# APPLICATIONS OF SMOOTHNESS-INCREASING ACCURACY-CONSERVING FILTERING TO PARTICLE-IN-CELL DATA DENOISING, ENHANCED MULTIRESOLUTION ANALYSIS, AND ENTROPY-CORRECTION SCHEMES

by

Matthew John Picklo Jr.

© **Copyright by Matthew John Picklo Jr., 2023**

All Rights Reserved

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Doctor of Philosophy (Computational and Applied Mathematics).

Golden, Colorado

Date _____

Signed: _____

Matthew John Picklo Jr.

Signed: _____

Dr. Jennifer K. Ryan
Thesis Advisor

Signed: _____

Dr. Ayaboe Edoh
Thesis Advisor

Golden, Colorado

Date _____

Signed: _____

Dr. Gus Greivel
Professor and Department Head
Department of Applied Mathematics and Statistics

ABSTRACT

The numerical solution of partial differential equations can be very challenging, and while different numerical methods have their own advantages, there is no single numerical method without drawbacks. For example, data arising from Particle-In-Cell (PIC) methods have noise in their numerical solutions owing to the use of finitely many particles. The underlying mesh and corresponding approximation in Discontinuous Galerkin (DG) methods may not be of sufficient spatial resolution to capture localized solution features. Lastly, additional conservation equations such as entropy conditions satisfied by analytic solutions to PDEs are often not satisfied by their discrete counterparts. These deficiencies can be ameliorated by auxiliary techniques and method augmentations. It is the purpose of this thesis to detail three such improvements making use of Smoothness-Increasing Accuracy-Conserving (SIAC) filters, proceeding in order of increasing embedment of SIAC methodologies within the underlying numerical process. First, we detail the application of SIAC filters as denoisers of data arising from PIC simulations and demonstrate how careful tuning of these filters enables noise-reduction in the presence of non-periodic boundaries. Furthermore, in the computation of Bohm speed estimates from PIC data, we show that SIAC filters enable a dramatic reduction in the quantity of PIC data needed. Having applied SIAC filters to the equivalent of finite volume data, we next consider higher-order piecewise polynomial data. Here we show that SIAC filtering can be used to increase the resolution of coarse mesh polynomial data in multiple dimensions and over nonuniform meshes. The resulting enhancement procedure can be viewed within the framework of multiresolution analysis (MRA), making it a natural candidate for application in mesh adaptive DG schemes, where it provides a means of providing more accurate approximations with reduced degrees of freedom as compared to uniformly refined data. To conclude, we break the requirement of applying SIAC methodologies to effectively stationary data by fully integrating SIAC filters with DG numerical methods in the context of entropy-correction schemes. In this application, SIAC filters are applied to ensure that approximations produced by the baseline DG method satisfy additional physically motivated conservation conditions, namely energy conservation, for the duration of a simulation.

TABLE OF CONTENTS

ABSTRACT . . . iii

LIST OF FIGURES . . . ix

LIST OF TABLES . . . xv

ACKNOWLEDGMENTS . . . xvii

CHAPTER 1 INTRODUCTION . . . 1

1.1 Overview . . . 2

1.2 Literature Review . . . 3

1.2.1 SIAC Filters . . . 3

1.2.2 DG Methods . . . 4

1.2.3 Wavelets and MRA . . . 5

1.2.4 Entropy-Correction . . . 6

CHAPTER 2 BACKGROUND . . . 8

2.1 SIAC Filters . . . 8

2.1.1 One-Dimensional SIAC Kernels . . . 8

2.1.2 Tensor-Product SIAC . . . 11

2.1.3 LSIAC . . . 11

2.2 Discontinuous Galerkin Schemes . . . 13

2.2.1 Weak Formulation: 1st Order Terms . . . 13

2.2.2 Weak Formulation: Higher-Order Spatial Terms . . . 15

2.2.3 Computational and Implementation Details: Hyperbolic Component . . . 16

2.2.4 Computational and Implementation Details: Elliptic Component . . . 18

2.2.4.1 Combination of spatial discretization terms . . . 20

2.2.5 Time Evolution Schemes . . . 21

2.2.5.1 A note on time-stepping stability . . . 21

2.3 Multiresolution Analysis . . . 22

2.3.1 Extension to Two Dimensions . . . 25
2.3.2 Relationship between Legendre Modal Coefficients and Scaling Coefficients . . . 27
2.3.3 Local vs. Global MRA . . . 27
CHAPTER 3 APPLICATION OF SIAC FILTERS TO PARTICLE-IN-CELL DATA . . . 29
3.1 Chapter Introduction . . . 29
3.2 Data Initialization . . . 31
3.2.1 Constructing the mesh . . . 31
3.2.2 Constructing a piecewise-polynomial interpolant . . . 32
3.3 SIAC Filters: Kernel Construction and Boundary Treatments . . . 32
3.3.1 Knot Matrix Kernel formulation . . . 33
3.3.2 Shifting function . . . 34
3.3.3 Knot Matrix Examples . . . 35
3.3.4 Kernel coefficients . . . 35
3.3.5 Generalized Spline Kernels . . . 35
3.3.6 Adaptive kernel scaling . . . 36
3.3.7 Convolution evaluation . . . 37
3.3.8 Fourier representation . . . 37
3.4 Numerical Results . . . 38
3.4.1 Assessment of Fourier Effects . . . 40
3.4.2 Periodic boundary conditions . . . 40
3.4.3 Data with sharp gradients near the boundary . . . 42
3.4.4 Application to Bohm speed calculations . . . 46
3.5 Chapter Summary . . . 47
3.6 Co-Author Contributions . . . 48
CHAPTER 4 (L)SIAC-MRA ON UNIFORM MESHES . . . 49
4.1 Chapter Introduction . . . 49
4.2 Input Data Format . . . 49

4.3 Review of One-dimensional SIAC-MRA . . . 50

4.4 Multidimensional (L)SIAC-MRA . . . 51

4.5 Transition Operator Implementation . . . 52

4.5.1 Sifting Matrices . . . 54

4.5.2 Coefficient Matrices . . . 56

4.5.3 Projection Matrices . . . 57

4.5.3.1 Quadrant II . . . 57

4.5.3.2 Quadrant IV . . . 58

4.5.3.3 Quadrant III . . . 58

4.5.3.4 Quadrant I . . . 59

4.5.4 Coding Comment: . . . 60

4.6 Numerical Results . . . 60

4.7 Two-dimensional Test Problems . . . 61

4.8 Three-dimensional Test Problems . . . 72

4.9 Chapter Summary . . . 74

4.10 Co-Author Contributions . . . 74

CHAPTER 5 LSIAC-MRA ON NONUNIFORM MESHES AND APPLICATIONS TO MESH ADAPTIVE DG . . . 75

5.1 Chapter Introduction . . . 75

5.2 Background . . . 77

5.2.1 Data format . . . 77

5.2.2 Enhancement procedure . . . 77

5.2.3 $L^2$-projection . . . 78

5.2.4 Adaptive kernel scaling . . . 78

5.3 Application to Nonuniform Mesh Transition . . . 79

5.3.1 Mesh Refinement . . . 79

5.3.2 Numerical tests . . . 80

5.3.3 Impact of characteristic frequencies and lengths on error reduction . . . 81

5.3.4 Adaptive versus constant scaling . . . 83

5.4 Application to Adaptive DG . . . 85

5.4.1 Adaptivity procedure . . . 85

5.4.2 Test Problems . . . 87

5.4.3 Numerical Results: Viscous Burgers . . . 88

5.4.4 Numerical Results: Poisson Equation . . . 92

5.4.5 Summary of Adaptivity results . . . 94

5.5 Chapter Summary . . . 96

5.6 Co-Author Contributions . . . 96

CHAPTER 6 APPLICATION OF SIAC FILTERING TO ENTROPY-CORRECTION DG SCHEMES . . . 97

6.1 Chapter Introduction . . . 97

6.2 Baseline Numerical Method . . . 99

6.3 Entropy Correction . . . 101

6.3.1 An Adaptive Filter Selection Strategy . . . 104

6.3.2 Shock Regularization based on Artificial Viscosity . . . 105

6.3.3 Fully Discrete EC via relaxation Runge-Kutta time-stepping . . . 107

6.4 Filter Construction and Properties . . . 109

6.4.1 Filter Matrix Construction . . . 109

6.4.2 Conservation of the Filtering Procedure . . . 110

6.4.3 Fourier and Spectral Response . . . 113

6.5 Results . . . 114

6.5.1 One-dimensional entropy conservation . . . 114

6.5.2 One-dimensional regularization . . . 117

6.5.3 Two-dimensional regularization . . . 122

6.5.4 Confirmation of high-order accuracy . . . 122

6.6 Chapter Summary . . . 126

6.7 Co-Author Contributions . . . 127

CHAPTER 7 SUMMARY AND FUTURE WORK . . . . . 128

REFERENCES . . . . . 129

APPENDIX A LEGENDRE POLYNOMIALS . . . . . 139

APPENDIX B RUNGE-KUTTA SCHEMES . . . . . 140

APPENDIX C CO-AUTHOR PERMISSIONS . . . . . 142

APPENDIX D COPYRIGHT PERMISSIONS . . . . . 144

D.1 Journal Article Permissions . . . . . 144

APPENDIX E SIAC-MRA TRANSITION MATRICES . . . . . 146

E.1 SIAC-MRA: Filtered Basis Functions, Coefficients, and Transition Matrices: . . . . . 146

E.2 One-dimensional SIAC-MRA . . . . . 146

E.2.1 One-dimensional Transition Matrices . . . . . 146

E.3 Two-dimensional SIAC-MRA . . . . . 149

E.3.1 Two-dimensional SIAC-MRA Transition Matrices . . . . . 149

E.3.2 Construction of a permutation matrix . . . . . 152

E.4 Three-dimensional SIAC-MRA . . . . . 157

E.4.1 Three-dimensional SIAC-MRA Transition Matrices . . . . . 157

APPENDIX F LSIAC-MRA TRANSITION MATRICES . . . . . 159

F.1 Two-dimensional LSIAC-MRA . . . . . 159

F.1.1 Two-dimensional LSIAC-MRA Transition Matrices . . . . . 160

F.2 Three-dimensional LSIAC-MRA . . . . . 160

F.2.1 Three-dimensional LSIAC-MRA Transition Matrices . . . . . 169

F.2.2 Sifting Matrices . . . . . 169

F.2.3 Coefficient Matrices . . . . . 170

F.2.4 Global Filtered Coefficient Matrices . . . . . 172

F.2.5 Projection Matrices . . . . . 173

LIST OF FIGURES


Figure 2.1 B-splines of orders 1, 2, and 3. Note that each B-spline is nonzero on a finite interval. . . 9

Figure 2.2 Illustration of the two-dimensional multiwavelet idea. The approximation at the finest level, $\mathbf{V}_3^p$, can be represented as the approximation from the $\mathbf{V}_0^p$ scaling function space plus the multiwavelet coefficients from multiwavelet spaces $\mathbf{W}_0^p$,$\mathbf{W}_1^p$, and $\mathbf{W}_2^p$. . . . . 26

Figure 3.1 Example of partitioning pointwise PIC data into elements, and constructing an interpolant of the pointwise data ($f_j = f(x_j)$) on each element. The red partitions represent the superimposed mesh to the pointwise grid. The blue lines represent the piecewise linear interpolant reconstructed from a stencil containing two grid values. In (a) the elements $\tau_j = I_j$ consist of a single cell, and the stencil uses information from the right adjacent cell ($\ell = 0$ and $r = 1$). In (b) the elements $\tau_j = I_{2j-1} \cup I_{2j}$ are formed of a union of two neighboring cells and the stencil for each element only consists of those cells inside that element. . . . . 31

Figure 3.2 Demonstration of a shift function $\lambda(x^\star)$ for $K_H^{(3,2)}$ with $H(x^\star) = 0.1$ on the domain $\Omega = [-1, 1]$. . . . . 34

Figure 3.3 Shifted Kernels for $K^{(3,2)}(x)$. For $\lambda = \pm 2$, $x = 0$ corresponds to the boundary. Note that owing to the reflection of $y$ in the kernel argument during the convolution, $\lambda = -2$ ($\lambda = 2$) corresponds to the left-sided (right-sided) kernel in the physical space. . . . . 35

Figure 3.4 Shifted Kernels for $K^{(3,2)}(x)$ including a generalized spline. Here $x = 0$ corresponds to a boundary. . . . . 36

Figure 3.5 An adaptive kernel scaling example for $K_H^{(3,2)}(x)$ over the domain $\Omega = [0, 1]$ with $H_{\text{int}} = 0.2$, $h_{\text{grid}} = 0.1$. . . . . 38

Figure 3.6 Variation of the magnitude of the Fourier representation of the $K_H^{(r+1,\ell)}$ filter with respect to (a) spline order, with higher orders corresponding to smoother and more heavily damped filtered data, (b) moment conditions, with more moment conditions allowing the capturing of more information about a point, and (c) kernel scaling, with a larger scaling resulting in more information being used in the filtering process. Note that the kernel is symmetric with respect to $k$ and that unless indicated otherwise in the legends, $H = 12.8$, $r + 1 = 3$, and $\ell = 2$. . . . . 39

Figure 3.7 Application of SIAC filtering to periodic magnetic field for the whistler instability driven by the trapped electrons: (upper-left) filtered vs. unfiltered data over the whole domain. (right) zoomed-in physical space plot displaying spatial noise damping. (lower-left) single-sided amplitude spectrum of filtered and unfiltered data with analytic SIAC kernel response superimposed. . . . . 41

Figure 3.8 Effect of generalized splines and adaptive kernel scaling on boundary behavior preservation. The underlying data is electron density data $n_e$ with $\Omega = [0, 800]$ and $h_{grid} = \Delta x = 0.1$. Here the solid blue lines denote the unfiltered data and the dashed red lines the filtered data. . . . . 43

Figure 3.9 Effect of generalized splines and adaptive kernel scaling on the single-sided amplitude spectrum of electron density data $n_e$ with $\Omega = [0, 800]$ and $h_{grid} = \Delta x = 0.1$. Here the solid blue lines denote the unfiltered data and the dashed red lines the filtered data. Note that the underlying data was non-periodic so a Hanning window function was applied. . . . . 43

Figure 3.10 Effect of generalized splines and adaptive kernel scaling on boundary behavior preservation. The underlying data is electron heat flux data $q_e^n$ with $\Omega = [0, 800]$ and $h_{grid} = \Delta x = 0.1$. Here the solid blue lines denote the unfiltered data and the dashed red lines the filtered data. . . . . 44

Figure 3.11 Effect of generalized splines and adaptive kernel scaling on the single-sided amplitude spectrum of electron heat flux data $q_e^n$ with $\Omega = [0, 800]$ and $h_{grid} = \Delta x = 0.1$. Here the solid blue lines denote the unfiltered data and the dashed red lines the filtered data. Note that the underlying data was non-periodic so a Hanning window function was applied. . . . . 45

Figure 3.12 Effect of generalized splines and adaptive kernel scaling on electron heat flux gradient data $dq_e^n/dx$ in physical and frequency space. Here $\Omega = [0, 800]$, $h_{grid} = \Delta x = 0.1$, and the dashed red line (Typ.) denotes a filtering procedure with the typical kernel without generalized splines, while (Gen.) denotes a filtering with a kernel including a generalized spline. Finally, (Adap.) denotes that an adaptive kernel scaling is employed. In physical space, the effects of the differing boundary treatments are most apparent closest to the boundary, so the interior of domain is not shown. In Fourier space the interior scaling once again determines the magnitude of damping . . . . . 45

Figure 3.13 Comparison of Bohm speed and ion exit flow using filtered and unfiltered data time-averaged over 2,600 snapshots. The unfiltered long-time averaged data over 200,000 snapshots is also plotted. A $K_H^{(3,2)}$ kernel with scalings as indicated in Table 3.1 was used. Note that the modulus of $u_{Bohm}$ is plotted for the unfiltered short-time average data as the noise was sufficient to induce complex values. . . . . 47

Figure 4.1 Depiction of the varying element overlap of the LSIAC kernel in two dimensions resulting in varying filtered approximation forms for $p = 0$. The element is shaded in gray, the filter region has diagonal lines, and the kernel support is encapsulated by dashed lines. The filter region (labeled by quadrant and orientation relative to the element diagonal) represents the location of possible filtering points, and the kernel support represents the overlap of the kernel support for those filtering points. . . . . 53

Figure 4.2 Purpose of transition matrix components during refinement procedure: (1) $\rightarrow$ (2) Select modes from elements relevant to refining element $(i, j)$, (2) $\rightarrow$ (3) Select only the modes relevant to filtering a given region, in this case $2T$, (3) $\rightarrow$ (4) Use that modal information to obtain filtered approximation coefficients, (4) $\rightarrow$ (5) Obtain fine mesh modes by projecting the filtered approximation onto the refined mesh. Notice that $K_{q_x,q_y}^{i,j} = A\mathscr{C}^{q_x,q_y}\mathscr{S}^{i,j}$. . . . . 54

Figure 4.3 Comparison of the Log-Log plots of the $L^2$ and $L^\infty$ errors of the function $u_0(x, y) = \sin(2\pi x) + \sin(2\pi y)$ with $p = 4$ under refinement. In the left plot, the errors are computed on the mesh level of the iteration, i.e on the mesh size indicated by the data points abscissa. In the right plot, the errors are computed on the $140 \times 140$ mesh which is the finest mesh encountered during the procedure. We observe that this latter approach allows for the errors to remain constant under $L^2$-projection alone, which agrees with the intuition of the approximation being reproduced. . . . . 61

Figure 4.4 Surface plots of $u_0(x,y) = f_1(x)f_1(y)$ (Left) and $u_0(x,y) = f_2(x)f_2(y)$ (Right). The initial condition defined as a tensor-product of a discontinuous function $f_1$ represents a "strong discontinuity", while that using a tensor-product of a continuous but not globally differentiable function $f_2$ represents a continuous function with discontinuous derivatives $\frac{\partial u_0}{\partial x}$ and $\frac{\partial u_0}{\partial y}$. . . . . 62

Figure 4.5 Contour plots depicting pointwise |Error| for initial condition $u_0(x,y) = \sin(2\pi(x+y))$ on a series of three meshes using 2D LSIAC-MRA for $p = 4$. Notice that the filtering procedure regularizes and reduces the errors when it is applied at each refinement. . . . . 63

