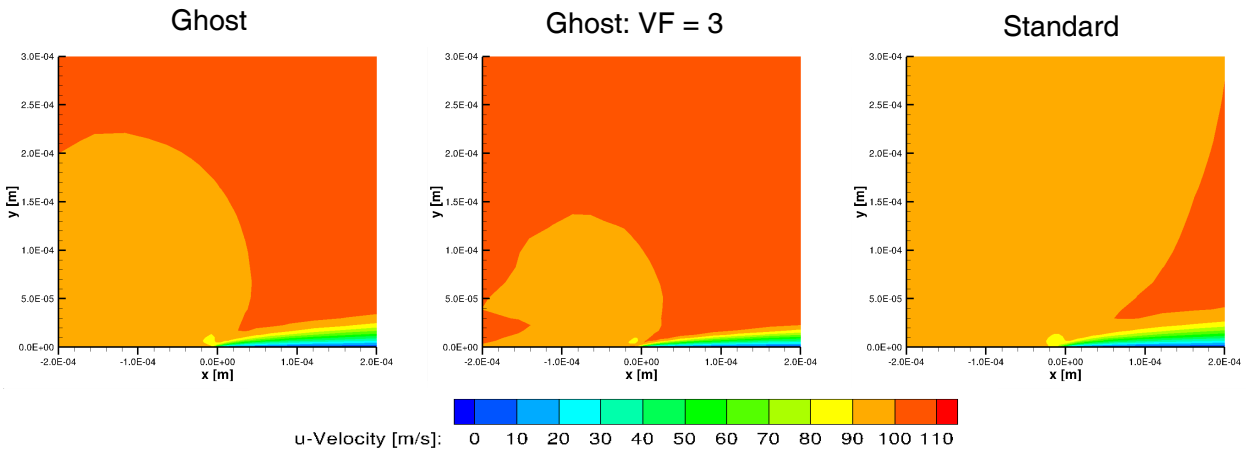


Figure 7.94: $u-$Velocity Contours at Leading-Edge of Laminar Adiabatic Flat Plate.

Both ghost boundaries have a smaller region of slower flow leading up to the plate compared to the standard boundary. Additionally, the flow field above the plate appears to reach the freestream velocity sooner than the standard boundary. The $v-$velocity is shown next in Figure 7.95

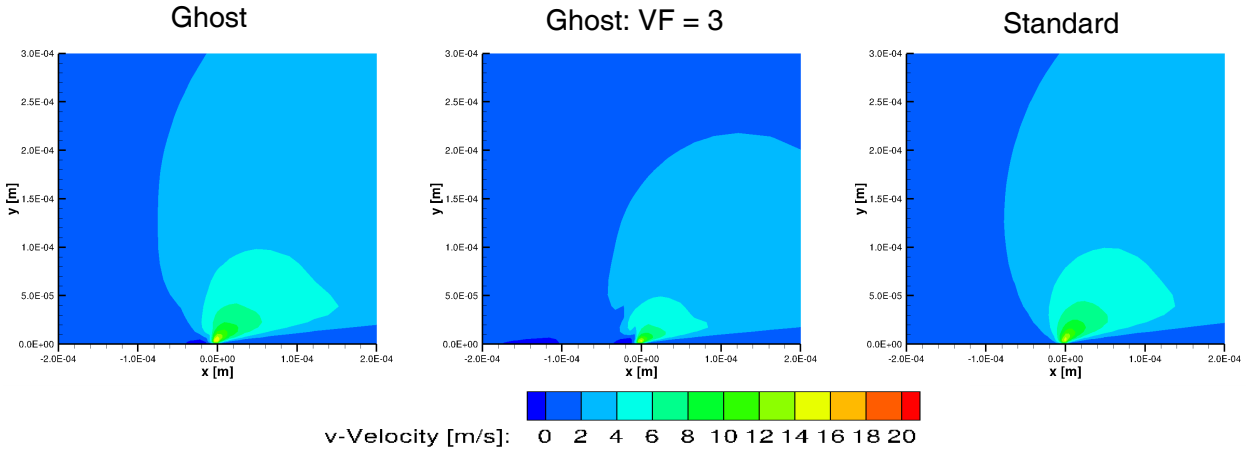


Figure 7.95: $v-$Velocity Contours at Leading-Edge of Laminar Adiabatic Flat Plate.

The wall normal velocity is similar between both the ghost the standard boundary. The ghost boundary using the viscous stencil has two noticeable features. First, the area of $v-$ velocity is

smaller than the other two cases. Additionally, there appears to be some disturbances coming from the leading edge traveling upstream for this case. This is most likely due to the symmetry/wall interface, which is an unphysical boundary. The higher-order viscous flux reconstruction is less forgiving in this situations, and when performing third- or fourth-order simulations this interface may best be avoided to circumvent numerical issues. Figure 7.96 shows the Mach contours.

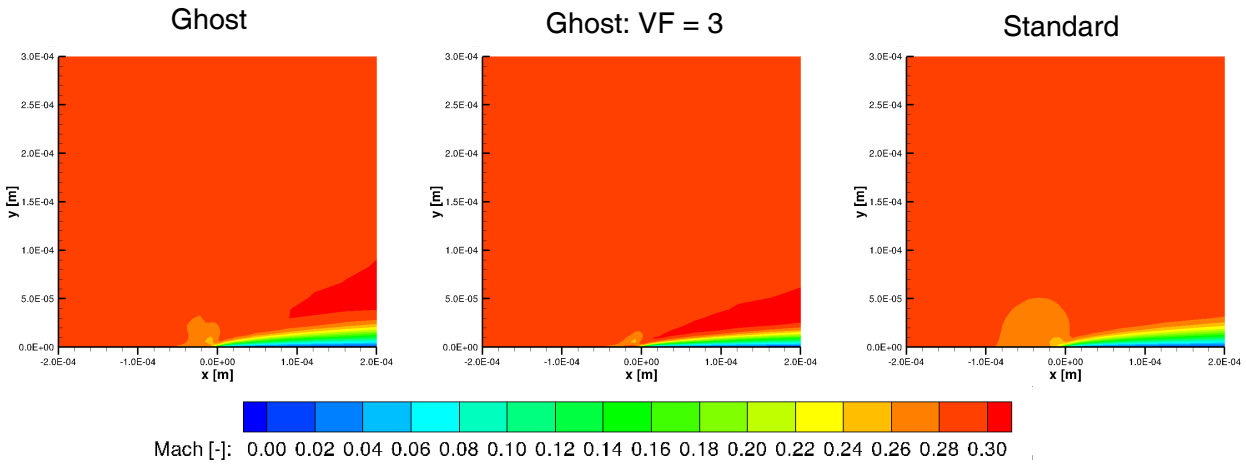


Figure 7.96: Mach Contours at Leading-Edge of Laminar Adiabatic Flat Plate.

Both ghost boundaries slow the flow down less before reaching the plate compared to the standard boundary, with the VF = 3 ghost boundary results having the smallest slow down. Figure 7.97 compares the density contours.

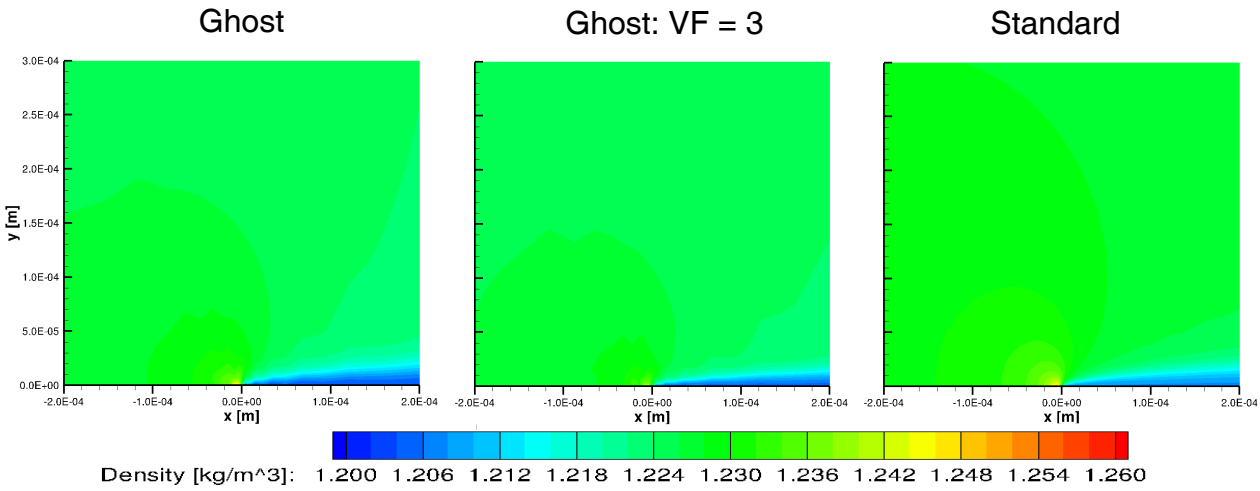


Figure 7.97: Density Contours at Leading-Edge of Laminar Adiabatic Flat Plate.

The density contours between both ghost methods are similar, with the standard boundary again showing a larger impact on the flow at the leading edge. Figure 7.98 compares the temperatures at the leading edge.

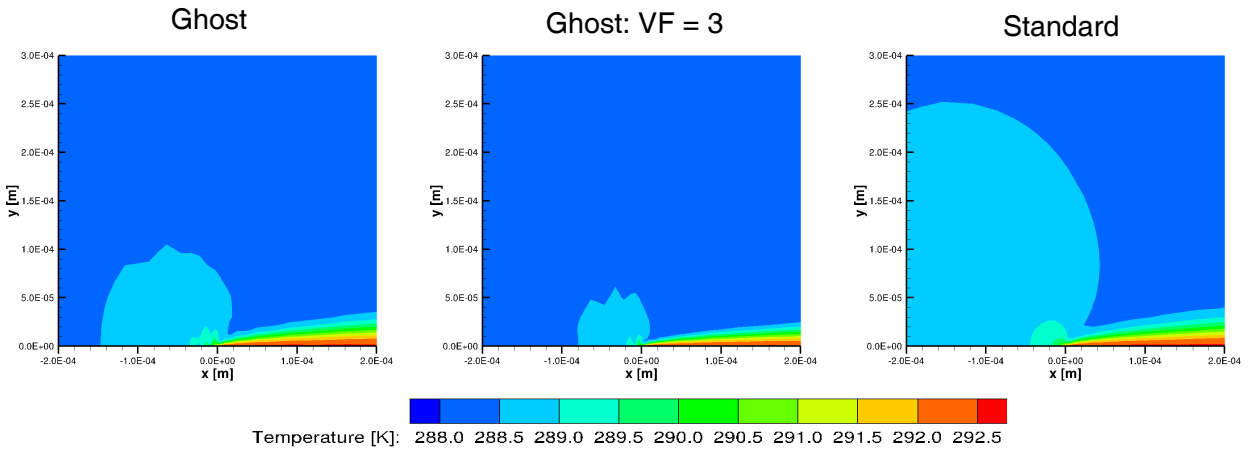


Figure 7.98: Temperature Contours at Leading-Edge of Laminar Adiabatic Flat Plate.

As with the density, the temperature field is near the leading edge is heated less with the ghost

boundary results when compared to the standard boundary. Some instability near the leading edge (non-smooth contour) is also present with the ghost results, but as stated earlier, this is probably due to the unphysical interaction of the symmetry and wall boundaries. Figure 7.99 presents the pressure contours.

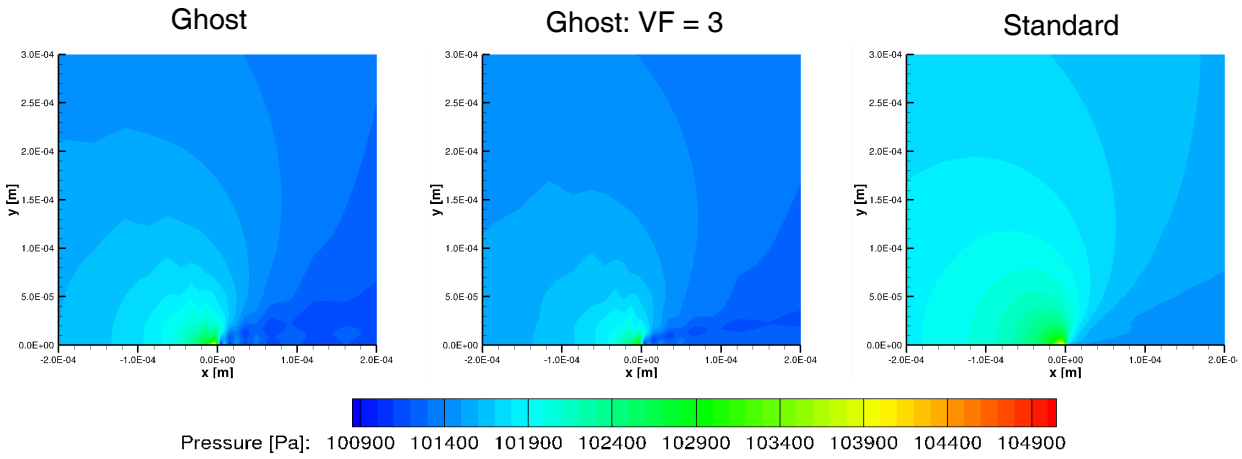


Figure 7.99: Pressure Contours at Leading-Edge of Laminar Adiabatic Flat Plate.

The instability caused by the unphysical symmetry/wall boundary interface is most apparent in the pressure contours of the ghost boundaries. It would appear that the VF = 3 ghost results have a smaller instability. The residuals should show the instability of the flow. Figure 7.100 shows the residuals for each of the methods.

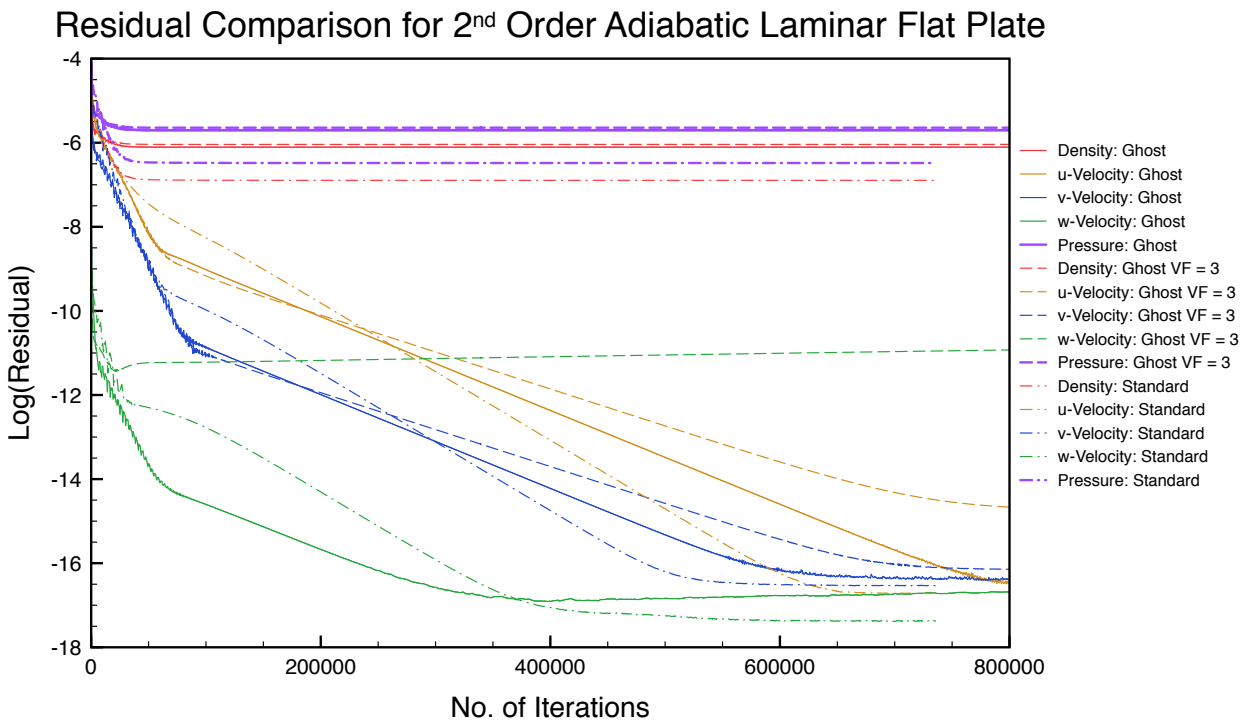


Figure 7.100: Residuals for Laminar Adiabatic Flat Plate.

Rather interestingly, the methods all tend to converge at roughly the same rate. Some issues, though, for the VF = 3 ghost results do appear, as the $w-$ velocity stalls at only -11, which is probably indicative of the instability. Figure 7.101 compares the axial boundary layer velocities for each of the methods at x =2.39E-3 m

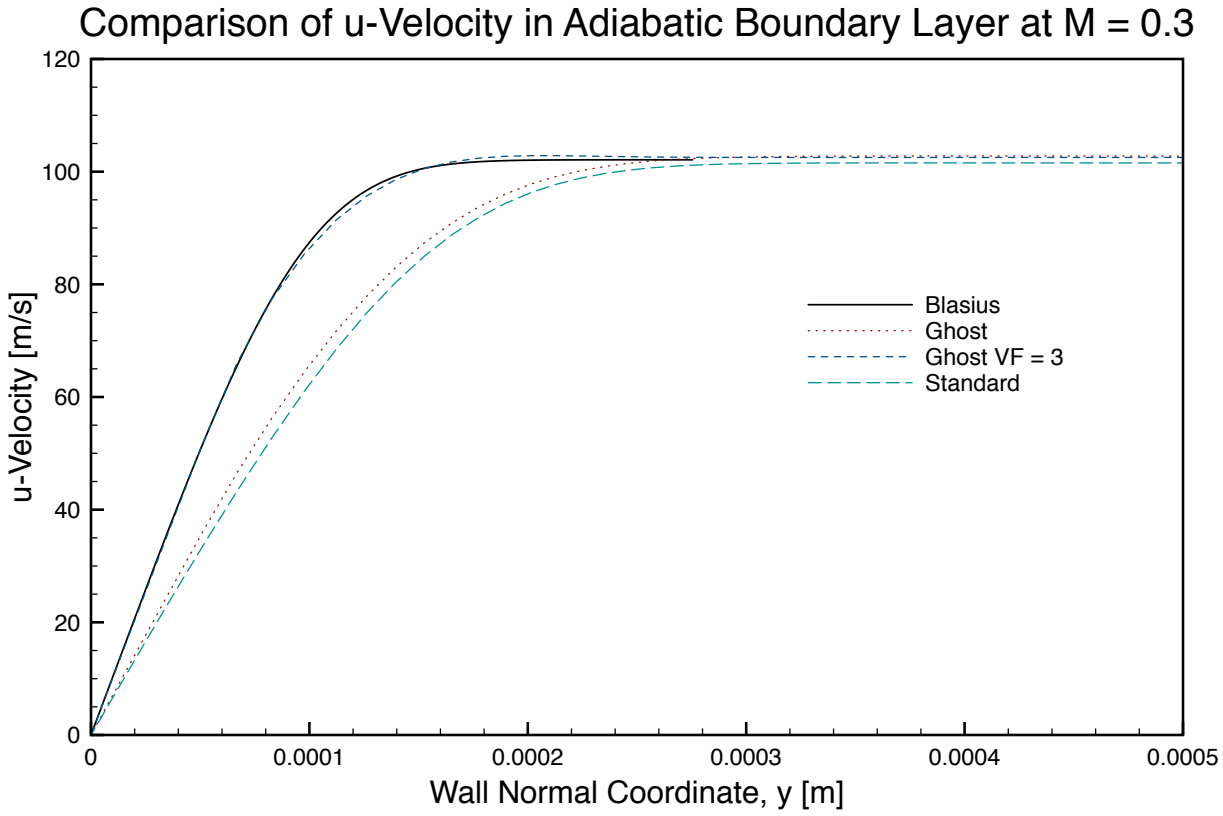


Figure 7.101: Comparison of $u-$Velocity Boundary Layer Profile for the Laminar Adiabatic Flat Plate.

First, the standard boundary least matches Blasius profile and fails to reach the predicted maximum velocity. The ghost boundary condition using the original viscous flux has a similar profile to that of the standard boundary conditions, but does reach the maximum velocity predicted by Blasius. The VF = 3 ghost boundary conditions match almost exactly the Blasius profile. This leads the reader to draw the following conclusion: how the boundary is defined seems to have less of an impact than the viscous flux reconstruction strategy. The obvious reason for this is that the standard reconstruction method is less accurate, but it may also be more dissipative. This is hard to prove here, but can be shown later when solving the TGV cases. The $v-$velocity profiles are shown next in Figure 7.102.

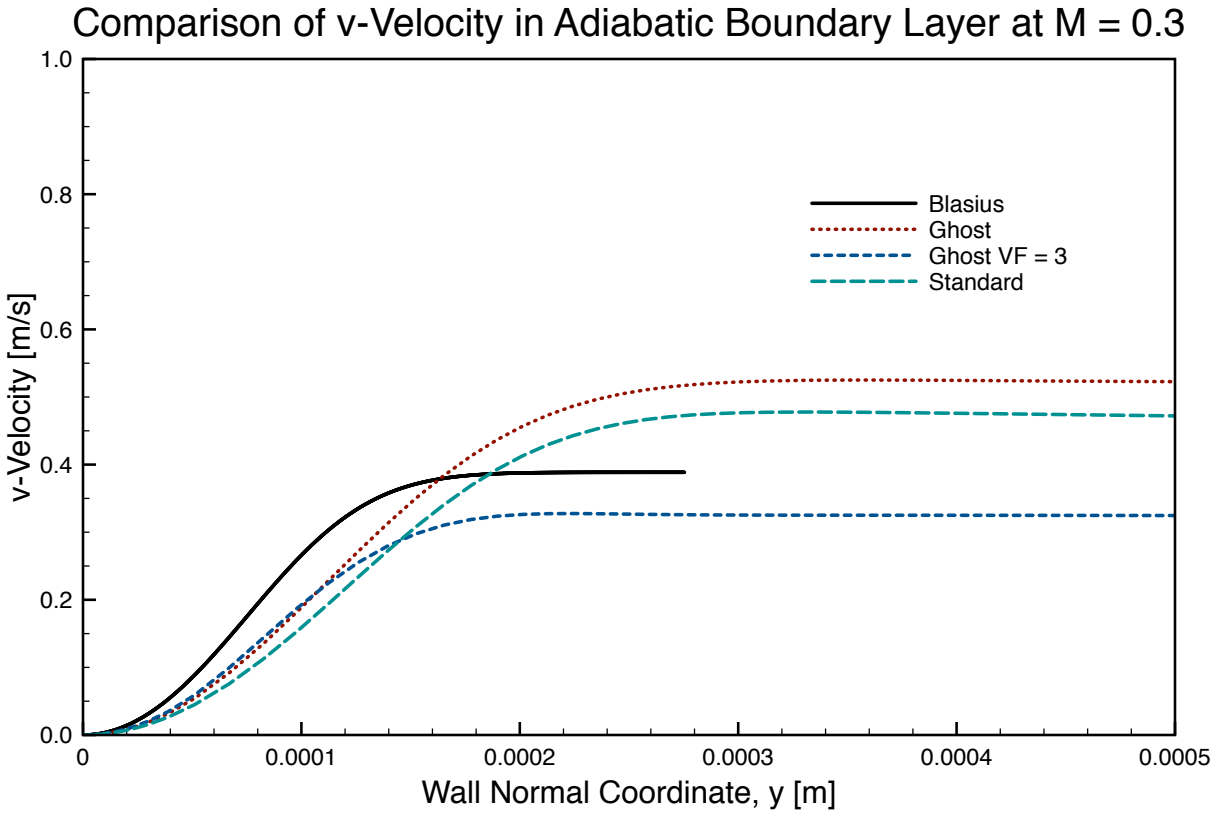


Figure 7.102: Comparison of $v-$Velocity Boundary Layer Profile for the Laminar Adiabatic Flat Plate.

All of the methods do not accurately predict the wall normal velocity in the boundary layer. Both the standard and ghost boundary over-predict the velocity, while the VF = 3 ghost boundary results under-predict the velocity. This would point to a possible need to further study the enforcement of wall-normal velocity. However, since the $v-$ velocity is orders of magnitude smaller than the $u-$velocity so the error should not have a sizable impact on the overall solution. Finally, the thermal profiles are shown in Figure 7.103.

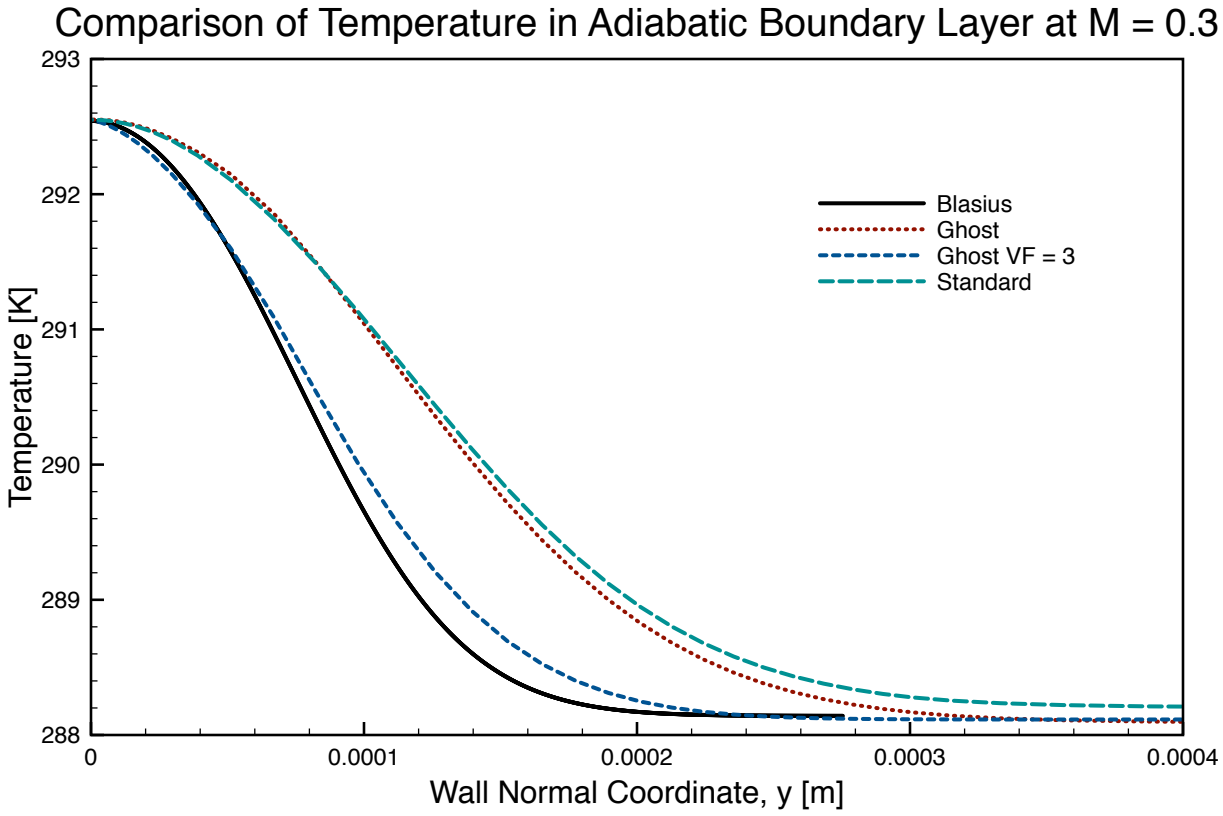


Figure 7.103: Comparison of Thermal Boundary Layer Profile for the Laminar Adiabatic Flat Plate.

As with the $u-$velocity profile, the ghost boundaries do a better job at modeling the thermal profile in the boundary layer. Both ghost methods reach the freestream temperature, while the standard boundary does not. The VF = 3 ghost boundary results actually does a respectable job of following the Blasius solution. If the instability could be removed, that is the inlet is given a profile and the symmetry boundary is removed, the VF = 3 ghost results would probably do even better at modeling the thermal profile. The last thing to check is the wall-normal temperature gradient, $\frac{dT}{d\mathbf{n}}$. Recall that for an adiabatic wall boundary, $\frac{dT}{d\mathbf{n}}$ should be zero. Figure 7.104 compares the $\frac{dT}{d\mathbf{n}}$ for each boundary method.