Figure 4.6 Contour plots depicting pointwise |Error| for initial condition $u_0(x,y) = \sin(2\pi(x+y))$ on a series of three meshes using 2D SIAC-MRA for $p = 4$. Notice that the filtering procedure regularizes and reduces the errors when it is applied at each refinement. . . . . 63

Figure 4.7 Log-Log plots of $N$ vs. Error for a high frequency function $u_0(x,y) = \sin(10\pi x)\sin(10\pi y)$ with wavenumber $k = 5$ on a series of three meshes using 2D LSIAC-MRA. We see that the effectiveness somewhat degrades with increasing polynomial order. . . . . 64

Figure 4.8 Contour plots depicting pointwise |Error| for the initial condition $u_0(x,y) = f_2(x)f_2(y)$ which contains a discontinuity in its first derivatives $\frac{\partial u_0}{\partial x}$ and $\frac{\partial u_0}{\partial y}$. 2D LSIAC-MRA was performed over a series of three meshes for $p = 4$. Notice that owing to the jump discontinuity in the derivatives at $x, y = 1/4, 3/4$, the polluted region where filtering adds artificial smoothness grows with each enhancement. Away from these locations, we still have error improvement. A zoomed in view is given in Figure Figure 4.10. . . . . 66

Figure 4.9 Contour plots depicting pointwise |Error| for the initial condition $u_0(x,y) = f_2(x)f_2(y)$ which contains a discontinuity in its first derivatives $\frac{\partial u_0}{\partial x}$ and $\frac{\partial u_0}{\partial y}$. 2D SIAC-MRA was performed over a series of three meshes for $p = 4$. Notice that owing to the jump discontinuity in the derivatives at $x, y = 1/4, 3/4$, the polluted region where filtering adds artificial smoothness grows with each enhancement. Away from these locations, we still have error improvement. A zoomed in view is given in Figure Figure 4.11. . . . . 66

Figure 4.10 Zoomed in contour error plots for the LSIAC-MRA twice enhanced $p = 4$ approximation of a discontinuous function (Top) and a continuous function that is discontinuous in its $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$ derivatives (Bottom). Both approximations were initialized on a $140 \times 140$ mesh. . . . . 67

Figure 4.11 Zoomed in contour error plots for the SIAC-MRA twice enhanced $p = 4$ approximation of a continuous function that is discontinuous in its $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$ derivatives. The approximation was initialized on a $140 \times 140$ mesh. . . . . 68

Figure 4.12 Log-Log plots of $N$ vs. Error for a continuous function that has discontinuous $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$ derivatives on a series of three meshes using 2D LSIAC-MRA. Errors were calculated by excluding the polluted regions around the discontinuous derivatives. Notice that performance is better than for the discontinuous function. . . . . 68

Figure 4.13 Log-Log plots of $N$ vs. Error for a continuous function that has discontinuous $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$ derivatives on a series of three meshes using 2D SIAC-MRA. Errors were calculated by excluding the polluted regions around the discontinuous derivatives. Notice that performance is better than for the discontinuous function. . . . . 70

Figure 4.14 Log-Log plots of $N$ vs. Error for the function $u_0(x, y, z) = \sin(2\pi x) + \sin(2\pi y) + \sin(2\pi z)$ on a series of three meshes using 3D LSIAC-MRA. Notice that procedure improves the approximation and provides error reduction. . . . . 73

Figure 4.15 Log-Log plots of $N$ vs. Error for the function $u_0(x, y, z) = \sin(2\pi x) + \sin(2\pi y) + \sin(2\pi z)$ on a series of three meshes using 3D SIAC-MRA. Notice that procedure improves the approximation and provides error reduction. . . . . 73

Figure 5.1 Illustration of enhancement . . . . . 78

Figure 5.2 Examples of nonuniform meshes with varying ratios of boundary element to interior element sizes. The color map indicates the kernel scaling used at each point in the domain. . . . . 79

Figure 5.3 Refinement procedure for quadrilateral and triangular elements. . . . . 80

Figure 5.4 Depiction of nonuniform quadrilateral and triangular meshes (left) and the refined meshes resulting from a uniform subdivision (right). . . . . 80

Figure 5.5 Log-Log Error plots for $u(x, y) = \sin(2\pi kx)\sin(2\pi ky)$ computed over the domain $[0, 1]^2$. On the x-axis is plotted the points per wavelength in a single coordinate direction $\sqrt{N}(p + 1) * 2^{\#enh}/k$. . . . . 82

Figure 5.6 Errors under refinement of the double hyperbolic tangent and Gaussian functions for varying scaling factors $k$. Here $N$ is the number of elements in a single coordinate direction $40 * 2^{\#enh}$. . . . . 84

Figure 5.7 Comparison of errors when using adaptive kernel scalings versus constant max edge length kernel scalings for $u(x, y) = \sin(2\pi x)\sin(2\pi y)$ on the SVR100 and ISR2 meshes over the domain $[0, 1]^2$. . . . . 84

Figure 5.8 $L^2$-errors, effectivity indices, and final meshes achieved under different refinement indicators for the one-dimensional Sine IC. Here a level equal to $n$ indicates an element width of $2^{-n+1}$. . . . . 90

Figure 5.9 $L^2$-errors, effectivity indices, and final meshes achieved under different refinement indicators for the one-dimensional hyperbolic tangent IC. Here a level equal to $n$ indicates an element width of $2^{-n+1}$. . . . . 91

Figure 5.10 Meshes produced by the multiwavelet-based error indicator for $p = 2$. Over repeated enhancements, the resolution about the sharp Gaussian spike in the interior increases while the resolution near the boundary remains coarse. . . . . 92

Figure 5.11 The first column depicts the $L^2$-errors and second the effectivity indices achieved under different refinement indicators for the two-dimensional Gaussian IC. . . . . 93

Figure 5.12 Comparison of final meshes generated after the last refinement for the Gaussian test problem with $p = 2$ for different error indicators. . . . . 94

Figure 5.13 The first and second column depict the $L^2$-errors and effectivity indices achieved under different refinement indicators for the two-dimensional tensor sine IC. . . . . 95

Figure 6.1 Left: Depiction of the LSIAC filter procedure for the $K^{(1,1)}_{\Gamma,\Delta x}$ filter with $\theta = \pi/4$ applied to $p = 5$. Right: Depiction of the sparsity pattern of the resultant $K^{(1,1)}_{\Gamma,\Delta x}$ filtering matrix for $p = 3$ on a $4 \times 4$ uniform mesh with $\theta = \pi/4$. . . . . 110

Figure 6.2 Eigenvector corresponding to $\lambda = 1$ for varying corrections of $\mathbf{K}_{glob.avg}$. Note that here a 20 element mesh using a 6 node per element LGL quadrature matrix $\mathbf{M}$ was used. The eigenvectors $\mathbf{v}$ have been normalized so that $\|\mathbf{v}\|_{\ell^\infty} = 1$. . . . . 112

Figure 6.3 Comparison of the effect of number of moments, spline order, and kernel scaling on the Fourier response of the SIAC filter. . . . . 113

Figure 6.4 Comparison of the effect of number of moments, spline order, and kernel scaling on the spectrum and singular values of the discrete filtering matrix for a $p = 3$ approximation on a $N_{elm} = 10$ mesh. Note only real values of the eigenvalues are plotted here as the imaginary components are, if not 0, on the order of machine precision. . . . . 114

Figure 6.5 Comparison of SIAC corrected and uncorrected approaches using central fluxes and LLF fluxes. First row: Relative magnitude of the correction to the RHS, subcell entropy metric, and energy over time of the method. Second Row: Solutions at time of shock formation $t = \frac{1}{\pi}$ and then afterwards at $t = \frac{3}{2\pi}$ (the unplotted uncorrected central flux solution has crashed) and $t = \frac{2}{\pi}$ (Only corrected solutions remain). . . . . 115

Figure 6.6 Comparison of SIAC and Local EC approaches, and their LS combinations with the EC Skew-Symmetric DGSEM formulation. First row: Relative magnitude of the correction to the RHS, subcell entropy metric, and weighing of constrained LS combination of corrections with $\theta = 1$ being only a SIAC correction and $\theta = 0$ being only a local correction). Second Row: Solutions at time of shock formation $t = \frac{1}{\pi}$ and then afterwards at $t = \frac{3}{2\pi}$ and $t = \frac{2}{\pi}$. . . . . 117

Figure 6.7 Comparison of SIAC correction regularization results for varying kernel parameterizations: number of moments/splines (Left), spline order/smoothness (Center), and kernel scaling (Right). Here $K^{(1,1)}_{\Delta x}$ is the baseline kernel with changed parameters as indicated. For all tests $c_e = 10$ and $c_{max} = 1$ was used. Increasing moments degrades solution quality when the approximation loses smoothness. Increasing the order and scaling makes the regularization more dissipative. . . . . 118

Figure 6.8 Comparison of the relative conservation error (Row 1) and energy dissipation (Row 2) of the SIAC correction regularization for varying kernel parameterizations. Here $K^{(1,1)}_{\Delta x}$ is the baseline kernel with changed parameters as indicated. For all tests $c_e = 10$ and $c_{max} = 1$ was used. . . . . 119

Figure 6.9 Reference finite volume solution to Burgers equation with $N_{elm} = 2000$ elements and a Local Lax-Friedrichs flux. . . . . 119

Figure 6.10 Comparison of Local and SIAC correction approaches to regularization. Here the tuning parameters used were those that qualitatively gave the best performance for the local correction. . . . . 121

Figure 6.11 Comparison of Local and SIAC correction approaches to regularization. Here the tuning parameters used were those that qualitatively gave the best performance for the SIAC correction. . . . . 121

Figure 6.12 Comparison of uncorrected and corrected 2D Modal DG approximations of the periodic inviscid Burgers equation with $u_0(x, y) = \sin(2\pi(x + y)) + 0.01$ on $[0, 1]^2$ with $p = 5$ and $N_{elm} = 8 \times 8 = 64$ elements. For regularization we used $c_E = 0.1$ and $c_{max} = 0.15$. . 123

Figure 6.13 Cross-sections ($y = x$) of uncorrected and corrected 2D Modal DG approximations of the periodic inviscid Burgers equation with $u_0(x, y) = \sin(2\pi(x + y)) + 0.01$ on $[0, 1]^2$ with $p = 5$ and $N_{elm} = 8 \times 8 = 64$ elements. Solutions depicted at shock formation $t = \frac{1}{2\pi}$ and afterwards at $t = \frac{1}{\pi}$. . . . . 123

Figure 6.14 Spatial order of convergence results for the periodic one-dimensional Burgers problem with $u(x, 0) = \sin(\pi x)$ on $\Omega = [0, 2]$ prior to shock formation. Exact solution obtained by Newton iteration. . . . . 125

Figure 6.15 Spatial order of convergence results for the periodic two-dimensional constant advection problem. Here $N$ is the number of elements in a given Cartesian direction. . . 125

Figure 6.16 Relative energy change over time using energy non-conserving time-stepping methods. . 126

Figure C.1 Co-author approvals for inclusion of Chapter 3. . . . . 143

Figure D.1 Reproduction approval from SIAM. . . . . 145

Figure F.1 Partition of domain for both 2D LSIAC (left) and the octant numbering scheme in 3D (right). . . . . 159

Figure F.2 Depiction of how the domain of dependence of the filtered approximation form varies from octant to octant. Octants 1 and 8 have six regions (Top), while octants 2 through 7 have only a pair of regions each (Bottom). The form of these regions is a consequence of the filtering coordinate inequalities given in Table F.4 coupled with the relative location of each octant in the reference element. . . . . 163

LIST OF TABLES

Table 3.1 Variable specific scalings for the 2,600 short-time averaged data set which were found to produce qualitative agreement with respect to the long-time averaged data. . . . . 46

Table 4.1 Table of $L^2-$ and $L^\infty-$errors for the high frequency initial condition $u(x,y) = \sin(10\pi x)\sin(10\pi y)$ on a series of three meshes using 2D LSIAC-MRA. Because it is high frequency, we have to initialize our approximation on a more refined grid. The shaded regions of the same shade represent the same approximation. For higher wavenumbers, the enhancement procedure requires greater initial mesh resolution for error reduction. . . . . 64

Table 4.2 Table of $L^2$- and $L^\infty$-errors for an initial condition that has discontinuities along $x, y = 1/4, 3/4$, $u(x,y) = f_1(x)f_1(y)$. The 2D LSIAC-MRA procedure was performed on a series of three meshes. Errors were calculated by excluding the polluted regions around the discontinuities. The LSIAC-MRA procedure reduces errors in these unpolluted regions. . . . . 69

Table 4.3 Shown are the $L^2$- and $L^\infty$-errors for a function $u_0(x,y) = f_2(x)f_2(y)$ that has discontinuous derivatives $\frac{\partial u_0}{\partial x}$ and $\frac{\partial u_0}{\partial y}$. The 2D LSIAC-MRA procedure was performed on a series of three meshes. Errors were calculated by excluding the polluted regions around the discontinuous derivatives. We observe that the LSIAC-MRA is effective for error reduction in unpolluted regions. . . . . 69

Table 4.4 Table of $L^2$- and $L^\infty$-errors for an initial condition that has discontinuities along $x, y = 1/4, 3/4$, $u(x,y) = f_1(x)f_1(y)$. The 2D SIAC-MRA procedure was performed on a series of three meshes. Errors were calculated by excluding the polluted regions around the discontinuities. The SIAC-MRA procedure reduces errors in unpolluted regions and does so to a greater extent than LSIAC-MRA. . . . . 70

Table 4.5 Shown are the $L^2$- and $L^\infty$-errors for a function $u_0(x,y) = f_2(x)f_2(y)$ that has discontinuous derivatives $\frac{\partial u_0}{\partial x}$ and $\frac{\partial u_0}{\partial y}$. The 2D SIAC-MRA procedure was performed on a series of three meshes. Errors were calculated by excluding the polluted regions around the discontinuous derivatives. The SIAC-MRA procedure reduces errors in unpolluted regions and does so to a greater extent than LSIAC-MRA. . . . . 71

Table 4.6 Table of $L^2$- and $L^\infty$-errors for initial condition $u(x,y,z) = \sin(2\pi x) + \sin(2\pi y) + \sin(2\pi z)$ on a series of three meshes using 3D LSIAC-MRA. We observe error reduction with each enhancement. . . . . 72

Table 4.7 Table of $L^2$- and $L^\infty$-errors for initial condition $u(x,y,z) = \sin(2\pi x) + \sin(2\pi y) + \sin(2\pi z)$ on a series of three meshes using 3D SIAC-MRA. We observe error reduction with each enhancement. . . . . 74

Table 5.1 Average element-wise $L^2$-errors of the $N_{elm}$ uniform element approximations used as user supplied error tolerances $\eta_{TOL}$. . . . . 89

Table 6.1 $L^2-$ orders of convergence for DGSEM solution to Burgers equation with initial data $\sin(\pi x) + 0.01$ on $[0,2]$. Errors computed at a final time of $t = 1/2\pi$. . . . . 124

Table B.1 Coefficients for the low storage 5-stage 4th order RK method. . . . . 141

Table E.1 Table of basis functions and associated coefficients resulting from the analytical evaluation of the convolution in $u_h^\star(x)$. Here $P_k$ denote Legendre polynomials. . . . . 146

Table E.2 Two-dimensional tensor-product filter coefficients. Note $m, r = 0, \ldots, p+1$. . . . . 150

Table E.3 Orientation of quadrants in the numbering scheme with respect to the origin. The orientation indicates which $\chi$ functions to use with respect to which variable by quadrant. The coefficient parameters indicate how coefficients adjust by quadrant. . . . 150

Table E.4 Orientation of octants in the numbering scheme with respect to the origin. The orientation indicates which $\chi$ functions to use with respect to which variable by octant. The shifting parameters for the coefficients are also determined by the octant and correspond to the shifting of the kernel support depending on the filtering point location. The octant numbering scheme is given in Figure F.1. . . . . 157

Table E.5 Three-dimensional tensor-product filter coefficients. Note $m, r, \ell = 0, \ldots, p+1$. . . . . 158

Table F.1 Two-dimensional LSIAC basis functions. For notational convenience suppose $\mathbf{x}$ belongs to element $(i, j)$. . . . . 160

Table F.2 Table describing how the shifting of the arguments of the $a$ coefficients vary by region. . 160

Table F.3 Regions present by octant. . . . . 162

Table F.4 Classification of regions. . . . . 162

Table F.5 Three-dimensional LSIAC basis functions for octant 1. For notational convenience suppose that $\mathbf{x}$ belong to element $(i, j, k)$ and that the region $S$ and position $P$ of the function are as indicated in the table rows. . . . . 164

Table F.6 Three-dimensional LSIAC basis functions for octants 2 through 4. For notational convenience suppose that $\mathbf{x}$ belong to element $(i, j, k)$ and that the region $S$ and position $P$ of the function are as indicated in the table rows. . . . . 165

Table F.7 Three-dimensional LSIAC basis functions for octants 5 through 7. For notational convenience suppose that $\mathbf{x}$ belong to element $(i, j, k)$ and that the region $S$ and position $P$ of the function are as indicated in the table rows. . . . . 166

Table F.8 Three-dimensional LSIAC basis functions for octant 8. For notational convenience suppose that $\mathbf{x}$ belong to element $(i, j, k)$ and that the region $S$ and position $P$ of the function are as indicated in the table rows. . . . . 167

Table F.9 Table describing how the shifting arguments $(q_x, q_y, q_z)$ of the $a$ coefficients vary by region. . . . . 168

Table F.10 Projection integrals and transformations by region for 3D LSIAC-MRA. . . . . 176

ACKNOWLEDGMENTS

I would like to thank my advisors Dr. Ryan and Dr. Edoh for all their help and I am very grateful for the great many things they have taught me. This work would not have come to fruition without their support and to them I am indebted. I would also like to thank Dr. Wu Fung, Dr. Wakin, and Dr. Fasshauer who have been very generous with their time in serving on my graduate committee and have provided feedback that has substantially improved this thesis. I am grateful to the team at LANL and would like to thank Dr. Qi Tang, Dr. Yanzeng Zhang, and Dr. Xian-Zhu Tang for the opportunity to work with them.

Many great friends, both in and outside the department have been supportive of me during this PhD. and I owe a big thanks to Dave, Josh, Johnny, Shelby, Spencer, Laura, Lucas, and Soraya, and many others too. To my family, Mom and Dad who have always encouraged me, and to Ian and Rose who have always brightened the day, I am truly grateful. Most of all, I want to acknowledge my wife Anna, whose patience and understanding during the completion of this work have been without compare.

# CHAPTER 1
# INTRODUCTION

Partial differential equations (PDEs) represent powerful mathematical models which are used to describe numerous phenomena in all areas of science and engineering. Understandably, their broad applicability and versatility results in situations where analytical solutions are either non-existent, or too unwieldy to prove useful in a given application. For these reasons, numerical methods are considered to discretize these models, and by solving them discretely, aid in our understanding of their behavior. However, the numerical solution of partial differential equations can be very challenging, and while different numerical methods have their own advantages and subproblems in which they excel, there is no single numerical method without drawbacks. For example, data arising from Particle-In-Cell (PIC) methods have noise in their numerical solutions owing to the use of finitely many particles. The underlying mesh and corresponding approximation in Discontinuous Galerkin (DG) methods may not be of sufficient spatial resolution to capture localized solution features. Lastly, additional conservation equations such as entropy conditions satisfied by analytic solutions to PDEs are often not satisfied by their discrete counterparts. That being said, these deficiencies can be ameliorated by auxiliary techniques and method augmentations. It is the purpose of this thesis to detail three such improvements making use of Smoothness-Increasing Accuracy-Conserving (SIAC) filters, proceeding in order of increasing embedment of SIAC methodologies within the underlying numerical process.

First, we detail the application of SIAC filters as post-processing denoisers of data arising from PIC simulations and demonstrate how careful tuning of these filters enables the desired noise-reduction in the presence of non-periodic boundaries. Furthermore, in the computation of Bohm speed estimates from PIC data, we show that SIAC filters enable data compression in the sense that utilizing filtering enables a dramatic reduction in the quantity of PIC data needed.

Having applied SIAC filters to the spatial equivalent of finite volume data, we then consider higher-order piecewise polynomial data such as that arising from discontinuous Galerkin schemes. Here we show that SIAC filtering can be used to increase the resolution of coarse mesh polynomial data in multiple dimensions and over nonuniform meshes. The resulting enhancement procedure can be viewed within the framework of multiresolution analysis (MRA), making it a natural candidate for application in mesh adaptive DG schemes, where it provides a means of providing more accurate approximations with reduced degrees of freedom as compared to uniformly refined data.

Historically, SIAC filtering has been applied after the final time of the numerical simulation has been computed. To conclude, we break the requirement of applying SIAC methodologies to effectively stationary data. We cast off this restriction in the context of entropy-correction schemes by fully integrating SIAC filters with method-of-lines DG numerical methods. In this application, SIAC filters are applied to ensure that the numerical approximations produced by the baseline DG method satisfy additional physically motivated conservation conditions, namely energy conservation, for the duration of a simulation.

### 1.1 Overview

In the remainder of this chapter a survey of select literature relevant to this thesis is provided. Technical background information on SIAC filters, Discontinuous Galerkin methods, and Multiresolution Analysis is contained in Chapter 2.

In Chapter 3 we construct a framework for the denoising of PIC-data by means of SIAC filters. Beginning with a discussion of the presumed form of the data, knot-matrix formulations [1] of SIAC filters are introduced. We show how this more-flexible expression for the filter enables straightforward treatment of finite domains via position-dependent knot-matrices and associated boundary structures by position-dependent kernel scaling. Numerical experiments are performed to validate the denoising characteristics of the PIC data SIAC filters. The chapter concludes with a demonstration of SIAC denoising in the computation of Bohm speeds estimates from limited noisy data.

In Chapter 4 we discuss approximation refinement by SIAC enhancement on uniform two-dimensional meshes for higher-order piecewise polynomial data analogous to that generated by finite element simulations. Owing to the nested hierarchy of approximation spaces the procedure can be, from the perspective of multiresolution analysis, thought of as generating multiwavelet details with which to construct fine mesh approximations. Using both tensor-product and Line SIAC filters, we construct explicit transition matrices for realizing the enhancement procedure and we demonstrate accuracy improvements under the developed enhancement procedure.

In Chapter 5 we extend the LSIAC-MRA procedure to nonuniform meshes in the form of perturbed quadrilateral meshes and Delaunay triangulations. We investigate the performance of the enhancement procedure on an array of challenging test problems and discuss possible improvements. We also consider the influence of mesh-informed adaptive kernel scaling. Lastly, we apply the enhancement procedure and the implicitly defined multiwavelet details in service to mesh adaptation schemes for the one-dimensional inviscid Burgers and two-dimensional Poisson equations. We show that SIAC filtering techniques are useful in the application of adaptive mesh refinement schemes.

The extension of SIAC methodologies to entropy-correction schemes is considered in Chapter 6, where we introduce SIAC filters in this context and apply them to construct spatially energy conserving DG schemes. Leveraging relaxation Runge-Kutta methods for temporal energy conservation we develop fully discrete energy conserving DG schemes. We apply these methods in pursuit of energy conservation and solution regularization in both one and two dimensions.

Finally, in Chapter 7 we summarize the contributions of this thesis and discuss possible future research directions.

## 1.2 Literature Review

To better understand the topics of this thesis, we begin with a brief interlude into the historical foundations of SIAC and MRA methodologies. Detailing the literature relevant to this thesis, also described is work conducted in adaptive numerical techniques for DG-style approximations and the entropy-correction schemes of Abgrall [2, 3]. Further details will be given in the Background section.

### 1.2.1 SIAC Filters

Smoothness-Increasing Accuracy-Conserving filters are a family of convolution kernel post-processors used to enrich numerical data. The origin of what would become known as SIAC filters began in the 1970s. In a finite difference context, Mock and Lax considered post-processing of numerical approximations to linear hyperbolic problems [4]. They noticed that improved accuracy could be obtained via convolution of the approximation with compact functions satisfying certain moment conditions. Independently, Bramble and Schatz in their seminal 1977 work [5] developed post-processing for finite element data. They obtained higher order accuracy from finite element numerical solutions to elliptic boundary value problems by application of a convolution kernel post-processor. Furthermore, they established a precedent for using B-spline functions in the kernel construction. This choice enabled efficient computation of the kernel, compact kernel support, and various theoretical properties relevant to error estimates. Superconvergence results were then generalized to finite element methods for hyperbolic problems in 2002 by Cockburn et al. [6]. In 2009, the family of convolution kernels gained the SIAC moniker with their application to streamline integration [7]. Interest in SIAC filters has grown steadily since. Position-dependent SIAC filters have been introduced to deal with nonperiodic boundaries by shifting the B-spline knot sequences away from the boundary [1, 7–11]. The knot-matrix formulation of SIAC kernels introduced in [1] is particularly useful in the application of position-dependent filters and will be discussed further in Chapter 3. SIAC filtering has also been extended to nonuniform meshes and unstructured triangulations where maintaining consistent integration over these more complicated mesh geometries has resulted in the incorporation of algorithms

from computational geometry to delineate quadrature regions [12–14]. Additionally, an approach given in [15] considers the construction of Hexagonal SIAC (HSIAC) filters for application on hexagonal meshes. The HSIAC itself composed of translates of hex splines which themselves can be viewed as linear combinations of box splines.

The use of different functions in kernel construction has also been investigated. The author of [16] has considered the use of non-spline functions of compact support. By selecting a compactly supported function, one can generate the equivalents of higher-order B-splines by repeated convolution by a first order B-spline. In so doing, the new kernel component functions inherit the crucial divided difference properties of the original B-spline components while admitting the possibility of reduced kernel support. SIAC kernels have also been analyzed through the framework of quasi-interpolation [17].

The generalization of SIAC filtering to higher dimensions need not require a tensor product formulation as discovered by Docampo-Sánchez et al. (2017), where the dimension of the kernel support could be reduced via line filtering. These Line-SIAC (LSIAC) filters are one-dimensional SIAC filters rotated to align with non-Cartesian axes. They represent a more computationally efficient alternative to traditional tensor-product SIAC filters as the curse of dimensionality associated with higher dimensional quadrature is avoided by the linear support. In addition, the LSIAC filtering methodology has been shown to obtain the same order of superconvergence as the traditional tensor product formulation with reduced error constants. Extensive use of LSIAC filters will be made in Chapters 4 and 5, but also in Chapter 6 as the small support of the kernel leads to sparse filtering matrices.

More recent developments have been concerned with the improvement of the computational efficiency of the filtering procedure using mesh-based adaptive scalings [18, 19], applications to streamline visualization [20], shock-capturing [21, 22], and of particular importance to Chapters 4 and 5, the enrichment of coarse information via post-processing and scale transition [23]. It is this later work which we generalize to multidimensional space via LSIAC filters and more complicated mesh geometries in Chapters 4 and 5 respectively. Furthermore, in Chapters 3 and 6 we will also apply SIAC methodologies to new application in entropy-correction and PIC data denoising respectively.

### 1.2.2 DG Methods

Many of the papers discussed above have dealt with filtering data arising from Discontinuous Galerkin (DG) methods. In the course of this thesis, we apply both our (L)SIAC-MRA and SIAC entropy-correction schemes to DG data, though they would by no means be restricted to that data type. We provide here a brief history of DG methods with a preference for hyperbolic problems. For the reader with further interest in DG methods, we refer them to the excellent survey paper [24], or the books [25–28] for additional

information.

DG methods were initially developed for studying neutron transport in 1973 by Reed and Hill [29]. In 1974 Le Saint and Raviart analyzed the method and obtained convergence results for triangulations and Cartesian meshes [30]. Interest grew greatly in DG around the 1990s as Cockburn, Shu, and others wrote a series of papers [31–35] analyzing DG for scalar and systems of hyperbolic conservation laws. They employed a method-of-lines approach in which spatial derivatives were discretized using DG, while time-stepping was performed using Runge-Kutta methods, yielding the aptly-named Runge-Kutta DG (RKDG) methods which we will apply in Chapters 5 and 6. For an introduction to DG methods for conservation laws, advection-diffusion, and higher order equations see [36]. It has been observed that DG discretizations can exhibit superconvergence in a variety of ways such as superconvergence at the roots of special polynomials [37–39], or in negative-order norms [6]. For further discussion of superconvergent phenomena associated with DG, see [24]. For our purposes we focus on DG methods for conservation laws in one and two dimensions as they will serve as the base schemes for both nonuniform SIAC-MRA and entropy-correction. We now turn our attention to the history of the MRA framework which we will use in mesh adaptivity for DG techniques in Chapter 5.

### 1.2.3 Wavelets and MRA

Wavelets and their corresponding multiresolution analyses (MRA) represent a cohesive framework for the decomposition of multi-scale information. This cohesive framework has made them powerful tools in a myriad of areas in mathematics, engineering, and physics. We seek to use this framework as a means of characterizing our (L)SIAC MRA procedures. We provide here a brief historical overview of MRA.

In the mid-1980s, Meyer and Mallat developed the powerful concept of a multiresolution analysis, defining scaling functions and wavelets in terms of constraints upon a single "scaling filter" [40, 41], and enabling scale-specific decomposition of functions. Alpert [42] introduced the concept of multiwavelets to generalize the ideas of wavelet theory to bases consisting of multiple scaling functions, and as result extending the applicability of multiresolution analyses to high-order polynomial data. Specifically, Alpert's multiwavelets play a critical role in establishing DG representations in a multiresolution framework. This link will be described more in Chapter 2 and used for mesh adaptation in Chapter 5.

The beneficial characteristics of space localization have made wavelets powerful tools in the numerical solution of PDEs, (see [43] for an excellent survey article of the diverse application wavelet-based schemes in fluid dynamics). In many adaptive methods, the general idea is to use the magnitude of wavelet (or multiwavelet) coefficients in a specified region as an indicator of the variability of a solution in that region. This relates to the idea of the magnitude of coefficients in a multiwavelet expansion being tied to the

smoothness of the function being approximated, with the faster decay of multiwavelet coefficients under scale refinement in a region being tied to the function's smoothness there [44, 45].

A prevalent use of this measure in the numerical PDEs literature is for reasons of computational efficiency. By truncating small magnitude coefficients, the mesh under consideration is effectively coarsened where possible and the numerical scheme made more efficient with fewer degrees of freedom. See the works of [46–50] for adaptivity in the finite volume context, [51–53] for adaptivity in a purely wavelet framework, and the works of [54–59] for adaptivity in a DG context. The DG formulations listed above are based upon grid coarsening for computational efficiency and require fine mesh information from which to start.

In this thesis we instead pursue an approach whereby coarse mesh approximations are enhanced into more accurate fine mesh approximations. Examples of this in the literature include [60], where higher-order polynomial reconstruction based off least-squares polynomials serve as a proxy for the exact solution and giving an error-type indicator for refinement (note this article eschews a MRA framework), and the work of Bautista et al. [61, 62], where a higher-order least-squares reconstruction based on information from neighboring elements is used to define an enhanced approximation. Unlike [60] this approach makes use of the multiresolution framework by using the computed detail coefficients from the reconstruction for mesh adaptivity indication. We will use detail coefficients resulting from SIAC enhancements for mesh adaptivity indication in Chapter 5.

### 1.2.4 Entropy-Correction

As will be described further in Chapter 6, a sought-after property in designing numerical methods for hyperbolic conservation laws is satisfaction of discrete entropy conditions. Numerical schemes which satisfy these conditions are often more robust, and in the case of simulation of physical phenomena, better respect underlying physical laws.

There are numerous approaches to analyzing and augmenting the entropic properties of numerical schemes, see [63] for an overview with respect to difference methods. Here we focus on the DG entropy-correction schemes in which a correction term is applied to a DG numerical scheme to ensure that it satisfies a discrete entropy condition. We use SIAC filters in determination of this correction term, and to that purpose, we briefly describe the relevant works of Abgrall and others in this area.

Abgrall [2] developed entropy-correction terms for steady-state residual distribution schemes which ensured entropy conservation with respect to a chosen convex entropy of the numerical approximation of arbitrary polynomial order. The correction itself relied on local-element averages of the entropy variable. In [3], these methods were generalized to unsteady problems, and also formulated in the context of the spatial discretization of DG schemes, again relying on local-element averages of the approximation data. In

Chapter 6 we will discuss a formulation for entropy-correction that makes use of averaging via multi-element SIAC filters.

# CHAPTER 2

# BACKGROUND

In this chapter we provide technical details on SIAC filtering, Discontinuous Galerkin methods, and Multiresolution Analysis. Other necessary, but chapter specific, information will appear in later chapters. The ordering of the background information is as follows. In Section 2.1 we discuss SIAC filters, detailing their uniform formulation, as well as their extensions to higher dimensions via tensor-product and line filter variants. In Section 2.2 we provide a derivation of Discontinuous Galerkin methods for hyperbolic, elliptic, and by extension parabolic partial differential equations. We also discuss the method-of-lines approach and different time-stepping methods used for time evolution of spatial DG discretizations. Lastly, considering the multiresolution framework in which SIAC-MRA can be viewed, we also provide in Section 2.3 an overview of one- and two-dimensional multiresolution analysis for the Alpert multiwavelets.

## 2.1 SIAC Filters

The SIAC post-processor is a convolution kernel filter originally designed to enable superconvergence extraction in finite element methods [6, 64]. They have since been extended for derivatives, boundaries and nonlinear hyperbolic equations [1, 8, 11, 65, 66] as well as mesh adaptivity [23]. The SIAC filter has been generalized for application to higher dimensional data using three methods [15, 64, 67]: the first is a tensor-product SIAC filter, whereby a convolution kernel is constructed by taking the tensor-product of multiple one-dimensional SIAC kernels; the second is to construct a Line SIAC filter (LSIAC), which is simply a one-dimensional SIAC kernel that has been rotated to align with a non-Cartesian axis. The last is the HSIAC filter which we do consider further here. All of these techniques rely on understanding the one-dimensional kernel presented below. We present here the standard uniform definition of SIAC kernels and leave the general knot formulation of SIAC kernels and their discussion to Chapter 3.

### 2.1.1 One-Dimensional SIAC Kernels

Given initial data $u_h(x)$, the one-dimensional the expression for a SIAC filtered approximation is given by

$$u_h^\star(x) = K_H^{(r+1,\ell)} \star u_h = \frac{1}{H}\int_{-\infty}^{\infty} K^{(r+1,\ell)}\Big(\frac{x-y}{H}\Big)u_h(y)\,dy, \tag{2.1}$$

where

$$K_H^{(r+1,\ell)}(t) = \frac{1}{H}K^{(r+1,\ell)}(t/H), \tag{2.2}$$

and

$$K^{(r+1,\ell)}(t) = \sum_{\gamma=0}^{r} c_\gamma B^{(\ell)}\Big(t - \gamma + \frac{r}{2}\Big). \tag{2.3}$$

The kernel scaling $H$ is typically chosen based off a characteristic length scale of the underlying data, such as element width $h$ in the context of finite-element data. The translated function occurring in the kernel definition, $B^{(\ell)}(t)$, is the order $\ell$ central uniform B-splines given by the recurrence relation

$$B^{(1)}(t) = \chi_{[-1/2,1/2)}(t), \tag{2.4}$$

$$B^{(\ell)}(t) = (B^{(\ell-1)} \star B^{(1)})(t), \tag{2.5}$$

$$= \frac{1}{\ell-1}\Big[\Big(\frac{\ell-1}{2} + t\Big)B^{(\ell-1)}(t+1/2) + \Big(\frac{\ell-1}{2} - t\Big)B^{(\ell-1)}(t-1/2)\Big]. \tag{2.6}$$

A depiction of the first three orders of central B-spline is given in Figure 2.1.

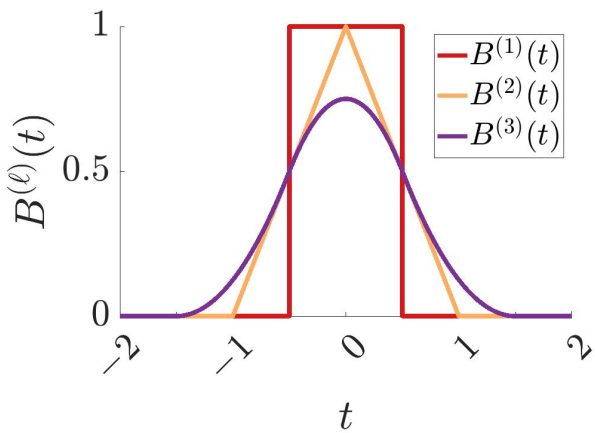


Figure 2.1 B-splines of orders 1, 2, and 3. Note that each B-spline is nonzero on a finite interval.