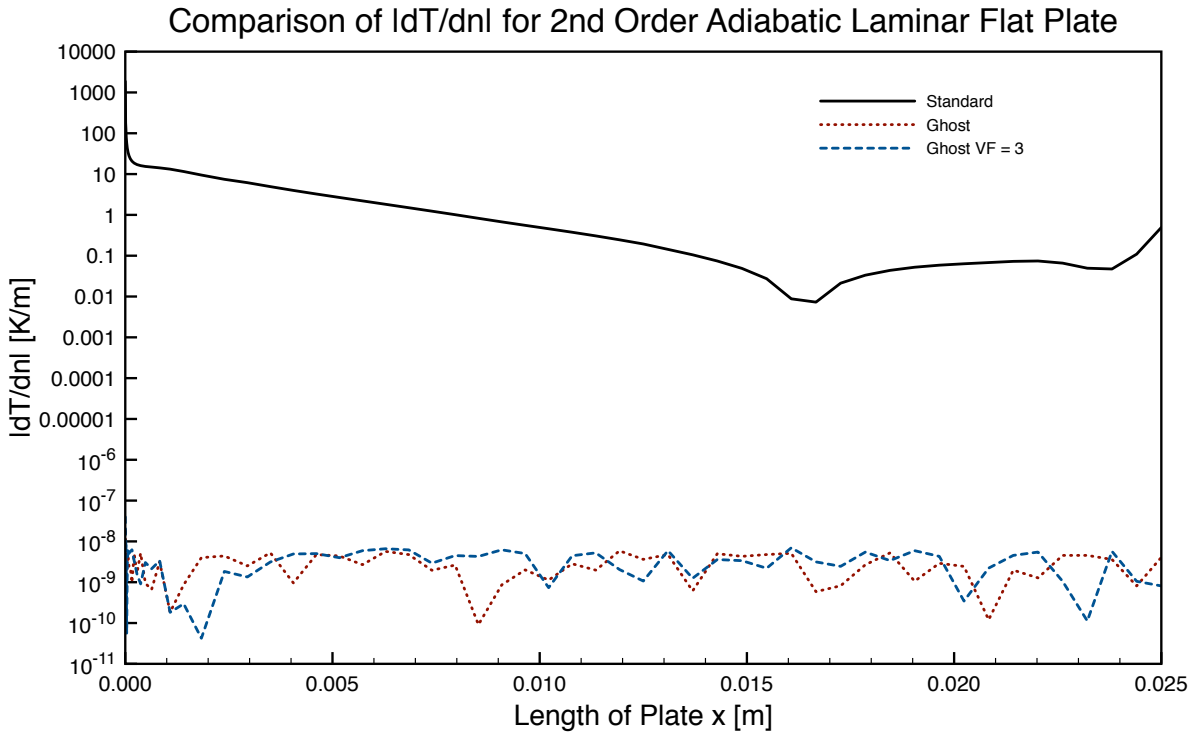


Figure 7.104: Comparison of Wall-Normal Temperature Gradient $\frac{dT}{d\mathbf{n}}$ for the Laminar Adiabatic Flat Plate Case.

The ghost boundary methods have temperature gradients on the order of 10^{-9}, so still not quite zero but reasonably close to it. The standard boundary condition, though, at best has a minimum gradient on the order of 0.01. In fact, one could argue that the standard boundary condition does not model an adiabatic wall. It may produce results that in some cases are similar to an adiabatic wall, but as shown in Figure 7.104, it is not, so any convergence advantage it may have over the ghost methods is moot. Overall, two things should be apparent from these results. First, the ghost boundaries are true adiabatic conditions. Second, the higher-order viscous flux does a much better job of modeling the near-wall flow than the viscous flux presented in Section 4.2.2.

7.2.5.2 Isothermal Laminar Flat Plate

This section presents the results comparing the standard and ghost boundary conditions for the isothermal wall boundary condition with a wall temperature of $T_{\text{wall}} = 315\text{K}$. The flow fields are

compared at the leading edge of the plate for the standard boundaries, ghost boundaries, and ghost boundaries using the viscous flux stencils described in Section 5.3.2 (VF = 3). The $u-$velocity is presented in Figure 7.105

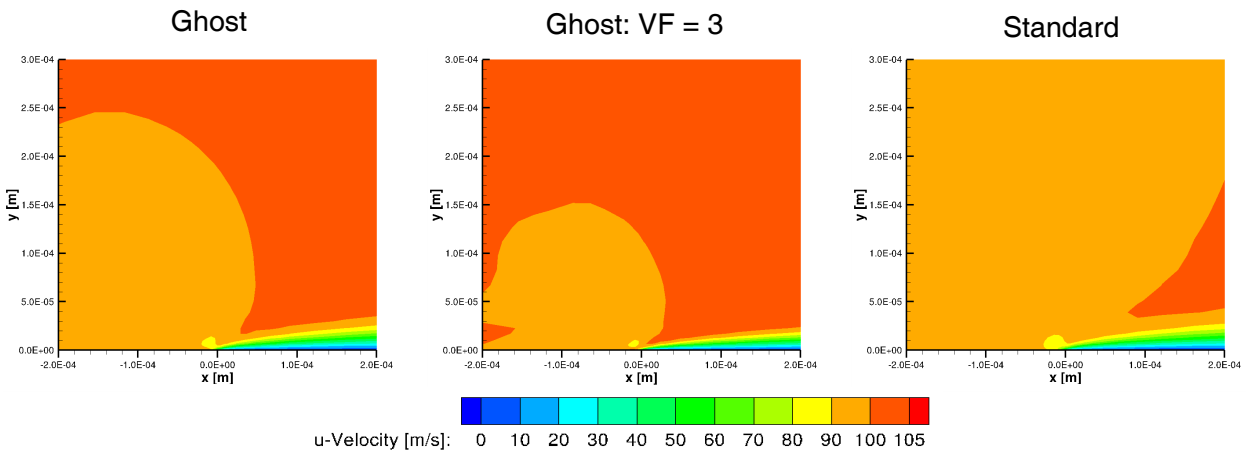


Figure 7.105: $u-$Velocity Contours at Leading-Edge of Laminar Isothermal Flat Plate.

Both ghost boundaries have a smaller region of slower flow leading up to the plate compared to the standard boundary. Additionally, the flow field above the plate appears to reach the freestream velocity sooner than the standard boundary. The $v-$velocity is shown next in Figure 7.106

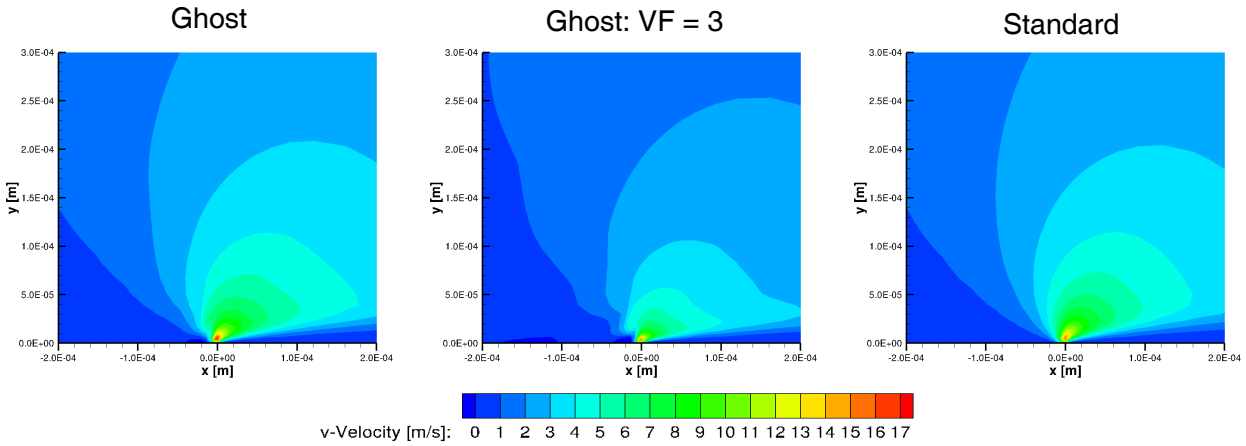


Figure 7.106: $v-$Velocity Contours at Leading-Edge of Laminar Isothermal Flat Plate.

The wall normal velocity is similar between both the ghost the standard boundary. The ghost boundary using the viscous stencil has two noticeable features. First, the area of $v-$ velocity is smaller than the other two cases. Additionally, there appears to be some disturbances coming from the leading edge traveling upstream for this case. This is most likely due to the symmetry/wall interface, which is an unphysical boundary. The higher-order viscous flux reconstruction is less forgiving in this situations, and when performing third- or fourth-order simulations this interface may best be avoided to circumvent numerical issues. Figure 7.107 shows the Mach contours.

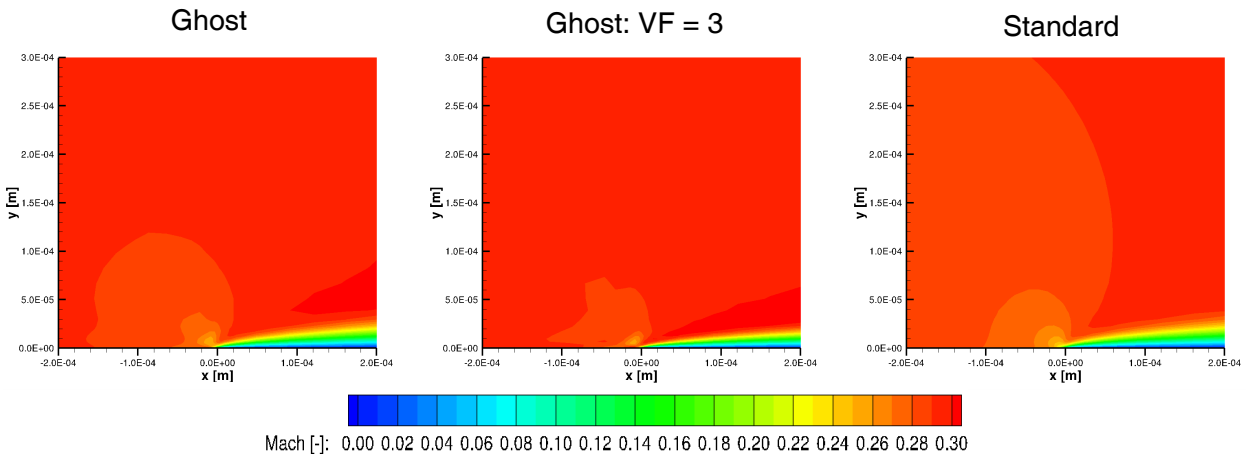


Figure 7.107: Mach Contours at Leading-Edge of Laminar Isothermal Flat Plate.

Both ghost boundaries slow the flow down less before reaching the plate compared to the standard boundary, with the VF = 3 ghost boundary results having the smallest slow down. Figure 7.108 compares the density contours.

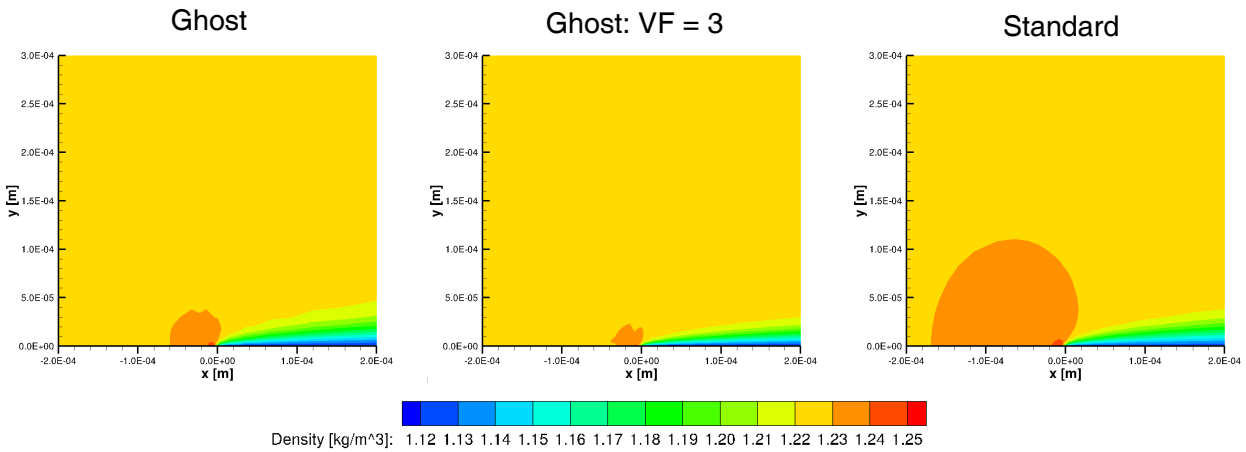


Figure 7.108: Density Contours at Leading-Edge of Laminar Isothermal Flat Plate.

The density contours between both ghost methods are similar, with the standard boundary again showing a larger impact on the flow at the leading edge. Figure 7.109 compares the temperatures at the leading edge.

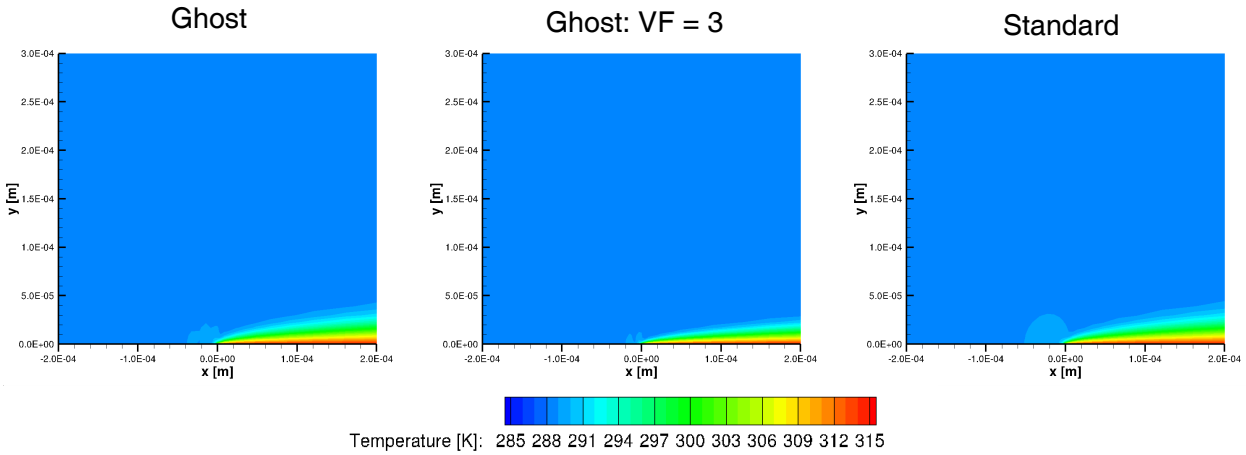


Figure 7.109: Temperature Contours at Leading-Edge of Laminar Isothermal Flat Plate.

As with the density, the temperature field is near the leading edge is heated less with the ghost boundary results when compared to the standard boundary. Some instability near the leading edge

(non-smooth contour) is also present with the ghost results, but as stated earlier, this is probably due to the unphysical interaction of the symmetry and wall boundaries. Figure 7.110 presents the pressure contours.

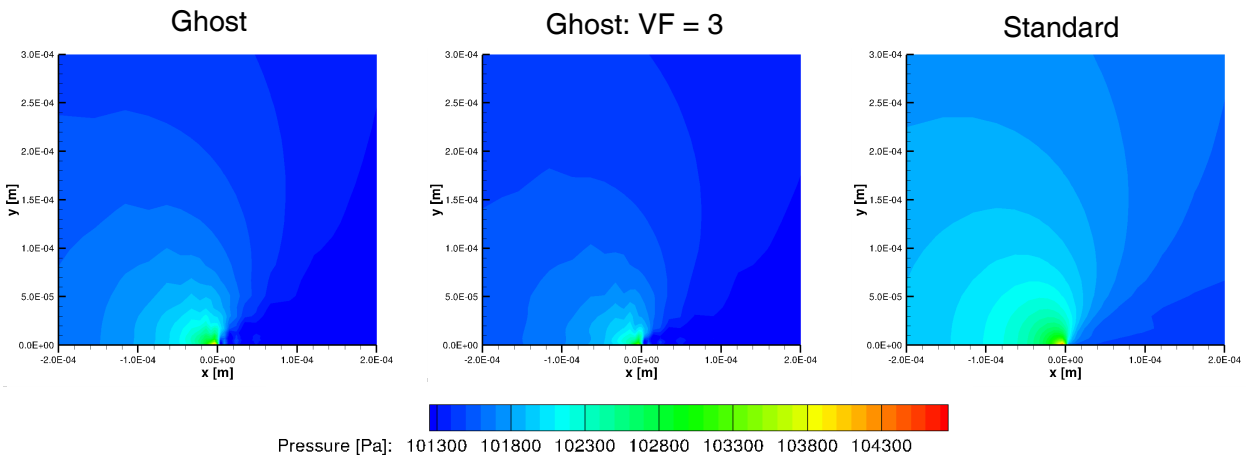


Figure 7.110: Pressure Contours at Leading-Edge of Laminar Isothermal Flat Plate.

Unlike the adiabatic results shown in Figure 7.99, there does not appear to be a noticeable disturbance in the pressure field just above the wall for the ghost boundary results. That would point to the isothermal boundaries being more stable than the adiabatic boundary conditions. This follows from the fact that the wall temperature is directly specified. Figure 7.111 shows the residuals for each of the methods.

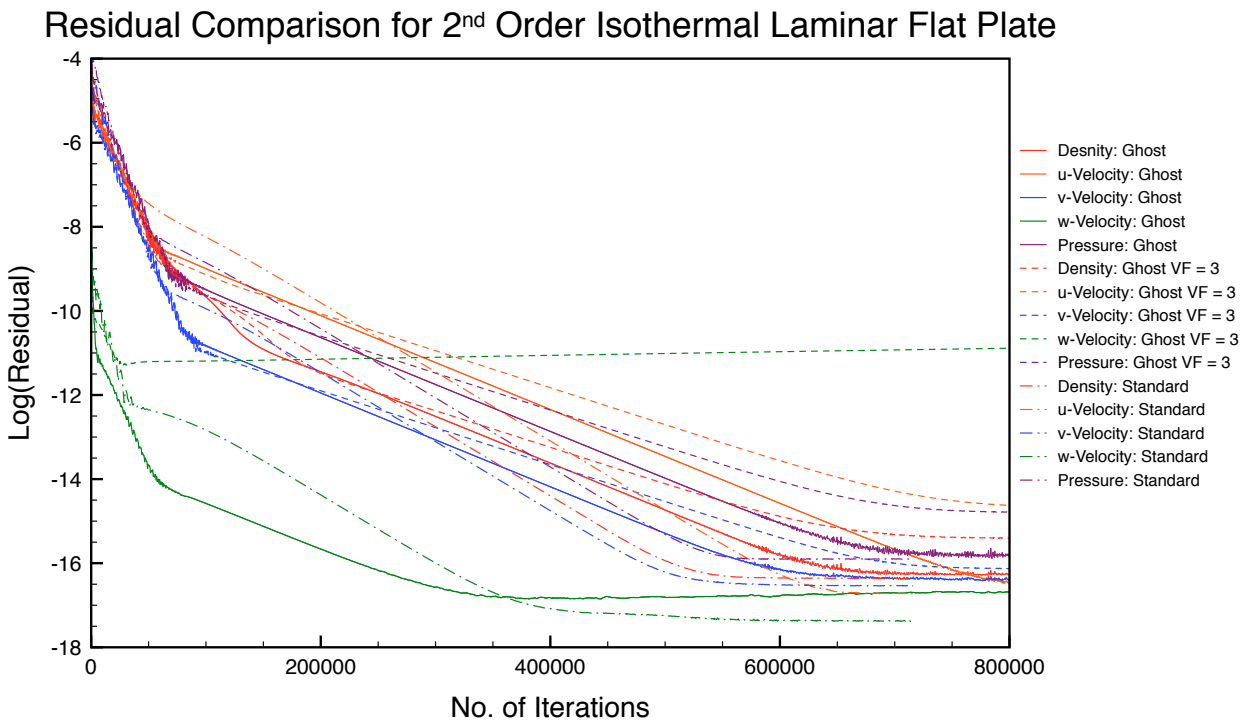


Figure 7.111: Residuals for Laminar Isothermal Flat Plate.

Rather interestingly, the methods all tend to converge at roughly the same rate. As with the adiabatic case, shown in Figure 7.100, $w-$ velocity stalls at only -11 for the VF = 3 ghost boundary results. This is again indicative of an underlying instability. Figure 7.112 compares the axial boundary layer velocities for each of the methods at $x =$ 2.39E-3 m. .

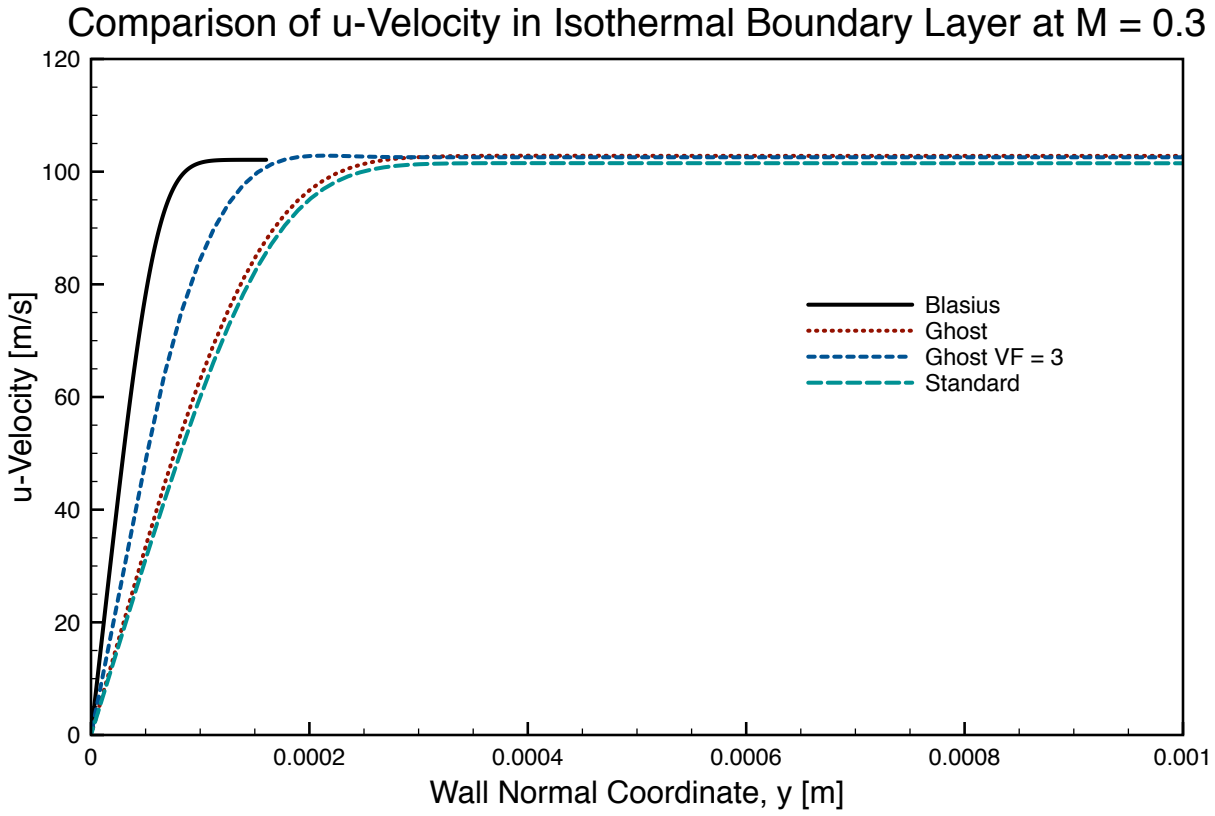


Figure 7.112: Comparison of $u-$Velocity Boundary Layer Profile for the Laminar Isothermal Flat Plate.

First, the standard boundary performs the worst of the three methods. However, unlike the adiabatic case, the VF = 3 ghost boundary profile does not match the Blasius solution. It is the best of the three methods, however. The ghost boundary condition using the original viscous flux has a similar profile to that of the standard boundary conditions, but does reach the maximum velocity predicted by Blasius. The $v-$velocity profiles are shown next in Figure 7.113.

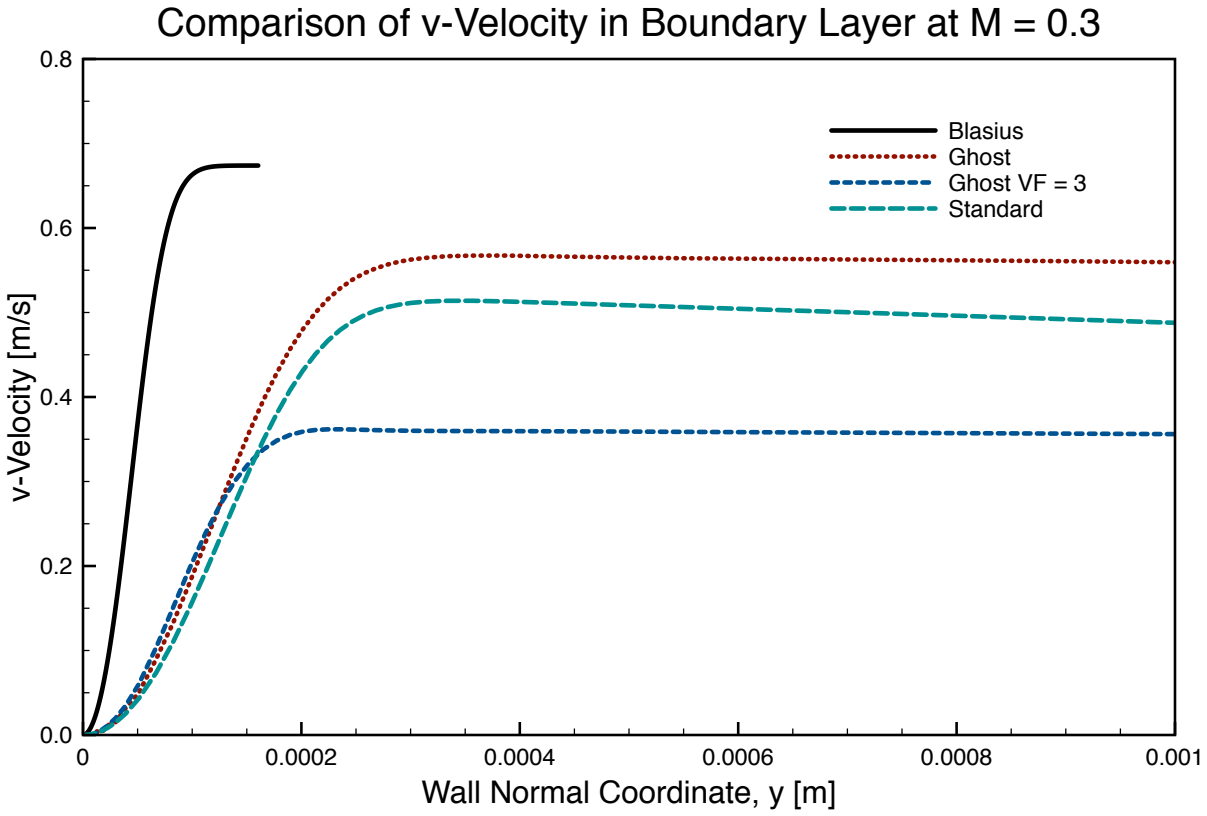


Figure 7.113: Comparison of $v-$Velocity Boundary Layer Profile for the Laminar Isothermal Flat Plate.

All of the methods do not accurately predict the wall normal velocity in the boundary layer. All of the methods under-predict the velocity. Also worth noting, the standard boundary condition shows a decreasing wall-normal velocity soon after reaching the maximum value, which is incorrect. Overall, the $v-$ velocity is orders of magnitude smaller than the $u-$velocity so the error shown here should not have a sizable impact on the overall solution. Finally, the thermal profiles are shown in Figure 7.114.