We construct the SIAC kernel (2.3) from B-Splines for the following reasons:

1. They have compact support

$$\mathrm{supp}(B^{(\ell)}(t)) = \Big[-\frac{\ell}{2}, \frac{\ell}{2}\Big],$$

   which localizes our filter and reduces the expense of the convolution.

2. Their derivatives can be written in terms of divided differences of lower order splines:

$$D^\alpha B^{(\ell)}(t/H) = \partial_H^\alpha B^{(\ell-\alpha)}(t/H),$$

where $\partial_H^\alpha$ is the $H$-scaled $\alpha$-th order central divided difference given by

$$\partial_H f(t) = \frac{f(t+H/2) - f(t-H/2)}{H},$$

with higher order divided differences define recursively via

$$\partial_H^\alpha(f(t)) = \partial_H(\partial_H^{\alpha-1} f(t)).$$

This property is crucial for maintaining or improving the order of accuracy of the approximation.

3. They are simple to compute through the given recurrence relation (2.6).

Additional properties and generalizations of B-splines can be found in [68, 69]. The coefficients $c_\gamma$ are chosen so that the SIAC kernel satisfies consistency and moment conditions. These are equivalent to polynomial reproduction:

$$\begin{aligned}
&\text{(Consistency)} \quad \int_{\mathbb{R}} K(x)\,dx = 1 \quad \Rightarrow \quad \int_{\mathbb{R}} K(x-y)\,dy = 1 \quad \text{(Polynomial Reproduction)} \\
&\text{(Moments)} \quad \int_{\mathbb{R}} K(x)x^k\,dx = 0 \quad \Rightarrow \quad \int_{\mathbb{R}} K(x-y)y^k\,dy = x^k,\ k \in \mathbb{Z},\ 1 \le k \le r,
\end{aligned}$$

where $r$ is the number of moment conditions that the kernel is chosen to satisfy. Note that increasing the number of moments increases the number of splines composing the kernel, as well as the kernel support. To see how these requirements generate a linear system to solve for the coefficients we simply substitute our kernel definition into the polynomial reproduction constraint:

$$\begin{aligned}
K^{(r+1,\ell)} \star x^k &= x^k \ \text{ for } \ k = 0, \ldots, r \\
\int_{\mathbb{R}} \sum_{\gamma=0}^{r} c_\gamma B^{(\ell)}(x - y - \gamma + \frac{r}{2}) y^k\,dy &= x^k \ \text{ for } \ k = 0, \ldots, r
\end{aligned}$$

Choosing $x = 0$ gives a linear system

$$\begin{bmatrix} b_{0,0} & \ldots & b_{0,r} \\ \vdots & & \vdots \\ b_{r,0} & \ldots & b_{r,r} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_r \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

where

$$b_{k,\gamma} = \int_{\mathbb{R}} B^{(\ell)}(-y - \gamma + \frac{r}{2}) y^k\,dy \ \text{ for } \ k, \gamma = 0, \ldots, r.$$

For large $r$ this system becomes ill-conditioned; however, for the values of $r$ considered in this thesis (typically $r + 1 \le 9$), no issues arise. The use of central B-splines with a uniform knot sequence will result in a symmetric filter, though this is not the case for general knot sequences as discussed in Chapter 3.

For uniform meshes we typically choose the kernel scaling parameter $H$ to be equal to the uniform element width $h$. To enable superconvergence extraction in DG approximations, we require $r = 2p$ and $\ell = p + 1$, where $p$ is the degree of the approximation. This allows for our filtered approximation to obtain $\mathcal{O}(h^{2p+1})$ errors in the $L^2$-norm and $L^\infty$-norm for linear hyperbolic equations [6]. For the purposes of our enhancement procedure described in later chapters under mesh refinement we will instead choose $r = 2p$ and $\ell = 1$ as based off [23] it appears to produce the lowest errors and avoids over-smoothing.

### 2.1.2 Tensor-Product SIAC

It is intuitive to generalize the SIAC methodology to higher dimensions via tensor-products, and this approach was considered in [64]. The filtered approximation for $\mathbf{x} \in \mathbb{R}^d$ is given by

$$u_h^\star(\mathbf{x}) = \int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty} K_{\mathbf{H}}^{(\mathbf{r}+1,\ell)}(\mathbf{x}-\mathbf{y}) u_h(\mathbf{y})\, dy_1 \, \dots \, dy_d, \tag{2.7}$$

where

$$K_{\mathbf{H}}^{(\mathbf{r}+\mathbf{1},\ell)}(\mathbf{t}) = K_{H_{x_1}}^{(r_1+1,\ell_1)}(t_1) \times \dots \times K_{H_{x_d}}^{(r_d+1,\ell_d)}(t_d).$$

Here $\mathbf{r}$ and $\ell$ are $d \times 1$ vectors containing the number of moments and splines orders for each component of the tensor-product kernel. For uniform meshes we choose the scaling factors $\mathbf{H} = (H_{x_1}, \dots, H_{x_d}) = (h_{x_1}, \dots, h_{x_d})$, where $h_{x_i}$ represents the uniform element length in the $x_i$-direction.

Though an intuitive means of extension in higher dimensions, the main cause of concern for generalizing the filter using a tensor-product is the increased computational cost associated with quadrature over the increased dimensionality of the kernel support. Specifically, performing exact integration of the convolution is very expensive as polygon clipping algorithms [70] are required to determine quadrature regions over which the integrand is strictly polynomial. Both breaks in continuity associated with element boundaries (mesh breaks), and breaks in smoothness associated with the piecewise polynomial kernel (kernel breaks) contribute to the delineation of these regions. On nonuniform meshes, they must be computed for every data point being filtered. This adds additional cost onto an already higher-dimensional quadrature. We describe next a more computationally efficient alternative.

### 2.1.3 LSIAC

A more computationally efficient formulation is the Line-SIAC filter (LSIAC) [67], which is simply a one-dimensional SIAC kernel that has been rotated to align with a non-Cartesian axis. In two dimensions, LSIAC filter is given by

$$u_h^\star(\mathbf{x}) = \left(K_{\Gamma,H}^{(r+1,\ell)} \star u_h\right)(\mathbf{x}) = \int_\Gamma K_H^{(r+1,\ell)}(t) u_h(\Gamma(t))\, dt, \tag{2.8}$$

where

$$\Gamma(t) = \mathbf{x} + t(\cos(\theta), \sin(\theta)).$$

The angle of rotation $\theta$ and the scaling parameter $H$ are selected to be

$$\theta = \tan^{-1}\left(\frac{h_{x_2}}{h_{x_1}}\right), \quad H = h_{x_1}\cos(\theta) + h_{x_2}\sin(\theta),$$

which aligns the kernel support with element diagonals for the case of quadrilateral meshes. On uniform meshes we choose the uniform mesh diagonal, $\theta = \pi/4$, and subsequently $H = \sqrt{2}h_{x_1} = \sqrt{2}h_{x_2}$. In higher dimensions we simply need to choose a new orientation and scaling factor. In three-dimensional space, the filtered approximation is similarly formulated where the line $\Gamma$ parameterized by $t$ is given by

$$\Gamma(t) = \mathbf{x} + t\big(\cos(\theta)\sin(\phi), \sin(\theta)\sin(\phi), \cos(\phi)\big) = \mathbf{x} + t\mathbf{v}, \tag{2.9}$$

with $\theta, \phi$ being the orientation of the line in spherical coordinates. Because our approximation lies on a uniform three-dimensional mesh, the line filter will be aligned with one of the four diagonals connecting antipodal vertices of the cubic element. Hence, we choose $H = \sqrt{3}h$, where $h$ is the uniform mesh spacing, and $\theta = \pi/4$, $\phi = \tan^{-1}\left(\sqrt{2}\right)$. This results in the direction vector $\mathbf{v} = \frac{1}{\sqrt{3}}(1,1,1)$. Note that alternative direction choices are possible, and on nonuniform meshes there is not theory for the choice of kernel orientation.

The utility of the LSIAC filter is its ability to filter higher-dimensional data without increasing the dimensionality of the kernel support. This makes it a computationally efficient alternative to the tensor-product filter when evaluating the convolution via quadrature sums. For example, in two dimensions for a quadratic approximation it can reduce the number of quadrature sums from 4000 to 10 [67]. Additionally, the constant in the error estimate is reduced for the filtered solution under the same assumptions on the initial data as for the tensor-product filter. We consider the LSIAC filter specifically because of its reduced and rotated kernel support in the filtering context. The ability to manipulate the kernel orientation and dimensionality is beneficial, owing to the reduction in computational expenses incurred by consistent integration over more complicated meshes. We do note that outside of the uniform quadrilateral mesh case, there is no theory developed for optimal choices of kernel scaling and orientation, and while improved error reduction is observed for some functions using the LSIAC filter rather than the tensor-product filter [67], we see that with respect to mesh refinement in Chapter 4 that this is not always the case. A more in-depth investigation of the choice of orientation and scaling for line filter variants is outside the scope of this thesis, but an important area needing further study. Next we describe the formulation for modal DG methods employed in this work.

## 2.2 Discontinuous Galerkin Schemes

For application in Chapters 5 and 6, we will be considering the numerical solution of the partial differential equation initial-value problem

$$u_t + \nabla \cdot \mathbf{F}(u) = \gamma(\nabla \cdot (\nabla u)) + f(\mathbf{x}, t), \quad (\mathbf{x}, t) \in \Omega \times (0, T], \tag{2.10}$$

$$u(\mathbf{x}, 0) = u_0(\mathbf{x}), \quad \mathbf{x} \in \Omega \subset \mathbb{R}^d, \quad d = 1, 2, \tag{2.11}$$

where $f$ denotes a source term and the solution is subject to appropriate Dirichlet or periodic boundary conditions. Where applicable we will denote the boundary of $\Omega$ by $\partial\Omega$ and the portion of that boundary upon which Dirichlet boundary conditions are specificed by $\Gamma_D$. We describe here a modal DG numerical method and leave the discussion of the DG spectral element method (DGSEM) until Chapter 6. We will first describe the discretization for the convective terms without sources, before discussing the discretization of higher-order terms. Combining terms from each discretization, we will be able to solve the parabolic and elliptic PDEs arising in Chapter 5 and hyperbolic PDEs in Chapter 6. We conclude this DG background section by discussing relevant implementation details.

### 2.2.1 Weak Formulation: 1st Order Terms

Our goal is to construct $u_h$ a piecewise polynomial approximation to $u$. To do so we will take a method-of-lines approach, that is we will discretize in space via DG to obtain a system of ordinary differential equations, and then evolve our approximation in time via a time-stepping scheme. To that end, we focus only on the spatial discretization here, leaving the choice of time-stepping to be decided later.

Partition the domain $\Omega \subset \mathbb{R}^d$, $d = 1, 2$, into elements $\{\Omega_e\}_{e=1}^N$ such that $\Omega = \cup_{e=1}^N \Omega_e$. For a degree $p$ approximation, we define the approximation space in which to search for $u_h$ at a given time step by

$$V_h^p = \{v_h \in L^2(\Omega) : v_h\big|_{\Omega_e} \in \mathbb{Q}^p(\Omega_e),\ e = 1, \ldots, N\}, \tag{2.12}$$

where $v_h\big|_{\Omega_e}$ denotes the restriction of $v_h$ to the element $\Omega_e$, and $\mathbb{Q}^p(\Omega_e)$ denotes the space of polynomials of degree $p$ or less defined on $\Omega_e$. To obtain our weak formulation we multiply both the first order portion of (2.10), and (2.11) by a test function $v$ and integrate over an element:

$$\int_{\Omega_e} u_t v \, d\Omega_e + \int_{\Omega_e} \nabla \cdot \mathbf{F}(u) v \, d\Omega_e = 0, \tag{2.13}$$

$$\int_{\Omega_e} u(\mathbf{x}, 0) v \, d\Omega_e = \int_{\Omega_e} u_0(\mathbf{x}) v \, d\Omega_e. \tag{2.14}$$

Applying integration by parts to our spatial derivative term, we obtain the weak form

$$\int_{\Omega_e} u_t v \, d\Omega_e - \int_{\Omega_e} \mathbf{F}(u) \cdot (\nabla v) \, d\Omega_e + \int_{\Gamma_e} \mathbf{n} \cdot (\mathbf{F} v) d\Gamma_e = 0, \tag{2.15}$$

$$\int_{\Omega_e} u(\mathbf{x},0)v \; d\Omega_e = \int_{\Omega_e} u_0(\mathbf{x})v \; d\Omega_e. \tag{2.16}$$

Here $\mathbf{n}$ denotes the outward pointing normal to the element boundary. At this point we now restrict our numerical approximation to exist in the discontinuous polynomial space $V_h^p$ defined in (2.12), that is we substitute $u$ with $u_h$ in the weak formulation. We further assume $v = v_h$ is also an element of $V_h^p$. We then obtain the final form

$$\frac{d}{dt}\int_{\Omega_e} u_h v_h \; d\Omega_e - \int_{\Omega_e} \mathbf{F}(u_h)\cdot(\nabla v_h) \; d\Omega_e + \int_{\Gamma_e} \mathbf{n}\cdot\left(\hat{\mathbf{F}}(u_h^-, u_h^+)v_h\right) \; d\Gamma_e = 0, \quad \forall v_h \in V_h^p, \quad e = 1,\ldots,N, \tag{2.17}$$

$$\int_{\Omega_e} u_h(\mathbf{x},0)v_h \; d\Omega_e = \int_{\Omega_e} u_0(\mathbf{x})v_h \; d\Omega_e \quad \forall v_h \in V_h^p, \quad e = 1,\ldots,N, \tag{2.18}$$

where we have introduced the numerical flux $\hat{\mathbf{F}}$ to handle the fact that the solution is doubly defined at element interfaces as continuity of the approximation is not enforced there. Here we use $v(x^+)$ (resp. $v(x^-)$) to denotes the limit of $v(x_n)$ and as $x_n \to x$ from the plus (resp. minus) side of an element. By convention, on a given edge, we use the minus superscript to denote the local element $\Omega_e$ , and the plus superscript to denote its neighbor. To clarify further, $v(x^-)$ is the value approaching the boundary from the interior of the element, and $v(x^+)$ the value approaching from the exterior.

Returning to the definition of the numerical flux, which in turn defines the DG method, we note that following properties must be satisfied:

1. Consistency: if $u^- = u^+ = u_h$ then $\hat{\mathbf{F}}(u,u) = \mathbf{F}(u)$.
2. Conservation: $n^- \cdot \hat{\mathbf{F}}(u^-,u^+) = -n^+ \cdot \hat{\mathbf{F}}(u^+,u^-)$.

The first requirement ensures consistency with the actual flux of the PDE, while the second ensure conservation of the numerical method, as the choice of $v_h = 1$ (equivalent to a finite volume method in this case) and summation over elements of the mesh in (2.17) yields

$$\frac{d}{dt}\int_{\Omega} u_h \; d\Omega = \sum_{\Gamma_e \subset \partial\Omega} \int_{\Gamma_e} \mathbf{n}\cdot\hat{\mathbf{F}}(u_h^-, u_h^+),$$

which states that the time rate of change of the conserved quantity $u_h$ will only depend on fluxes in and out of the domain along domain boundary as fluxes along internal edges cancel out. For discretization of convective flux terms, we will use the Local Lax-Friedrichs flux (LLF):

$$\hat{\mathbf{F}}(u_h^-, u_h^+) = \frac{1}{2}(f(u_h^-) + f(u_h^-)) + \frac{1}{2}\alpha_{LLF}[\![u_h]\!] \tag{2.19}$$

where $\alpha_{LLF} = \max_{u^-,u^+}(|\mathbf{n}\cdot\frac{d\mathbf{f}}{du}|)$, and $[\![u_h]\!] = \mathbf{n}^- u_h^- + \mathbf{n}^+ u_h^+$. Having specified the numerical flux, this formulation is complete for hyperbolic conservation laws up to computational and implementation details.

We next consider the discretization of the higher-order derivative terms.

### 2.2.2 Weak Formulation: Higher-Order Spatial Terms

Now suppose we wish to discretize an elliptic equation $\nabla \cdot (\nabla u) = f(\mathbf{x})$ subject to Dirichlet boundary conditions $u(\mathbf{x}) = g_D(\mathbf{x})$ for $\mathbf{x} \in \Gamma_D$. To do so, we will introduce an auxiliary variable $\mathbf{Q} = \nabla u$ and rewrite the second order problem as a first order system:

$$\mathbf{Q} = \nabla u \tag{2.20}$$

$$\nabla \cdot \mathbf{Q} = f. \tag{2.21}$$

We then proceed in an analogous manner to the first-order case. Multiplying by a test function, integrating by parts, and restricting the functions considered to the polynomial space, we have the 1st order semi-discrete system

$$\int_{\Omega_e} \tau_h \cdot \mathbf{Q}_h \, d\Omega_e = \int_{\Gamma_e} \mathbf{n} \cdot (\tau_h \hat{u}) \, d\Gamma_e - \int_{\Omega_e} (\nabla \cdot \tau_h) u_h \, d\Omega_e, \quad \forall v_h \in V_h^p, \quad e = 1, \ldots, N, \tag{2.22}$$

$$\int_{\Omega_e} f v_h \, d\Omega_e + \int_{\Omega_e} \nabla v_h \cdot (\mathbf{Q}_h) \, d\Omega_e - \int_{\Gamma_e} \mathbf{n} \cdot (v_h(\hat{\mathbf{Q}}_h)) \, d\Gamma_e = 0, \quad \forall v_h \in V_h^p, \quad e = 1, \ldots, N. \tag{2.23}$$

where $\mathbf{Q}_h$ has $d$ components, and $\tau_h = v_h \mathbf{I}_d$ is a tensor-style test function. Effectively, (2.22) is really $d$ equations, one for each component of $\mathbf{Q} = (q_1, \ldots, q_d)$, see the text by Giraldo for more details [71].

For the choice of numerical fluxes, the work of Arnold et al. [72] lists several types of numerical fluxes applicable for elliptic DG discretizations. In one dimension when we append the elliptic discretization to the convective discretization to solve parabolic problems, we will use central fluxes for both variables: $\hat{u} = \frac{1}{2}(u^- + u^+)$ and $\hat{\mathbf{Q}} = \frac{1}{2}(\mathbf{Q}^- + \mathbf{Q}^+)$. This is known as the Bassi-Rebay method [73], which is unfortunately unstable for purely elliptic problems [72]. As a result, for our two-dimensional tests we will consider the following alternative numerical fluxes which yield what is termed a Symmetric Interior Penalty (SIP) method. For the conserved variable we choose $\hat{u} = \frac{1}{2}(u_h^- + u_h^+)$, while for the auxiliary variable we choose $\hat{\mathbf{Q}} = \frac{1}{2}(\nabla u_h^- + \nabla u_h^+) - \alpha_{SIP} [\![ u_h ]\!]$. Here $\alpha_{SIP}$ weighs how much the jump at element interfaces is penalized, and is selected here as

$$\alpha_{SIP} = \frac{\eta_f (p+1)^2}{2|\mathbf{n} \cdot (\max(\Delta x^-, \Delta x^+), \max(\Delta y^-, \Delta y^+))|} \tag{2.24}$$

where $\eta_f = 1$ for $p > 0$ and $\eta_f = 2$ for $p = 0$ [74]. The Dirichlet boundary conditions are incorporated into the numerical flux and read if $\Gamma_e$ is boundary edge, then set $u^+ = g_D$. If information does not exist, such as for gradient values with Dirichlet boundaries, we simply set the exterior value equal to the interior value. For the case considered above we have

$$\hat{\mathbf{Q}}(u^-, g_D) = \nabla(u_h)^- - \alpha(u^- n^- + g_D n^+),$$
$$\hat{u}(u^-, g_D) = \frac{1}{2}(u^- + g_D).$$

With the boundary conditions accounted for, the numerical method is now defined up to implementation details.

### 2.2.3 Computational and Implementation Details: Hyperbolic Component

At this stage the semi-discrete numerical method is defined, and in a sense the whole numerical method up through time integration. From a computational and implementation standpoint however, it is beneficial to pin down the choice of polynomial basis and quadrature to actually realize the scheme. As used in the adaptive mesh DG experiments in Chapter 5, we will presume either a one-dimensional mesh, or a two-dimensional quadrilateral mesh where element edges are aligned with the Cartesian axes. We are not presuming uniformity of elements, but we will presume a 2:1 rule is satisfied by the mesh, meaning that any element is either the same size, half the size, or twice the size of a given neighboring element [75]. We will further make a choice of basis functions. We express $u_h$ in terms of an element local polynomial basis:

$$u_h\Big|_{\Omega_e}(\mathbf{x}, t) = \sum_{i=1}^{N_p} u_i^{(e)}(t)\Phi_i^{(e)}(\mathbf{x}). \tag{2.25}$$

where $N_p$ is the number of degrees in the polynomial space. For $\mathbb{Q}^p$ approximation spaces, we choose $N_p = (p+1)^d$ for $d = 1, 2$. We also select $v_h$ to be one of the basis functions: $v_h = \Phi_j^{(e)}$, $j = 1, \ldots, N_p$.

We select for basis functions in one dimension the orthonormal Legendre polynomials $\phi_{i-1}(x) = \sqrt{i - 1/2}P_{i-1}(x)$ defined in Appendix A, which are a hierarchical basis of polynomial orthogonal over $[-1, 1]$. Introducing the mapping $\xi_e(x) = \frac{2}{\Delta x_e}(2 - x_e)$ where $x_e$ is the center of the interval $\Omega_e$ and $\Delta x_e = |\Omega_e|$ is the width of the interval, we choose the element-wise orthogonal basis functions $\Phi_i^{(e)}(x) = \phi_i(\xi_e(x))$ so that

$$\int_{\Omega_e} \Phi_i^{(e)}(x)\Phi_j^{(e)}(x)\, d\Omega = |J^e|^{-1}\delta_{i,j}.$$

Here $|J^e| = \frac{2}{\Delta x_e}$ is the determinant of the Jacobian associated with the mapping $\xi_e(x)$. In two dimensions we consider the tensor-product orthonormal Legendre basis with

$$\Phi_i^{(e)}(\mathbf{x}) = \Phi_i(\Xi_e(\mathbf{x})) = \phi_{i_x}(\xi_e(x))\phi_{i_y}(\eta_e(y)).$$

Here $\Xi_e(\mathbf{x}) = [\xi_e(x), \eta_e(y)]$, where $\xi_e$ is as in the one-dimensional case, and $\eta_e = \frac{2}{\Delta y_e}(y - y_e)$ with $\Delta y_e$ being the height of the element, and $y_e$ the $y$-coordinate of the element center. Again orthogonality manifests as

$$\int_{\Omega_e} \Phi_i^{(e)}(\mathbf{x})\Phi_j^{(e)}(\mathbf{x})\, d\Omega = |J^e|^{-1}\delta_{i,j}$$

where $|J^e| = \frac{4}{\Delta x_e \Delta y_e}$ is the determinant of the Jacobian associated with the mapping $\Xi_e(\mathbf{x})$.

We focus first on the hyperbolic case. Substitution of the ansatz for $u_h$ and the choice of $v_h$ into (2.17), we obtain over a specific element the system

$$\frac{d}{dt}\left(\sum_{i=1}^{N_p} u_i^{(e)}(t) \int_{\Omega_e} \Phi_i^{(e)} \Phi_j^{(e)} \, d\Omega_e\right) - \int_{\Omega_e} \mathbf{F}(u_h) \cdot (\nabla \Phi_j^{(e)}) \, d\Omega_e + \int_{\Gamma_e} \mathbf{n} \cdot (\hat{\mathbf{F}}(u_h^-, u_h^+) \Phi_j^{(e)}) \, d\Gamma_e = 0, \; j = 1, \ldots, N_p. \tag{2.26}$$

For the choices of basis function described above, we will have orthogonality over a reference element so the first integral can be computed exactly:

$$\left(\sum_{i=1}^{N_p} u_i^{(e)}(t) \int_{\Omega_e} \Phi_i^{(e)} \Phi_j^{(e)} \, d\Omega_e\right) = \sum_{i=1}^{N_p} |J^e|^{-1} \delta_{i,j} u_i^{(e)} = |J^e|^{-1} u_j^{(e)}. \tag{2.27}$$

Put another way, the Mass-Matrix

$$(\mathbf{M}^{(e)})_{i,j} = \int_{\Omega_e} \Phi_i^{(e)} \Phi_j^{(e)} \, d\Omega_e = |J^e|^{-1} \delta_{i,j}$$

is diagonal for the orthonormal bases.

The second and third integrals, however, are generally not exactly integrable for arbitrary fluxes, hence we resort to quadrature to compute them. We make the approximation

$$\int_{\Omega_e} \mathbf{F}(u_h) \cdot (\nabla \Phi_j^{(e)}) \, d\Omega_e = (\mathbf{F}_V^{(e)})_j \approx \sum_{q=1}^{NQ_{vol}} \omega_q \mathbf{F}(u_h(\mathbf{x}_q^e)) \cdot (\nabla \Phi_j^{(e)}(\mathbf{x}_q^e)) |J_q^e|^{-1} \tag{2.28}$$

and

$$\int_{\Gamma_e} \mathbf{n} \cdot (\hat{\mathbf{F}}(u_h^-, u_h^+) \Phi_j^{(e)}) \, d\Gamma_e = (\mathbf{F}_S^{(e)})_j \approx \sum_{f_\Gamma \in \text{edges of } \Gamma_e} \sum_{q=1}^{NQ_{edge}} \omega_q \mathbf{n} \cdot (\hat{\mathbf{F}}(u_h^\pm(\mathbf{x}_q^e)) \Phi_j^{(e)}(\mathbf{x}_q^e)) |J_q^{f_\Gamma, e}|^{-1}. \tag{2.29}$$

Here $\{\omega_q, \mathbf{x}_q^e\}_{q=1}^{NQ_{vol}}$ are quadrature weights and nodes associated with the volume integral over $\Omega_e$, $\{\omega_q, \xi_q^e\}_{q=1}^{NQ_{face}}$ are the quadrature weights and nodes associated with the surface integral over of $\Gamma_e$, and $|J_q^e|^{-1}$ denotes the determinant of the Jacobian of the global-to-local coordinate transformation $(\Xi_e)^{-1} : [-1,1]^d \to \Omega_e$ at node $\xi_q$.

Presuming $d > 1$ (otherwise the surface integral needs no quadrature) $|J_q^{f_\Gamma, e}|^{-1}$ denotes the determinant of the Jacobian of the local-to-global coordinate transformation $(\xi_e)^{-1} : [-1,1]^{d-1} \to f_\Gamma$. For the modal DG methods employed in this work we will use Gauss-Legendre (GL) quadrature [76] which using $N$ points is exact for polynomials of degree $2N-1$, and whose quadrature points $\{\xi_{GL}\}$ lie in the interval $[-1,1]$. To construct higher dimensional quadrature we simply take tensor-products of the nodes and weights over $[-1,1]^d$ and then map these nodes to to the physical element by the inverse transforms $(\Xi_e)^{-1}$ and $(\xi_e)^{-1}$.

For the DGSEM method employed in Chapter 6 we use Legendre-Gauss-Lobatto (LGL) quadrature which for $N$ points is exact for polynomials of degree $2N-3$. This reduced exactness relative to GL quadrature is resultant from the method requiring that the quadrature nodes $\{\xi_{LGL}\}$ contain the boundary points $-1$ and 1.

Note: for the case of 2:1 quadrilateral meshes [75] where a larger element could neighbor two separate smaller elements, conservation is maintained via over-integration of the larger element's numerical fluxes. That is, the physical location of quadrature nodes for surface integral computation on the larger element is collocated with the node locations for its smaller shared edge neighbors.

Having addressed the issue of quadrature, we can express the scheme as

$$\mathbf{M}^{(e)}_{j,j}\frac{d}{dt}u^{(e)}_j = (\mathbf{F}^{(e)}_V)_j - (\mathbf{F}^{(e)}_S)_j, \quad j = 1,\ldots,N_p, \quad e = 1,\ldots,N. \tag{2.30}$$

Concatenating along the $j$ index, we obtain the matrix-vector expression for the scheme

$$\mathbf{M}^{(e)}\frac{d}{dt}\mathbf{u}^{(e)} = \mathbf{F}^{(e)}_V - \mathbf{F}^{(e)}_S, \quad e = 1,\ldots,N. \tag{2.31}$$

The initialization of the approximation from the initial data is given by the same procedure and in matrix-vector form is given by

$$\mathbf{M}^{(e)}\mathbf{u}^{(e)} = \mathbf{b}^{(e)}, \quad e = 1,\ldots,N \tag{2.32}$$

$$\tag{2.33}$$

where

$$(\mathbf{b}^{(e)})_j = \int_{\Omega_e} u_0(\mathbf{x})\Phi^{(e)}_j(\mathbf{x})\, d\Omega_e, \quad j = 1,\ldots,N_p \quad e = 1,\ldots,N \tag{2.34}$$

and the volume integral in (2.34) is treated in the same manner as (2.28).

### 2.2.4 Computational and Implementation Details: Elliptic Component

Now we consider the more complicated case of implementing the elliptic discretization. In the parabolic case where we have initial data, we can in fact solve first for $\mathbf{Q}_h$ and then for $u_h$ on an element-by-element basis. Unfortunately, in the elliptic case we cannot and will instead need to form and solve a global linear system. We sketch the implementation for the one-dimensional case, noting that the two-dimensional case follows a similar trend. Further details can be found in the text by [71].

Again we assume an orthonormal Legendre ansatz for our approximation

$$u_h\Big|_{\Omega_e}(x,t) = \sum_{i=1}^{N_p} u^{(e)}_i(t)\Phi^{(e)}_i(x), \tag{2.35}$$

but also for the auxiliary variable

$$Q_h\Big|_{\Omega_e}(x,t) = \sum_{i=1}^{N_p} q_i^{(e)}(t)\Phi_i^{(e)}(x), \tag{2.36}$$

and the forcing function

$$f_h\Big|_{\Omega_e}(x,t) = \sum_{i=1}^{N_p} f_i^{(e)}(t)\Phi_i^{(e)}(x). \tag{2.37}$$

We let $v_h = \tau_h = \Phi_j^{(e)}(x)$. The semi-discrete elliptic system of (2.22) and (2.23) becomes

$$\sum_{i=1}^{N_p} q_i^{(e)} \int_{\Omega_e} \Phi_i^{(e)}\Phi_j^{(e)}\, d\Omega_e = \int_{\Gamma_e} \mathbf{n}\cdot(\Phi_j^{(e)}\hat{u})\, d\Gamma_e - \sum_{i=1}^{N_p} u_i^{(e)} \int_{\Omega_e} \frac{d\Phi_j^{(e)}}{dx}\Phi_i^{(e)}\, d\Omega_e, \quad j=1,\ldots,N_p, \quad e=1,\ldots,N, \tag{2.38}$$

$$\sum_{i=1}^{N_p} f_i^{(e)} \int_{\Omega_e} \Phi_i^{(e)}\Phi_j^{(e)}\, d\Omega_e + \sum_{i=1}^{N_p} q_i^{(e)} \int_{\Omega_e} \frac{d\Phi_j^{(e)}}{dx}\Phi_i^{(e)}\, d\Omega_e - \int_{\Gamma_e} \mathbf{n}\cdot(\Phi_j^{(e)}\hat{\mathbf{Q}}_h)\, d\Gamma_e = 0, \quad j=1,\ldots,N_p, \quad e=1,\ldots,N. \tag{2.39}$$

In matrix-vector form we have

$$\mathbf{M}^{(e)}\mathbf{q}^{(e)} = \mathbf{D}_u^{(e)}\mathbf{u} - \mathbf{S}^{(e)}\mathbf{u}^{(e)} + \mathbf{b}_u^{(e)}, \quad e=1,\ldots,N, \tag{2.40}$$

$$\mathbf{M}^{(e)}\mathbf{f}^{(e)} = \mathbf{D}_q^{(e)}\mathbf{q} - \mathbf{S}^{(e)}\mathbf{q}^{(e)} + \mathbf{b}_q^{(e)}, \quad e=1,\ldots,N, \tag{2.41}$$

where again $\mathbf{M}^{(e)}$ is a element local mass matrix

$$(\mathbf{M}^{(e)})_{i,j} = \int_{\Omega_e} \Phi_i^{(e)}\Phi_j^{(e)}\, d\Omega_e = |J^e|^{-1}\delta_{i,j}, \tag{2.42}$$

and the stiffness matrix $\mathbf{S}^{(e)}$ is given by

$$(\mathbf{S}^{(e)})_{i,j} = \int_{\Omega_e} \frac{d\Phi_j^{(e)}}{dx}\Phi_i^{(e)}\, d\Omega_e = \int_{-1}^{1} \frac{d\Phi_j}{d\xi}\Phi_i\, d\xi. \tag{2.43}$$

Here $\mathbf{D}_u^{(e)}$, and $\mathbf{D}_q^{(e)}$ are flux matrices operating on the global vectors of $\mathbf{u}$ and $\mathbf{q}$, respectively, owing to the inter-element communication imposed by the numerical flux. Lastly, the boundary vectors $\mathbf{b}_u^{(e)}$, and $\mathbf{b}_q^{(e)}$ are responsible for the imposition of the Dirichlet boundary conditions through the numerical flux. In general, the choice of mesh and element indexing will affect the form of the flux matrices and the boundary vector, but we note that both are sparse, see [71]. In forming the global matrix vector system we concatenate the matrices and vectors to obtain:

$$\mathbf{M}\mathbf{q} = (\mathbf{D}_u - \mathbf{S})\mathbf{u} + \mathbf{b}_u$$
$$\mathbf{M}\mathbf{f} = (\mathbf{D}_q - \mathbf{S})\mathbf{q} + \mathbf{b}_q$$

where

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}^{(1)} & & \\ & \ddots & \\ & & \mathbf{M}^{(N)} \end{bmatrix}, \tag{2.44}$$

$$\mathbf{D}_u = \begin{bmatrix} (\mathbf{D}_u^{(1)})^T & \dots & (\mathbf{D}_u^{(N)})^T \end{bmatrix}^T, \tag{2.45}$$

$$\mathbf{D}_q = \begin{bmatrix} (\mathbf{D}_q^{(1)})^T & \dots & (\mathbf{D}_q^{(N)})^T \end{bmatrix}^T, \tag{2.46}$$

and

$$\mathbf{b}_w = [(\mathbf{b}_w^{(1)})^T, \dots, \mathbf{b}_w^{(N)})^T]^T \tag{2.47}$$

$$\tag{2.48}$$

for $w = u, q$. Setting up the linear system for $\mathbf{u}$ we obtain

$$\mathbf{A}\mathbf{u} = \mathbf{y}, \tag{2.49}$$

where

$$\mathbf{A} = \mathbf{C}_q \mathbf{M}^{-1} \mathbf{C}_u \tag{2.50}$$

and

$$\mathbf{y} = \mathbf{M}\mathbf{f} - \mathbf{C}_q \mathbf{M}^{-1} \mathbf{b}_u - \mathbf{b}_q \tag{2.51}$$

with

$$\mathbf{C}_w = \mathbf{D}_w - \mathbf{S},$$

for $w = u, q$. Owing to the sparsity of all the matrices under consideration, typically far fewer than the $N(p+1) \times N(p+1)$ total entries in the linear system $\mathbf{A}$ need be stored and sparse arrays can be utilized.

#### 2.2.4.1 Combination of spatial discretization terms

Having described the spatial discretization for both first- and second-order derivative terms we can now combine the local discretization equations to solve the original PDE (2.10). We provide the form for the one-dimensional case for purposes of conciseness. Combining the element-local equations (2.40), (2.41), and (2.31) and appending the element-local auxiliary equation (2.40), we obtain the semi-discrete system

$$\mathbf{M}^{(e)}\mathbf{q}^{(e)} = \mathbf{d}_{S,u}^{(e)}\mathbf{u} - \mathbf{S}^{(e)}\mathbf{u}^{(e)}, \quad e = 1, \dots, N, \tag{2.52}$$

$$\mathbf{M}^{(e)}\frac{d}{dt}\mathbf{u}^{(e)} = \mathbf{F}_V^{(e)} - \mathbf{F}_S^{(e)} + \gamma\Big(\mathbf{M}^{(e)}\mathbf{f}^{(e)} - \mathbf{d}_{S,q}^{(e)}\mathbf{q}^{(e)} + \mathbf{S}^{(e)}\mathbf{q}^{(e)}\Big), \quad e = 1, \dots, N, \tag{2.53}$$

where as the vector $\mathbf{u}$ is known at a given time step, the vectors $\mathbf{d}_{S,q}^{(e)}$ and $\mathbf{d}_{S,u}^{(e)}$ account for the numerical surface fluxes of the elliptic components and associated boundary conditions in the same way as the

hyperbolic surface fluxes $\mathbf{F}_S^{(e)}$. The solution procedure first solves for the auxiliary variable vector $\mathbf{q}$ from the known data and then for the time derivative of $\frac{d}{dt}\mathbf{u}$.