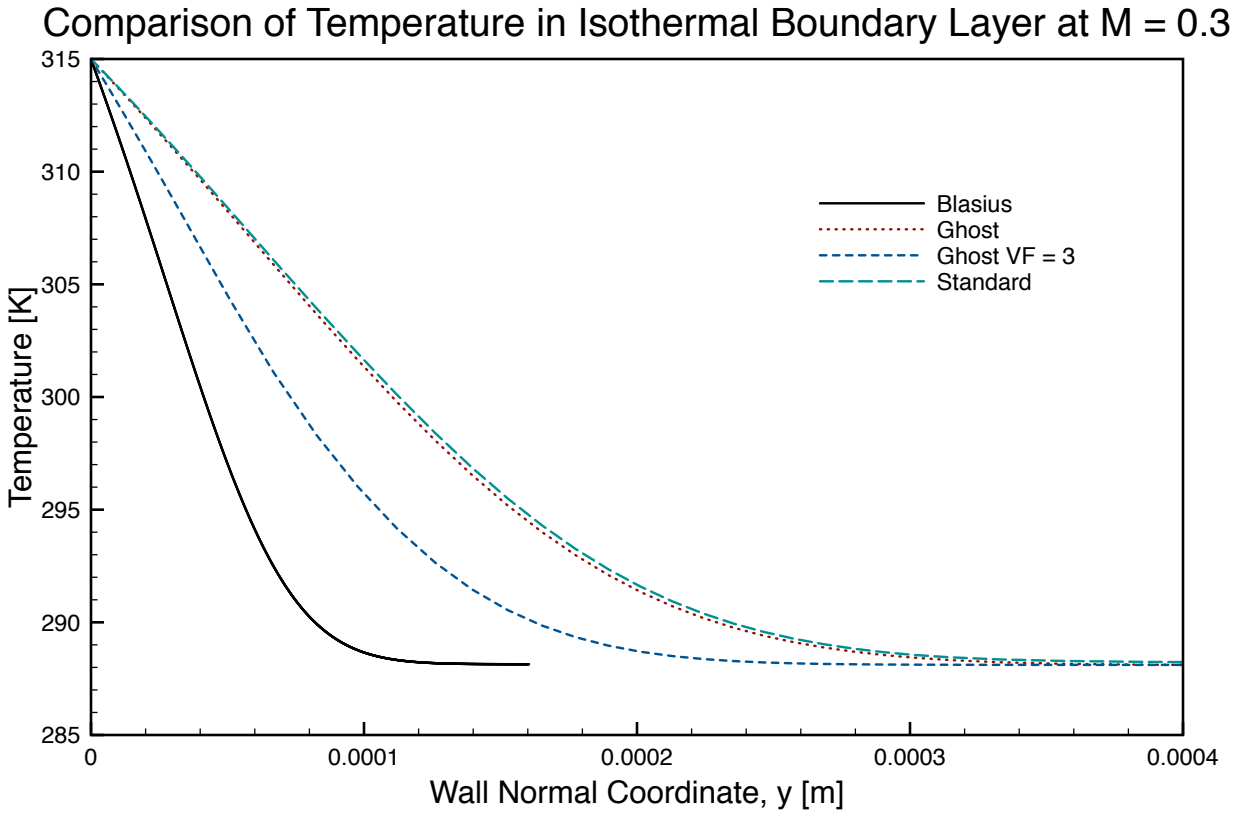


Figure 7.114: Comparison of Thermal Boundary Layer Profile for the Laminar Isothermal Flat Plate.

As with the $u-$velocity profile, the ghost boundaries do a better job at modeling the thermal profile in the boundary layer. Both ghost methods reach the freestream temperature, while the standard boundary comes just short. The VF = 3 ghost boundary results produces the best profile compared to the Blasius solution. This may point to some additional work being required for the isothermal wall, but it is difficult to determine the wall normal thermal gradient *a priori*. The last thing to check and compare is the wall-normal temperature gradient, $\frac{dT}{d\mathbf{n}}$. Figure 7.115 compares the $\frac{dT}{d\mathbf{n}}$ for each boundary method.

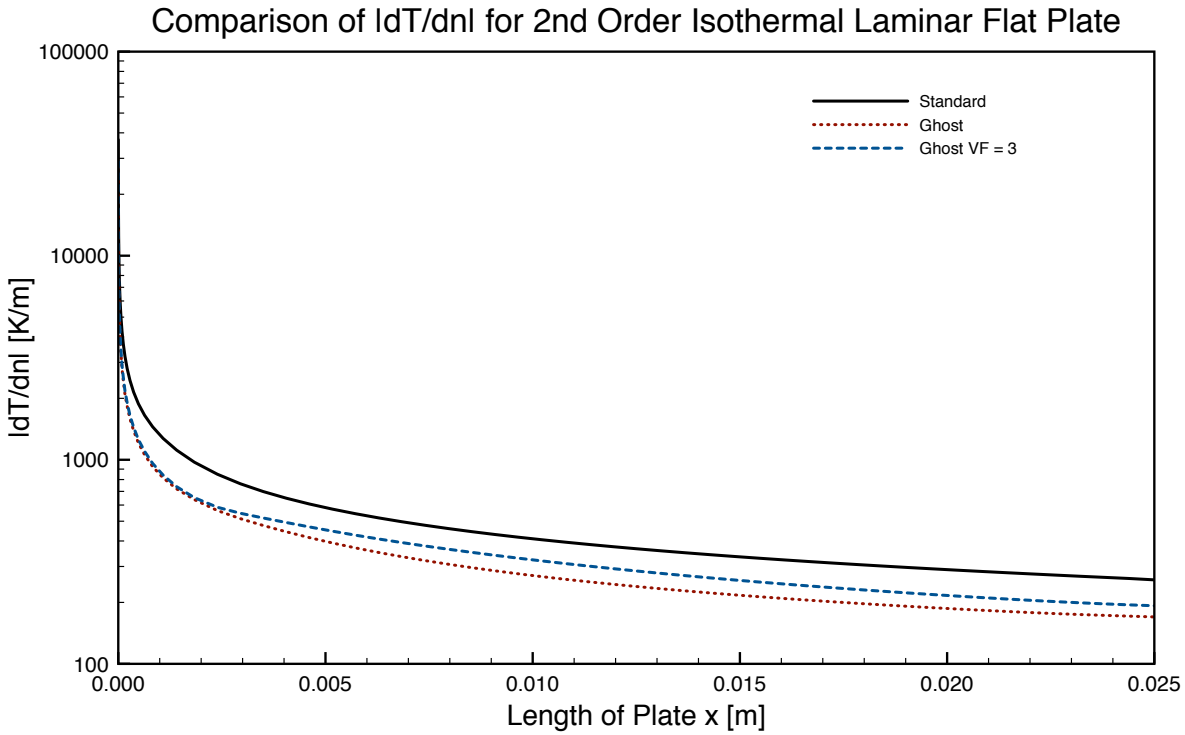


Figure 7.115: Comparison of Wall-Normal Temperature Gradient $\frac{dT}{d\mathbf{n}}$ for the Laminar Isothermal Flat Plate Case.

All three methods have similar wall-normal temperature gradients. The standard method has the largest thermal gradient, while the ghost boundary methods slightly lower values. Both ghost boundary results have the same thermal gradient near the leading edge of the plate, but diverge from one another after about $x = 0.002$m. For isothermal walls, the ghost boundaries produce overall better results than the standard boundaries, and using the higher-order viscous flux does a much better job of modeling the near-wall flow than the viscous flux presented in Section 4.2.2, as was the case for the adiabatic laminar flat plate.

7.3 Taylor–Green Vortex

This section presents the benchmarking results of the Implicit Large-Eddy Simulation model. The classical Taylor–Green Vortex (TGV) problem described by the 4^{th} *International Workshop on High-Order CFD Methods* was used to benchmark to determine the performance characteristics

of the ILES model [20]. The TGV flow field is described as:

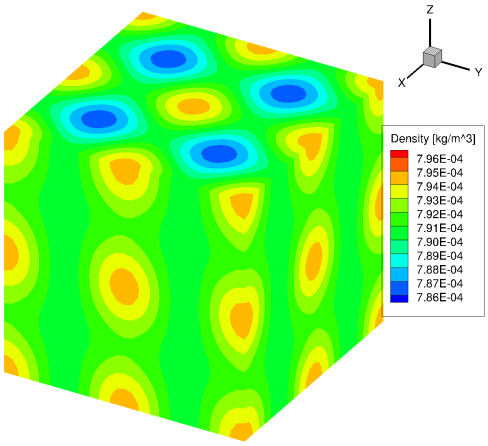
$$u = V_0 \sin\frac{x}{L}\cos\frac{y}{L}\cos\frac{z}{L} \tag{7.12}$$

$$v = -V_0 \cos\frac{x}{L}\sin\frac{y}{L}\cos\frac{z}{L} \tag{7.13}$$

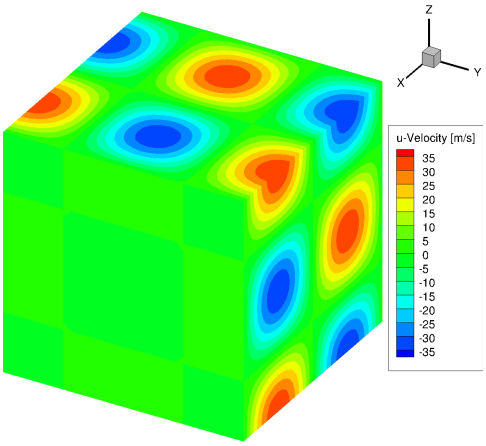
$$w = 0 \tag{7.14}$$

$$p = p_0 + \frac{\rho_0 V_0^2}{16}\left(\left(\cos\frac{2x}{L} + \cos\frac{2y}{L}\right)\left(\cos\frac{2z}{L} + 2\right)\right) \tag{7.15}$$

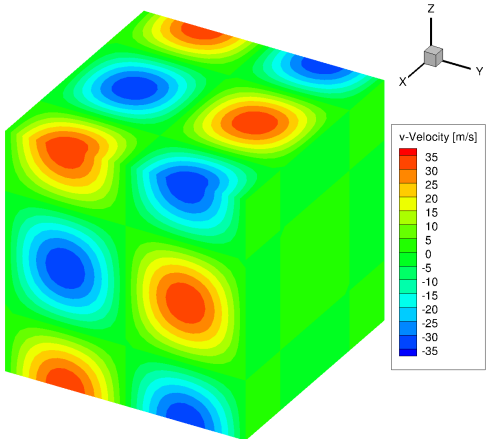
where V_0, p_0, and ρ_0 are the initial reference states and L is the length of the domain. The domain is a periodic square box defined as $-\pi L \leq x, y, z \leq \pi L$, where $L = 1$. The Reynolds number for the TGV is $Re = \frac{\rho_0 V_0 L}{\mu} = 1600$. The Mach number is $M_0 = \frac{V_0}{c_0} = 0.10$, with a sound speed c_0 determined from the initial uniform temperature field $T_0 = 300$ K. The dynamic viscosity is $\mu = 1.716 \times 10^{-5} \frac{\text{kg}}{\text{m s}}$. The results presented herein where generated on meshes shown in Figure 7.1 using a $CFL = 0.25$. The initial flow field is shown in Figure 7.116.



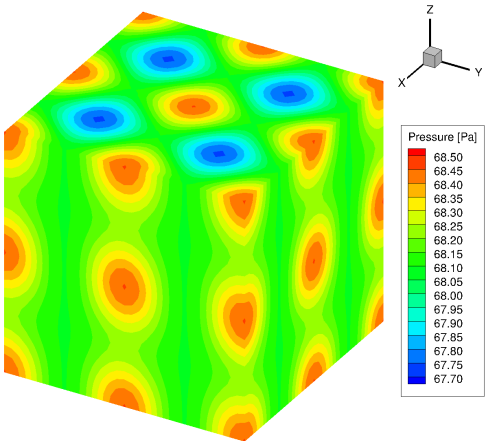
(a) ρ



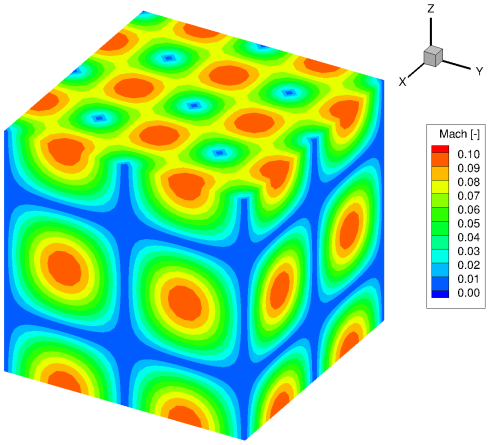
(b) u



(c) v



(d) p



(e) M

Figure 7.116: Initial Flow Field for the Taylor–Green Vortex Case.

The results are presented for second- through fourth-order comparing the turbulent kinetic energy:

$$k = \frac{1}{\rho_0 \Omega} \int_{\Omega} \rho \frac{\mathbf{v} \cdot \mathbf{v}}{2} \, d\Omega \tag{7.16}$$

computed for each case to full resolution spectral results [20]. Figure 7.117 shows the second-order results.

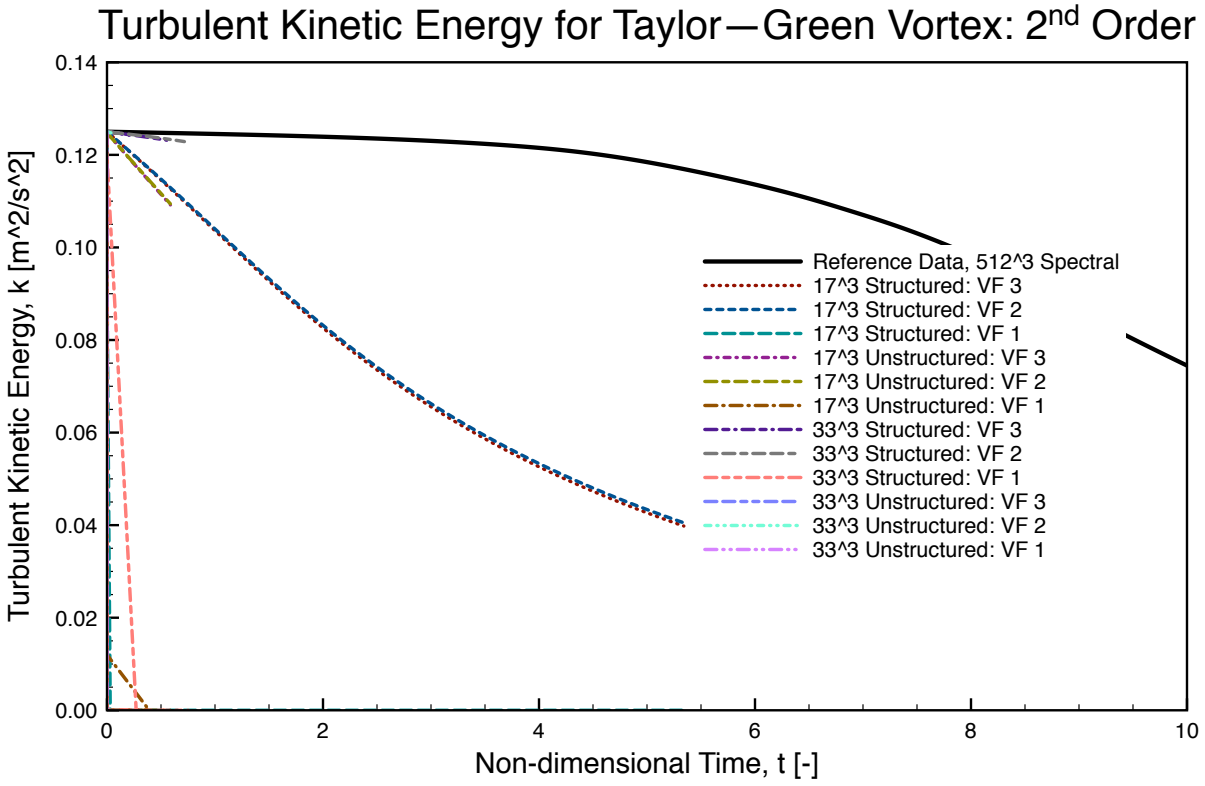


Figure 7.117: Comparison of the Turbulent Kinetic Energy for Coarse Meshes: 2^{nd}-Order.

Even with the limited data available, some important conclusions can be drawn. First, it is readily apparent that the baseline viscous flux reconstruction described in Section 4.2.2 is extremely dissipative. In fact, the vortexes have fully dissipated within a few thousand iterations (or less than $t^* = 0.25$ for all of the meshes. This is more support for the dissipation hypothesis discussed Section 7.2.5.1. As one would also expect, the finer case has less dissipation. Additionally, the

unstructured mesh has more dissipation than the structured case, which is also to be expected. Finally, both of the higher-order viscous flux reconstructions produce nearly the same level of dissipation, which is good since the full reconstruction (VF = 3) is more expensive to compute than the extrapolation method (VF = 2). Figure 7.118 shows the third-order results.

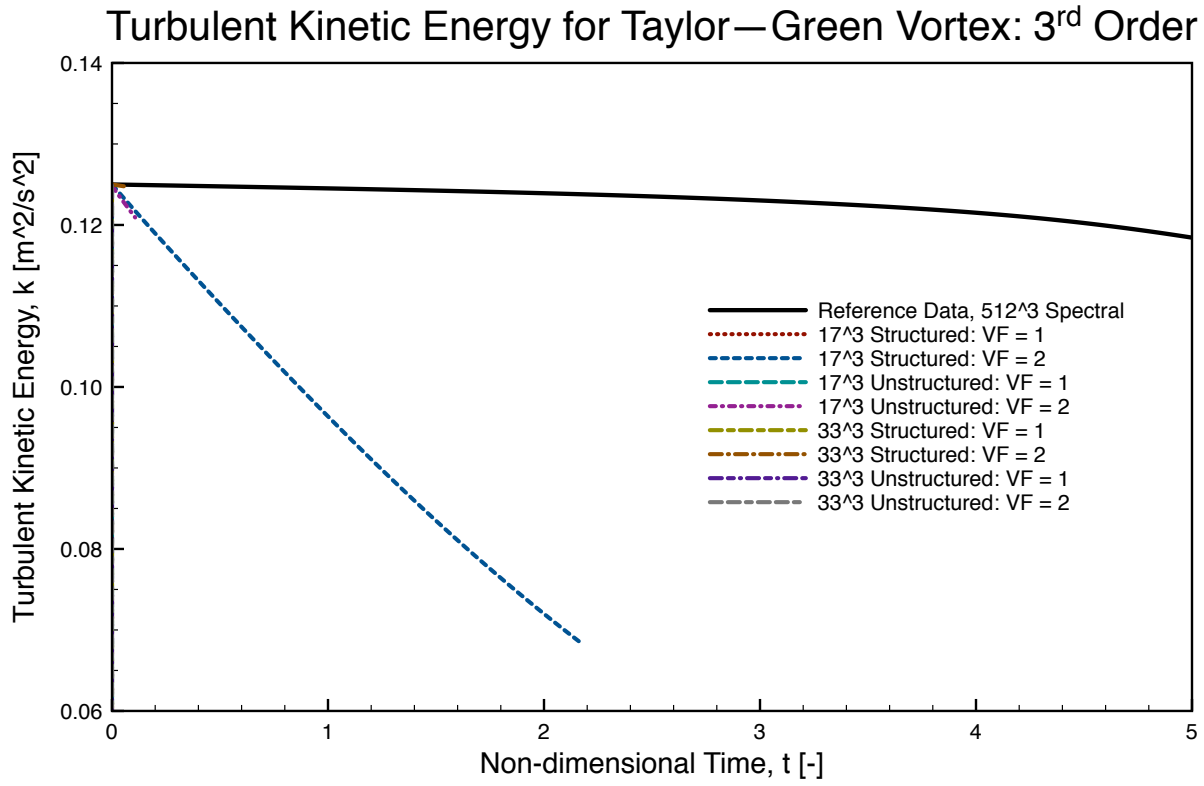


Figure 7.118: Comparison of the Turbulent Kinetic Energy for Coarse Meshes: 3^{rd}-Order.

Again, the baseline viscous flux reconstruction is overly dissipative and should not be used. The full viscous flux reconstruction was not used here as the method is too slow to run sequentially. As shown previously in Section 7.2.1.2, the third-order results tend to be more dissipative than the second-order results. Though barely noticeable, the 33^3 results appear to produce less dissipation than the 17^3 results. It should be noted that these results have been obtained after running for several months, so it is clear that any further work on higher-order methods should be carried out

after parallelizing the code. Figure 7.119 shows the fourth-order results.

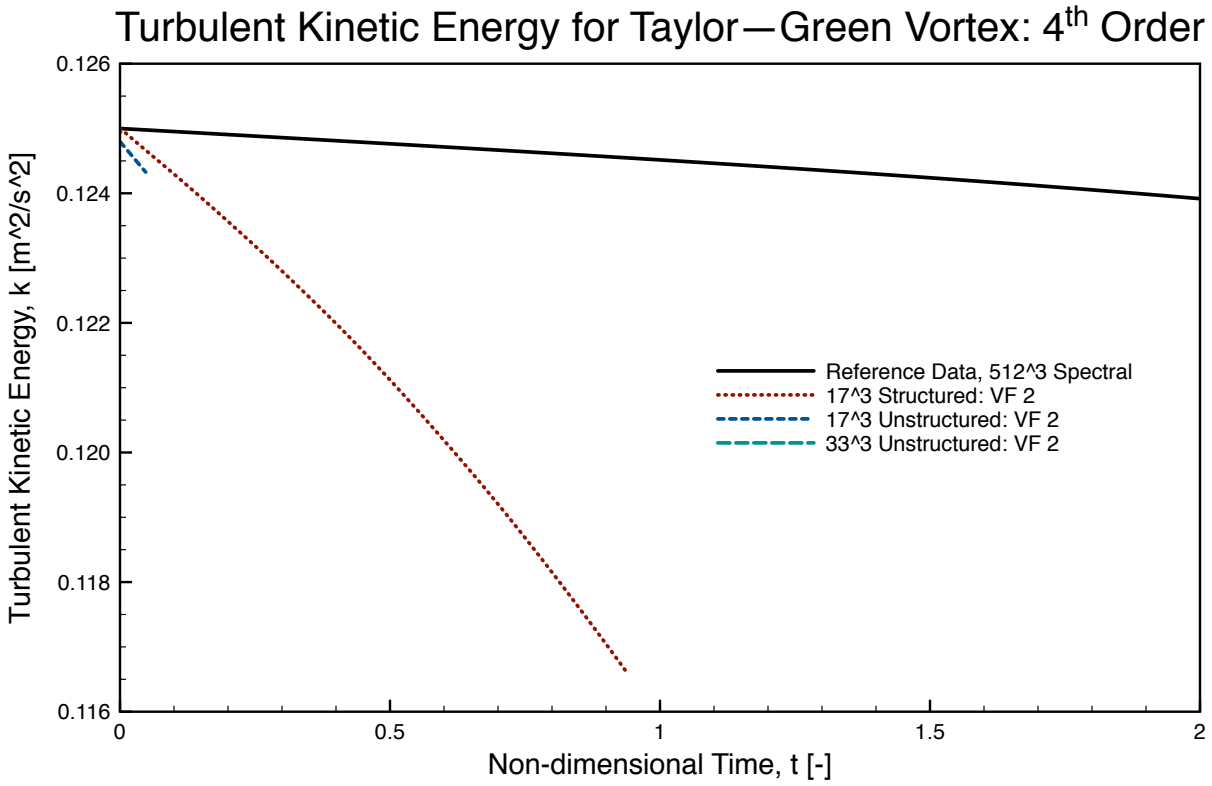


Figure 7.119: Comparison of the Turbulent Kinetic Energy for Coarse Meshes: 4^{th}-Order.

Unfortunately, there is not much that can be gleamed from the fourth-order results alone. However, some conclusions can be drawn from Figure 7.120, which compares results for each order on the 17^3 mesh.

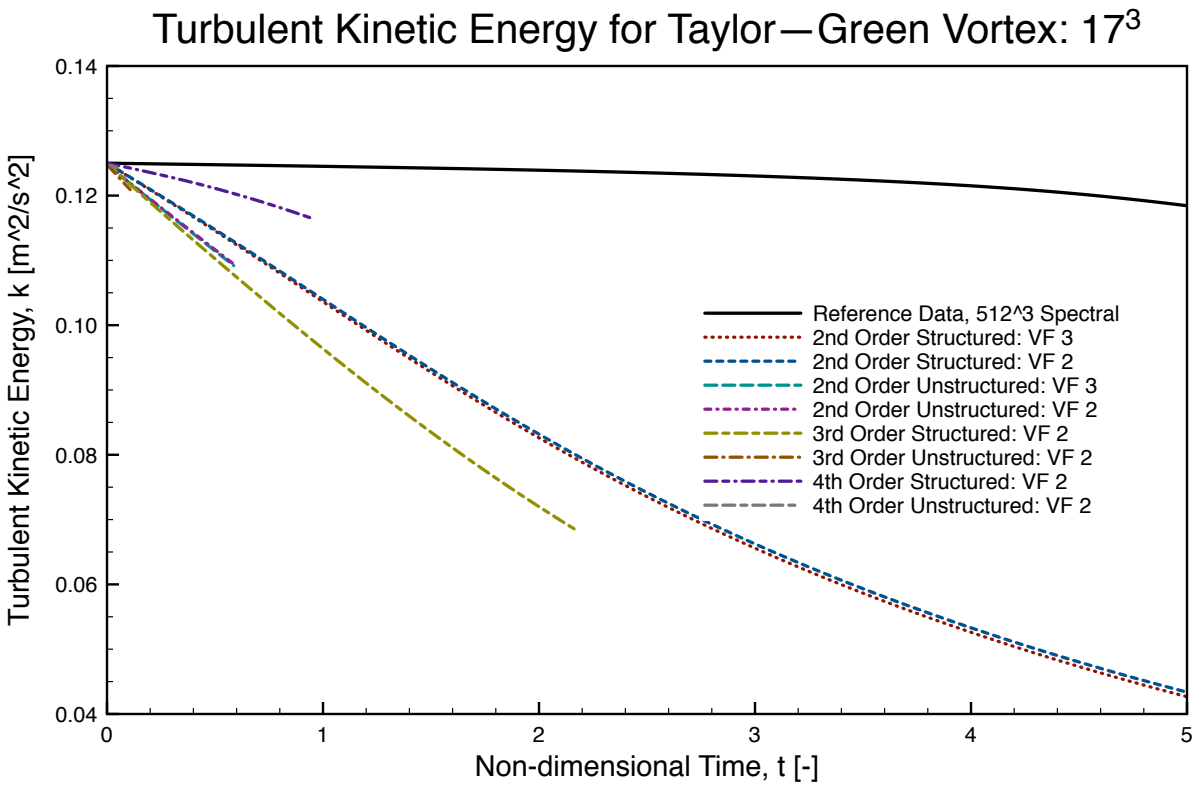


Figure 7.120: Comparison of the Turbulent Kinetic Energy for 17^3 Meshes.

As shown previously in Section 7.2.1.2, the fourth-order method is the least dissipative whereas the third-order results are the most dissipative. The unstructured results also are more dissipative than the structured results. For these simulations, the unstructured results take significantly longer to acquire due to the smaller cell sizes. While it would be interesting to know how different the results would be overall between the structured and unstructured meshes, this cannot be carried out until the code is parallelized.

7.4 Summary

The focus of this chapter was on the usage of the MLS method to compute the gradients for more practical three-dimensional meshes and the application of MLS with ghost nodes for the boundary conditions. Reviewing Section 7.1, the results of Chapter 6 carry over. The results of Section 7.1 show that the Pivoting QR method will generally produce the best conditioned system

regardless of the weighting strategy. Additionally, the Affine MLS, when coupled with the Gegenbauer basis, will produce significantly better conditioned moment matrix $\mathbf{M}$. This is especially true for the laminar flat plate grids (or stretched grids) shown in Figure 7.5, where for any order over second, only the Affine MLS strategy produced moment matrices $\mathbf{M}$ with condition numbers even remotely acceptable. The other weighting strategies produce extremely ill-conditioned systems. In fact, from these results, one could speculate that the Affine MLS could overcome most mesh deficiencies, which arise from either user error in generating the mesh or from the requirements of the geometry. Some additional work needs to be performed to see how much of a beneficial affect the Affine MLS has on these types of mesh topologies.

The ghost node boundary conditions have a marked improvement on the flow, as shown in Sections 7.2.2, 7.2.3 7.2.4, and 7.2.5. In terms of the NSCBC ghost boundary results of Section 7.2.2, the conditions perform well at convecting out the vortex, for any order. When compared to the Riemann invariant based boundary conditions shown in Figures 7.15-7.17, the ghost boundary conditions show their worth. Having the ghost node allows more control with the characteristics than the Riemann invariants, which effectively smash the downstream traveling vortex and throw the information upstream. The NSCBC results also show that using the diffuse derivatives to compute the flow gradients have negligible differences between the full derivatives, as shown in Section 7.2.2.2. This is a great result, as there are situations (moving meshes) when using diffuse derivatives is beneficial to save computational time.

The ghost boundaries show their greatest benefit though for the bump-in-channel cases of Sections 7.2.3 and 7.2.4. The ghost boundary condition results converge significantly faster than the standard boundary conditions due to the usage of the NSCBC significantly reducing the reflections in the domain. Additionally, the ghost boundary conditions for the wall are actually zero-normal pressure gradient, as shown in Figures 7.75 and 7.93. Having the ghost node allows the user to correctly prescribe the boundaries, unlike the standard boundary conditions.