### 2.2.5 Time Evolution Schemes

Having described the discretization of spatial derivative terms for the unsteady equations, we are left with an ordinary differential equation (ODE) initial value problem (IVP) of the form

$$\frac{d}{dt}\mathbf{u} = \mathbf{r}, \tag{2.54}$$

$$\mathbf{u}(t_0) = \mathbf{u}_0. \tag{2.55}$$

For time-stepping, we consider explicit Runge-Kutta methods. For an ODE IVP

$$\frac{d}{dt}u = f(u(t), t), \tag{2.56}$$

$$u(t_0) = u_0, \tag{2.57}$$

a general RK discretization method is of the form

$$\begin{aligned} y_i &= u^n + \Delta t \sum_{j=1}^{s} a_{ij} f(y_j, t_n + c_j \Delta t), \\ u^{n+1} &= u^n + \Delta t \sum_{j=1}^{s} b_j f(y_j, t_n + c_j \Delta t), \end{aligned} \tag{2.58}$$

where $\mathbf{A} \in \mathbb{R}^{s \times s}$, and $\mathbf{b}, \mathbf{c} \in \mathbb{R}^s$. Up to a change of naming convention ($u \to \mathbf{u}$, $f \to \mathbf{r}$), and a choice of Butcher Tableau ($\mathbf{A}$, $\mathbf{b}$, and $\mathbf{c}$) we have a time evolution scheme for our PDE discretization. A list of RK methods employed in this thesis and their associated definitions is given in Appendix B.

#### 2.2.5.1 A note on time-stepping stability

For explicit method-of-lines approaches, the time step must be chosen sufficiently small so that numerical wavespeed of the underlying discretization is greater than the actual wave speed of the underlying system. The amount by which the numerical wavespeed must exceed this physical wave speed is known as the Courant-Friedrichs-Lewy (CFL) condition and serves as one constraint on the fully discrete numerical method for stability. Denoting the magnitude of the physical system's wavespeed by $|c|$, the CFL condition requires $|c|\frac{\Delta t}{\Delta x} \leq CFL$. For explicit time-stepping of hyperbolic DG approximations the CFL condition is typically set as $\frac{1}{2p+1}$ [77]. For parabolic systems, the denominator scales as $\Delta x^2$ divided by the diffusion coefficient $\gamma$ [26]. In the case of parabolic systems with convective terms, we then choose the time step to satisfy

$$\Delta t \leq CFL \min\Big\{\frac{\Delta x_{min}}{|c|}, \frac{\Delta x_{min}^2}{\gamma}\Big\},$$

where $\Delta x_{min} = \min_e\{\Delta x_e\}$. In all explicit time-stepping results presented in this work, an additional safety factor of at least $\frac{1}{10}$ was applied to the CFL to minimize any effects of time-stepping on the spatial discretization or enhancement technique being considered.

### 2.3 Multiresolution Analysis

In order to contextualize the enhancement procedure and understand the multiwavelet indicator used in Chapter 5, we introduce the multiresolution analysis (MRA) framework. While the enhancement procedure does not actually make explicit use of multiwavelet details, it implicitly defines them upon application. Later, we will utilize these details for adaptivity indication. These details are important in the adaptive solution of the PDEs considered in Section 5.4 where the multiwavelet detail indicator of [61] is used. For simplicity, we consider multiresolution analysis specifically for the case of hierarchies of dyadically uniformly subdivided meshes, and, in the multidimensional case, we will restrict ourselves to quadrilateral elements. Note that a multiresolution analyses naturally exists and can be constructed for other hierarchies (see Yu et al. for triangular-mesh multiwavelets, and Gerhard et al. for wavelet-free methods for nonuniform meshes [56, 78]). For more in-depth treatments of MRA we refer the reader to [42, 52, 78–80].

For clarity, we first consider the one-dimensional case and follow the presentation of [81, 82]. Define an initial mesh $\mathcal{T}^{(0)} = [-1, 1]$ to consist of a single element. Let $\mathcal{T}^{(n)}$ denote the mesh obtained by uniformly subdividing every element in $\mathcal{T}^{(n-1)}$, i.e.

$$\mathcal{T}^{(n)} = \cup_{j=0}^{2^n-1} I_j^n, \tag{2.59}$$

where $I_j^n = (-1 + 2^{1-n}j, -1 + 2^{1-n}(j+1))$. Over each mesh, we define the approximation space

$$V_n^p = \{v \ : v\big|_{I_j^n} \in \mathbb{P}^p(I_j^n), j = 0, \ldots, 2^n - 1\}. \tag{2.60}$$

Owing to the dyadic hierarchy of the meshes, the approximation spaces are nested:

$$V_0^p \subset V_1^p \subset \ldots \subset V_n^p \subset \ldots. \tag{2.61}$$

Denote by $\{\phi_k(x)\}_{k=0}^p = \{\phi_{k,0}^0(x)\}_{k=0}^p$ an orthonormal basis for the coarsest mesh consisting of one element. These basis functions are called *scaling functions*. We can obtain bases for any $V_n^p$ by scaling and translating this initial basis of scaling functions. The approximation space $V_n^p$ can also be defined as the span of $\{\phi_{k,j}^n\}_{k,j}$ where

$$\phi_{k,j}^n = 2^{n/2}\phi_k\big(2^n(x+1) - 2j - 1\big), \quad k = 0, \ldots, p, \quad j = 0, \ldots, 2^n - 1. \tag{2.62}$$

As such, a function $u$ can be approximated in $V_n^p$ by

$$u_h^{(n)}(x) = (\mathcal{P}^n u)(x) = \sum_{j=0}^{2^n-1} \sum_{k=0}^{p} s_{k,j}^n \phi_{k,j}^n(x), \tag{2.63}$$

where

$$s_{k,j}^n = \langle u, \phi_{k,j}^n \rangle_{I_j^n}, \quad k = 0, \ldots, p, \qquad j = 0, \ldots, 2^n - 1. \tag{2.64}$$

and $\mathcal{P}^n$ denotes the projection operator onto $V_n^p$.

To transition from a coarse approximation space $V_n^p$ to a finer approximation space $V_{n+1}^p$ additional information is required. This information is contained within the multiwavelet space $W_n^p$ defined to be the orthogonal compliment of $V_n^p$ in $V_{n+1}^p$:

$$V_{n+1}^p = V_n^p \oplus W_n^p, \qquad W_n^p \subset V_{n+1}^p, \qquad W_n^p \perp V_n^p. \tag{2.65}$$

Inductively, this hierarchy allows the expression of the fine mesh approximation space as the direct sum of the coarsest mesh approximation spaces and the intermediary wavelet spaces:

$$V_n^p = V_0^p \oplus W_0^p \oplus W_1^p \oplus \cdots \oplus W_{n-1}^p.$$

An orthonormal basis for these multiwavelet spaces can be obtained via a Gram-Schmidt orthogonalization procedure and obtain uniqueness by imposing additional conditions [80]. Here the special case of Alpert's multiwavelets (see [42]) is considered, where orthonormal Legendre polynomials are chosen as scaling functions,

$$\{\phi_k(x)\}_{k=0}^p = \{\phi_{k,0}^0\}_{k=0}^p = \begin{cases} \sqrt{\frac{2k+1}{2}} P_k(x), & x \in [-1, 1] \\ 0, & \text{Else} \end{cases} \tag{2.66}$$

and the multiwavelet bases for the spaces $W_n^p$ satisfy the properties given below. Denoting the multiwavelets on the coarsest scale by

$$\{\psi_k(x)\}_{k=0}^p = \{\psi_{k,0}^0\}_{k=0}^p,$$

multiwavelet bases $\{\psi_{k,j}^n\}_{k,j}$ are obtained for each of the multiwavelet spaces $W_n^p$ by dilation and scaling where

$$\psi_{k,j}^n = 2^{n/2} \psi_k(2^n(x+1) - 2j - 1), \quad k = 0, \ldots, p, \qquad j = 0, \ldots, 2^n - 1. \tag{2.67}$$

These multiwavelets satisfy the following properties relevant to our analysis:

1. Orthogonality conditions: $\langle \phi_{k,i}^n, \psi_{\ell,j}^n \rangle_{I_j^n} = 0$ and $\langle \psi_{k,i}^n, \psi_{\ell,j}^n \rangle_{I_j^n} = \delta_{i,j}\delta_{k,\ell}$.

2. Support conditions: $\text{supp}(\psi_{k,j}^n) = \text{supp}(\phi_{k,j}^n) = I_j^n$.

3. Moment conditions: $\langle x^m, \psi_{k,j}^n \rangle_{I_j^n} = 0$ for $m \leq k + p + 1$.

We refer the interested reader to the original work of Alpert [42] for details as to their construction and additional properties satisfied by these piecewise polynomial multiwavelets, and to the thesis of Vuik [81] for explicit formulas for $\psi$ for the first few polynomials degrees. We note that by (2.65) and the orthogonality conditions, the detail necessary for transition from a coarse mesh approximation space to a fine mesh approximation space

$$\sum_{j=0}^{2^n-1} d_{k,j}^n \psi_{k,j}^n(x) = \left(\mathcal{P}^{n+1}u\right)(x) - \left(\mathcal{P}^n u\right)(x), \tag{2.68}$$

are obtained simply by projection of the initial data onto the multiwavelet space:

$$d_{k,j}^n = \langle u, \psi_{k,j}^n \rangle_{I_j^n}.$$

Hence, a fine mesh representation is simply a coarse mesh representation with additional multiwavelet details:

$$u_h^{(n+1)}(x) = \sum_{j=0}^{2^n-1} \sum_{k=0}^{p} s_{k,j}^n \phi_{k,j}^n(x) + \sum_{j=0}^{2^n-1} \sum_{k=0}^{p} d_{k,j}^n \psi_{k,j}^n(x). \tag{2.69}$$

When considering the enhancement technique for adaptive PDEs in later sections, it is beneficial to compute the coarse mesh scaling and detail coefficients from the fine mesh scaling coefficients. To that end we define the quadrature mirror filters [81] useful in performing this exact task. Denote the vectors of scaling and detail coefficients by $\mathbf{s}_j^n = [s_{0,j}^n, \ldots, s_{p,j}^n]^T$ and $\mathbf{d}_j^n = [d_{0,j}^n, \ldots, d_{p,j}^n]^T$, respectively, then

$$\mathbf{s}_j^{n-1} = H^{(0)}\mathbf{s}_{2j}^n + H^{(1)}\mathbf{s}_{2j+1}^n \tag{2.70}$$

$$\mathbf{d}_j^{n-1} = G^{(0)}\mathbf{s}_{2j}^n + G^{(1)}\mathbf{s}_{2j+1}^n, \tag{2.71}$$

where $H^{(\,)}$ and $G^{(\,)}$ are $(p+1)\times(p+1)$ matrices with entries given by

$$H^{(0)}(\ell, r) = \langle \phi_{\ell,0}^0, \phi_{r,0}^1 \rangle_{I_0^1}, \qquad H^{(1)}(\ell, r) = \langle \phi_{\ell,0}^0, \phi_{r,1}^1 \rangle_{I_1^1}, \tag{2.72}$$

$$G^{(0)}(\ell, r) = \langle \psi_{\ell,0}^0, \phi_{r,0}^1 \rangle_{I_0^1}, \qquad G^{(1)}(\ell, r) = \langle \psi_{\ell,0}^0, \phi_{r,1}^1 \rangle_{I_1^1}, \tag{2.73}$$

for $\ell, r = 0, \ldots, p$.

One benefit of restricted uniform hierarchical meshes of the present analysis is that these matrices for transitioning detail and scaling coefficients between scales are fixed and only need be computed once. For more general meshes these quadrature mirror filters may be spatially varying, which increases the computational cost of the decomposition into multiwavelet and scaling coefficients.

### 2.3.1 Extension to Two Dimensions

The same techniques generalize to multiple dimensions easily via tensor-products of the approximation spaces and their associated bases. Here we follow the presentation of [61]. To simplify notation, denote element indices on level $n$ by $\mathbf{j} = (j_x, j_y)$ for $j_x, j_y \in \{0, \ldots, 2^{n-1}\}$ and modal indices by $\mathbf{k} = (k_x, k_y)$ for $k_x, k_y \in \{0, \ldots, p\}$. Let $\mathcal{T}^{(0)} = [-1, 1]^2$ be the coarsest level mesh consisting of a single element, and $\mathcal{T}^{(n)}$ the mesh obtained by uniform subdivision into quadrants of each element of $\mathcal{T}^{(n-1)}$. The elements on mesh level $n$ are given by $Q^n_{\mathbf{j}} = I^n_{j_x} \otimes I^n_{j_y}$. Define the piecewise polynomial approximation space over mesh level $n$ to be

$$\mathbf{V}^p_n = \{v \ : v\big|_{I^n_j} \in \mathbb{Q}^p(Q^n_{\mathbf{j}}),\ Q^n_{\mathbf{j}} \in \mathcal{T}^{(n)}\}. \tag{2.74}$$

A basis for this approximation space is given by $\{\phi^n_{\mathbf{k},\mathbf{j}} \ : \ \|\mathbf{k}\|_{\ell^\infty} \leq p,\ Q^n_{\mathbf{j}} \in \mathcal{T}^{(n)}\}$, where

$$\phi^n_{\mathbf{k},\mathbf{j}}(\mathbf{x}) = \phi^n_{k_x,j_x}(x)\phi^n_{k_y,j_y}(y). \tag{2.75}$$

The bases for the multiwavelet subspaces are obtained similar to the one-dimensional case except now there are three multiwavelet spaces denoted by $\mathbf{W}^p_{n,\alpha}, \mathbf{W}^p_{n,\beta}$, and $\mathbf{W}^p_{n,\gamma}$. The bases for these subspaces are given by

$$\{\psi^{n,\alpha}_{\mathbf{k},\mathbf{j}} \ : \ \|\mathbf{k}\|_{\ell^\infty} \leq p,\ Q^n_{\mathbf{j}} \in \mathcal{T}^{(n)}\}, \tag{2.76}$$

$$\{\psi^{n,\beta}_{\mathbf{k},\mathbf{j}} \ : \ \|\mathbf{k}\|_{\ell^\infty} \leq p,\ Q^n_{\mathbf{j}} \in \mathcal{T}^{(n)}\}, \tag{2.77}$$

and

$$\{\psi^{n,\gamma}_{\mathbf{k},\mathbf{j}} \ : \ \|\mathbf{k}\|_{\ell^\infty} \leq p,\ Q^n_{\mathbf{j}} \in \mathcal{T}^{(n)}\}, \tag{2.78}$$

respectively, where

$$\psi^{n,\alpha}_{\mathbf{k},\mathbf{j}}(\mathbf{x}) = \psi^n_{k_x,j_x}(x)\phi^n_{k_y,j_y}(y), \tag{2.79}$$

$$\psi^{n,\beta}_{\mathbf{k},\mathbf{j}}(\mathbf{x}) = \phi^n_{k_x,j_x}(x)\psi^n_{k_y,j_y}(y), \tag{2.80}$$

and

$$\psi^{n,\gamma}_{\mathbf{k},\mathbf{j}}(\mathbf{x}) = \psi^n_{k_x,j_x}(x)\psi^n_{k_y,j_y}(y). \tag{2.81}$$

Note that $W^p_{n,\alpha}$ which has multiwavelet functions for the x-component of the tensor-product basis can be thought of as corresponding to the x-direction, similarly, $W^p_{n,\beta}$ and $W^p_{n,\gamma}$ correspond to the $y$- and $xy$-directions respectively [81].

Similar to the one-dimensional case, the approximation in $\mathbf{V}^p_{n+1}$ is expressible as a sum of approximations in $\mathbf{V}^p_n, \mathbf{W}^p_{n,\alpha}, \mathbf{W}^p_{n,\beta}$, and $\mathbf{W}^p_{n,\gamma}$:

$$\mathcal{P}^{n+1}u = \sum_{Q_{\mathbf{j}}^{n+1}\in\mathcal{T}^{(n+1)}} \sum_{\|\mathbf{k}\|_{\ell^\infty}\le p} s_{\mathbf{k},\mathbf{j}}^{n+1}\phi_{\mathbf{k},\mathbf{j}}^{n+1} \tag{2.82}$$

$$= \sum_{Q_{\mathbf{j}}^{n}\in\mathcal{T}^{(n)}} \sum_{\|\mathbf{k}\|_{\ell^\infty}\le p} s_{\mathbf{k},\mathbf{j}}^{n}\phi_{\mathbf{k},\mathbf{j}}^{n} \tag{2.83}$$

$$+ \sum_{Q_{\mathbf{j}}^{n}\in\mathcal{T}^{(n)}} \sum_{\|\mathbf{k}\|_{\ell^\infty}\le p} d_{\mathbf{k},\mathbf{j}}^{n,\alpha}\psi_{\mathbf{k},\mathbf{j}}^{n,\alpha} + d_{\mathbf{k},\mathbf{j}}^{n,\beta}\psi_{\mathbf{k},\mathbf{j}}^{n,\beta} + d_{\mathbf{k},\mathbf{j}}^{n,\gamma}\psi_{\mathbf{k},\mathbf{j}}^{n,\gamma}.$$

Again, the coefficients in the expansions are determined by projection of $u$ onto the scaling function and multiwavelet bases:

$$s_{\mathbf{k},\mathbf{j}}^{n} = \langle u, \phi_{\mathbf{k},\mathbf{j}}^{n}\rangle_{Q_{\mathbf{j}}^{n}}, \quad \|\mathbf{k}\|_{\ell^\infty} \le p, \qquad Q_{\mathbf{j}}^{n} \in \mathcal{T}^{(n)}, \tag{2.84}$$

$$d_{\mathbf{k},\mathbf{j}}^{n} = \langle u, \psi_{\mathbf{k},\mathbf{j}}^{n,w}\rangle_{Q_{\mathbf{j}}^{n}}, \quad \|\mathbf{k}\|_{\ell^\infty} \le p, \qquad Q_{\mathbf{j}}^{n} \in \mathcal{T}^{(n)}, \tag{2.85}$$

where $w = \alpha, \beta, \gamma$. Inductively, a decomposition in multiwavelet approximation spaces direct-summed with the coarsest scaling approximation space exists. An illustration of the decomposition idea in two-dimensions is given in Figure 2.2 where $\mathbf{W}_n^p$ denotes the direct sum of the multiwavelet spaces $\mathbf{W}_{n,\alpha}^p, \mathbf{W}_{n,\beta}^p$, and $\mathbf{W}_{n,\gamma}^p$.