The laminar flat plate results of Section 7.2.5 also show how well the ghost boundary conditions perform compared to the standard boundary conditions. Specifically, the adiabatic flat plate

using ghost nodes correctly has a zero-normal thermal gradient, whereas the standard boundary conditions are incorrect, as shown in Figure 7.104. For both the adiabatic and isothermal flat plate cases with ghost nodes, the boundary layer profiles are closer to the Blasius solutions than the standard boundary condition. The direct viscous flux computes the best results for either case. It is speculated that the difference between the viscous fluxes is a result of excessive dissipation. This is backed up by the results for the second-order TGV case in Section 7.3, which show the standard viscous flux dissipating all of the turbulent kinetic energy soon after the simulation started. Finally, the TGV case showed that the higher-order schemes produce lower dissipation than the second-order scheme, but additional work in parallelizing the higher-order method needs to be performed before extracting more significant information from the case.

8. CONCLUSIONS AND FUTURE WORK

This chapter presents the conclusions from results obtained herein from this dissertation. This chapter concludes with suggestions for future work based upon this dissertation.

8.1 Conclusions

In this dissertation, a higher-order unstructured finite volume computational fluid dynamics model was implemented using Moving Least-Squares as the main backbone of the method. The full method was developed in the historical context of the state-of-the-art CFD methods, and the case was made for developing the higher-order method. Due to current limitations, the Moving Least-Squares method was selected to overcome mesh topology limitations present in other methods. However, as it was shown herein, the Moving Least-Squares method as shown has it's own set of limitations. To this effect, the Affine MLS method was developed. The Affine MLS method, as shown herein, produced significantly lower condition numbers for the MLS moment matrix M. Without the Affine MLS method, computing accurate gradients would be impossible for some mesh topologies. In conjunction with the Affine MLS method, an anisotropic weighting scheme was developed using the MVEE algorithm. This weighting scheme does a better job at describing the relative locations of nodes in stencils than the general isotropic weighting scheme. Finally, a Pivoting QR method was implemented, which helped reduce the condition number of the MLS systems.

The MLS method was analyzed using four analytic two-dimensional functions. Effectively, the method produces accurate results on structured and unstructured meshes. On stretched grids, the results were accurate for the Gaussian, but not for the other functions when performing a fourth-order reconstruction. This is interesting because it gives a clear indication that the higher-order derivatives of C can be detrimental to the accuracy of the reconstruction. So in addition to being faster, the diffuse derivatives in may produce more accurate results, which was unexpected. The scaling parameter tests were interesting as well. It was shown that in general, the scaling

parameter k that produced the most accurate results generally had the worst conditioned moment matrix $\mathbf{M}$. The range of scaling parameters which gave the 'best' results ($k \in [0.6 - 0.85]$) were the best comprise of the condition number and accuracy. The orthogonal bases helped further improve the condition number of the system, with the super-spherical Gegenbauer basis producing a significantly lower conditioned system compared to any other basis without any loss in accuracy.

The ghost boundary conditions generated herein are also a dramatic improvement to the standard boundary conditions. The ghost nodes are essential to the MLS method, since this radial method (or any other method) cannot model information beyond the boundary accurately. The ghost nodes are defined using finite differences that are coupled normal to the boundary, which better enforces the gradient conditions. For example, the new ghost NSCBC allow information to exit the domain with minimal disturbance, on par with the results reported by Granet et al. [70] for each order. The ghost nodes can also dramatically improve the convergence of the solution, as shown in Sections 7.2.3 and 7.2.4, on the order of ten or greater times faster. Also, the ghost boundaries can better enforce the physical boundary conditions, such as the inviscid and viscous walls. In fact, the ghost boundaries actually enforce the zero-pressure gradient boundary in the inviscid case, and the zero-normal thermal gradient for the adiabatic case. For the laminar cases, it was also interesting to see the effect of the viscous flux reconstruction methods. The standard method is much too dissipative to accurately model the boundary layer. Finally, though the current implementation is limited to sequential runs, effectively limiting the sizes of the cases that can be run, the preliminary TGV cases show that the finer mesh and higher-order do have lower rates of dissipation.

8.2 Future Work

This section presents possible future avenues of research and work that can build off of the work presented herein.

First, while the periodic boundaries and MLS stencils are fully parallelized, none of the boundary conditions nor the auxiliary stencils (image, intercept, ghost, and viscous flux) are. These are difficult to parallelize with the current parallelization paradigm. The current paradigm takes geom-

etry output from a grid generator, pre-processes it to generate mesh relations, stencils, and ghosts, and splits the preprocessed mesh and auxiliary information for the number of processors desired. This process is sufficient for smaller geometries; however, this process can become intractable as the number of nodes grows, especially since the current methodology generates $n_{\text{vector size}} \times n_{\text{processor}}$ arrays to build the parallel geometries, which severely limits the size or order of meshes that can be parallelized. For example, when attempting to pre-process grids for the TGV case, a the 128^3 case takes approximately 3 days to pre-process on a supercomputer using 64 Gbs of memory. A 256^3 case takes at least 7 days to pre-process, but since the limit of usage on the supercomputer for a single processor job is 1 week, this case not could be generated. Additionally, all of the TGV cases only have MLS stencils. One can only estimate that the addition of non-periodic conditions, and all the auxiliary information required for higher-order methods, would limit the size of cases that would be possible to run. To avoid this bottleneck, the parallelization paradigm must be changed. The recommended approach would be to at least switch the splitting and preprocessor steps. Effectively, the splitting would reduce the amount of information that must be generated during the preprocessor step. This would require parallelizing the preprocessor step, which would still be a challenging undertaking but would allow for larger cases to be generated for running. Additionally, parallelizing the preprocessor would allow for intermediate adjustments of certain grid parameters, such as during mesh adaptation or mesh movement (with significant movement) during runtime. This change would also make using viscous flux stencils much more feasible, since file size for even small meshes is prohibitive for third- or higher-order cases. For example, the laminar flat plate grid shown in Figure 7.5, has a reduced viscous stencil file (reduced means a second-order stencil) of 2.7 Gigabytes, which is impossible to load on a single processor. The viscous flux stencils have been shown to be very effective in the cases presented. Additional study should be made into determining a minimum size and more efficient storage for the viscous flux stencils as well.

Next, the ghost node boundaries will need some additional work for usage in real flows. Parallelizing the conditions should be a top priority. To parallelize the ghost node boundary conditions, the interface for UMFpack/SuiteSparse [31] needs to be parallelized. While there is a deployment

of a parallel UMFpack/SuiteSparse, this is only written in C/C++. In fact, even the sequential UMFpack/SuiteSparse is written in C/C++. For this dissertation, a FORTRAN interface was written for the sequential UMFpack/SuiteSparse. It is unclear how a parallel interface would work, but it is necessary if the higher-order ghost boundary conditions are to be utilized in higher-order methods. This is especially true for 'real' geometries, where corners and sharp edges are prevalent, that will have coupling of the ghost node conditions.

Following the last point, the ghost node boundaries built here can handle corner topologies, but a further study on how corner ghost nodes that are coupled to one another would need to be investigated. Some of this work has been performed in the literature in terms of the Immersed Boundary Method, so that work could again be adapted into the current methodology. One thing that remains unclear is if, or how, corners exterior to the domain should be defined. Obviously, the easiest approach is to average the adjacent boundary conditions, as suggested by Blazek [12, Chapter 8]. This would be insufficient for the non-reflecting boundary conditions described in Sections 5.4.2.3-5.4.2.6. Granet et al. [70] did not include nodes in the corner region when developing the boundary condition, but as was done herein the corner conditions are coupled in Section 5.4.2.6. This may not be sufficient for some flows, so some additional work could be done on this subject.

Once the code is parallelized, the RANS boundary conditions can be tested. The RANS boundary conditions where discussed in Chapter 5, and these were implemented in the solver. However, running even a turbulent flat plate case was intractable, as only a few hundred iterations could be managed. This brings up the issue of testing boundary conditions. As it currently stands, the way the boundary conditions (or any other issue in the code) are tested is with a full simulation. This can quickly become an issue, such as turbulent flows. Implementing a Method of Manufactured Solutions (MMS) [144] would help alleviate many of these issues. MMS will allow a much better assessment of the flow solver accuracy and validity without the need to run more realistic test cases to 'shake' out the code before full deployment. This also creates a numerical sandbox, where more advanced and new methods can be tested. To wrap up discussion on the RANS method, the higher-order inviscid and viscous fluxes were implemented. **?**] claimed that beyond first-order accuracy

was not necessary for the SST model, and this has been accepted ever sense. With the work herein, it would be worthwhile to see if first-order accuracy for the RANS model really is sufficient.

Geometric considerations must also be considered for future work. In general, curved boundaries present a problem for higher-order methods. The standard approach is to generate the geometric description in a CAD software or similar and generate the mesh topology by approximating surfaces with nodes. This process tends to provide an inexact description of the surface, which then further provides inexact geometric quantities that lead to first-order errors on curved boundaries [113]. Researchers attempt to correct this error by generating approximating curves on the surface to generate the more accurate geometric descriptions, such as cell-volumes, edge- and surface-areas, and surface and nodal normals. This, however, may not be sufficient to fully overcome the error on curved boundaries, since an additional error may be introduced when the surface is refined, leading to different surface descriptions. In order to best reduce the errors caused by surface approximations, analytic geometry must be used to describe a surface and related geometric quantities. This can either be done from the known description of the surface or from using NURBS [128] or method to generate an analytic description of the surface. For either case, the analytic geometry would eliminate a potential first-order source of error that undoubtedly reduces the accuracy of the method. This potential future work becomes even more critical when considering LES, especially ILES, where the grid provides the underlying sub-grid model. If the surface description is off, there is no hope for correctly modeling a flow with ILES using standard geometric descriptions of the surface.

Additional work needs to be done with the higher-order method to allow for mesh movement, which would be required for cases having mesh movement, especially those with aeroelasticity. It is unclear if the relative locations in $\Omega_{\mathbf{x}_I}$ would need to be redetermined, which is a complicated process if periodicity is involved, or if the MLS basis functions can be 'moved' without a loss in accuracy of the method. The generation of ghost meshes in these situations would also need to be explored.

Finally, more extensive work into higher-order limiters needs to done. Though not shown

herein, there are issues with using higher-order limiters on the cases presented. This is presumably due to issues coupling the ghost fluxes and main fluxes. Additionally, there are limited options for the higher-order limiter, as this is very much an active area of research. Furthermore, there are limited discussions on how limiters are implemented at even standard boundaries, much less ghost boundaries.

REFERENCES

[1] Abramowitz, M., & Stegun, I. A. (1964). *Handbook of mathematical functions: with formulas, graphs, and mathematical tables* volume 55. Courier Corporation.

[2] Ahipasaoglu, S. D., Sun, P., & Todd, M. J. (2008). Linear convergence of a modified frank-wolfe algorithm for computing minimum-volume enclosing ellipsoids. *Optimization Methods Software*, *23*, 5–19.

[3] Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., & Sorensen, D. (1999). *LAPACK Users' Guide*. (3rd ed.). Philadelphia, PA: Society for Industrial and Applied Mathematics.

[4] Arden, M. (2009). Benchmarkfcns.

[5] Aslam, T. D. (2004). A partial differential equation approach to multidimensional extrapolation. *Journal of Computational Physics*, *193*, 349 – 355.

[6] Atwood, C. L. (1973). Sequences converging to d-optimal designs of experiments. *The Annals of Statistics*, *1*, 342–352.

[7] Balsara, D. S., & Shu, C.-W. (2000). Monotonicity Preserving Weighted Essentially Non-oscillatory Schemes with Increasingly High Order of Accuracy. *Journal of Computational Physics*, *160*, 405 – 452.

[8] Bandringa, H. (2010). *Immersed boundary methods*. Master's thesis University of Groningen Groningen, Netherlands.

[9] Berthelsen, P. A., & Faltinsen, O. M. (2008). A local directional ghost cell approach for incompressible viscous flow problems with irregular boundaries. *Journal of Computational Physics*, *227*, 4354 – 4397.

[10] Betke, U., & Henk, M. (1993). Approximating the volume of convex bodies. *Discrete & Computational Geometry*, *10*, 15–21.

[11] Biswas, D. (2006). Studies on Unsteady Laminar-Turbulent Transition in a Low Pressure Turbine Flow Based on a Higher Order LES model. In *36th AIAA Fluid Dynamics Confer-*

ence and Exhibit (pp. 1–18). San Francisco, California.

[12] Blazek, J. (2001). *Computational Fluid Dynamics: Principles and Applications*. Amsterdam: Elsevier.

[13] Blazek, J., Irmisch, S., & Haselbacher, A. (1999). Unstructured Mixed-grid Navier-Stokes Solver for Turbomachinery Applications. AIAA Paper 99-0664.

[14] Boris, J. P. (1990). On Large Eddy Simulation using subgrid turbulence models: Comment 1. In *Whither Turbulence? Turbulence at the Crossroads* (pp. 344–353). Springer.

[15] Bosshard, C., Dehbi, A., Deville, M., Leriche, E., Puragliesi, R., & Soldati, A. (2013). Large Eddy Simulation of the differentially heated cubic cavity flow by the spectral element method. *Computers & Fluids*, *86*, 210 – 227.

[16] Breitkopf, P., & Huerta, A. (2004). *Meshfree and particle based approaches in computational mechanics*. Kogan Page Science.

[17] Carley, M. (2010). Moving least squares via orthogonal polynomials. *SIAM Journal on Scientific Computing*, *32*, 1310–1322.

[18] Carpenter, F. (2016). *Practical Aspects of Computational Fluid Dynamics for Turbomachinery*. Ph.D. thesis.

[19] Carpenter, M. H., & Kennedy, C. A. (1994). Fourth-order 2n-storage runge-kutta schemes. *NASA TM-109112*, .

[20] Cenaero (2016). 4th international workshop on higher-order cfd methods.

[21] Chassaing, J.-C., Khelladi, S., & Nogueira, X. (2013). Accuracy assessment of a high-order moving least squares finite volume method for compressible flows. *Computers & Fluids*, *71*, 41 – 53.

[22] Chassaing, J.-C., Nogueira, X., & Khelladi, S. (2013). Moving Kriging reconstruction for high-order finite volume computation of compressible flows. *Computer Methods in Applied Mechanics and Engineering*, *253*, 463–478.

[23] Chenoweth, S. K., Soria, J., & Ooi, A. (2009). A singularity-avoiding moving least squares scheme for two-dimensional unstructured meshes. *Journal of Computational Physics*, *228*,

5592 – 5619.

[24] Cizmas, P. G. A. (1995). *A Simultaneously Coupled Potential/Boundary Layer Model of Stall Flutter in Turbomachinery*. Ph.D. thesis Duke University Durham, NC.

[25] Cockburn, B., & Shu, C.-W. (2001). Runge–Kutta Discontinuous Galerkin Methods for Convection-Dominated Problems. *Journal of Scientific Computing*, *16*, 173–261.

[26] Cueto-Felgueroso, L. (2005). *Partículas, Volúmenes Finitos y Mallas No Estructuradas: Simulación Numerica de Problemas de Dinámica de Fluidos*. Ph.D. thesis Universidade de Coruña La Coruña.

[27] Cueto-Felgueroso, L., & Colominas, I. (2008). High-order Finite Volume Methods and Multiresolution Reproducing Kernels. *Archives of Computational Methods in Engineering*, *15*, 185–228.

[28] Cueto-Felgueroso, L., & Colominas, I. (2008). High-order finite volume methods and multiresolution reproducing kernels. *Archives of Computational Methods in Engineering*, *15*, 185–228.

[29] Cueto-Felgueroso, L., Colominas, I., Fe, J., Navarrina, F., & Casteleiro, M. (2006). High-order finite volume schemes on unstructured grids using moving least-squares reconstruction. application to shallow water dynamics. *International Journal for Numerical Methods in Engineering*, *65*, 295–331.

[30] Dairay, T., Lamballais, E., Laizet, S., & Vassilicos, J. C. (2017). Numerical dissipation vs. subgrid-scale modelling for large eddy simulation. *Journal of Computational Physics*, *337*, 252–274.

[31] Davis, T. A. (2004). Algorithm 832: Umfpack v4.3—an unsymmetric-pattern multifrontal method. *ACM Trans. Math. Softw.*, *30*, 196–199.

[32] Dax, A. (2000). A modified Gram–Schmidt algorithm with iterative orthogonalization and column pivoting. *Linear Algebra and its Applications*, *310*, 25–42.

[33] Deardorff, J. W. (1970). A numerical study of three-dimensional turbulent channel flow at large Reynolds numbers. *Journal of Fluid Mechanics*, *41*, 453–480.

[34] Deardorff, J. W. (1974). Three-dimensional numerical study of the height and mean structure of a heated planetary boundary layer. *Boundary-Layer Meteorology*, *7*, 81–106.

[35] Deniau, H., & Nybelen, L. (2009). Strategy for spatial simulation of co-rotating vortices. *International journal for numerical methods in fluids*, *61*, 23–56.

[36] Di Pasquale, D., Rona, A., & Garrett, S. (2009). A selective review of transition modelling for cfd. In *39th AIAA fluid dynamics conference* (p. 3812).

[37] Ducros, F., Ferrand, V., Nicoud, F., Weber, C., Darracq, D., Gacherieu, C., & Poinsot, T. (1999). Large-Eddy Simulation of the Shock/Turbulence Interaction. *Journal of Computational Physics*, *152*, 517 – 549.

[38] Dumbser, M., Hidalgo, A., & Zanotti, O. (2014). High order space–time adaptive ADER-WENO finite volume schemes for non-conservative hyperbolic systems. *Computer Methods in Applied Mechanics and Engineering*, *268*, 359 – 387.

[39] Dumbser, M., Iben, U., & Munz, C.-D. (2013). Efficient implementation of high order unstructured WENO schemes for cavitating flows. *Computers & Fluids*, *86*, 141 – 168.

[40] Dunkl, C. F., & Xu, Y. (2014). *Orthogonal polynomials of several variables*. 155. Cambridge University Press.

[41] Ekaterinaris, J. A. (2005). High-order accurate, low numerical diffusion methods for aerodynamics. *Progress in Aerospace Sciences*, *41*, 192–300.

[42] Erlebacher, G., Hussaini, M., Speziale, C., & Zang, T. A. (1992). Toward the Large-Eddy Simulation of compressible turbulent flows. *Journal of Fluid Mechanics*, *238*, 155–185.

[43] Fasshauer, G. E. (2005). Meshfree methods. In *Handbook of Theoretical and Computational Nanotechnology*. American Scientific Publishers.

[44] Fasshauer, G. E. (2007). *Meshfree approximation methods with MATLAB* volume 6. World Scientific.

[45] Fedorov, V. V. (1972). *Theory of optimal experiments*. Elsevier.

[46] Ferziger, J. (1985). Large Eddy Simulation: Its role in turbulence research. In D. Dwoyer, M. Hussaini, & R. Voigt (Eds.), *Theoretical Approaches to Turbulence* (pp. 51–72). Springer

New York volume 58 of *Applied Mathematical Sciences*.

[47] Fontenot, R. L. (2012). *Advances in reduced-order modeling based on proper orthogonal decomposition for single and two-phase flows*. Master's thesis Texas A&M University.

[48] Fornberg, B. (1988). Generation of finite difference formulas on arbitrarily spaced grids. *Mathematics of computation*, *51*, 699–706.

[49] Fornberg, B. (1998). Calculation of Weights in Finite Difference Formulas. *SIAM Review*, *40*, 685–691.

[50] Fornberg, B. (1998). Classroom note: Calculation of weights in finite difference formulas. *SIAM review*, *40*, 685–691.

[51] Fornberg, B., Driscoll, T., Wright, G., & Charles, R. (2002). Observations on the Behavior of Radial Basis Function Approximations Near Boundaries. *Computers and Mathematics with Applications*, *43*, 473–490.

[52] Frank, M., & Wolfe, P. (1956). An algorithm for quadratic programming. *Naval Research Logistics (NRL)*, *3*, 95–110.

[53] Freno, B. A., Fontenot, R., Matula, N., & Cizmas, P. G. (2014). The use of dynamic basis functions in proper orthogonal decomposition. In *52nd Aerospace Sciences Meeting* (p. 1436).

[54] Freno, B. A., Matula, N. R., Fontenot, R. L., & Cizmas, P. G. (2015). The use of dynamic basis functions in proper orthogonal decomposition. *Journal of Fluids and Structures*, (pp. 332–360).

[55] Fukuchi, T. (2014). Finite difference method and algebraic polynomial interpolation for numerically solving poisson's equation over arbitrary domains. *AIP Advances*, *4*, 060701.

[56] Galperin, B., & Orszag, S. A. (1993). *Large Eddy Simulation of Complex Engineering and Geophysical Flows*. Cambridge University Press.

[57] Gargoloff, J. I. (2007). *A Numerical Method for Fully Nonlinear Aeroelastic Analysis*. Ph.D. thesis Texas A&M University College Station, Texas.

[58] Garnier, E., Adams, N., & Sagaut, P. (2009). *Large Eddy Simulation for Compressible*

Flows. Scientific Computation. Berlin: Springer.

[59] Gassner, G. J., & Beck, A. D. (2013). On the accuracy of high-order discretizations for underresolved turbulence simulations. *Theoretical Computational Fluid Dynamics*, *27*, 201–237.

[60] Germano, M. (1992). Turbulence - The filtering approach. *Journal of Fluid Mechanics*, *238*, 325–336.

[61] Germano, M., Piomelli, U., Moin, P., & Cabot, W. (1991). A dynamic subgrid-scale eddy viscosity model. *Physics of Fluids*, *A 3*, 1760–1765.

[62] Ghosal, S. (1996). An Analysis of Numerical Errors in Large-Eddy Simulations of Turbulence. *Journal of Computational Physics*, *125*, 187 – 206.

[63] Ghosal, S., Lund, T. S., Moin, P., & Akselvoll, K. (1995). A dynamic localization model for Large-Eddy Simulation of turbulent flows. *Journal of Fluid Mechanics*, *286*, 229–255.

[64] Gibou, F., & Fedkiw, R. (2004). A fourth order accurate discretization for the laplace and heat equations on arbitrary domains, with applications to the stefan problem. *J. Comput. Phys*, *202*, 2005.

[65] Girimaji, S. S., & Abdol-Hamid, K. S. (2005). Partially-Averarged Navier–Stokes Model for Turbulence: Implementation and Validation. In *43rd AIAA Aerospace Sciences Meeting & Exhibit* AIAA Paper 2005-0502. Reno, NV.

[66] Golub, G. H., & Van Loan, C. F. (1996). *Matrix Computations (3rd Ed.)*. Baltimore, MD, USA: Johns Hopkins University Press.

[67] Gordnier, R., Chimakurthi, S., Cesnik, C., & Attar, P. (2010). Implicit les simulations of a flexible flapping wing. In *51st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference 18th AIAA/ASME/AHS Adaptive Structures Conference 12th* (p. 2960).

[68] Gordnier, R. E., Chimakurthi, S. K., Cesnik, C. E., & Attar, P. J. (2013). High-fidelity aeroelastic computations of a flapping wing with spanwise flexibility. *Journal of Fluids and Structures*, *40*, 86–104.

[69] Gossler, A. (2001). *Moving Least-Squares: a numerical differentiation method for irregularly spaced calculation points*. Technical Report June Sandia National Laboratories Albuquerque, New Mexico.

[70] Granet, V., Vermorel, O., Léonard, T., Gicquel, L., & Poinsot, T. (2010). Comparison of nonreflecting outlet boundary conditions for compressible solvers on unstructured grids. *AIAA journal*, *48*, 2348–2364.

[71] Grinstein, F., & Fureby, C. (2005). LES studies of the flow in a swirl gas combustor. *Proceedings of the Combustion Institute*, *30*, 1791 – 1798.

[72] Grinstein, F. F., Margolin, L. G., & Rider, W. J. (2007). *Implicit Large Eddy Simulation: Computing Turbulent Fluid Dynamics*. Cambridge University Press.

[73] Gruber, P. M., & Schuster, F. E. (2005). An arithmetic proof of john's ellipsoid theorem. *Archiv der Mathematik*, *85*, 82–88.

[74] Hahn, M., & Drikakis, D. (2009). Assessment of Large-Eddy Simulation of internal separated flow. *Journal of Fluids Engineering*, *131*, 071201.

[75] Hahn, M., & Drikakis, D. (2009). Implicit Large-Eddy Simulation of swept-wing flow using high-resolution methods. *AIAA Journal*, *47*, 618–630.

[76] Hajek, J. (2009). qrupdate.

[77] Han, Z., & Cizmas, P. G. A. (2003). Prediction of axial thrust load in centrifugal compressors. *International Journal of Turbo & Jet-Engines*, *20*, 1–16.

[78] Harman, R., & Pronzato, L. (2007). Improvements on removing non-optimal support points in D-optimum design algorithms. *Statistics and Probability Letters*, *77*, 90–94.

[79] Harten, A., & Hyman, J. M. (1983). Self adjusting grid methods for one-dimensional hyperbolic conservation laws. *Journal of computational Physics*, *50*, 235–269.

[80] Hindenlang, F., Gassner, G. J., Altmann, C., Beck, A., Staudenmaier, M., & Munz, C.-D. (2012). Explicit Discontinuous Galerkin methods for unsteady problems. *Computers & Fluids*, *61*, 86 – 93.

[81] Hoare, C. A. (1962). Quicksort. *The Computer Journal*, *5*, 10–16.

[82] Hubbard, S. A., Fontenot, R. L., McFarland, D. M., Cizmas, P. G., Bergman, L. A., Strganac, T. W., & Vakakis, A. F. (2014). Transonic aeroelastic instability suppression for a swept wing by targeted energy transfer. *Journal of Aircraft*, *51*, 1467–1482.

[83] Hughes, T. J., Mazzei, L., & Jansen, K. E. (2000). Large Eddy Simulation and the variational multiscale method. *Computing and Visualization in Science*, *3*, 47–59.

[84] Jiang, L., Shan, H., Liu, C., & Visbal, M. R. (1999). Non-reflecting boundary conditions for dns in curvilinear coordinates. In D. Knight, & L. Sakell (Eds.), *Recent Advances in DNS and LES* (pp. 219–233). Dordrecht: Springer Netherlands.

[85] Kansa, E. (1990). Multiquadrics–a scattered data approximation scheme with applications to computational fluid-dynamics–i surface approximations and partial derivative estimates. *Computers & Mathematics with Applications*, *19*, 127 – 145.

[86] Karamanos, G.-S., & Karniadakis, G. (2000). A Spectral Vanishing Viscosity Method for Large-Eddy Simulations. *Journal of Computational Physics*, *163*, 22 – 50.

[87] Kato, C., Kaiho, M., & Manabe, A. (2001). *Industrial applications of LES in mechanical engineering*. Technical Report DTIC Document.

[88] Kemm, F. (2011). A comparative study of tvd-limiters–well-known limiters and an introduction of new ones. *International Journal for Numerical Methods in Fluids*, *67*, 404–440.

[89] Kennedy, C. A., Carpenter, M. H., & Lewis, R. M. (2000). Low-Storage, explict Runge–Kutta schemes for the compressible Navier–Stokes equations. *Applied Numerical Mathematics*, *35*, 177–219.

[90] Khachiyan, L. G. (1996). Rounding of polytopes in the real number model of computation. *Mathematics of Operations Research*, *21*, 307–320.

[91] Kim, K. S. (2003). *Three-dimensional hybrid grid generator and unstructured flow solver for compressors and turbines*. Ph.D. thesis Texas A&M University College Station, TX.

[92] Kincaid, D., & Cheney, W. (2002). *Numerical Anaylsis: Mathematics of Scientific Computing*. (3rd ed.). Pacific Grove, CA: American Mathematical Society.

[93] Krath, E., Carpenter, F., & Cizmas, P. (2017). *A Zeta-Coordinate Proper Orthogonal De-*

composition Method with Dynamic Basis Functions for Turbomachinery Aeroelastic Analysis. Technical Report Texas Engineering Experiment Station College Station, TX.

[94] Kumar, P., & Yildirim, E. A. (2005). Minimum-volume enclosing ellipsoids and core sets. *Journal of Optimization Theory and Applications*, *126*, 1–21.

[95] Lancaster, P., & Salkauskas, K. (1981). Surfaces generated by moving least squares methods. *Mathematics of computation*, *37*, 141–158.

[96] Larchevêque, L., Sagaut, P., & Labbé, O. (2007). Large-Eddy Simulation of a subsonic cavity flow including asymmetric three-dimensional effects. *Journal of Fluid Mechanics*, *577*, 105–126.

[97] Larchevêque, L., Sagaut, P., Le, T.-H., & Comte, P. (2004). Large-Eddy Simulation of a compressible flow in a three-dimensional open cavity at high Reynolds number. *Journal of Fluid Mechanics*, *516*, 265–301.

[98] Larchevêque, L., Sagaut, P., Mary, I., Labbé, O., & Comte, P. (2002). Large-Eddy Simulation of a compressible flow past a deep cavity. *Physics of Fluids (1994-present)*, *15*, 193–210.

[99] Lee, M., Malaya, N., & Moser, R. D. (2013). Petascale Direct Numerical Simulation of Turbulent Channel Flow on up to 786k Cores. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis* SC '13 (pp. 61:1–61:11). New York, NY, USA: ACM.

[100] Lévêque, E., Toschi, F., Shao, L., & Bertoglio, J.-P. (2007). Shear-improved Smagorinsky model for Large-Eddy Simulation of wall-bounded turbulent flows. *Journal of Fluid Mechanics*, *570*, 491–502.

[101] Levin, D. (1998). The approximation power of moving least-squares. *Mathematics of Computation of the American Mathematical Society*, *67*, 1517–1531.

[102] Levin, D. (2004). Mesh-independent surface interpolation. In *Geometric modeling for scientific visualization* (pp. 37–49). Springer.

[103] Lilly, D. K. (1966). The Representation of Small-Scale Turbulence in Numerical Simula-

tion Experiments. In I. H. Goldstine (Ed.), *Proceedings of the IBM Scientific Computing Symposium on Environmental Sciences* (pp. 195–210). Yorktown Heights, NY.

[104] Liu, G., & Gu, Y. (2005). *An Introduction to Meshfree Methods and Their Programming*. Springer Netherlands.

[105] Lodato, G. (2008). *Tridimensional Boundary Conditions for Direct and Large-Eddy Simulation of Turbulent Flows. Sub-Grid Scale Modeling for Near-Wall Region Turbulence*. Ph.D. thesis INSA de Rouen.

[106] Lohner, R. (2012). Improved error and work estimates for high-order elements. *International Journal for Numerical Methods in Fluids*, *72*, 1207–1218.

[107] MacNeal, R. H. (1993). *Finite Elements: Their Design and Performance*. New York: Marcel Dekker.

[108] Mahesh, K., Constantinescu, G., & Moin, P. (2004). A numerical method for Large-Eddy Simulation in complex geometries. *Journal of Computational Physics*, *197*, 215–240.

[109] Marcellán, F., Pérez, T. E., & Piñar, M. A. (1994). Gegenbauer-sobolev orthogonal polynomials. In *Nonlinear Numerical Methods and Rational Approximation II* (pp. 71–82). Springer.

[110] Margolin, L. G., Rider, W. J., & Grinstein, F. F. (2006). Modeling turbulent flow with implicit LES. *Journal of Turbulence*, *7*, 1–27.

[111] Mathar, R. J. (2008). Zernike basis to cartesian transformations. *arXiv preprint arXiv:0809.2368*, .

[112] McMullan, W., & Page, G. (2012). Towards Large Eddy Simulation of gas turbine compressors. *Progress in Aerospace Sciences*, *52*, 30 – 47. Applied Computational Aerodynamics and High Performance Computing in the UK.

[113] Mei, R., Luo, L.-S., & Shyy, W. (1999). An accurate curved boundary treatment in the lattice boltzmann method. *Journal of computational physics*, *155*, 307–330.

[114] Meneveau, C., & Katz, J. (2000). Scale-invariance and turbulence models for Large-Eddy Simulation. *Annual Review of Fluid Mechanics*, *32*, 1–32.

[115] Menter, F. R. (1994). Two-equation eddy-viscosity turbulence models for engineering applications. *AIAA Journal*, *32*, 1598–1605.

[116] Michalak, C., & Ollivier-Gooch, C. (2009). Accuracy preserving limiter for the high-order accurate solution of the euler equations. *Journal of Computational Physics*, *228*, 8693 – 8711.

[117] Mirzaei, D., Schaback, R., & Dehghan, M. (2012). On generalized moving least squares and diffuse derivatives. *IMA Journal of Numerical Analysis*, *32*, 983–1000.

[118] Mittal, R., & Iaccarino, G. (2005). Immersed Boundary Methods. *Annual Review of Fluid Mechanics*, *37*, 239–261.

[119] Modenov, P., Parkhomenko, A., Booker, H., Bromley, D., & DeClaris, N. (2014). *Euclidean and Affine Transformations: Geometric Transformations*. Academic paperbacks. Elsevier Science.

[120] Motheau, E., Almgren, A., & Bell, J. B. (2017). Navier–stokes characteristic boundary conditions using ghost cells. *AIAA Journal*, (pp. 1–10).

[121] Nealen, A. (2004). An as-short-as-possible introduction to the least squares, weighted least squares and moving least squares methods for scattered data approximation and interpolation. *URL: http://www. nealen. com/projects*, *130*, 150.

[122] Nejat, A., & Ollivier-Gooch, C. (2008). A high-order accurate unstructured finite volume newton–krylov algorithm for inviscid compressible flows. *Journal of Computational Physics*, *227*, 2582 – 2609.

[123] Nogueira, X., Cueto-Felgueroso, L., Colominas, I., & Gomez, H. (2010). Implicit Large Eddy Simulation of non-wall-bounded turbulent flows based on the multiscale properties of a high-order finite volume method. *Computer Methods in Applied Mechanics and Engineering*, *199*, 615 – 624.

[124] Nogueira, X., Cueto-Felgueroso, L., Colominas, I., Gomez, H., Navarrina, F., & Casteleiro, M. (2009). On the accuracy of finite volume and discontinuous Galerkin discretizations for compressible flow on unstructured grids. *International Journal for Numerical Methods in*

Engineering, *78*, 1553–1584.

[125] Nogueira, X., Cueto-Felgueroso, L., Colominas, I., Navarrina, F., & Casteleiro, M. (2010). A new shock-capturing technique based on Moving Least Squares for higher-order numerical schemes on unstructured grids. *Computer Methods in Applied Mechanics and Engineering*, *199*, 2544 – 2558.

[126] Nogueira, X., Khelladi, S., Colominas, I., Cueto-Felgueroso, L., París, J., & Gómez, H. (2011). High-resolution finite volume methods on unstructured grids for turbulence and aeroacoustics. *Archives of Computational Methods in Engineering*, *18*, 315–340.

[127] O'Mahoney, T., Hills, N., & Chew, J. (2012). Sensitivity of LES results from turbine rim seals to changes in grid resolution and sector size. *Progress in Aerospace Sciences*, *52*, 48 – 55. Applied Computational Aerodynamics and High Performance Computing in the UK.

[128] Piegl, L., & Tiller, W. (2012). *The NURBS book*. Springer Science & Business Media.

[129] Piomelli, U. (1999). Large-Eddy Simulation: achievements and challenges. *Progress in Aerospace Sciences*, *35*, 335 – 362.

[130] Poinsot, T., & Lele, S. (1992). Boundary conditions for direct simulations of compressible viscous flows. *Journal of Computational Physics*, *101*, 104 – 129.

[131] Pope, S. (2000). *Turbulent Flows*. Cambridge, UK: Cambridge University Press.

[132] Pope, S. B. (2008). Algorithms for ellipsoids. *Cornell University Report No. FDA*, (pp. 08–01).

[133] Pouangué, A. F., Deniau, H., & Lamarque, N. (2010). A Sixth-Order Compact Finite-Volume scheme for Aeroacoustics: Application to a Large Eddy Simulation of a Jet. In *V European Conference on Computational Fluid Dynamics*. Lisbon, Portugal.

[134] Qu, Y., & Batra, R. C. (2017). Constrained moving-least squares immersed boundary method for fluid-structure interaction analysis. *International Journal for Numerical Methods in Fluids*, .

[135] Raverdy, B., Mary, I., Sagaut, P., & Liamis, N. (2003). High-resolution Large-Eddy Simulation of flow around low-pressure turbine blade. *AIAA Journal*, *41*, 390–397.

[136] Reddy, J. N. (2006). *An introduction to the finite element method* volume 3. McGraw-Hill New York.

[137] Reynolds, W. (1990). The potential and limitations of Direct and Large Eddy Simulations. In J. Lumley (Ed.), *Whither Turbulence? Turbulence at the Crossroads* (pp. 313–343). Springer Berlin Heidelberg volume 357 of *Lecture Notes in Physics*.

[138] Roe, P. L. (1981). Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, *43*, 357–372.

[139] Rudy, D. H., & Strikwerda, J. C. (1980). A nonreflecting outflow boundary condition for subsonic navier-stokes calculations. *Journal of Computational Physics*, *36*, 55–70.

[140] Rudy, D. H., & Strikwerda, J. C. (1981). Boundary conditions for subsonic compressible navier-stokes calculations. *Computers & Fluids*, *9*, 327–338.

[141] Sagaut, P. (2006). *Large Eddy Simulation for Incompressible Flows: An Introduction*. Scientific Computation. Berlin: Springer.

[142] Sagaut, P., & Deck, S. (2009). Large Eddy Simulation for aerodynamics: status and perspectives. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *367*, 2849–2860.

[143] Sagaut, P., & Drikakis, D. (2010). Large Eddy Simulation. In *Encyclopedia of Aerospace Engineering* chapter 7. (pp. 1–7). John Wiley & Sons, Ltd.

[144] Salari, K., & Knupp, P. (2000). *Code verification by the method of manufactured solutions*. Technical Report Sandia National Labs., Albuquerque, NM (US); Sandia National Labs., Livermore, CA (US).

[145] Samet, H. (1988). An overview of quadtrees, octrees, and related hierarchical data structures. *NATO ASI Series*, *40*, 51–68.

[146] Schaback, R. (2011). The missing Wendland functions. *Advances in Computational Mathematics*, *34*, 67–81.

[147] Shu, C.-W. (2003). High-order finite difference and finite volume WENO schemes and Discontinuous Galerkin methods for CFD. *International Journal of Computational Fluid*

Dynamics, *17*, 107–118.

[148] Smagorinsky, J. (1963). General circulation experiments with the primitive equations: I. The basic experiment. *Monthly Weather Review*, *91*, 99–164.

[149] Smirnov, S., Lacor, C., & Baelmans, M. (2001). A Finite Volume Formulation for Compact Scheme with Applications to LES. In *15th AIAA Computational Fluid Dynamics Conference*. Anaheim, California.

[150] Subbareddy, P. K., & Candler, G. V. (2009). A fully discrete, kinetic energy consistent finite-volume scheme for compressible flows. *Journal of Computational Physics*, *228*, 1347–1364.

[151] Suman, S., & Girimaji, S. S. (2010). On the Invariance of Compressible Navier-–Stokes and Energy Equations Subject to Density-Weighted Filtering. *Flow, Turbulence and Combustion*, *85*, 383–396.

[152] Sutherland, J. C., & Kennedy, C. A. (2003). Improved boundary conditions for viscous, reacting, compressible flows. *Journal of computational physics*, *191*, 502–524.

[153] Szeg, G. (1939). *Orthogonal polynomials* volume 23. American Mathematical Soc.

[154] Thompson, K. W. (1987). Time dependent boundary conditions for hyperbolic systems. *Journal of computational physics*, *68*, 1–24.

[155] Titarev, V., & Toro, E. (2004). Finite-volume WENO schemes for three-dimensional conservation laws. *Journal of Computational Physics*, *201*, 238 – 260.

[156] Todd, M. J. (2016). *Minimum-volume ellipsoids: Theory and algorithms*. SIAM.

[157] Tseng, Y.-H., & Ferziger, J. H. (2003). A ghost-cell immersed boundary method for flow in complex geometry. *Journal of computational physics*, *192*, 593–623.

[158] Tucker, P. (2011). Computation of unsteady turbomachinery flows: Part 2–LES and hybrids. *Progress in Aerospace Sciences*, *47*, 546 – 569.

[159] Tyacke, J., Jefferson-Loveday, R., & Tucker, P. (2012). On LES Methods Applied to Seal Geometries. In *ASME Turbo Expo 2012: Turbine Technical Conference and Exposition* (pp. 2053–2065). American Society of Mechanical Engineers.

[160] Tyacke, J., Jefferson-Loveday, R., & Tucker, P. G. (2011). Application of LES to labyrinth

seals. In *Proceedings of the AIAA CFD Conference, AIAA Paper no. AIAA-2011-3861*.

[161] Unger, F., & Friedrich, R. (1991). Large Eddy Simulation of Fully-Developed Turbulent Pipe Flow. In *8th Symposium on Turbulent Shear Flows*. Munich, Germany.

[162] Venkatakrishnan, V. (1995). Convergence to Steady State Solutions of the Euler Equations on Unstructured Grids with Limiters. *Journal of Computational Physics*, *118*, 120–130.

[163] Wang, G., Papadogiannis, D., Duchaine, F., Gourdain, N., & Gicquel, L. (2013). Towards massively parallel Large Eddy Simulation of turbine stages. In *ASME TURBO EXPO 2013 Gas Turbine Technical Congress & Exposition*. San Antonio, USA. Conf.

[164] Wang, J. G., & Liu, G. R. (2002). A point interpolation meshless method based on radial basis functions. *International Journal for Numerical Methods in Engineering*, *54*, 1623–1648.

[165] Wei, M., & Liu, Q. (2003). Roundoff error estimates of the modified Gram–Schmidt algorithm with column pivoting. *BIT Numerical Mathematics*, *43*, 627–645.

[166] Wendland, H. (1995). Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics*, *4*, 389–396.

[167] White, F. M., & Corfield, I. (2006). *Viscous fluid flow* volume 3. McGraw-Hill New York.

[168] Wilcox, D. (2006). *Turbulence Modeling for CFD*. La Cañada, CA: DCW Industries, Incorporated.

[169] Wilcox, D. C. (2001). Turbulence Modeling: An Overview. In *39th Aerospace Sciences Meeting & Exhibit* AIAA Paper 2001-0724. Reno, NV.

[170] Williamson, J. (1980). Low-storage runge-kutta schemes. *Journal of Computational Physics*, *35*, 48–56.

[171] Wissink, J., & Rodi, W. (2006). Direct Numerical Simulations of transitional flow in turbomachinery. *Journal of Turbomachinery*, *128*, 668–678.

[172] Wold, S., Esbensen, K., & Geladi, P. (1987). Principal component analysis. *Chemometrics and intelligent laboratory systems*, *2*, 37–52.

[173] Wolfe, P. (1970). Convergence theory in nonlinear programming. *Integer and Nonlinear*

Programming, (pp. 1–36).

[174] Wynn, H. P. et al. (1970). The sequential generation of d-optimum experimental designs. *The Annals of Mathematical Statistics*, *41*, 1655–1664.

[175] Xu, X., Lee, J. S., & Pletcher, R. H. (2005). A compressible finite volume formulation for large eddy simulation of turbulent pipe flows at low Mach number in Cartesian coordinates. *Journal of Computational Physics*, *203*, 22–48.

[176] Yakhot, V. (1999). LES of Turbulence and Turbulent Combustion: Advances and Theoretical limitations. In S. Biringen, H. Örs, A. Tezel, & J. H. Ferziger (Eds.), *Industrial and Environmental Applications of Direct and Large-Eddy Simulation* (pp. 263–278). Springer Berlin Heidelberg volume 529 of *Lecture Notes in Physics*.

[177] Yoo, C. S., & Im, H. G. (2007). Characteristic boundary conditions for simulations of compressible reacting flows with multi-dimensional, viscous and reaction effects. *Combustion Theory and Modelling*, *11*, 259–286.

[178] Yoo, C. S., Wang, Y., Trouvé, A., & Im, H. G. (2005). Characteristic boundary conditions for direct simulations of turbulent counterflow flames. *Combustion Theory and Modelling*, *9*, 617–646.

[179] Zhang, Y.-T., & Shu, C.-W. (2009). Third order WENO scheme on three dimensional tetrahedral meshes. *Communications in Computational Physics*, *5*, 836–848.

[180] Zwillinger, D. (2002). *CRC Standard Mathematical Tables and Formulae, 31st Edition*. Discrete Mathematics and Its Applications. Taylor & Francis.

APPENDIX A

MINIMUM VOLUME ENCLOSING ELLIPSOID FOR GRADIENTS

A.1 Introduction

This appendix presents the algorithm for determining the ellipsoidal coefficients for the stencil scaling function s (5.6) used for anisotropic $\Omega_{\mathbf{x}_I}$. For an in-depth discussion on how optimal ellipsoids can be generated from a convex hull of points, refer to [156], specifically Chapters 1 through 3. This appendix will summarize the main points and algorithms required to determine the ellipsoidal coefficients from the Minimum Volume Enclosing Ellipsoid.

A.2 Definition

If the position of any point in $\Omega_{\mathbf{x}_I}$ is defined as:

$$\Delta_{jI}\mathbf{x} = \mathbf{x}_j - \mathbf{x}_I, \tag{A.1}$$

then the centered minimum-volume ellipsoid enclosing $\Omega_{\mathbf{x}_I}$ is defined as:

$$\begin{aligned} &\min_{\mathbf{H}} \; -\ln\left[\det(\mathbf{H})\right] \\ &\quad \Delta_{jI}\mathbf{x}^T\mathbf{H}\Delta_{jI}\mathbf{x} \leq n,\; j = 1, 2, \ldots, m \end{aligned} \tag{A.2}$$

where n is the dimension of the problem and m is the total number of points in $\Omega_{\mathbf{x}_I}$. Although n is not in general restricted, herein, n is either 2 or 3. The matrix $\mathbf{H}$ describes the shape of the ellipsoid centered at $\mathbf{x}_I$:

$$E(\mathbf{H}, \mathbf{x}_I) \coloneqq \left\{\mathbf{x} \in R^n : \Delta_{jI}\mathbf{x}^T\mathbf{H}\Delta_{jI}\mathbf{x} \leq n,\; j = 1, 2, \ldots, m\right\}. \tag{A.3}$$

$E(\mathbf{H}, \mathbf{x}_I)$ is also known as the volume of the ellipsoid. $\mathbf{H}$ has the following properties:

- $\mathbf{H}$ is symmetric positive definite.
- $\mathbf{H}$ is of the order n.
- $\mathbf{H} = \mathbf{H}^T$.
- $\mathbf{v}^T\mathbf{H}\mathbf{v} > 0$ for all $\mathbf{v} \in R^n$.

It is important to note that $\mathbf{H}$ is not strictly optimized for a minimum volume, but is instead designed to have a minimum volume for the convex hull derived from the design points $\Omega_{\mathbf{x}_I}$ such that both $\mathbf{H}$ and $\Delta_{jI}\mathbf{x}$ are jointly convex. The problem is convex because the objective function constraints are linear in $\mathbf{H}$, and the objective function itself is convex. To solve (A.2), both the convex set of points from $\Omega_{\mathbf{x}_I}$ and the convex $\mathbf{H}$ must be determined simultaneously. For the general case of $\Omega_{\mathbf{x}_I}$ presented herein, finding the optimal convex set of points is relatively simple compared to finding the optimal minimum volume $\mathbf{H}$ for the optimal convex point set.

A.3 The Dual Problem

The problem (A.2) can be optimized (and solved) via a dual problem. This section describes the dual problem, which is generally easier to solve than the main problem. Recall the problem (A.2)

$$\begin{aligned} &\min_{\mathbf{H}\in S^n} \; -\ln\left[\det(\mathbf{H})\right[\\ &\qquad \Delta_{jI}\mathbf{x}^T\mathbf{H}\Delta_{jI}\mathbf{x} \leq n,\; j = 1, 2, \ldots, m \end{aligned} \tag{A.4}$$

where now explicitly $\mathbf{H}$ lies in the span of the data points $\Omega_{\mathbf{x}_I}$. Also, we assume that $\mathbf{X}$ has full row rank, such that $\mathbf{X}$ is a matrix with columns containing all coordinates of points in $\Omega_{\mathbf{x}_I}$:

$$\mathbf{X} = [\mathbf{x}_{1I}, \mathbf{x}_{2I}, \ldots, \mathbf{x}_{mI}] \in R^{nxm}. \tag{A.5}$$

Using Lagrange multipliers u_j for the constraints, where $\mathbf{u} \in R^m$, a finite Lagrangian is obtained:

$$L(\mathbf{H}, \mathbf{u}) = -\ln\left[\det(\mathbf{H})\right] + \sum_j u_j(\Delta_{jI}\mathbf{x}^T\mathbf{H}\Delta_{jI}\mathbf{x} - n) \tag{A.6}$$

for all positive definite $\mathbf{H}$ satisfying the constraints. Additionally, the Lagrangian is a strictly convex function of $\mathbf{H}$. Denoting

$$\mathbf{U} = \text{diag}(\mathbf{u}) \in S^m \tag{A.7}$$

and using the unit vector $\mathbf{e} \in R^m$, the Lagrangian is rewritten as:

$$L(\mathbf{H}, \mathbf{u}) = -\ln\left[\det(\mathbf{H})\right] + \mathbf{H} \cdot \mathbf{X}\mathbf{U}\mathbf{X}^T - n\mathbf{e}^T\mathbf{u}. \tag{A.8}$$

For any non-negative $\mathbf{u}$ and $\mathbf{H}$ that satisfies the constraints then

$$-\ln\left[\det(\mathbf{H})\right] \geq \min_H L(\mathbf{H}, \mathbf{u}) \tag{A.9}$$

which implies

$$\mathbf{v}(E) \geq \min_H L(\mathbf{H}, \mathbf{u}) \tag{A.10}$$

where $\mathbf{v}(E)$ denotes the optimal solution to the ellipsoid problem E. The Lagrangian has the derivative

$$\nabla_{\mathbf{H}} L(\mathbf{H}, \mathbf{u}) = -\mathbf{H}^{-1} + \mathbf{X}\mathbf{U}\mathbf{X}^T \tag{A.11}$$

and is minimized if and only if

$$\mathbf{H} = (\mathbf{X}\mathbf{U}\mathbf{X}^T)^{-1} \tag{A.12}$$

and $\mathbf{X}\mathbf{U}\mathbf{X}^T$ is positive definite. If instead of being positive definite, $\mathbf{X}\mathbf{U}\mathbf{X}^T$ is allowed to be positive *semidefinite*. Let $\mathbf{v} \in R^n$ satisfy $\mathbf{X}\mathbf{U}\mathbf{X}^T\mathbf{v} = 0$ with a unit norm, and consider $\mathbf{H}(\lambda) = \mathbf{I} + \lambda\mathbf{v}\mathbf{v}^T$ such that the Lagrangian becomes:

$$\begin{aligned} L(\mathbf{H}(\lambda), \mathbf{u}) &= -\ln\left\{\det[\mathbf{I} + \lambda\mathbf{v}\mathbf{v}^T]\right\} + (\mathbf{I} + \lambda\mathbf{v}\mathbf{v}^T) \cdot \mathbf{X}\mathbf{U}\mathbf{X}^T - n\mathbf{e}^T\mathbf{u} \\ &= -\ln(1 + \lambda) + Tr[\mathbf{X}\mathbf{U}\mathbf{X}^T] - n\mathbf{e}^T\mathbf{u}. \end{aligned} \tag{A.13}$$

Letting $\lambda \to \infty$, $L(\mathbf{H}(\lambda), \mathbf{u}) \to -\infty$. From this, the minimum value of the Lagrangian is:

$$\begin{aligned}\min_{H} L(\mathbf{H}, \mathbf{u}) &= -\ln\left\{\det[(\mathbf{X}\mathbf{U}\mathbf{X}^T)^{-1}]\right\} + (\mathbf{X}\mathbf{U}\mathbf{X}^T)^{-1} \cdot \mathbf{X}\mathbf{U}\mathbf{X}^T - n\mathbf{e}^T\mathbf{u} \\ &= \ln\left\{\det[\mathbf{X}\mathbf{U}\mathbf{X}^T]\right\} + n - n\mathbf{e}^T\mathbf{u}. \end{aligned} \tag{A.14}$$

To bound the optimal solution $\mathbf{v}(E)$ (A.10), let $\mathbf{u} \in R_+^m$ maximize the right-hand side of (A.14), which will yield the first dual problem:

$$\max_{u \in R^m} -\ln\left\{\det[\mathbf{X}\mathbf{U}\mathbf{X}^T]\right\} + n - n\mathbf{e}^T\mathbf{u} : \mathbf{u} \geq \mathbf{0} \tag{A.15}$$

$\mathbf{u}$ is further constrained $\mathbf{u}$ such that $\mathbf{e}^T\mathbf{u} = 1$. With any $\hat{\mathbf{u}} \geq \mathbf{0}$ written as $\lambda\mathbf{u}$ if $\lambda \geq 0$, and if $\mathbf{e}^T\mathbf{u} = 1$ for $\mathbf{u} \geq \mathbf{0}$, then:

$$\begin{aligned}\ln\left\{\det[\mathbf{X}\hat{\mathbf{U}}\mathbf{X}^T)]\right\} + n - n\mathbf{e}^T\hat{\mathbf{u}} &= \ln\left\{\det[\lambda\mathbf{X}\mathbf{U}\mathbf{X}^T)]\right\} + n - n\lambda \\ &= n\ln\lambda + \ln\left\{\det[\mathbf{X}\mathbf{U}\mathbf{X}^T)]\right\} + n - n\lambda \end{aligned} \tag{A.16}$$

which is at a maximum when $\lambda = 1$. The best bound for $\mathbf{v}(E)$ (A.10) is therefore determined from the final dual problem:

$$\begin{aligned}\max_{\mathbf{u} \in R^M} \; & g(\mathbf{u}) = \ln\left\{\det[\mathbf{X}\mathbf{U}\mathbf{X}^T]\right\} \\ & \mathbf{e}^T\mathbf{u} = 1 \\ & \mathbf{u} \geq \mathbf{0}, \end{aligned} \tag{A.17}$$

therefore:

$$f(\mathbf{H}) \geq g(\mathbf{u}). \tag{A.18}$$

The result From the above equations is that the optimal problem (A.10) has a unique optimal solution, the dual problem (A.17) has an optimal solution, and that between (A.10) and (A.17)

there is no gap in duality. The lack of a duality gap and strong duality is not proven here, but the reader is referred to Section 2.1 in [156].

From the strong duality of the problem, the following conditions are necessary and sufficient for optimal solutions $\mathbf{H}$ and $\mathbf{u}$, for $\mathbf{v}(E)$ (A.10) and the dual problem (A.17):

- $\mathbf{e}^T\mathbf{u} = 1$, $\mathbf{u} \geq 0$, and $\Delta_{jI}\mathbf{x}^T\mathbf{H}\Delta_{jI}\mathbf{x} \leq n,\ j = 1, 2, \ldots, m$
- $\mathbf{H} = (\mathbf{X}\mathbf{U}\mathbf{X}^T)^{-1}$
- $\Delta_{jI}\mathbf{x}^T\mathbf{H}\Delta_{jI}\mathbf{x} = n$ if $\mathbf{u}_j > 0$, $j = 1, 2, \ldots, m$

Using item (2) from the conditions given above, the optimal conditions with respect to the dual problem (A.17) and solution $\mathbf{u}$ are described if and only if:

- $\mathbf{H}(\mathbf{u}) = (\mathbf{X}\mathbf{U}\mathbf{X}^T)^{-1}$ is a feasible for $\mathbf{v}(E)$
- $\Delta_{jI}\mathbf{x}^T\mathbf{H}(\mathbf{u})\Delta_{jI}\mathbf{x} = n$ if $\mathbf{u}_j > 0$, $j = 1, 2, \ldots, m$.

Finally, an approximate optimality condition is introduced as a condition for convergence in the later algorithm to solve the dual problem such that $\mathbf{u}$ is ϵ-primal feasible. This optimality condition holds if $\mathbf{H}(\mathbf{u}) = (\mathbf{X}\mathbf{U}\mathbf{X}^T)^{-1}$ satisfies

$$\Delta_{jI}\mathbf{x}^T\mathbf{H}(\mathbf{u})\Delta_{jI}\mathbf{x} \leq (1+\epsilon)n,\ j = 1, 2, \ldots, m \tag{A.19}$$

and

$$\Delta_{jI}\mathbf{x}^T\mathbf{H}(\mathbf{u})\Delta_{jI}\mathbf{x} \geq (1-\epsilon)n \text{ if } \mathbf{u}_i > \mathbf{0},\ j = 1, 2, \ldots, m. \tag{A.20}$$

The solution to the suboptimal problem is then within δ of being optimal if it is both feasible and δ away of the optimal value. Using both (A.19) and (A.20), the suboptimal problem duality gap is $n\ln(1+\epsilon)$, and the suboptimal ellipsoid is within $(1+\epsilon)^{n/2}$ of the optimal minimum volume ellipsoid [156].

A.4 Algorithm for Suboptimal Problem

This section presents the algorithm of the suboptimal ellipsoid problem defined by (A.19) and (A.20), since we cannot directly solve the optimal (A.10) or dual problem (A.17). The algorithm presented here is that presented in [156].