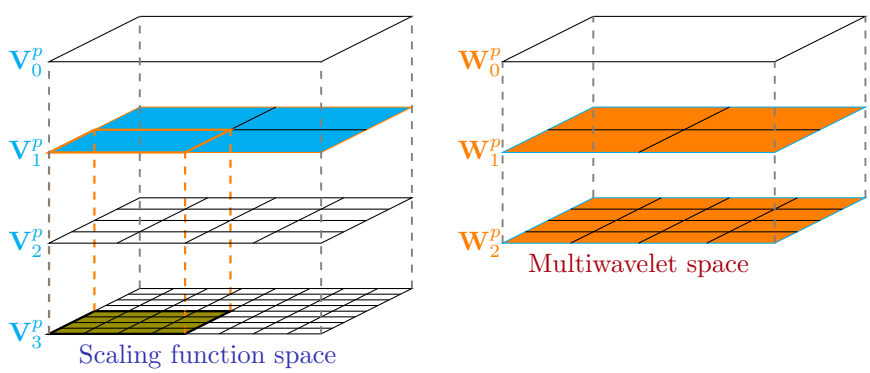


Figure 2.2 Illustration of the two-dimensional multiwavelet idea. The approximation at the finest level, $\mathbf{V}_3^p$, can be represented as the approximation from the $\mathbf{V}_0^p$ scaling function space plus the multiwavelet coefficients from multiwavelet spaces $\mathbf{W}_0^p$,$\mathbf{W}_1^p$, and $\mathbf{W}_2^p$.

The quadrature mirror filters previously defined can also be used in higher dimensions to compute the coarse mesh scaling and multiwavelet coefficients from the fine mesh scaling coefficients [83]:

$$s^n_{\mathbf{k},\mathbf{j}} = \sum_{\tilde{j}_x,\tilde{j}_y=0}^{1} \sum_{r_x,r_y=0}^{p} H^{(\tilde{j}_x)}(k_x,r_x)H^{(\tilde{j}_y)}(k_y,r_y)s^{n+1}_{\mathbf{r},2\mathbf{j}+\tilde{\mathbf{j}}}, \tag{2.86}$$

$$d^{n,\alpha}_{\mathbf{k},\mathbf{j}} = \sum_{\tilde{j}_x,\tilde{j}_y=0}^{1} \sum_{r_x,r_y=0}^{p} G^{(\tilde{j}_x)}(k_x,r_x)H^{(\tilde{j}_y)}(k_y,r_y)s^{n+1}_{\mathbf{r},2\mathbf{j}+\tilde{\mathbf{j}}}, \tag{2.87}$$

$$d^{n,\beta}_{\mathbf{k},\mathbf{j}} = \sum_{\tilde{j}_x,\tilde{j}_y=0}^{1} \sum_{r_x,r_y=0}^{p} H^{(\tilde{j}_x)}(k_x,r_x)G^{(\tilde{j}_y)}(k_y,r_y)s^{n+1}_{\mathbf{r},2\mathbf{j}+\tilde{\mathbf{j}}}, \tag{2.88}$$

$$d^{n,\gamma}_{\mathbf{k},\mathbf{j}} = \sum_{\tilde{j}_x,\tilde{j}_y=0}^{1} \sum_{r_x,r_y=0}^{p} G^{(\tilde{j}_x)}(k_x,r_x)G^{(\tilde{j}_y)}(k_y,r_y)s^{n+1}_{\mathbf{r},2\mathbf{j}+\tilde{\mathbf{j}}}. \tag{2.89}$$

### 2.3.2 Relationship between Legendre Modal Coefficients and Scaling Coefficients

It would be natural to presume that there is some relationship between the scaling coefficients used in the scaling function expansion (2.82) and the modal coefficients used in the DG expansion (2.25), as both are effectively built on the scaled and shifted Legendre polynomials. A relationship between these is also important for enabling the computation of detail coefficients when refining modal DG approximations. To that end we briefly describe the relationship between the two sets of coefficients. Suppose that the modal element index $(e)$ and the MRA hierarchical indices $n, j$ in one dimension ($n, \mathbf{j}$ in two dimensions) are such that $\Omega_e = I^n_j$ ($\Omega_e = Q^n_{\mathbf{j}}$). Then it can be shown by projection of the DG approximation onto the scaling function basis for $\Omega_e$ (see [81],[62]) that in one dimension:

$$s^n_{k,j} = 2^{-n/2}u^{(e)}_k,\ k = 0,\ldots,p \tag{2.90}$$

where to match prior notation of indexing we have $i = k + 1$, and in two dimensions:

$$s^n_{\mathbf{k},\mathbf{j}} = 2^{-n}u^{(e)}_i,\ \|\mathbf{k}\|_{\ell^\infty} \leq p, \tag{2.91}$$

where $i = k_x + 1 + (p+1)(k_y - 1)$.

### 2.3.3 Local vs. Global MRA

We note for use in the adaptive mesh DG experiments in Chapter 5 that the enhancement procedure we develop need not start on a mesh resultant from dyadic splitting. As result rather than start a global MRA described above, i.e. the mesh for $n = 0$ is $[-1,1]^d$, we can instead choose to compute local multiresolution analyses (see [61]) where we initialize a level $n = 0$ multiresolution analysis for each element $\Omega_e$ of some initial coarse mesh. We describe the idea in one dimension as the two-dimensional case follows like in the initial presentation of MRA. Choose scaling functions $\tilde{\phi}^0_{k,0} = \sqrt{\frac{2}{\Delta x_e}}\phi_k(\xi_e(x))$ and in turn multiwavelets $\tilde{\psi}^0_{k,0} = \sqrt{\frac{2}{\Delta x_e}}\psi_k(\xi_e(x))$. The multiresolution analyses generated by these choices of scaling and

multiwavelet functions satisfy the same orthogonality properties described previously except $I_j^n$ now represents the $j$th interval in the $n$-times repeated dyadic subdivision of $\Omega_e$. We use the tilde notation $\tilde{d}_{k,j}^n$ and $\tilde{s}_{k,j}^n$ to denote the detail and scaling coefficients resultant from this local subdivision. The relationship between DG modes and scaling coefficients (2.90) and (2.91) is the same except now $n = 0$ corresponds to the original element $\Omega_e$. For more details, see [61].

CHAPTER 3

# APPLICATION OF SIAC FILTERS TO PARTICLE-IN-CELL DATA

Portions of this chapter are based on a manuscript "Denoising Particle-In-Cell Data via Smoothness-Increasing Accuracy-Conserving Filters with Application to Bohm Speed Computation" submitted to the Journal of Computational Physics by Matthew J. Picklo, Qi Tang, Yanzeng Zhang, Jennifer K. Ryan, and Xian-Zhu Tang. In Section 3.6, the co-author contributions to the chapter are described. In this chapter, I have elected to focus primarily on my contribution to the aforementioned work in the form of filter construction and application.

## 3.1 Chapter Introduction

With the high-dimensionality of the equations governing plasma kinetics in six dimensional phase space [84], traditional numerical techniques for solving partial differential equations (PDE) suffer from the curse of dimensionality. This motivated the development and application of Particle-In-Cell (PIC) methods that have been a staple in plasma kinetic simulations [85, 86]. The particle-based approach provides simplicity and geometric flexibility, but these benefits come at the expense of significant amounts of noise, which scales as $1/\sqrt{N}$ with $N$ the number of particle markers in a cell. Remarkably, finite amounts of noise, even at substantial level, does not generally prevent a physics-wise meaningful PIC simulation. Unfortunately, post-run physics analysis is hampered by the noisy diagnostics. The difficulty is aggravated by the fact that the underlying physics is usually described, in the standard statistical physics approach, by velocity moments of the particle distribution function $f(\mathbf{x}, \mathbf{v}, t)$,

$$\left\langle v_i^k \right\rangle = \int v_i^k f(\mathbf{x}, \mathbf{v}, t)\, d^3\mathbf{v}, \tag{3.1}$$

with $v_i$ the Cartesian component of $\mathbf{v}$. For example, the thermodynamic state variables have the density $n(\mathbf{x}, t) \equiv \left\langle \mathbf{v}^0 \right\rangle$, the flow $\mathbf{U}(\mathbf{x}, t) \equiv \langle \mathbf{v} \rangle$, and the temperature $T(\mathbf{x}, t) \equiv (m/2)\left(\left\langle \mathbf{v}^2 \right\rangle - \mathbf{U}^2\right)$. Closure quantities such as the plasma heat flux involve even higher order velocity moments, for example,

$$\mathbf{q} \equiv \frac{m}{2}\left\langle (\mathbf{v} - \mathbf{U})^3 \right\rangle. \tag{3.2}$$

The general rule of thumb is that the particle noise becomes further exaggerated in higher order moment quantities. This high level of noise in the post-processed physical quantities hampers our ability to gain deep insights into the underlying physics.

A recent work on the Bohm criterion analysis [87] that establishes the lower bound of the plasma outflow speed, also known as the Bohm speed [88], in a wall-bounded plasma with finite collisionality, illustrates the extent one has to go to overcome the PIC noise in order to decipher the underlying physics. Specifically, the Bohm speed is predicted by theoretical analysis to take the form of

$$u_{Bohm} \equiv \sqrt{\frac{Z\beta T_{ex}^{se} + 3T_{ix}^{se}}{m_i}}, \tag{3.3}$$

where

$$\beta \equiv \frac{3 - \frac{3+2\alpha}{Ze\Gamma_i^{se}}\left(\frac{\partial q_n^i}{\partial \phi} + \frac{Q_{ii}}{E}\right) + \frac{\alpha}{e\Gamma_e^{se}}\left(\frac{\partial q_n^e}{\partial \phi} + \frac{Q_{ee}+Q_{ei}}{E}\right)}{1 + \frac{1+\alpha}{e\Gamma_e^{se}}\left(\frac{\partial q_n^e}{\partial \phi} + \frac{Q_{ee}+Q_{ei}}{E}\right)}, \tag{3.4}$$

with $\Gamma_{e,i} = n_{e,i}u_{e,ix}$, $\partial q_n^{e,i}/\partial\phi = -(\partial q_n^{e,i}/\partial x)/E$, and $\alpha = -R_T/(n_e\partial T_{ex}/\partial x)$. Here the superscript $se$ denotes the spatial location at which all quantities are to be evaluated to find the local Bohm speed. The plasma transport physics are in the form of local electric field $E$, which is proportional to spatial gradient of the potential ($\phi$), particle and heat fluxes (which are denoted by $\Gamma_{e,i}$ and $q_n^{e,i}$, respectively), collisional energy exchange between electrons ($Q_{ee}$) and between electrons and ions ($Q_{ei}$), and the thermal force ($R_T$). Most of these quantities are simply velocity moments of the computed distribution function, or the velocity moments of the Coulomb collisional integral, and quantities that are derived from them. The PIC noise thus makes it an extremely difficult task to directly compare the numerical simulation results with theoretical predictions. The authors of [87] overcame this hurdle by performing long-time simulations after a steady state is reached, and then applying time averaging over an extremely long-period of PIC data to reduce the PIC noise. With the noise at a sufficiently low level, a comparison of the predicted (3.3) and actual speeds could be performed. This ensemble averaging effectively increases $N$ from $10^{3-4}$ markers per cell to tens of millions per cell by accumulating data over time. Powerful and computationally expensive as this approach is, many systems do not admit steady state solutions. To decipher subtle transport physics from PIC simulations, one must therefore explore alternative approaches to effectively deal with PIC noise when modestly large $N$ and limited time-window averaging are available. Our work aims to propose and explore a distinctive approach tailored for PIC moments.

Here we address this issue through the use of SIAC filters which were originally developed as a post-processor for extracting superconvergence from finite-element method (FEM) based numerical approximations [5]. The purpose of this chapter is to demonstrate the effectiveness of SIAC filtering in reducing noise in pointwise data arising from PIC simulations. We will show that the application of specially designed SIAC convolution kernels can selectively remove high frequency oscillations without polluting physically-relevant low-frequency information, and thereby improve approximation quality.

Extensions of the filtering procedure for non-periodic data are constructed, and results are shown for how SIAC filtering schemes can reduce the quantity of data needed in computing the Bohm speed, an important parameter in plasma physics bounding from below the ion velocity to a charged boundary.

The rest of the chapter is organized as follows. In Section 3.2 we describe a procedure for initializing an approximation in a continuous variable from discrete data using Lagrange polynomials. In Section 3.3 we describe the construction of symmetric and position-dependent SIAC kernels which we convolve with our initialization dampen high-frequency oscillations. Boundary treatments as well as adaptive kernel scalings are also discussed. Lastly, in Section 3.4 we provide numerical examples detailing the damping effects of symmetric kernel on periodic data, adaptively scaled and position-dependent generalized spline kernels for non-periodic data, and data reduction enabled by SIAC filtering in Bohm speed computation.

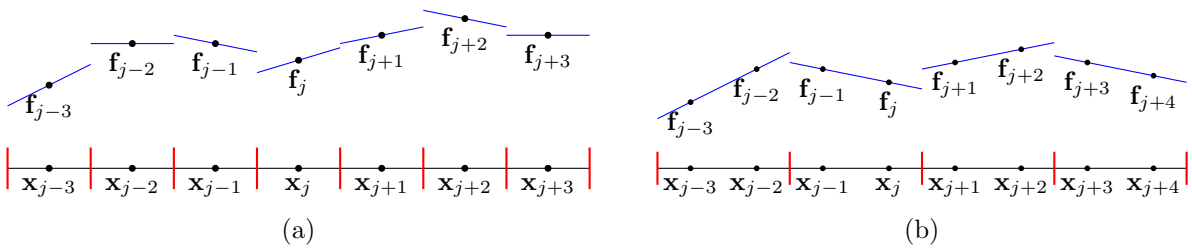


Figure 3.1 Example of partitioning pointwise PIC data into elements, and constructing an interpolant of the pointwise data ($f_j = f(x_j)$) on each element. The red partitions represent the superimposed mesh to the pointwise grid. The blue lines represent the piecewise linear interpolant reconstructed from a stencil containing two grid values. In (a) the elements $\tau_j = I_j$ consist of a single cell, and the stencil uses information from the right adjacent cell ($\ell = 0$ and $r = 1$). In (b) the elements $\tau_j = I_{2j-1} \cup I_{2j}$ are formed of a union of two neighboring cells and the stencil for each element only consists of those cells inside that element.

## 3.2 Data Initialization

Unlike traditional FEM data which is defined over the entirety of a given mesh, the data considered here arises from moments of PIC simulations and is discrete in nature. SIAC filtering makes use of a continuous convolution, therefore to apply SIAC filtering a global representation must be constructed. The methodology presented here is to superimpose a mesh over the pointwise grid where the data is defined. For simplicity, within each element of this mesh, we construct a Lagrange interpolant from the pointwise data using a stencil of a chosen size. For ease of illustration we consider the one-dimensional case.

### 3.2.1 Constructing the mesh

Consider a domain $\Omega = [a, b]$, and assume that the pointwise data $\{f_j\}_{j=1}^{N}$ is given on the cell centers of a grid

$$a < x_1 < x_2 < \ldots < x_{N-1} < x_N < b.$$

Thinking about the data from a finite volume perspective, introduce cells $I_j = [x_{j-1/2}, x_{j+1/2}]$, $j = 1, \ldots, N$, where $x_{j-1/2} = \frac{1}{2}(x_j - x_{j-1})$, with $x_{1/2} = a$ and $x_{N+1/2} = b$. Denote the cell widths by $\Delta x_j = x_{j+1/2} - x_{j-1/2}$. It is possible to choose the elements of the superimposed mesh, $\mathcal{T} = \{\tau_k\}_{k=1}^{N_\tau}$, to just be these cells, in which case the mesh will be similar to that depicted in Figure 3.1(a). Alternatively, the elements can be selected to be a union of adjacent cells, an example of which is given in Figure 3.1(b). This choice will cause no difficulty in what follows, it only alters the number of polynomial interpolants needed in the construction, $N_\tau$, and the minimum stencil size of these interpolants.

### 3.2.2 Constructing a piecewise-polynomial interpolant

To convert the discrete data into a continuous approximation on a given element $\tau_k$, we choose nonnegative integers $\ell$ and $r$ and introduce the stencil about element $I_j$ :

$$S(j) = \{I_{j-\ell}, \ldots, I_j, \ldots, I_{j+r}\}.$$

Assume here that the center cell of the stencil, $I_j$, satisfies $I_j \subset \tau_k$. The pointwise data contained within the cells of this stencil will be used to construct the interpolant over $\tau_k$. Note that the lack of assumptions about boundary conditions necessitates that the stencil is reduced in size near domain boundaries in order to not exceed the boundaries of our domain. Setting $p = \ell + r$, the Lagrange basis specific to the element is $\{L_q^j(x)\}_{q=0}^p$, with

$$L_q^j(x) = \prod_{\substack{n=0 \\ n \neq q}}^{p} \frac{x - x_{n+j-\ell}}{x_{q+j-\ell} - x_{n+j-\ell}}.$$

The element-specific interpolant is then given by

$$u_h(x)\Big|_{\tau_j} = \sum_{q=0}^{p} f_{q+j-\ell} L_q^j(x).$$

Iterating through the elements, and restricting the stencils where appropriate, a piecewise polynomial representation of the pointwise data over the whole domain is obtained. In the work to follow we find there is no significant difference in using larger stencils for the initialization procedure. To that end, here a finite volume perspective is used and the discrete data is treated as cell-center values on a finite volume style mesh. This corresponds to a $p = 0$ piecewise-constant initialization.

## 3.3 SIAC Filters: Kernel Construction and Boundary Treatments

This section describes a convolution denoising procedure and the construction of the SIAC kernel used in the convolution. For ease of illustration we again consider the one-dimensional case.