A.4.1 Initialization

The suboptimal ellipsoid algorithm begins with the initialization of $\mathbf{u}$, or an approximation of the volume of the minimum volume ellipsoid. There are two methods to determine the initial $\mathbf{u}$: 1) Khachiyan [90] and 2) Kumar and Yildirim [94]. The Khachiyan algorithm applies a simple yet quality initialization of $\mathbf{u}$ [156]. The Khachiyan algorithm is:

Algorithm A.1 Khachiyan Algorithm

Require: $\mathbf{u}$ to be equal for all points m in $\Omega_{\mathbf{x}_I}$

$$\mathbf{u} = \frac{1}{m}\mathbf{1}. \tag{A.21}$$

The Khachiyan algorithm A.1 is first-order. The Kumar and Yildirim second-order algorithm [94] uses a modified algorithm of Betke and Henk [10]. The Betke and Henk algorithm is as follows:

Algorithm A.2 Betke–Henk Algorithm

1: Choose an arbitrary direction in the set $\Omega_{\mathbf{x}_I}$ and set the counter to $j = 1$.

2: Compute the absolute value didactic product of the j^{th} column vector of orthogonal matrix $\mathbf{Q}$. j^{th} columns of $\mathbf{Q}$ span the same space as the points in $\Omega_{\mathbf{x}_I}$, with all the points in $\Omega_{\mathbf{x}_I}$. The maximum of this product is assigned $u_{\max dx} = 1$. If $j = n$, the algorithm is complete.

3: Project $\mathbf{Q}$ onto the $\mathbf{y} = \Delta_{\max dx,I}\mathbf{x}$:

$$\mathbf{z} = \mathbf{Q}^T\mathbf{y} \tag{A.22}$$

such that $\bar{z}_j$ and $\underline{z}_j$ both maximize and minimize the projection for all points in $\Omega_{\mathbf{x}_I}$. This is done by computing the norm of $\mathbf{z}$ and modifying the z_j:

$$z_j = z_j + \text{sign}(z_j)||\mathbf{z}|| \tag{A.23}$$

for all z_k for $k = 1, \ldots j - 1$, $z_k = 0$.

4: Update the $\mathbf{Q}$ with $\mathbf{z}$:

$$\mathbf{Q} = \mathbf{Q} - \frac{1}{\text{sign}(z_j)||\mathbf{z}||z_j}(\mathbf{Q}\mathbf{z})^T\mathbf{Q}\mathbf{z} \tag{A.24}$$

choose another arbitrary direction, $j + 1$, and go to step 1.

The Kumar–Yildirim algorithm is as follows:

Algorithm A.3 Kumar–Yildirim Algorithm

1: Run the Betke–Henk Algorithm A.2

2: Set initial $\mathbf{u}$ to

$$\mathbf{u} = \frac{1}{d}\mathbf{u}_{BH} \tag{A.25}$$

Kumar and Yildirim [94] use the $\mathbf{u}$ computed from the algorithm as the initialization. The

default herein is the Kumar–Yildirim algorithm, though the Khachiyan algorithm is used for intermediate updates within the primary algorithm, due to its lower cost.

A.4.2 Solution

This section describes the algorithm to solve the suboptimal problem. There are again two options to solve the problem. The classical approach is attributed to Frank and Wolfe [52], though a later update was made by Fedorov [45] and Wynn [174], which is commonly known as the Frank–Wolfe algorithm. This algorithm is not described here as it is not the algorithm used in this work, though the option is available; see [156] for implementation aspects and details. The second algorithm, used herein, is attributed to Wolfe [173] and Atwood [6] (independent of Wolfe for the dual problem). It is commonly known as the Wolfe–Atwood algorithm.

The Wolfe–Atwood algorithm is described by the following steps:

Algorithm A.4 Wolfe–Atwood Algorithm

1: Initialize $\mathbf{u}$ using either the Khachiyan A.1 or Kumar–Yildirim A.3 algorithm.

2: Compute the variance $\boldsymbol{\omega}(\mathbf{u})$ and a scaled Cholesky factorization of $\mathbf{XUX}^T$ using $\mathbf{u}$. The scaled Cholesky factorization is:

$$\mathbf{XUX}^T = \phi^{-1}\mathbf{LL} = \mathbf{R}^T\mathbf{R} \tag{A.26}$$

with $\phi = 1$, and $\mathbf{R}$ determined from the QR decomposition

$$\mathbf{QR} = \text{diag}(\sqrt{\mathbf{u}})\mathbf{X}^T \tag{A.27}$$

An element of the variance is computed as:

$$\boldsymbol{\omega}(u_j) = \sum_i (\mathbf{R}^{-T}\mathbf{X})_{i,j}^2 \tag{A.28}$$

3: Using the current iterate of $\mathbf{u}$ and its corresponding variance, compute the maximum and minimum bounds. The maximum bound is :

$$\epsilon_+ = \max_i(\omega_i - n)/n, \tag{A.29}$$

and the minimum bound is:

$$\epsilon_- = \max_j[(n - \omega_j)/n : u_j > 0], \tag{A.30}$$

where the indices i and j give the maximum bounds, respectively. If both bounds are less than a specified tolerance, then $\mathbf{u}$ is suboptimal, and the MVEE is determined. Otherwise, the algorithm goes to Step 4 to continue iterating.

Algorithm A.4 Wolfe–Atwood Algorithm, Continued

4: Determine how the variance and scaled Cholesky factorization are updated based on the variance error. The variance error is determined by:

$$\epsilon_{\boldsymbol{\omega}} = \frac{|\epsilon_k - \epsilon_{+,-}|}{\max 1, \epsilon_{+,-}} \tag{A.31}$$

where $\epsilon_{+,-}$ is the maximum or minimum variance error determined in Step 3. If $\epsilon_+ + \epsilon_- > 2d$, then ϵ_+ is used and $u_k = u_i$, otherwise, ϵ_- is used and $u_k = u_j$. ϵ_k is determined using the partial Cholesky reconstruction at the $k = (i, j)$ index:

$$\epsilon_k = \phi \left(\mathbf{R}^{-T} \Delta \mathbf{x}_{kI} \right)^2 . \tag{A.32}$$

5: Set u_k using

$$u_k = \max(u_k + \lambda^*, 0) \tag{A.33}$$

where λ^* is the optimal step size computed using the variance from ϵ_+:

$$\lambda^* = \frac{\omega_k - n}{(n-1)\omega_k} \tag{A.34}$$

such that $\mathbf{u}$ is always positive. $\mathbf{u}$ is updated using:

$$\mathbf{u} = \frac{\mathbf{u} + \lambda^* \mathbf{e}}{1 + \lambda^*}. \tag{A.35}$$

6: If $\epsilon_{\boldsymbol{\omega}} > \epsilon$, the variance and scaled Cholesky factorization are recomputed using the method in Step 2, and Step 3 follows.

Algorithm A.4 Wolfe–Atwood Algorithm, Continued

7: If $\epsilon_{\boldsymbol{\omega}} \leq \epsilon$, the scaled Cholesky factorization is updated or downdated first. The factorization is updated if $\lambda^* < 0$ and

$$\mathbf{R}_{up} = \mathbf{R}^T\mathbf{R} + \mathbf{v}\mathbf{v}^T. \tag{A.36}$$

$\mathbf{v}$ is the vector determining the rank-1 update. If $\lambda^* \leq 0$, the downdate is:

$$\mathbf{R}_{dn} = \mathbf{R}^T\mathbf{R} - \mathbf{v}\mathbf{v}^T. \tag{A.37}$$

Both the update and downdate are performed using the library *qrupdate* [76]. The scaled Cholesky factor ϕ is updated also:

$$\phi = (1 + \lambda^*)\phi. \tag{A.38}$$

The variance is recomputed using the updated scaled Cholesky factorization:

$$\omega_l = (1 + \lambda^*)(\omega_k - \frac{\lambda^*}{1 + \lambda^*\omega_k}\sum_i(\phi\mathbf{R}^{-T}(\mathbf{R}^{-T}\Delta_{kI}\mathbf{x})\mathbf{X})^2_{i,l}). \tag{A.39}$$

The algorithm returns to Step 3.

The Wolfe–Atwood algorithm is modified in two key ways. First, the algorithm A.4 uses all the points in $\Omega_{\mathbf{x}_I}$, though many of them may not be necessary to determine the suboptimal ellipsoid. Within the algorithm, $\Omega_{\mathbf{x}_I}$ is reduced using the Harman–Prozato algorithm [78]. The Harman–Prozato algorithm is described as follows:

Algorithm A.5 Harman–Prozato Algorithm

1: Compute the threshold for eliminating points, using the maximum variance ω_i (from ϵ_+) as:

$$\Theta = n(1 + .5(\omega_i - n) - \sqrt{.5(\omega_i - n)(4 + (\omega_i = n) - 4/n)}) \tag{A.40}$$

and mark points in $\Omega_{\mathbf{x}_I}$ or the current reduced set (of size p), whose variances are greater than the threshold or $u_j > 0$. If the number of points marked is less than the current set, the current algorithm continues. Otherwise, no points are eliminated and A.4 continues.

2: $\mathbf{X}$ is reduced to only contain p points from $\Omega_{\mathbf{x}_I}$. If $p = n$, both the scaled Cholesky factorization and variance are updated as in step 4 of the main algorithm. If $p > n$, then only $\mathbf{u}$ is updated and the variances are reduced to contain only p points.

3: For $p > n$, $\mathbf{u}$ is updated by summing $\mathbf{u}$ for the active p points and dividing $\mathbf{u}$ by this sum. Algorithm A.4 continues using the reduced p set of points.

The Harman–Prozato algorithm is beneficial to use when $\Omega_{\mathbf{x}_I}$ contains interior (non-convex) points, which typically happens for third- and fourth-order stencils.

The second modification to A.4 is to recompute, rather than just update, the scaled Cholesky factorization. This modification was suggested by [156]. This is an expensive process, but when the recomputation of the scaled Cholesky factorization is done on a limited basis, the modification has been shown to improve convergence [156]. The scaled Cholesky factorization is computed as in Step 2 in the Wolfe–Atwood algorithm A.4. [156] also suggests updating the variance, if the L_1 norm from the recomputed scaled Cholesky factorization versus the old Cholesky factorization is

greater than a user-specified tolerance. The L_1 norm of the difference is determined by:

$$\begin{aligned} \mathbf{M} &= \phi \mathbf{R}_{\text{rec}}^T \mathbf{R}_{\text{rec}} - \mathbf{R}^T \mathbf{R} \\ \mathbf{M} &= \frac{1}{\phi ||\mathbf{M}||} \mathbf{M} \\ \text{norm}_M &= ||\mathbf{M}|| \end{aligned} \tag{A.41}$$

where $\mathbf{R}_{\text{rec}}$ is the rescaled Cholesky matrix.

With the two modifications suggested by [156], the Todd Algorithm for the MVEE problem is then:

Algorithm A.6 Todd Algorithm for MVEE

1: Do steps 1 and 2 of the Wolfe–Atwood algorithm A.4.
2: Use the Harman–Prozato algorithm A.5 to determine p convex points from $\Omega_{\mathbf{x}_I}$. Update $\mathbf{u}$ and the variance as needed.
3: Start the iteration for MVEE with step 3 of the Wolfe–Atwood algorithm A.4.
4: Perform Steps 4 and 5 of Wolfe–Atwood algorithm A.4.
5: If desired, recompute the scaled Cholesky factorization using Step 2 of the Wolfe–Atwood algorithm A.4 and determined if the variance is to be updated using (A.41).
6: Perform either Step 6 or 7 of the Wolfe–Atwood algorithm A.4, but only return to Step 3 if a point reduction is not desired.
7: If a point reduction is desired, attempt to reduce p using the Harman–Prozato algorithm A.5 to determine p convex points from $\Omega_{\mathbf{x}_I}$. Update $\mathbf{u}$ and $\boldsymbol{\omega}$ as needed and return to Step 3.

At the completion of Todd's algorithm A.6, the covariance matrix $\mathbf{M}$ is determined:

$$\mathbf{M} = \frac{1}{\phi} \mathbf{R}^T \mathbf{R}. \tag{A.42}$$

Note that the Cholesky factor $\neg\Box(\phi = 1)$ at the end of A.6. Additionally, $\mathbf{R}$ could be either the original Cholesky matrix or the rescaled Cholesky matrix $\mathbf{R}_{\text{rec}}$, depending on the algorithmic outcome. The ellipsoid matrix $\mathbf{H}(\mathbf{u})$ for the suboptimal problem is determined, from which the

ellipsoid coefficients can be extracted:

$$\mathbf{H}(\mathbf{u}) = \frac{1}{n}\mathbf{M}^{-1} \tag{A.43}$$

Todd's algorithm [**?**] has linear convergence [2], and often can stall when the initial ellipsoid is nearly the final ellipsoid. Since the use of the ellipsoid is to fit the data with some padding, a truly optimal MVEE is not completely necessary. A sub-optimal MVEE can be obtained sooner if an additional stopping criteria is applied. With this consideration, the rate change of the maximum and minimum variances can be tracked over a few iterations:

$$\epsilon_{\max\omega,i} = ||\max\omega_i - \max\omega_{i-2}| - |\max\omega_{i-1} - \max\omega_{i-3}|| \tag{A.44}$$

$$\epsilon_{\min\omega,i} = ||\min\omega_i - \min\omega_{i-2}| - |\min\omega_{i-1} - \min\omega_{i-3}|| \tag{A.45}$$

If $\epsilon_{\max\omega,i}$ or $\epsilon_{\min\omega,i}$ are less than the given tolerance $\epsilon_{\delta\omega}$, implying that the change in variance is minimal, then the ellipsoid can be considered a sub-optimal MVEE. Typically, $\epsilon_{\delta\omega} = 1e^{-8}$.

A.4.3 Extracting Ellipsoid Coefficients

This section presents how the ellipsoidal coefficients from $\mathbf{H}(\mathbf{u})$ are determined. Recall that the minimum volume ellipsoid satisfies:

$$\mathbf{X}^T\mathbf{H}(\mathbf{u})\mathbf{X} = 1 \tag{A.46}$$

from the dual problem. To find the semi-axis lengths for the ellipsoid, a singular value decomposition of $\mathbf{H}(\mathbf{u})$ is performed. Let $\mathbf{H}(\mathbf{u}) = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. $\mathbf{U}$ is a symmetric matrix whose columns represent the axes of the ellipsoid, $\mathbf{V}$ is the matrix containing the axis rotations, and $\mathbf{\Sigma}$ is a diagonal matrix with the diagonal equal to the inverse square of the semi-axis coefficients a, b, c, or alternatively the eigenvalues of $\mathbf{H}(\mathbf{u})$. An element in $\mathbf{\Sigma}$ is:

$$\Sigma_{i,i} = \frac{1}{||\mathbf{r}_i||^2}, \tag{A.47}$$

such that

$$||\mathbf{r}_i||^2 = \frac{1}{\Sigma_{i,i}}, \tag{A.48}$$

and the maximum length r_{max} is

$$r_{\text{max}} = \sqrt{\max||\mathbf{r}_i||^2}. \tag{A.49}$$

The ellipsoidal coefficients are extracted from $\mathbf{H}(\mathbf{u})$ as:

$$\mathbf{H}(\mathbf{u}) = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \begin{bmatrix} c_x & c_{xy} & c_{xz} \\ c_{xy} & c_y & c_{yz} \\ c_{xz} & c_{yz} & c_z \end{bmatrix} \tag{A.50}$$

such that the ellipsoid describing $\Omega_{\mathbf{x}_I}$ is:

$$c_x x^2 + c_y y^2 + c_z z^2 + c_{xy} xy + c_{yz} yz + c_{xz} xz = 1. \tag{A.51}$$

A.5 Example

This section provides an example of the minimum volume ellipsoid problem and the extraction of the ellipsoid coefficients used for s (5.6). The problem consists of the following simple $\Omega_{\mathbf{x}_I}$:

$$\Omega_{\mathbf{x}_I} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 1.5 \\ 0 & 0 & -1.5 \end{pmatrix} \tag{A.52}$$

with $k = .75$, the $\Omega_{\mathbf{x}_I}$ used for the problem is:

$$\Omega_{\mathbf{x}_I} = \begin{pmatrix} 0 & 0 & 0 \\ 1.3 & 0 & 0 \\ -1.3 & 0 & 0 \\ 0 & 2.6 & 0 \\ 0 & -2.6 & 0 \\ 0 & 0 & 2 \\ 0 & 0 & -2 \end{pmatrix} \tag{A.53}$$

In the standard spherical case, the coefficients would be $(c_x, c_y, c_z) = .1406$, with $r_{\max} = .375$. Using the Kumar–Yildirim algorithm A.3, $\mathbf{u}$ is

$$\mathbf{u} = \begin{pmatrix} 0 \\ \frac{1}{3} \\ 0 \\ \frac{1}{3} \\ 0 \\ \frac{1}{3} \\ 0 \end{pmatrix}. \tag{A.54}$$

The initial variance and Cholesky factorization ($\phi = 1$) are (Step 2 of the Wolfe–Atwood algorithm A.4):

$$\boldsymbol{\omega} = \begin{pmatrix} 0 \\ 3 \\ 3 \\ 3 \\ 3 \\ 3 \\ 3 \end{pmatrix} \tag{A.55}$$

and

$$\mathbf{R} = \begin{pmatrix} 0.7698 & 0 & 0 \\ 0 & 1.5396 & 0 \\ 0 & 0 & 1.1547 \end{pmatrix} \tag{A.56}$$

Points are removed in Step 2 of the Todd algorithm A.6 using Harman–Prozato algorithm A.5. In this case, only the first node is removed from $\Omega_{\mathbf{x}_I}$, such that

$$\Omega_{\mathbf{x}_I} = \begin{pmatrix} 1.3 & 0 & 0 \\ -1.3 & 0 & 0 \\ 0 & 2.6 & 0 \\ 0 & -2.6 & 0 \\ 0 & 0 & 2 \\ 0 & 0 & -2 \end{pmatrix} \tag{A.57}$$

The maximum and minimum variance are 3.. The initial error ϵ_+ and ϵ_- are $4.441e^{-16}$ and $-4.441e^{-16}$ as determined in Step 3 of the Todd algorithm A.6. If the initial tolerance is $1e^{-8}$,

then the solution has converged without any iterations. $\mathbf{H}(\mathbf{u})$ for the suboptimal problem is

$$\mathbf{H}(\mathbf{u}) = \begin{pmatrix} 0.5625 & 0 & 0 \\ 0 & 0.1406 & 0 \\ 0 & 0 & 0.25 \end{pmatrix}. \tag{A.58}$$

The maximum radii in this situation is $r_{\max} = 1.333$, where the SVD yields $\mathbf{\Sigma}$ as:

$$\mathbf{\Sigma} = \begin{pmatrix} 0.5625 & 0 & 0 \\ 0 & 0.25 & 0 \\ 0 & 0 & 0.1406 \end{pmatrix}. \tag{A.59}$$

In Figure A.1, the ellipsoid described by the coefficients is shown. In Figure A.2, the sphere for the isotropic weight is plotted over the ellipsoid. Note how much larger the sphere is, and how well the ellipsoid tightly closes $\Omega_{\mathbf{x}_I}$. In the next appendix, the isotropic weighting and anisotropic weighting will be compared to show the difference in the derivatives computed with MLS.

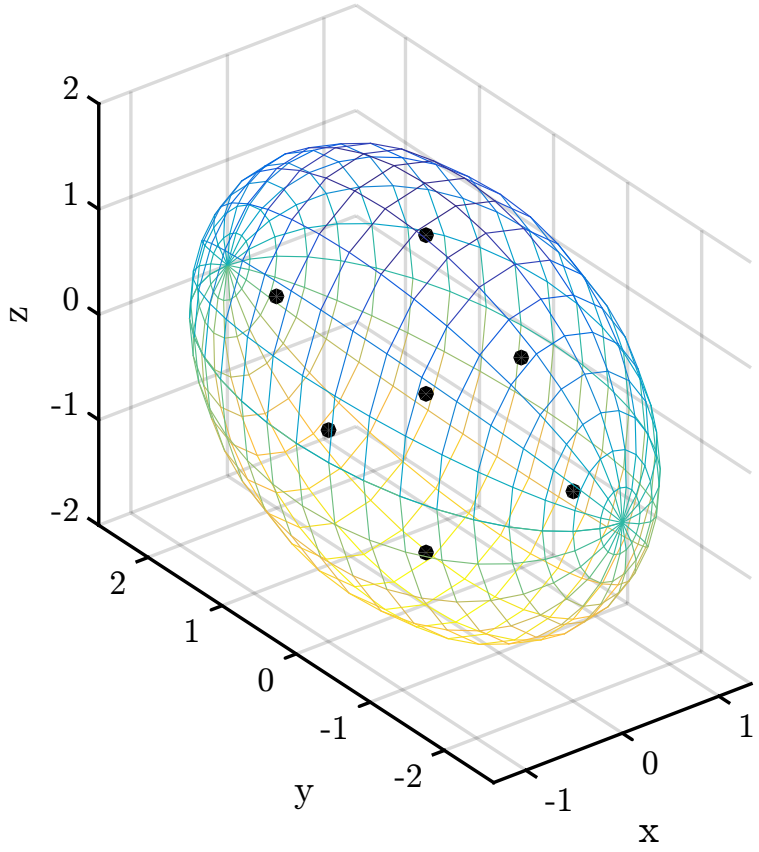


Figure A.1: Ellipsoid Computed Using MVEE Algorithm.

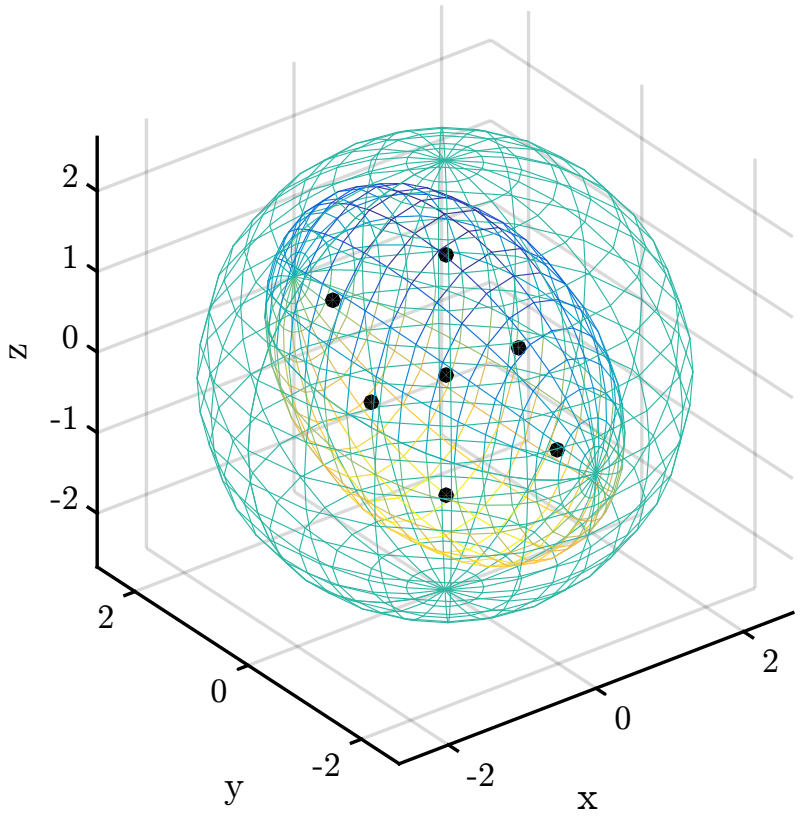


Figure A.2: Ellipsoid Computed Using MVEE Algorithm with Sphere Computed Using Isotropic Weights Super-Imposed.

APPENDIX B

MOVING LEAST-SQUARES EXAMPLES

B.1 Introduction

This appendix presents examples for computing gradients using Moving Least-Squares presented in Chapter 5. Two cases will be presented. The first will compute and compare the isotropic and anisotropic MLS basis functions and derivatives for the points used in A.5. The second case will compute and compare the isotropic and anisotropic MLS basis functions and derivatives for a scattered (unstructured) data set.

B.2 Structured Data Set

This section computes and compares the MLS basis functions and derivatives for the points used in A.5:

$$\Omega_{\mathbf{x}_I} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 1.5 \\ 0 & 0 & -1.5 \end{pmatrix} \tag{B.1}$$

The radial coefficients for the isotropic and anisotropic MLS reconstruction are: Using (5.6), the

Table B.1: Quadratic Coefficients for Radial Location of Points in $\Omega_{\mathbf{x}_I}$.

Weight	x^2	y^2	z^2	xy	xz	yz
Isotropic	0.140625	0.140625	0.140625	0	0	0
Anisotropic	0.56250	0.140625	0.250000	0	0	0

isotropic radii for the points in $\Omega_{\mathbf{x}_I}$ are:

$$\mathbf{s}_{iso} = \begin{pmatrix} 0 \\ 0.375 \\ 0.375 \\ 0.75 \\ 0.75 \\ 0.5625 \\ 0.5625 \end{pmatrix} \tag{B.2}$$

and the anisotropic radii are

$$\mathbf{s}_{ani} = \begin{pmatrix} 0 \\ 0.75 \\ 0.75 \\ 0.75 \\ 0.75 \\ 0.75 \\ 0.75 \end{pmatrix}. \tag{B.3}$$

The weights, using the Wendland function (5.7), are:

$$\mathbf{w}_{iso} = \begin{pmatrix} 1 \\ 0.38147 \\ 0.38147 \\ 0.015625 \\ 0.015625 \\ 0.119068 \\ 0.119068 \end{pmatrix} \tag{B.4}$$

and

$$\mathbf{w}_{ani} = \begin{pmatrix} 1 \\ 0.015625 \\ 0.015625 \\ 0.015625 \\ 0.015625 \\ 0.015625 \\ 0.015625 \end{pmatrix}. \tag{B.5}$$

The derivatives of s for the isotropic and anisotropic weights are:

$$\left(\frac{\partial \mathbf{s}}{\partial \mathbf{x}}\right)_{iso} = \left(\frac{\partial \mathbf{s}}{\partial x}, \frac{\partial \mathbf{s}}{\partial y}, \frac{\partial \mathbf{s}}{\partial z}\right) = \begin{pmatrix} 0 & 0 & 0 \\ 0.375 & 0 & 0 \\ -0.375 & 0 & 0 \\ 0 & 0.375 & 0 \\ 0 & -0.375 & 0 \\ 0 & 0 & 0.375 \\ 0 & 0 & -0.375 \end{pmatrix} \tag{B.6}$$

and

$$\left(\frac{\partial \mathbf{s}}{\partial \mathbf{x}}\right)_{ani} = \left(\frac{\partial \mathbf{s}}{\partial x}, \frac{\partial \mathbf{s}}{\partial y}, \frac{\partial \mathbf{s}}{\partial z}\right) = \begin{pmatrix} 0 & 0 & 0 \\ 0.75 & 0 & 0 \\ -0.75 & 0 & 0 \\ 0 & 0.375 & 0 \\ 0 & -0.375 & 0 \\ 0 & 0 & 0.5 \\ 0 & 0 & -0.5 \end{pmatrix} \tag{B.7}$$

The derivatives of the weights $\mathbf{w}$ are then:

$$\left(\frac{\partial \mathbf{w}}{\partial \mathbf{x}}\right)_{iso} = \left(\frac{\partial \mathbf{w}}{\partial x}, \frac{\partial \mathbf{w}}{\partial y}, \frac{\partial \mathbf{w}}{\partial z}\right) = \begin{pmatrix} 0 & 0 & 0 \\ -0.686648 & 0 & 0 \\ 0.686646 & 0 & 0 \\ 0 & -0.0878906 & 0 \\ 0 & 0.0878906 & 0 \\ 0 & 0 & -0.353279 \\ 0 & 0 & 0.375279 \end{pmatrix} \tag{B.8}$$

and

$$\left(\frac{\partial \mathbf{w}}{\partial \mathbf{x}}\right)_{ani} = \left(\frac{\partial \mathbf{w}}{\partial x}, \frac{\partial \mathbf{w}}{\partial y}, \frac{\partial \mathbf{w}}{\partial z}\right) = \begin{pmatrix} 0 & 0 & 0 \\ -0.175781 & 0 & 0 \\ 0.175781 & 0 & 0 \\ 0 & -0.0878906 & 0 \\ 0 & 0.0878906 & 0 \\ 0 & 0 & -0.117187 \\ 0 & 0 & -0.117187 \end{pmatrix}. \tag{B.9}$$

The polynomial matrix $\mathbf{P}$ for both the isotropic and anisotropic cases are:

$$\mathbf{P} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0.75 & -0.75 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.5 & -1.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.125 & -1.125 \end{pmatrix} \tag{B.10}$$

The polynomial and it's derivatives at $(0, 0, 0)$ for both cases are

$$\mathbf{p}(\mathbf{0}) = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}, \tag{B.11}$$

$$\frac{\partial \mathbf{p}}{\partial x}(\mathbf{0}) = \begin{pmatrix} 0 & 0.75 & 0 & 0 \end{pmatrix}, \tag{B.12}$$

$$\frac{\partial \mathbf{p}}{\partial y}(\mathbf{0}) = \begin{pmatrix} 0 & 0 & 0.75 & 0 \end{pmatrix}, \tag{B.13}$$

and

$$\frac{\partial \mathbf{p}}{\partial z}(\mathbf{0}) = \begin{pmatrix} 0 & 0 & 0 & 0.75 \end{pmatrix}. \tag{B.14}$$

With the necessary preliminary calculations complete, the matrices can now be determined. Using the standard approach to compute $\mathbf{C}$ (5.20) and $\frac{\partial \mathbf{C}}{\partial \mathbf{x}}$ (5.52) required for $\mathbf{\Psi}$ (5.19) and $\frac{\partial \mathbf{\Psi}}{\partial \mathbf{x}}$ (5.47), the moment matrix $\mathbf{M}$ (5.52) for the isotropic and anisotropic cases is:

$$\mathbf{M}_{iso} = \mathbf{P}\mathbf{W}_{iso}\mathbf{P}^T = \begin{pmatrix} 2.03233 & 0 & 0 & 0 \\ 0 & 0.429154 & 0 & 0 \\ 0 & 0 & 0.0703125 & 0 \\ 0 & 0 & 0 & 0.301391 \end{pmatrix} \tag{B.15}$$

and

$$\mathbf{M}_{ani} = \mathbf{P}\mathbf{W}_{ani}\mathbf{P}^T = \begin{pmatrix} 1.09375 & 0 & 0 & 0 \\ 0 & 0.0175781 & 0 & 0 \\ 0 & 0 & 0.0703125 & 0 \\ 0 & 0 & 0 & 0.0395508 \end{pmatrix}. \tag{B.16}$$

The inverse of the moment matrix $\mathbf{M}^{-1}$ for both cases is

$$\mathbf{M}_{iso}^{-1} = \begin{pmatrix} 0.492047 & 0 & 0 & 0 \\ 0 & 2.33017 & 0 & 0 \\ 0 & 0 & 14.2222 & 0 \\ 0 & 0 & 0 & 3.31794 \end{pmatrix} \tag{B.17}$$

and

$$\mathbf{M}_{ani}^{-1} = \begin{pmatrix} 0.914286 & 0 & 0 & 0 \\ 0 & 56.8889 & 0 & 0 \\ 0 & 0 & 14.2222 & 0 \\ 0 & 0 & 0 & 25.2839 \end{pmatrix}. \tag{B.18}$$

$\mathbf{C}$ (5.20) for the isotropic and anisotropic case is:

$$\mathbf{C}_{iso} = \mathbf{M}_{iso}^{-1}\mathbf{P}\mathbf{W}_{iso} =$$

$$\begin{pmatrix} 0.492047 & 0.187701 & 0.187701 & 0.00768824 & 0.00768824 & 0.0585871 & 0.058571 \\ 0 & 0.666667 & -0.666667 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.333333 & -0.333333 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.444444 & -0.444444 \end{pmatrix}$$

and

$$\mathbf{C}_{ani} = \mathbf{M}_{ani}^{-1}\mathbf{P}\mathbf{W}_{ani} =$$

$$\begin{pmatrix} 0.914286 & 0.0142857 & 0.0142857 & 0.0142857 & 0.0142857 & 0.0142857 & 0.0142857 \\ 0 & 0.666667 & -0.666667 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.333333 & -0.333333 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.444444 & -0.444444 \end{pmatrix}.$$

The MLS basis vector $\mathbf{\Psi}$ (5.19) for the isotropic and anisotropic weights is:

$$\mathbf{\Psi}_{iso} = \mathbf{p}(\mathbf{0})\mathbf{C}_{iso} = \begin{pmatrix} 0.492047 \\ 0.187701 \\ 0.187701 \\ 0.00768824 \\ 0.00768824 \\ 0.0585871 \\ 0.0585871 \end{pmatrix} \tag{B.19}$$

and

$$\mathbf{\Psi}_{ani} = \mathbf{p}(\mathbf{0})\mathbf{C}_{ani} = \begin{pmatrix} 0.914286 \\ 0.0142857 \\ 0.0142857 \\ 0.0142857 \\ 0.0142857 \\ 0.0142857 \\ 0.0142857 \end{pmatrix}. \tag{B.20}$$

The isotropic MLS basis function $\mathbf{\Psi}$ (B.19) is very different from the anisotropic MLS basis function (B.20). This difference is directly attributed to the difference between the radii (and description thereof) of the isotropic (B.28) and anisotropic (B.29). While $\sum_{j \in \Omega_{\mathbf{x}_I}} \Psi_j(\mathbf{x}_I) = 1$ for both cases, the anisotropic case is better at determining value at the node since it does not heavily weight the connected nodes in $\Omega_{\mathbf{x}_I}$.

The derivatives of $\mathbf{C}$ for the isotropic case are:

$$\frac{\partial \mathbf{C}}{\partial x}_{iso} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1.18091 & -0.749517 & -0.749517 & 0.0184518 & 0.0184518 & 0.140609 & 0.140609 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\frac{\partial \mathbf{C}}{\partial y}_{iso} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1.84518 & 0.703879 & 0.703879 & -1.84617 & -1.84617 & 0.219702 & 0.219702 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

and

$$\frac{\partial \mathbf{C}}{\partial z}_{iso} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1.29771 & 0.495036 & 0.0202767 & 0.0202767 & -1.16417 & -1.16417 & \end{pmatrix}$$

The derivatives of $\mathbf{C}$ for the anisotropic case are:

$$\frac{\partial \mathbf{C}}{\partial x}_{ani} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 13.7143 & -7.28571 & -7.28571 & 0.214286 & 0.214286 & 0.214286 & 0.214286 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\frac{\partial \mathbf{C}}{\partial y}_{ani} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3.42857 & 0.0535714 & 0.0535714 & -1.82143 & -1.82143 & 0.0535714 & 0.0535714 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

and

$$\frac{\partial \mathbf{C}}{\partial z}_{ani} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6.09524 & 0.0952381 & 0.0952381 & 0.0952381 & 0.0952381 & -3.23809 & -3.23809 \end{pmatrix}$$

The derivatives of the MLS basis function $\frac{\partial \mathbf{\Psi}}{\partial \mathbf{x}}$ (5.47) for the isotropic case are:

$$\frac{\partial \mathbf{\Psi}}{\partial x}_{iso} = \frac{\partial \mathbf{p}^T(\mathbf{0})}{\partial x}\mathbf{C} + \mathbf{p}^T(\mathbf{0})\frac{\partial \mathbf{C}}{\partial x} = \begin{pmatrix} 0 \\ 0.5 \\ -0.5 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{B.21}$$

$$\frac{\partial \mathbf{\Psi}}{\partial y}_{iso} = \frac{\partial \mathbf{p}^T(\mathbf{0})}{\partial y}\mathbf{C} + \mathbf{p}^T(\mathbf{0})\frac{\partial \mathbf{C}}{\partial y} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0.25 \\ -0.25 \\ 0 \\ 0 \end{pmatrix}, \tag{B.22}$$

and

$$\frac{\partial \mathbf{\Psi}}{\partial z}_{iso} = \frac{\partial \mathbf{p}^T(\mathbf{0})}{\partial z}\mathbf{C} + \mathbf{p}^T(\mathbf{0})\frac{\partial \mathbf{C}}{\partial z} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.333333 \\ -0.333333 \end{pmatrix} \tag{B.23}$$

The derivatives of the MLS basis function $\frac{\partial \mathbf{\Psi}}{\partial \mathbf{x}}$ (5.47) for the anisotropic case are:

$$\frac{\partial \mathbf{\Psi}}{\partial x}_{ani} = \frac{\partial \mathbf{p}^T(\mathbf{0})}{\partial x}\mathbf{C} + \mathbf{p}^T(\mathbf{0})\frac{\partial \mathbf{C}}{\partial x} = \begin{pmatrix} 0 \\ 0.5 \\ -0.5 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{B.24}$$

$$\frac{\partial \mathbf{\Psi}}{\partial y}_{ani} = \frac{\partial \mathbf{p}^T(\mathbf{0})}{\partial y}\mathbf{C} + \mathbf{p}^T(\mathbf{0})\frac{\partial \mathbf{C}}{\partial y} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0.25 \\ -0.25 \\ 0 \\ 0 \end{pmatrix}, \tag{B.25}$$

and

$$\frac{\partial \mathbf{\Psi}}{\partial z}_{ani} = \frac{\partial \mathbf{p}^T(\mathbf{0})}{\partial z}\mathbf{C} + \mathbf{p}^T(\mathbf{0})\frac{\partial \mathbf{C}}{\partial z} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.333333 \\ -0.333333 \end{pmatrix} \tag{B.26}$$

Unlike the MLS basis function $\mathbf{\Psi}$, the derivatives of $\mathbf{\Psi}$ for the isotropic and anisotropic case are equal, so there would be no difference in the derivatives computed using either method. $\sum_{j \in \Omega_{\mathbf{x}_I}} \frac{\partial \Psi_j}{\partial \mathbf{x}} = 0$ for both cases.

B.3 Unstructured Data Set

This section computes and compares the MLS basis functions and derivatives for an scattered, or unstructured data set that is more common in unstructured CFD grids. It may also be present in boundary stencils that include ghost and image nodes. The collection of points $\Omega_{\mathbf{x}_I}$ used here is:

$$\Omega_{\mathbf{x}_I} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -4 & 0 \\ -2 & 1.75 & 0 \\ 2 & -1.75 & 0 \\ 0 & 0 & -0.5 \\ 0 & 0 & 0.5 \\ 1.5 & 3.0 & 0 \\ -3.0 & -5.5 & 0 \\ -1.5 & 3.0 & 0 \\ 3.0 & 5.5 & 0 \end{pmatrix} \tag{B.27}$$

The radial coefficients for the isotropic and anisotropic MLS reconstruction are:

Table B.2: Quadratic Coefficients for Radial Location of Points in $\Omega_{\mathbf{x}_I}$.

Weight	x^2	y^2	z^2	xy	xz	yz
Isotropic	0.01433129	0.01433129	0.01433129	0	0	0
Anisotropic	0.07419659	0.02126654	2.25	-0.02268431	0	0

Using (5.6), the isotropic radii for the points in $\Omega_{\mathbf{x}_I}$ are:

$$\mathbf{s}_{iso} = \begin{pmatrix} 0 \\ 0.478852 \\ 0.318142 \\ 0.318142 \\ 0.0598565 \\ 0.0598565 \\ 0.40153 \\ 0.75 \\ 0.40153 \\ 0.75 \end{pmatrix} \tag{B.28}$$

and the anisotropic radii are

$$\mathbf{s}_{ani} = \begin{pmatrix} 0 \\ 0.583322 \\ 0.721599 \\ 0.721599 \\ 0.75 \\ 0.75 \\ 0.392661 \\ 0.75 \\ 0.75 \\ 0.75 \end{pmatrix}. \tag{B.29}$$

The weights, using the Wendland function (5.7), are for the isotropic case are:

$$\mathbf{w}_{iso} = \begin{pmatrix} 1 \\ 0.215052 \\ 0.491239 \\ 0.491239 \\ 0.968272 \\ 0.968272 \\ 0.334322 \\ 0.015625 \\ 0.334322 \\ 0.015625 \end{pmatrix} \tag{B.30}$$

The anisotropic weights are:

$$\mathbf{w}_{ani} = \begin{pmatrix} 1 \\ 0.100479 \\ 0.0233469 \\ 0.0233469 \\ 0.015625 \\ 0.015625 \\ 0.349758 \\ 0.015625 \\ 0.015625 \\ 0.015625 \end{pmatrix}. \tag{B.31}$$

The derivatives of s for the isotropic and anisotropic weights are:

$$\left(\frac{\partial \mathbf{s}}{\partial \mathbf{x}}\right)_{iso} = \left(\frac{\partial \mathbf{s}}{\partial x}, \frac{\partial \mathbf{s}}{\partial y}, \frac{\partial \mathbf{s}}{\partial z}\right) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -0.119713 & 0 \\ -0.0900932 & 0.0788316 & 0 \\ 0.0900932 & -0.0788316 & 0 \\ 0 & 0 & -0.11973 \\ 0 & 0 & 0.11973 \\ 0.0535373 & 0.107075 & 0 \\ -0.0573248 & -0.105096 & 0 \\ -0.0535373 & 0.107075 & \\ 0.0573248 & 0.105096 & \end{pmatrix} \tag{B.32}$$

and

$$\left(\frac{\partial \mathbf{s}}{\partial \mathbf{x}}\right)_{ani} = \left(\frac{\partial \mathbf{s}}{\partial x}, \frac{\partial \mathbf{s}}{\partial y}, \frac{\partial \mathbf{s}}{\partial z}\right) = \begin{pmatrix} 0 & 0 & 0 \\ 0.155553 & -0.145831 & 0 \\ -0.260658 & 0.114447 & 0 \\ 0.260658 & -0.114447 & 0 \\ 0 & 0 & -1.5 \\ 0 & 0 & 1.5 \\ 0.110126 & 0.0758241 & 0 \\ -0.130435 & -0.0652174 & 0 \\ -0.23913 & 0.130435 & 0 \\ 0.130435 & 0.0652174 & 0 \end{pmatrix} \tag{B.33}$$

The derivatives of the weights $\mathbf{w}$ are

$$\left(\frac{\partial \mathbf{w}}{\partial \mathbf{x}}\right)_{iso} = \left(\frac{\partial \mathbf{w}}{\partial x}, \frac{\partial \mathbf{w}}{\partial y}, \frac{\partial \mathbf{w}}{\partial z}\right) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0.162277 & 0 \\ 0.181729 & -0.159013 & 0 \\ -0.181729 & 0.159013 & 0 \\ 0 & 0 & 0.119087 \\ 0 & 0 & -0.119087 \\ -0.0921578 & -0.184316 & 0 \\ 0.0134355 & 0.0246318 & 0 \\ 0.0921578 & -0.184316 & 0 \\ -0.0134355 & -0.0246318 & 0 \end{pmatrix} \tag{B.34}$$

for the isotropic case and

$$\left(\frac{\partial \mathbf{w}}{\partial \mathbf{x}}\right)_{ani} = \left(\frac{\partial \mathbf{w}}{\partial x}, \frac{\partial \mathbf{w}}{\partial y}, \frac{\partial \mathbf{w}}{\partial z}\right) = \begin{pmatrix} 0 & 0 & 0 \\ -0.131286 & 0.12308 & 0 \\ 0.0811725 & -0.0356404 & 0 \\ -0.0811725 & 0.0356404 & 0 \\ 0 & 0 & 0.351562 \\ 0 & 0 & -0.351562 \\ -0.193745 & -0.133398 & 0 \\ 0.0305707 & 0.0152853 & 0 \\ 0.0560462 & -0.0305707 & 0 \\ -0.0305707 & -0.0152853 & 0 \end{pmatrix}. \tag{B.35}$$

for the anisotropic case. The polynomial matrix $\mathbf{P}$ for the isotropic and anisotropic cases is:

$$\mathbf{P} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & -0.239426 & 0.239426 & 0 \\ 0 & -0.478852 & 0.209498 & -0.209498 & 0 \\ 0 & 0 & 0 & 0 & -0.0598565 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0.17957 & -0.359139 & -0.17957 & 0.359139 \\ 0 & 0.359139 & -0.658422 & 0.359139 & 0.658422 \\ 0.0598565 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{B.36}$$

and

$$\mathbf{P} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & -0.226937 & 0.226937 & 0 \\ 0 & -0.453873 & 0.19857 & -0.19857 & 0 \\ 0 & 0 & 0 & 0 & -0.0567342 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0.170203 & -0.340405 & -0.170203 & 0.340405 \\ 0 & 0.340405 & -0.624076 & 0.340405 & 0.624076 \\ 0.0567342 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{B.37}$$

The differences between the isotropic and anisotropic case in $\mathbf{P}$ is due to the difference in the maximum radii computed by each method. The maximum radius for the isotropic case is $r_{\max} = 8.3533$, whereas for the anisotropic case the maximum radius is $r_{\max} = 8.8130$. It is this difference that will attribute to differences in $\frac{\partial \mathbf{\Psi}}{\partial \mathbf{x}}$ seen later on. The polynomial and it's derivatives at $(0, 0, 0)$ for the isotropic case are

$$\mathbf{p}(\mathbf{0}) = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}, \tag{B.38}$$

$$\frac{\partial \mathbf{p}}{\partial x}(\mathbf{0}) = \begin{pmatrix} 0 & 0.119713 & 0 & 0 \end{pmatrix}, \tag{B.39}$$

$$\frac{\partial \mathbf{p}}{\partial y}(\mathbf{0}) = \begin{pmatrix} 0 & 0 & 0.119713 & 0 \end{pmatrix}, \tag{B.40}$$

and

$$\frac{\partial \mathbf{p}}{\partial z}(\mathbf{0}) = \begin{pmatrix} 0 & 0 & 0 & 0.119713 \end{pmatrix}. \tag{B.41}$$

The polynomial and it's derivatives at $(0, 0, 0)$ for the anisotropic case are

$$\mathbf{p}(\mathbf{0}) = \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}, \tag{B.42}$$

$$\frac{\partial \mathbf{p}}{\partial x}(\mathbf{0}) = \begin{pmatrix} 0 & 0.113468 & 0 & 0 \end{pmatrix}, \tag{B.43}$$

$$\frac{\partial \mathbf{p}}{\partial x}(\mathbf{0}) = \begin{pmatrix} 0 & 0 & 0.113468 & 0 \end{pmatrix}, \tag{B.44}$$

and

$$\frac{\partial \mathbf{p}}{\partial x}(\mathbf{0}) = \begin{pmatrix} 0 & 0 & 0 & 0.113468 \end{pmatrix}. \tag{B.45}$$

As in the previous section, the standard approach is used to compute $\mathbf{C}$ (5.20) and $\frac{\partial \mathbf{C}}{\partial \mathbf{x}}$ (5.52) required for $\mathbf{\Psi}$ (5.19) and $\frac{\partial \mathbf{\Psi}}{\partial \mathbf{x}}$ (5.47). The moment matrix $\mathbf{M}$ (5.52) for the isotropic and anisotropic cases is:

$$\mathbf{M}_{iso} = \mathbf{P}\mathbf{W}_{iso}\mathbf{P}^T = \begin{pmatrix} 4.83397 & 0 & 0.137158 & 0 \\ 0 & 0.0819116 & -0.0418908 & 0 \\ 0.137158 & -0.0418908 & 0.192221 & 0 \\ 0 & 0 & 0 & 0.00693825 \end{pmatrix} \tag{B.46}$$

and

$$\mathbf{M}_{ani} = \mathbf{P}\mathbf{W}_{ani}\mathbf{P}^T = \begin{pmatrix} 1.57506 & 0.0568702 & 0.0787734 & 0 \\ 0.0568702 & 0.0166106 & 0.0238935 & 0 \\ 0.0787734 & 0.0238935 & 0.0770498 & 0 \\ 0 & 0 & 0 & 0.000100586 \end{pmatrix}. \tag{B.47}$$

The inverse of the moment matrix $\mathbf{M}^{-1}$ for the isotropic case is

$$\mathbf{M}_{iso}^{-1} = \begin{pmatrix} 0.211693 & -0.0869398 & -0.169999 & 0 \\ -0.0869398 & 13.7753 & 3.06409 & 0 \\ -0.169999 & 3.06409 & 5.9914 & 0 \\ 0 & 0 & 0 & 144.129 \end{pmatrix} \tag{B.48}$$

and for the anisotropic,

$$\mathbf{M}_{ani}^{-1} = \begin{pmatrix} 0.724569 & -2.55476 & 0.0514637 & 0 \\ -2.55476 & 117.69 & -33.8843 & 0 \\ 0.0514637 & -33.8843 & 23.4337 & 0 \\ 0 & 0 & 0 & 9941.69 \end{pmatrix}. \tag{B.49}$$

$\mathbf{C}$ (5.20) for the isotropic and anisotropic case is:

$$\mathbf{C}_{iso} = \mathbf{M}_{iso}^{-1}\mathbf{P}\mathbf{W}_{iso} = \begin{pmatrix} 0.211693 & 0.0630311 & 0.0967222 & 0.111262 & 0.204976 \\ -0.0869398 & -0.334231 & -1.34756 & 1.26214 & -0.0841814 \\ -0.169999 & -0.653541 & 0.172701 & -0.339721 & -0.164605 \\ 0 & 0 & 0 & 0 & -8.35331 \end{pmatrix}$$
$$\begin{pmatrix} 0.204976 & 0.0451428 & 0.00554449 & 0.0555815 & 0.00107092 \\ -0.0841814 & 1.16582 & -0.110182 & -0.488153 & 0.107465 \\ -0.164605 & 0.84649 & -0.081489 & 0.478591 & 0.0761766 \\ 8.35331 & 0 & 0 & 0 & 0 \end{pmatrix}$$

and

$$\mathbf{C}_{ani} = \mathbf{M}_{ani}^{-1}\mathbf{P}\mathbf{W}_{ani} = \begin{pmatrix} 0.724569 & 0.0704569 & 0.0306908 & 0.00314206 & 0.0113214 \\ -2.55476 & 1.28858 & -0.840287 & 0.720996 & -0.0399181 \\ 0.0514637 & -1.06351 & 0.289368 & -0.286965 & 0.00080412 \\ 0 & 0 & 0 & 0 & -8.81303 \end{pmatrix}$$
$$\begin{pmatrix} 0.0113214 & 0.107467 & 0.0244079 & 0.0183893 & -0.0017651 \\ -0.0399181 & 2.07827 & -0.33548 & -0.53313 & 0.255644 \\ 0.00080412 & 0.790875 & -0.0474771 & 0.215556 & 0.0490854 \\ 8.81303 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The MLS basis vector $\mathbf{\Psi}$ (5.19) for the isotropic and anisotropic weights is:

$$\mathbf{\Psi}_{iso} = \mathbf{p}(\mathbf{0})\mathbf{C}_{iso} = \begin{pmatrix} 0.211693 \\ 0.0630311 \\ 0.0967222 \\ 0.111262 \\ 0.204976 \\ 0.204976 \\ 0.0451428 \\ 0.00554449 \\ 0.0555815 \\ 0.00107092 \end{pmatrix} \tag{B.50}$$

and

$$\mathbf{\Psi}_{ani} = \mathbf{p}(\mathbf{0})\mathbf{C}_{ani} = \begin{pmatrix} 0.724569 \\ 0.0704569 \\ 0.0306908 \\ 0.00314206 \\ 0.0113214 \\ 0.0113214 \\ 0.107467 \\ 0.0244079 \\ 0.0183893 \\ -0.0017651 \end{pmatrix}. \tag{B.51}$$

As with the previous case, the anisotropic MLS basis function $\mathbf{\Psi}$ (B.51) places more emphasis on the center node in $\Omega_{\mathbf{x}_I}$. For the isotropic case, (B.50) places nearly equal weight on the fifth and sixth nodes in $\Omega_{\mathbf{x}_I}$ as it does the center node, which is not optimal. However, all $\mathbf{\Psi}_{iso}$ are positive, while the last node in $\Omega_{\mathbf{x}_I}$ has a negative MLS basis function. While it has never been reported,

the theory of MLS does not preclude a negative basis function. It is still more appropriate to use the MLS anisotropic basis function for fitting, though as in the previous case $\sum_{j\in\Omega_{\mathbf{x}_I}} \Psi_j(\mathbf{x}_I) = 1$ for both cases.

The derivatives of $\mathbf{C}$ for the isotropic case are:

$$\frac{\partial \mathbf{C}}{\partial x}_{iso} = \begin{pmatrix} -0.000218299 & 0.000170961 & -0.000960266 & -0.00463291 & -0.000211373 \\ 0.309625 & -0.0173489 & -0.34766 & -0.313576 & 0.299801 \\ -0.00211605 & -0.0437377 & 0.0102404 & 0.177247 & -0.00204891 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} -0.000211373 & 0.00581711 & 0.00304843 & -0.00359373 & 0.000791445 \\ 0.299801 & -0.0765371 & -0.102351 & 0.0233666 & -0.0751218 \\ -0.00204891 & -0.13465 & -0.0790011 & 0.132753 & -0.0566373 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\frac{\partial \mathbf{C}}{\partial y}_{iso} = \begin{pmatrix} -0.010464 & 0.0101813 & -0.00260924 & -0.00296506 & -0.010132 \\ 0.0107922 & -0.30394 & 0.221541 & 0.633819 & 0.0104498 \\ 0.341152 & -0.411525 & 0.0416867 & 0.127618 & 0.330328 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} -0.010132 & 0.0132233 & 0.00500673 & 0.00617251 & 0.00171855 \\ 0.0104498 & -0.43625 & -0.191998 & 0.195908 & -0.150771 \\ 0.330328 & -0.330752 & -0.125251 & -0.190948 & -0.112637 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

and

$$\frac{\partial \mathbf{C}}{\partial z}_{iso} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0.434974 & 0.129512 & 0.198739 & 0.228613 & -0.606197 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -0.606197 & 0.0927567 & 0.0113925 & 0.0114205 & 0.00220045 \end{pmatrix}.$$

The derivatives of $\mathbf{C}$ for the anisotropic case are:

$$\frac{\partial \mathbf{C}}{\partial x}_{ani} = \begin{pmatrix} -0.207795 & 0.0271745 & 0.040289 & 0.0457898 & -0.0032468 \\ 8.93163 & -1.52226 & -2.26484 & -2.74638 & 0.139557 \\ -3.07232 & 0.859919 & 0.871297 & 0.989039 & -0.048005 \\ 0 & 0 & 0 & 0 & 0 \\ -0.0032468 & 0.0337003 & 0.0269607 & 0.0226206 & 0.0177537 \\ 0.139557 & -0.300309 & -0.299424 & -1.49953 & -0.578008 \\ -0.048005 & 0.0249793 & -0.273985 & 0.708035 & -0.010952 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\frac{\partial \mathbf{C}}{\partial y}_{ani} = \begin{pmatrix} 0.0182596 & -0.0126146 & -0.0179523 & -0.0232499 & 0.000285306 \\ -3.44084 & 1.16305 & 1.05436 & 1.16837 & -0.0537631 \\ 2.2093 & -0.752931 & -0.431574 & -0.345072 & 0.0345203 \\ 0 & 0 & 0 & 0 & 0 \\ 0.000285306 & 0.0236042 & 0.0209205 & -0.0147922 & 0.00525409 \\ -0.0537631 & -0.0711793 & -0.552584 & 0.919571 & -0.133215 \\ 0.0345203 & -0.293825 & 0.0500454 & -0.429515 & -0.0754681 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

and

$$\frac{\partial \mathbf{C}}{\partial z}_{ani} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 287.354 & 27.9422 & 12.1716 & 1.2461 & -193.803 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -193.803 & 42.6199 & 9.67983 & 7.29293 & -0.700015 \end{pmatrix}.$$

The derivatives of the MLS basis function $\frac{\partial \mathbf{\Psi}}{\partial \mathbf{x}}$ (5.47) for the isotropic case are:

$$\frac{\partial \mathbf{\Psi}}{\partial x}_{iso} = \frac{\partial \mathbf{p}^T(\mathbf{0})}{\partial x}\mathbf{C} + \mathbf{p}^T(\mathbf{0})\frac{\partial \mathbf{C}}{\partial x} = \begin{pmatrix} -0.0106261 \\ -0.0398408 \\ -0.162281 \\ 0.146462 \\ -0.010289 \\ -0.010289 \\ 0.145381 \\ -0.0101418 \\ -0.062032 \\ 0.0136564 \end{pmatrix}, \tag{B.52}$$

$$\frac{\partial \mathbf{\Psi}}{\partial y}_{iso} = \frac{\partial \mathbf{p}^T(\mathbf{0})}{\partial y}\mathbf{C} + \mathbf{p}^T(\mathbf{0})\frac{\partial \mathbf{C}}{\partial y} = \begin{pmatrix} -0.0308151 \\ -0.0680561 \\ 0.0180654 \\ -0.0436341 \\ -0.0298373 \\ -0.0298373 \\ 0.114559 \\ -0.00474857 \\ 0.0634661 \\ 0.0108379 \end{pmatrix}, \tag{B.53}$$

and

$$\frac{\partial \mathbf{\Psi}}{\partial z}_{iso} = \frac{\partial \mathbf{p}^T(\mathbf{0})}{\partial z}\mathbf{C} + \mathbf{p}^T(\mathbf{0})\frac{\partial \mathbf{C}}{\partial z} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \tag{B.54}$$

The derivatives of the MLS basis function $\frac{\partial \mathbf{\Psi}}{\partial \mathbf{x}}$ (5.47) for the anisotropic case are:

$$\frac{\partial \mathbf{\Psi}}{\partial x}_{ani} = \frac{\partial \mathbf{p}^T(\mathbf{0})}{\partial x}\mathbf{C} + \mathbf{p}^T(\mathbf{0})\frac{\partial \mathbf{C}}{\partial x} = \begin{pmatrix} -0.497679 \\ 0.173388 \\ -0.055057 \\ 0.1276 \\ -0.00777624 \\ -0.00777624 \\ 0.269518 \\ -0.0111056 \\ -0.0378727 \\ 0.0467611 \end{pmatrix}, \tag{B.55}$$

$$\frac{\partial \mathbf{\Psi}}{\partial y}_{ani} = \frac{\partial \mathbf{p}^T(\mathbf{0})}{\partial y}\mathbf{C} + \mathbf{p}^T(\mathbf{0})\frac{\partial \mathbf{C}}{\partial y} = \begin{pmatrix} 0.0240991 \\ -0.13329 \\ 0.0148818 \\ -0.0558114 \\ 0.000376548 \\ 0.000376548 \\ 0.113344 \\ 0.0155334 \\ 0.00966663 \\ 0.0108237 \end{pmatrix}, \tag{B.56}$$

and

$$\frac{\partial \mathbf{\Psi}}{\partial z}_{ani} = \frac{\partial \mathbf{p}^T(\mathbf{0})}{\partial z}\mathbf{C} + \mathbf{p}^T(\mathbf{0})\frac{\partial \mathbf{C}}{\partial z} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \tag{B.57}$$

Unlike the previous case, only $\frac{\mathbf{\Psi}}{\partial z}_{iso} = \frac{\mathbf{\Psi}}{\partial z}_{ani}$. The other derivatives $\mathbf{\Psi}$ are not. $\frac{\mathbf{\Psi}}{\partial x}_{iso}$ and $\frac{\mathbf{\Psi}}{\partial y}_{iso}$ place more weight to the fifth and sixth nodes in $\Omega_{\mathbf{x}_I}$ than they should, since the isotropic radii for these nodes is 'closer' to the center node. $\frac{\mathbf{\Psi}}{\partial x}_{ani}$ places a larger weight to the center node than any other node in $\Omega_{\mathbf{x}_I}$, and this can be attributed to the center node not quite being in the 'center' of the collection of points in along the x-axis. $\sum_{j\in\Omega_{\mathbf{x}_I}} \frac{\partial \Psi_j}{\partial \mathbf{x}} = 0$ for both cases, so both sets of $\frac{\mathbf{\Psi}}{\partial \mathbf{x}}$ are still valid.

APPENDIX C

DERIVATIVES OF WEIGHT FUNCTIONS

This appendix presents the derivatives of the weighting function $W(s)$ in Cartesian coordinates for both the isotropic and anisotropic spline representations of $\Omega_{\mathbf{x}_I}$. Recall the spline parameter is

$$s = \frac{\sqrt{(\Delta_{jI}x)^2 + (\Delta_{jI}y)^2 + (\Delta_{jI}z)^2}}{r_{\max}} \tag{C.1}$$

for the isotropic (spherical) case or

$$s = \sqrt{\begin{array}{l} c_x(\Delta_{jI}x)^2 + c_y(\Delta_{jI}y)^2 + c_z(\Delta_{jI}z)^2 + \\ \quad 2(c_{yz}\Delta_{jI}y\Delta_{jI}z + c_{xz}\Delta_{jI}x\Delta_{jI}z + c_{xy}\Delta_{jI}x\Delta_{jI}y) \end{array}} \tag{C.2}$$

for the anisotropic (ellipsoidal) case. The first derivatives of $W(s)$ are then:

$$\frac{\partial W}{\partial x_i} = \frac{\partial W}{\partial s}\frac{\partial s}{\partial x_i}. \tag{C.3}$$

The second derivatives of $W(s)$ are:

$$\frac{\partial^2 W}{\partial x_i \partial x_j} = \frac{\partial s}{\partial x_i}\frac{\partial s}{\partial x_j}\frac{\partial^2 W}{\partial s^2} + \frac{\partial^2 s}{\partial x_i \partial x_j}\frac{\partial W}{\partial s}. \tag{C.4}$$

Finally, the third derivatives of $W(s)$ are:

$$\begin{aligned} \frac{\partial^3 W}{\partial x_i \partial x_j \partial x_k} &= \frac{\partial s}{\partial x_i}\frac{\partial s}{\partial x_j}\frac{\partial s}{\partial x_k}\frac{\partial^3 W}{\partial s^3} \\ &\quad + \frac{\partial^2 W}{\partial s^2}\left(\frac{\partial^2 s}{\partial x_i \partial x_k}\frac{\partial s}{\partial x_j} + \frac{\partial^2 s}{\partial x_j \partial x_k}\frac{\partial s}{\partial x_i} + \frac{\partial^2 s}{\partial x_i \partial x_j}\frac{\partial s}{\partial x_k}\right) \\ &\quad + \frac{\partial W}{\partial s}\frac{\partial^3 s}{\partial x_i \partial x_j \partial x_k} \end{aligned} \tag{C.5}$$

The first derivatives of s are:

$$\frac{\partial s}{\partial x_i} = \frac{1}{s}(c_i x_i + 2(c_{ik} x_k + c_{ij} x_j)) \tag{C.6}$$

One should note that the trailing terms can be dropped to give the isotropic (spherical) first derivative. The second derivatives of s are:

$$\frac{\partial^2 s}{\partial x_i \partial x_j} = \frac{\zeta_{ij}}{s} - \frac{1}{s}\frac{\partial s}{\partial x_i}\frac{\partial s}{\partial x_j} \tag{C.7}$$

where

$$\zeta_{ij} = c_i \delta_{ij} + 2(c_{ij}\delta_{jj} + c_{jk}\delta_{ki}). \tag{C.8}$$

The third derivatives of s are:

$$\frac{\partial^3 s}{\partial x_i \partial x_j \partial x_k} = \frac{3}{s^2}\frac{\partial s}{\partial x_i}\frac{\partial s}{\partial x_j}\frac{\partial s}{\partial x_k} - \frac{1}{s^2}\left(\zeta_{ij}\frac{\partial s}{\partial x_i} + \zeta_{ik}\frac{\partial s}{\partial x_j} + \zeta_{jk}\frac{\partial s}{\partial x_k}\right) \tag{C.9}$$

APPENDIX D

Affine Transformation for Image Nodes

D.1 Introduction

This appendix describes affine transformations [119], which are used when finding image nodes and for computing velocities at ghost nodes for symmetry and wall boundaries. The chapter begins with a quick definition of affine transformations. Next, an outline of how affine transformations work, using image node generation as a working example, is presented. Finally, an algorithm for affine transformations is given to generate image nodes.

D.2 Definition

Affine transformations are a class of transformations preserves points, lines, and planes through the transformation. Affine transformations are reversible, and preserve both collinearity and ratios of distances. This means that a point transformed through an affine transformation will lie on the same line it after the transformation. Additionally, sets of parallel lines remain parallel through the transformation. However, affine transforms do not necessarily preserve angles between lines or planes or distances between points.

An affine transformation in $\mathbb{R}^n$ maps any entity Σ to Ξ ($f : \Sigma \rightarrow \Xi$) via a linear transformation

$$f(\Sigma) = \mathbf{A}\Sigma + \Xi \tag{D.1}$$

where $\mathbf{A}$ is the linear transformation of $\mathbb{R}^n$. If the determinate of $\mathbf{A}$ is greater than zero, the transformation is an orientation-preserving transformation; otherwise, the transformation is orientation-reversing.

Common examples of affine transformations are translations, scalings, similarity transformations, reflections, rotations, and shears. For example, the affine rotation about the $x - axis$ in three

dimensions through the angle θ_x is

$$\mathbf{R}(\theta_x) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x & 0 \\ 0 & -\sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{D.2}$$

The affine translation matrix a distance $d\mathbf{x}$ is:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & -dx \\ 0 & 1 & 0 & -dy \\ 0 & 0 & 1 & -dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{D.3}$$

Finally, an affine reflection matrix about the y axis is of the point $\mathbf{x}$:

$$\mathbf{R}_{ref} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{D.4}$$

It is important to note the form of the affine matrices. Instead of being d x d, where d is the dimension, an affine matrix is $d+1$ x $d+1$. Affine matrices are represented by augmented matrices using homogeneous coordinates, where the linear transformation (translation, reflection, rotation, etc.) is contained within a matrix padded with zeros and one in the lower right corner. Therefore, a two-dimensional vector (x, y) is represented as $(x, y, 1)$. This technique allows for the affine transformation to be represented by a single matrix multiplication. Additionally, this means any combination of transformations can be built into a single matrix transformation to create a unique transformation matrix. For example, to combine a rotation, reflection, and translation into a single

matrix would look something like:

$$\mathbf{A} = \mathbf{T}^{-1}\mathbf{R}(-\theta_x)\mathbf{R}(\theta_z)\mathbf{R}_{ref_y}\mathbf{R}(-\theta_z)\mathbf{R}(\theta_x)\mathbf{T} \tag{D.5}$$

where $\mathbf{A}$ is the combined affine transformation. In general, the method is not commutative, since inverses are required, so ordering is important to build the proper transform.

D.3 Method

Affine transformations are used herein to generate image nodes. The particular transformation used is a reflection. The ghost node is reflected about a plane into the interior to generate the image node. In particular, an affine reflection works by rotating a vector through a plane. The plane of reflection need not be one of the coordinate planes. However, the reflection plane will be transformed into one of the coordinate planes, typically the $xy-$plane. The process of generating an image node is illustrated in Figure D.1. To generate the image node, the ghost node $\mathbf{x}_g$ and it's related boundary node $\mathbf{x}_b$ are used to construct the vector

$$d\mathbf{x}_{bg} = \mathbf{x}_b - \mathbf{x}_g. \tag{D.6}$$

Additionally, the nodal normal $\mathbf{n}$ at the boundary node is used to define the plane of reflection (recall a normal and point are all that is required to define a plane). The boundary plane p in Figure D.1 (a), can be described in normal-distance form as:

$$p(1)x + p(2)y + p(3)z + p(4) = d\mathbf{x}_{bg} \cdot \mathbf{x} + d\mathbf{x}_{bg} \cdot -\mathbf{x}_b = 0. \tag{D.7}$$

where $\mathbf{p} = \mathbf{n}$. With the plane constructed, the next step to build the affine transformation is to determine the rotation matrices. For the general boundary plane, the affine reflection requires two rotations. In this dissertation, the rotations chosen to be about the $x-$axis and $y-$axis. The angles of rotation, θ_x and θ_y, are determined from the position of the boundary point relative to origin. Note that the boundary node $\mathbf{x}_b$ is assumed to lie at the origin without any loss in generality, and

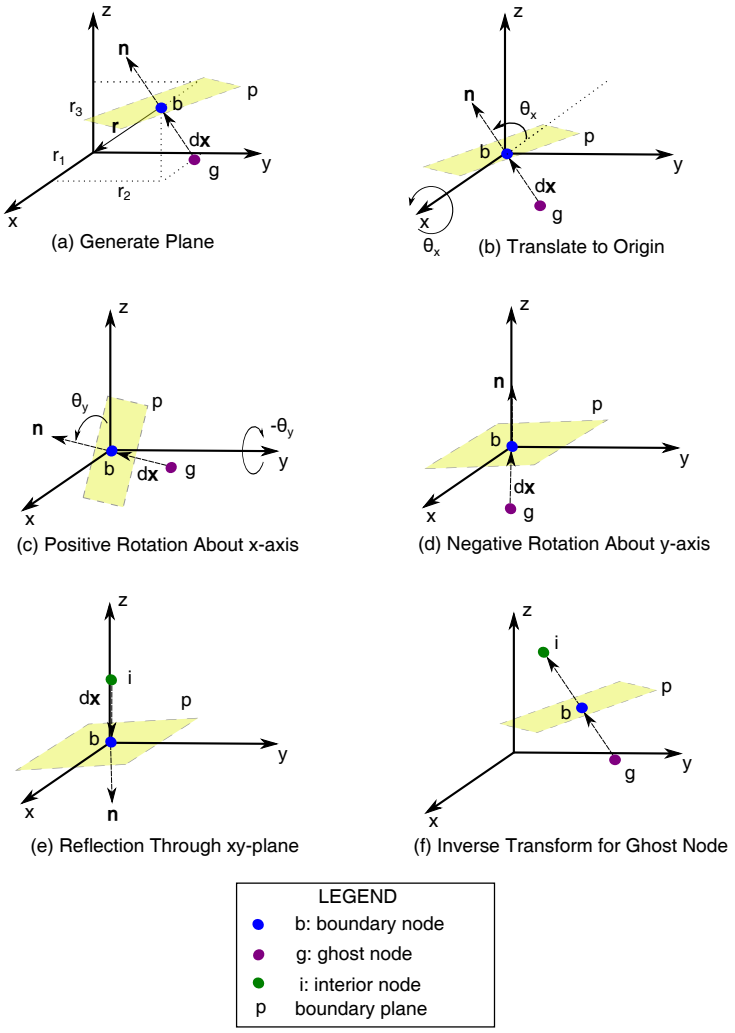


Figure D.1: Process of Generating an Image Node Using an Affine Transformation.

this assumption comes from defining the vector (D.6) in relative terms. Using $\mathbf{p}$, r_1, r_2, r_3, and r from Figure D.1 (a) are:

$$\begin{aligned} r_1 &= \sqrt{p(2)^2 + p(3)^2} \\ r_2 &= \sqrt{p(1)^2 + p(3)^2} \\ r_3 &= \sqrt{p(1)^2 + p(3)^2} \\ r &= \sqrt{p(1)^2 + p(2)^2 + p(3)^2}. \end{aligned} \tag{D.8}$$

The angle of rotation about the x-axis is then

$$\theta_x = \arcsin \frac{p(2)}{r_1}, \tag{D.9}$$

such that the components of $\mathbf{R}(\theta_x)$ are:

$$\begin{aligned} \sin\theta_x &= \frac{p(2)}{r_1} \\ \cos\theta_x &= \frac{p(3)}{r_1}. \end{aligned} \tag{D.10}$$

However, if $r_1 = 0$, then θ_x is zero and $\sin\theta_x = 0$ and $\cos\theta_x = 1$. $\mathbf{R}(\theta_x)$ is defined as:

$$\mathbf{R}(\theta_x) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x & 0 \\ 0 & -\sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{D.11}$$

In the same way, the angle of rotation about the y-axis is determined as

$$\theta_y = \arcsin \frac{p(1)}{r}, \tag{D.12}$$

such that the components of $\mathbf{R}(\theta_y)$ are

$$\begin{aligned} \sin\theta_y &= \frac{p(1)}{r} \\ \cos\theta_y &= \frac{r_1}{r}, \end{aligned} \tag{D.13}$$

and $\mathbf{R}(\theta_y)$ is:

$$\mathbf{R}(\theta_y) = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{D.14}$$

The reflection about the z-axis is just a 180^0 rotation:

$$\mathbf{R}(\theta_z) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{D.15}$$

The composite rotation (forward and backward) is then:

$$\mathbf{R}_{rot} = \mathbf{R}(-\theta_x)\mathbf{R}(\theta_y)\mathbf{R}(\theta_z)\mathbf{R}(-\theta_y)\mathbf{R}(\theta_x). \tag{D.16}$$

Since the boundary point is assumed to be at the origin, the translation matrix (and it's inverse) is simply the identity matrix

$$\mathbf{T} = \mathbf{I} \tag{D.17}$$

If the boundary point was not at the origin (or the assumption was not made), the translation matrix would take the form

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & -x_b \\ 0 & 1 & 0 & -y_b \\ 0 & 0 & 1 & -z_b \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{D.18}$$

with an inverse

$$\mathbf{T}^{-1} = \begin{bmatrix} 1 & 0 & 0 & x_b \\ 0 & 1 & 0 & y_b \\ 0 & 0 & 1 & z_b \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{D.19}$$

The full affine transformation matrix is then:

$$\mathbf{A} = \mathbf{T}^{-1}\mathbf{R}_{rot}\mathbf{T}. \tag{D.20}$$

With the affine transformation defined, the image node is determined from

$$\begin{Bmatrix} x_{im} \\ y_{im} \\ z_{im} \\ 1 \end{Bmatrix} = \mathbf{A} \begin{Bmatrix} dx_{bg} \\ dy_{bg} \\ dz_{bg} \\ 1 \end{Bmatrix}. \tag{D.21}$$

The image location determined in (D.26) is in reference to the origin. To define the image globally, the image location is shifted using the boundary node

$$\mathbf{x}_{im} = \mathbf{x}_{im} + \mathbf{x}_b \tag{D.22}$$

If (D.19) is used, then (D.22) is unnecessary.

D.4 Algorithm for Generating Image Nodes via Affine Transform

The algorithm to generate the image node using an affine transform, using Figure D.1 as a guide, is described by Algorithm D.1.

Algorithm D.1 Affine Image Nodes

1: Generate the plane p passing through the boundary point *b* assumed to lie at the origin defined by (D.7), using the boundary nodal normal $\mathbf{n}$ and boundary node $\mathbf{x}_b$ to define the plane.

2: Rotate the plane p and $\mathbf{x}_g$ about the x-axis using (D.11).

3: Rotate p and the points about the y-axis using (D.14).

4: Reflect *g* about the z-axis to generate the image node *im* using (D.15).

5: Perform the inverse rotations about the y- and x-axis to define the image node *im* in real space, but in reference to the origin.

6: Using (D.22), translate the image node *im* from the origin to it's real position relative to *b* and *g*.

This process is applied to each ghost node to generate the image nodes.

D.5 Algorithm for the Velocity of Symmetry and Inviscid Wall Ghost Nodes

The algorithm to determine the velocity at a ghost node using an affine transform is described in Algorithm D.2.

Algorithm D.2 Affine Velocity for Symmetry and Inviscid Wall Ghost Nodes

1: Generate the plane p passing through the boundary point *b* with the boundary nodal normal $\mathbf{n}$. Unlike the previous algorithm, velocity will not need to be translated, so the assumption of the boundary node at the origin is not required.

Algorithm D.2 Affine Velocity for Symmetry and Inviscid Wall Ghost Nodes, Continued

2: The velocity at the image node $\mathbf{v}_{im}$ is determined by:

(a) An interior node, where *i* and *im* are collocated such that

$$\mathbf{v}_{im} = \mathbf{v}_i. \tag{D.23}$$

(b) From exclusively interior nodes, where quantities at the image node are determined from a MLS fitting using a stencil containing no ghost nodes:

$$\mathbf{v}_{im} = \sum_{k \in \Omega_{\mathbf{x}_{im}}} \Psi_k q_k \tag{D.24}$$

(c) Using the higher-order boundary system described in 5.4.2.2. In this situation, the image velocity is defined as:

$$\mathbf{v}_{im} = \sum_{k \in \Omega_{\mathbf{x}_{im}}} \Psi_k q_k, \tag{D.25}$$

3: Follow steps 2 and 3 from Algorithm D.1 to generate the rotation matrices, and the reflection matrix is $\mathbf{A} = \mathbf{R}_{rot}$.

4: The velocity at the ghost is determined then as the reflection of the image velocity:

$$\left\{ \begin{array}{c} \mathbf{v}_g \\ 1 \end{array} \right\} = \mathbf{A} \left\{ \begin{array}{c} \mathbf{v}_{im} \\ 1 \end{array} \right\}. \tag{D.26}$$

This algorithm is applied to ghost nodes subject to symmetry or inviscid wall boundary conditions.

APPENDIX E

EXACT QUADRATURES FOR TRIANGLES AND QUADRILATERALS

This appendix provides the quadrature locations and weights for triangles and quadrilaterals. Figure E.1 shows the location of the quadrature for a first- or second-order quadrature. The

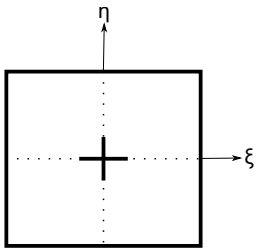


Figure E.1: Quadrature Location for a 1^{st}- and 2^{nd}-Order Accurate Quadrature on a Quadrilateral.

quadrature for a first- or second-order quadrature requires only a single quadrature point located at $(0, 0)$ with a weight $w_q = 1$.

Figure E.2 shows the location of the quadrature for a third- or fourth-order quadrature.

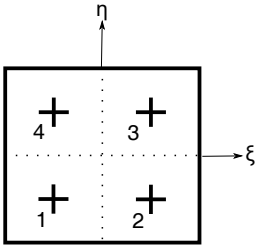


Figure E.2: Quadrature Location for a 3^{rd}- and 4^{th}-Order Accurate Quadrature on a Quadrilateral.

A third- and fourth-order accurate quadrature requires four quadrature nodes. The weights for the quadratures are given in Table E.1.

Table E.1: Quadrature Weights for a 3^{rd}- and 4^{th}-Order Accurate Quadrature for a Quadrilateral.

	Coordinates, (ξ, η)		Weight, w_q
1	-0.577350269189625	-0.577350269189625	0.25
2	0.577350269189625	-0.577350269189625	0.25
3	0.577350269189625	0.577350269189625	0.25
4	-0.577350269189625	0.577350269189625	0.25

Figure E.3 shows the location of the quadrature for a fifth- or sixth-order quadrature. A fifth-

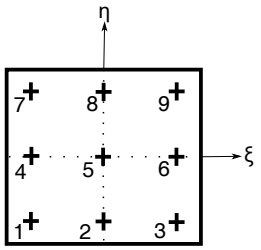


Figure E.3: Quadrature Location for a 5^{th}- and 6^{th}-Order Accurate Quadrature on a Quadrilateral.

and sixth-order accurate quadrature requires nine quadrature nodes. The weights for the quadratures are given in Table E.2.

Table E.2: Quadrature Weights for a 5^{th}- and 6^{th}-Order Accurate Quadrature for a Quadrilateral.

	Coordinates, (ξ, η)		Weight, $0.25w_q$
1	-0.774597	-0.774597	0.308641975308641
2	0	-0.774597	0.493827160493827
3	0.774597	-0.774597	0.308641975308641
4	-0.774597	0	0.493827160493827
5	0	0	0.790123456790123
6	0.774597	0	0.493827160493827
7	-0.774597	0.774597	0.308641975308641
8	0	0.774597	0.493827160493827
9	0.774597	0.774597	0.308641975308641

APPENDIX F

OCTREE

F.1 Introduction

This Appendix describes the octree algorithm used to determine where in the main mesh an image node is located. The octree algorithm is the three-dimensional version of a quadtree. Both work on the principle of subdivision into smaller domains. Octrees and quadtrees are types of tree data structures, where a tree is an abstract data type where the base reference is a root, and divisions of the root (or lower levels) are referred to as children, where each child holds a unique set of the root. In the case of an octree, each parent division has exactly 8 children. The octree has several uses, such as representing hierarchical data, storing data for efficient searching, representing sorted lists, routing algorithms, and compositing digital images for visual effects. The appendix is outlined as follows. First, the general algorithm for developing the octree is presented. Next, modifications for finding image nodes are given. The algorithm for transversing and searching the octree is given. Finally, the algorithm for using the octree to find image nodes is given.

F.2 General Algorithm

This section presents the general algorithm for the octree. The ultimate goal for the octree algorithm is to divide a given quantity into smaller, easier to handle groupings, or leaves. In terms of this dissertation, the octree divides the mesh nodes via their coordinates into leaves such that image and intercept nodes can be 'found' within the physical domain.

The first step in the octree algorithm is to perform a sort of the coordinates of the nodes. This sort could be performed with any sorting algorithm, but in this dissertation a heapsort [**?**] is used, which performs well when several quantities may be the same (such as a layer in the mesh). The heapsort algorithm is $O(n \log n)$ in the worse case, which is faster than the quicksort

algorithm [81]. Each coordinate direction is sorted and tracked, and all the sorted nodes are put into the root.

The root is then split into child bins. First, the top, bottom, and median in each coordinate direction for each child is determined. To find the median, the median is simply selected as the middle of the list. The median is adjusted to make the top and bottom list equal in length. If the median is close to the top or the bottom within a specified tolerance, and cannot be adjusted, the root is flagged for to no division into further children in the coordinate direction. If the median is far enough away from the top and bottom of the list, the root is available for splitting into further children. Next, if the root is available for splitting, the nodes in the root are segregated into 8 children using the median in each direction as the splitting point. It is possible for a child to have zero nodes, and in that case the root will have less than 8 children. Before further division, each child's bounds are determined. The child's bounds are determined by:

$$\begin{aligned} \text{low}_{\text{bound}} &= \max(\min(\mathbf{x}_{first}, \mathbf{x}_{\text{median}} - |\mathbf{x}_{first} - \mathbf{x}_{\text{median}}|), \mathbf{x}_{first,parent}) \\ \text{up}_{\text{bound}} &= \min(\max(\mathbf{x}_{last}, \mathbf{x}_{\text{median}} + |\mathbf{x}_{last} - \mathbf{x}_{\text{median}}|), \mathbf{x}_{last,parent}). \end{aligned} \tag{F.1}$$

Each child is then split, with the previous process for the root applied to each child to determine how or if the child is split. This recursiveness simplifies coding as well. The process of splitting into smaller and smaller bins continues until either a user minimum number of nodes in a bin is reached or if the child is determined to be unsplittable via the list size algorithm.

F.3 Algorithm Modifications for Locating Image Nodes

This section presents modifications made to the general octree algorithm. First, each child may be padded to ensure that nodes in the domain fall fully within a child and not on a border between children. Without the padding, nodes may not be correctly located in space via the transversal algorithm.

F.4 Octree Transversal Algorithm

Once the entire tree is built, the octree can be transversed to find whatever information is desired by the user. A transversal is begun by selecting a location in space, be it a physical node in the octree or a random location. To simplify the discussion, this node will be referred to as a flower. First, the algorithm determines if the flower is located within the bounds of the tree. The bounds of the tree, or each leaf (or child), is adjusted with a bin tolerance such that there is some padding to the bin. This padding ensures the algorithm does not run into an interface between leafs. If the flower is on the tree, then each child of the tree is checked to see if the flower is located on the leaf. If the flower is not located within the bounds of the leaf, the next leaf is checked to see if the flower is located within the leaf. This process continues until the top of the tree is reached and the leaf has no more children. Once this happens, the flower has been located, and the tree is transversed.

F.5 Using the Octree

This section describes how the octree is used within this dissertation. The octree algorithm is used herein to locate image and intercept nodes within the main or ghost domain. First, the tree information for both the main and ghost domain are built. Next, the image and intercept nodes are created as described in (reference). Once the image and intercept nodes are built, each image and intercept node 'flower' is pushed through the main tree to find the closet node using the transversal algorithm. To find the closet node on the leaf to the flower, the flower coordinates are compared to the coordinates of the nodes within the bounding leaf. The smallest difference is considered the closest node:

$$\begin{aligned} d\mathbf{x} &= \mathbf{x}_{\text{flower}} - \mathbf{x}_{\text{leaf node}} \\ d &= d\mathbf{x} \cdot d\mathbf{x} \\ \min_d &= \min(\min_d, d) \end{aligned} \tag{F.2}$$

If the flower is not on the main tree, then the ghost tree is checked. Once the closet node is found, the stencils can be built for the image and intercept nodes, as described in Section 5.4.2.1.

APPENDIX G

EQUIVALENCE OF CHOELSKY AND SVD DECOMPOSITION FOR A SYMMETRIC POSITIVE DEFINITE MATRIX

This appendix shows the equivalence between the Cholesky and SVD Decomposition for a symmetric positive definite (SPD) matrix. This is important for mapping an ellipsoid to the unit sphere (or disk in two dimensions) [132]. For an SPD matrix, of which the quadratic form of the ellipsoid, $\mathbf{H}$, is (see A, there exists an SVD decomposition of the form:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^{*} \tag{G.1}$$

where $\mathbf{U} = \mathbf{V}$ in the standard SVD decomposition and $()^{*}$ is the conjugate transpose. If a QR decomposition of the SVD system is performed, such that:

$$\mathbf{Q}\mathbf{R} = \mathbf{U}\sqrt{\mathbf{\Sigma}} \tag{G.2}$$

where $\mathbf{Q}$ is an unitary matrix and $\mathbf{R}$ upper right triangular, then:

$$\mathbf{A} = \mathbf{Q}\mathbf{R}(\mathbf{Q}\mathbf{R})^{*} = \mathbf{L}\mathbf{L}^{*} \tag{G.3}$$

which is the Cholesky decomposition of $\mathbf{A}$.