The filtered data, $u_h^\star$, is obtained via convolution of the interpolant, $u_h$, with a compactly supported scaled kernel function $K_H$:

$$u_h^\star(x) = (K_H \star u_h)(x) \equiv \int_{\mathbb{R}} K_H(x-y)\, u_h(y)\, dy. \tag{3.5}$$

Here, a family of kernel functions known as SIAC kernels (see Section 2.1) are considered. These SIAC kernels are piecewise polynomial functions of compact support that satisfy consistency and moment conditions, which will be discussed in Section 3.3.4. It is important to note that the compact support of the kernel function reduces the integral in (3.5) from the entire range of $\mathbb{R}$ to a shifted support of the kernel. In applying the filter to non-periodic data, it is exactly this restriction of the kernel's support that enables shifted or position-dependent kernels to constrict the convolution to within the domain of the data. In the following, a description of the knot matrix formulation of SIAC kernels for applications to both periodic and non-periodic data is given. In the latter case, we describe the position-dependent kernels developed in [10] and the generalized spline boundary kernels introduced in [1]. We also introduce a novel adaptive kernel scaling.

### 3.3.1 Knot Matrix Kernel formulation

We use in this chapter the knot matrix formulation of the SIAC kernel introduced in [1]. This description encompasses both the symmetric formulation for periodic data previously given in Section 2.1 and the position-dependent formulation for non-periodic data. The SIAC kernel composed of $(r+1)$ B-splines of order $\ell$ is defined via an $(r+1)\times(\ell)$ knot matrix $\mathbf{T}$ which will be given in Section 3.3.3. The composite kernel is given by

$$K_{\mathbf{T}}(x) = \sum_{\gamma=0}^{r} c_\gamma B^{(\ell)}_{\mathbf{T}_\gamma}(x),$$

where $\mathbf{T}_\gamma = \mathbf{T}(\gamma,:)$ denotes the $\gamma$-th row of $\mathbf{T}$ for $\gamma = 0, 1, \ldots, r$. These rows contain the knot sequences for each B-spline composing the kernel. As detailed in [68] the $j$-th B-spline of order $\ell$ is defined via its knot sequence $\mathbf{t}$ by the recursion relation:

$$B_j^{(\ell)}(x) = w_{j,\ell} B_j^{(\ell-1)}(x) + (1 - w_{j+1,\ell}) B_{j+1}^{(\ell-1)}(x),$$

where

$$w_{j,\ell}(x) = \frac{x - t_j}{t_{j+\ell-1} - t_j}.$$

This recursion ends with a simple characteristic function,

$$B_j^{(1)}(x) = \begin{cases} 1, & t_j \le x < t_{j+1} \\ 0, & \text{Else} \end{cases}.$$

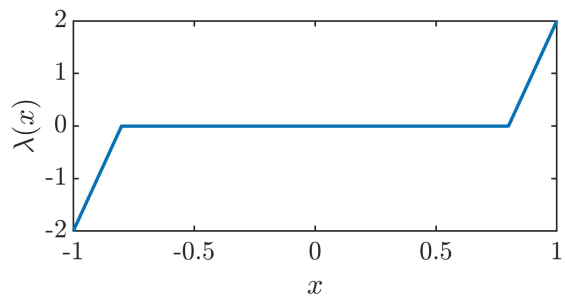


Figure 3.2 Demonstration of a shift function $\lambda(x^\star)$ for $K_H^{(3,2)}$ with $H(x^\star) = 0.1$ on the domain $\Omega = [-1, 1]$.

Note the alternative definition of B-splines given here versus in (2.6) as for position-dependent filters, nonuniform knot sequences may result.

For the purposes of the B-splines in the kernel, define $B_{\mathbf{T}_\gamma}^{(\ell)}$ to be the first B-spline of order $\ell$ corresponding to knot sequence $\mathbf{T}_\gamma$. The typical knot matrices considered here, with the exception of generalized spline knot matrices [1], are of the form

$$\mathbf{T}(i, j; x^\star) = -\frac{1}{2}(r + \ell - 1) + i + j + \lambda(x^\star), \quad i = 0, \ldots, r, \quad j = 0, \ldots, \ell. \tag{3.6}$$

where $\lambda(x^\star)$ is a mesh dependent shifting function that restricts the kernel support to the computational domain. Note that here and in what follows, the dependence of $\mathbf{T}_\gamma$ on $x^\star$ is suppressed in order to ease presentation.

When employing a scaled kernel, a multiplication of the knot matrix by a scaling function $H$ is done. $H$ may depend on the location of the filtering point $x^\star$. Note that, occasionally, use of the notation $K_H^{(r+1,\ell)}$ is made which refers to the kernel function composed of $(r + 1)$ B-splines of order $\ell$ with scaling $H$.

### 3.3.2 Shifting function

Given $\Omega = [a, b]$, consider the shifting function originally proposed in [8]:

$$\lambda(x^\star) = \begin{cases} \min\left\{0, -(r+\ell)/2 + (x^\star - a)/H(x^\star)\right\}, & x^\star \in [a, (a+b)/2), \\ \max\{0, (r+\ell)/2 + (x^\star - b)/H(x^\star)\}, & x^\star \in [(a+b)/2), b], \end{cases} \tag{3.7}$$

where $H$ is the kernel scaling function. If $\lambda(x^\star)$ is plotted over the domain $[-1, 1]$, as depicted in Figure 3.2, it is observed that the support of the kernel is being shifted away from the boundaries when getting close to both boundaries, while in the interior the kernel support remains symmetric. To understand the effect of the shift function on the kernel support, a plot of the fully shifted left ($\lambda = -2$), right ($\lambda = 2$), and symmetric kernels are given in Figure 3.3. The shifting function provides a continuous transition between the shifted and symmetric kernels, though the limited smoothness of the shifting function restricts the smoothness of the filtered approximation at the interface of shifted and unshifted kernels.

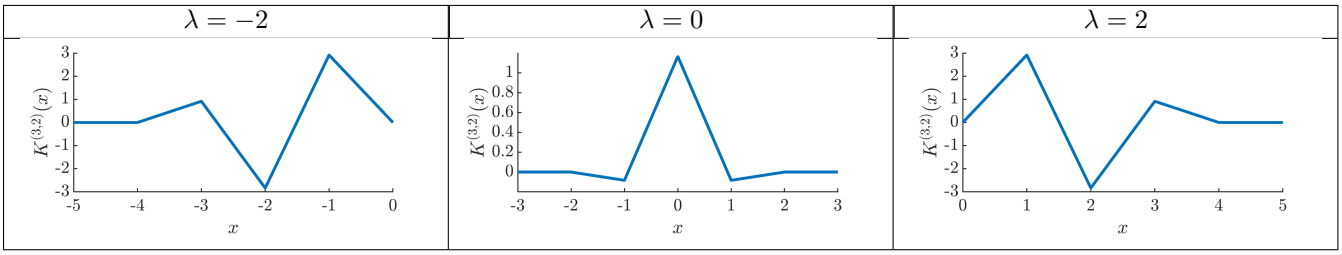


Figure 3.3 Shifted Kernels for $K^{(3,2)}(x)$. For $\lambda = \pm 2$, $x = 0$ corresponds to the boundary. Note that owing to the reflection of $y$ in the kernel argument during the convolution, $\lambda = -2$ ($\lambda = 2$) corresponds to the left-sided (right-sided) kernel in the physical space.

### 3.3.3 Knot Matrix Examples

Typical examples of knot matrices corresponding to the case of three B-splines of order two are given by

$$\mathbf{T}_{\text{Left}} = \begin{bmatrix} -4 & -3 & -2 \\ -3 & -2 & -1 \\ -2 & -1 & 0 \end{bmatrix}, \quad \mathbf{T}_{\text{Sym}} = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}, \quad \mathbf{T}_{\text{Right}} = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 2 & 3 \\ 2 & 3 & 4 \end{bmatrix},$$

where $\mathbf{T}_{\text{Left}}$ and $\mathbf{T}_{\text{Right}}$ denote the biased filters used for filtering at the left and right boundaries of the domain, respectively. The symmetric kernel $\mathbf{T}_{\text{Sym}}$ is used in the interior of the domain.

### 3.3.4 Kernel coefficients

To determine the coefficients, $c_\gamma$, we proceed the same as in Section 2.1.1. The only difference is that for non-symmetric kernels the knot matrices can change with each filtering point $x^\star$, and so the kernel coefficients need to be recomputed with every convolution evaluation, which is not the case in the interior. With the knot matrix notation, the kernel coefficients are obtain by solving

$$\begin{bmatrix} b_{0,0} & \cdots & b_{0,r} \\ \cdots & & \cdots \\ b_{r,0} & \vdots & b_{r,r} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_r \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3.8}$$

where

$$b_{k,\gamma} = \int_{\mathbb{R}} B^{(\ell)}_{\mathbf{T}_\gamma}(y)(-y)^k \, dy \;\text{ for }\; k, \gamma = 0, \ldots, r.$$

More explicit formulas for the coefficient system (3.8) in the uniform symmetric knot matrix case are provided in [17], while the general knot matrix case is considered in [89, 90].

### 3.3.5 Generalized Spline Kernels

An issue arising from position-dependent filters is that errors increase in boundary regions for FEM data and in the case of PIC denoising, the filter fails to preserve boundary layers. In the past for the FEM

case, these issues have been addressed by using larger numbers of B-splines or convex combinations of SIAC kernels [8, 11], but as detailed in [1], adding a single generalized spline can decrease errors at the boundaries without increasing the kernel support. For example, near the left boundary, write

$$K_{H\,\mathbf{T}}(x) = \sum_{\gamma=0}^{r} c_\gamma(x) B^{(\ell)}_{H\,\mathbf{T}_\gamma}(x) + c_{r+1} B^{(\ell)}_{H\,\mathbf{T}_{r+1}}(x).$$

Here the coefficients are still determined by requiring polynomial reproduction. The new knot matrices are defined based off the sign of $\lambda$ by

$$\lambda < 0 \to \begin{bmatrix} \mathbf{T} \\ \mathbf{s}_L \end{bmatrix}, \quad \lambda = 0 \to \mathbf{T}, \quad \lambda > 0 \to \begin{bmatrix} \mathbf{s}_R \\ \mathbf{T} \end{bmatrix}.$$

Here the $1 \times (\ell + 1)$ knot sequences for the left and right generalized splines are given by

$$\mathbf{s}_L = \left[\lambda + \frac{1}{2}(r+\ell) - 1, \lambda + \frac{1}{2}(r+\ell), \ldots, \lambda + \frac{1}{2}(r+\ell)\right],$$

and

$$\mathbf{s}_R = \left[\lambda - \frac{1}{2}(r+\ell), \ldots, \lambda - \frac{1}{2}(r+\ell), \lambda - \frac{1}{2}(r+\ell) + 1\right].$$

For instance, consider the case of $r = 2$ and $l = 2$. The new left and right knot matrices for $K_H^{(3,2)}$ are given by

$$\mathbf{T}_{Left} = \begin{bmatrix} -4 & -3 & -2 \\ -3 & -2 & -1 \\ -2 & -1 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \quad \mathbf{T}_{Right} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 2 \\ 1 & 2 & 3 \\ 2 & 3 & 4 \end{bmatrix}.$$

Plots of the these boundary kernels are given in Figure 3.4.

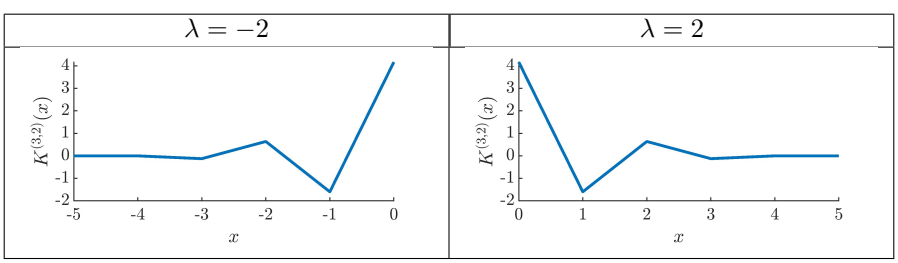


Figure 3.4 Shifted Kernels for $K^{(3,2)}(x)$ including a generalized spline. Here $x = 0$ corresponds to a boundary.

### 3.3.6 Adaptive kernel scaling

Although the addition of generalized splines improves the performance of SIAC denoising for moderate kernel scalings, as will be observed in the numerical results, some care is needed to preserve short-duration

sharp gradients near boundaries when the kernel scaling becomes large. To that end, we develop a remedy via an adaptive kernel scaling given by

$$H_{\text{adaptive}}(x) = \begin{cases} H_{\text{int}}, & x - H_{\text{int}}\frac{r+\ell}{2} \geq a \;\text{ and }\; x + H_{\text{int}}\frac{r+\ell}{2} \leq b, \\ \frac{h_{\text{grid}}}{r+\ell} + 2(x-a)\frac{H_{\text{int}} - h_{\text{grid}}/(r+\ell)}{H_{\text{int}}(r+\ell)}, & x - H_{\text{int}}\frac{r+\ell}{2} \leq a, \\ \frac{h_{\text{grid}}}{r+\ell} + 2(x-b)\frac{-H_{\text{int}} + h_{\text{grid}}/(r+\ell)}{H_{\text{int}}(r+\ell)}, & x + H_{\text{int}}\frac{r+\ell}{2} \geq b, \end{cases} \tag{3.9}$$

where $H_{\text{int}}$ is a large scale desired for the interior, and $h_{\text{grid}}$ is the size of the mesh elements. Typically $H_{\text{int}}$ is related to a physical length scale in the problem. This choice of adaptive scaling provides the advantage of preserving the boundary values of the underlying initialization by reducing the kernel support to a single element at the boundary locations, enabled by the polynomial-reproduction property of the kernel. Figure 3.5 shows a depiction of this adaptive scaling and its smoothing effect on the transition between the symmetric kernel and the position-dependent kernel with a shifting parameter. This adaptive scaling has the additional effect of creating a smoother shifting function $\lambda$ and thereby resulting in a smoother transition between different knot matrices.

### 3.3.7 Convolution evaluation

The consistent evaluation of convolution in 1D requires keeping track of all the breaks in continuity of the approximation and breaks in differentiability of the kernel within the scaled kernel support. By only integrating between these breaks in continuity of the data or the kernel, the convolution can be performed exactly using Gaussian quadrature. A detailed discussion of the implementation can be found in [91].

### 3.3.8 Fourier representation

As our primary focus is to denoise PIC data, it is critical to examine the Fourier transform of the convolution kernel to quantify the damping effect applied to each frequency. This examination will guide the choice of the kernels when denoising moments. Unfortunately, analytical computation of the kernel's Fourier response is only possible in the periodic case for the symmetric kernel with a constant kernel scaling. In such a case, the Fourier representation of the SIAC kernel composed of $(r+1)$ B-splines of order $\ell$ and constant scaling $H$ is given by

$$\hat{K}_H^{(r+1,\ell)}(k) = \left(\frac{\sin(\frac{kH}{2})}{\frac{kH}{2}}\right)^{\ell} \left(c_{\lceil \frac{r}{2} \rceil + 1} + 2\sum_{\gamma=1}^{\lceil \frac{r}{2} \rceil} c_\gamma \cos(\gamma k H)\right).$$

The first term in the product is controlled by the spline order, while the second is controlled by the number of B-splines, with the value of the coefficients, $c_\gamma$, being controlled by both the number of B-splines and the B-spline order.

In Figure 3.6 the Fourier representation is displayed for varying kernel parameters. We observe that increasing the spline order, which corresponds to an increase in smoothness, causes the filter to damp more heavily and reduce oscillations. This makes sense as the Fourier transform of smoother functions decay faster and increasing the spline order results in a smoother kernel. Increasing the number of splines, which corresponds to increasing the number of moments, does not change the shape drastically, but slightly alters how the kernel samples different modes. Specifically, it flattens the Fourier response at $k = 0$ by causing $\frac{d^n}{dk^n}\hat{K}^{(r+1,\ell)}(0) = 0$ for $n = 1, \ldots, r$. This follows from the polynomial reproduction condition on the kernel. Lastly, changing the scaling factor only causes a dilation in $k$ of kernel response and utilizes more (or fewer) elements in the filtering process. An understanding of the frequency effects of position-dependent and adaptively scaled filters remains an important area of further study. In particular, the standard methodology of computing the Fourier transform of the SIAC kernel is inappropriate owing to the spatial variability of the kernel function employed over the computational domain. One possible approach is the consideration of the spectrum of the discrete filtering operator, but that is beyond the scope of this work.

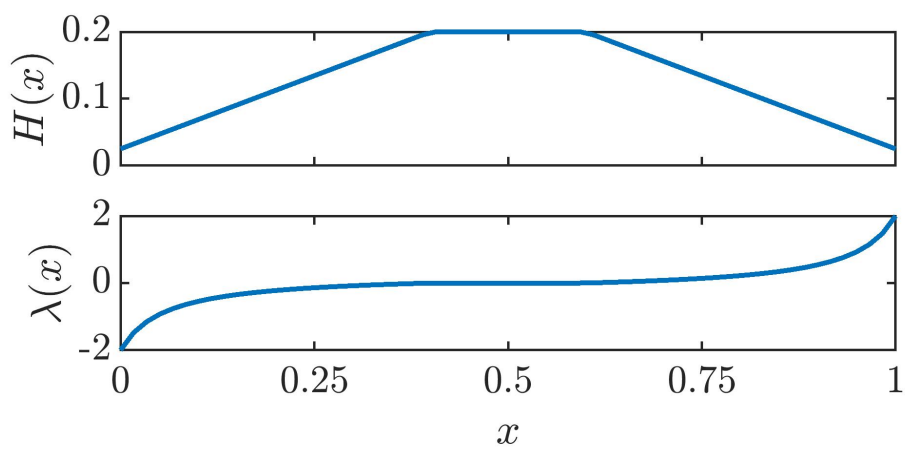


Figure 3.5 An adaptive kernel scaling example for $K_H^{(3,2)}(x)$ over the domain $\Omega = [0, 1]$ with $H_{\text{int}} = 0.2$, $h_{\text{grid}} = 0.1$.

### 3.4 Numerical Results

In this section we consider the application of SIAC filters to denoise PIC data. Here, both a periodic boundary condition and a boundary condition that introduces sharp gradients will be considered. For the former, an investigation of the damping of the noise using the uniformly scaled symmetric filter is performed, where it is expected that the filter should preserve the physically-relevant oscillations (see Section 3.4.2). For the latter case, several aspects are studied, including comparisons of the abilities of position-dependent filters with or without generalized splines as well as with or without adaptive kernel

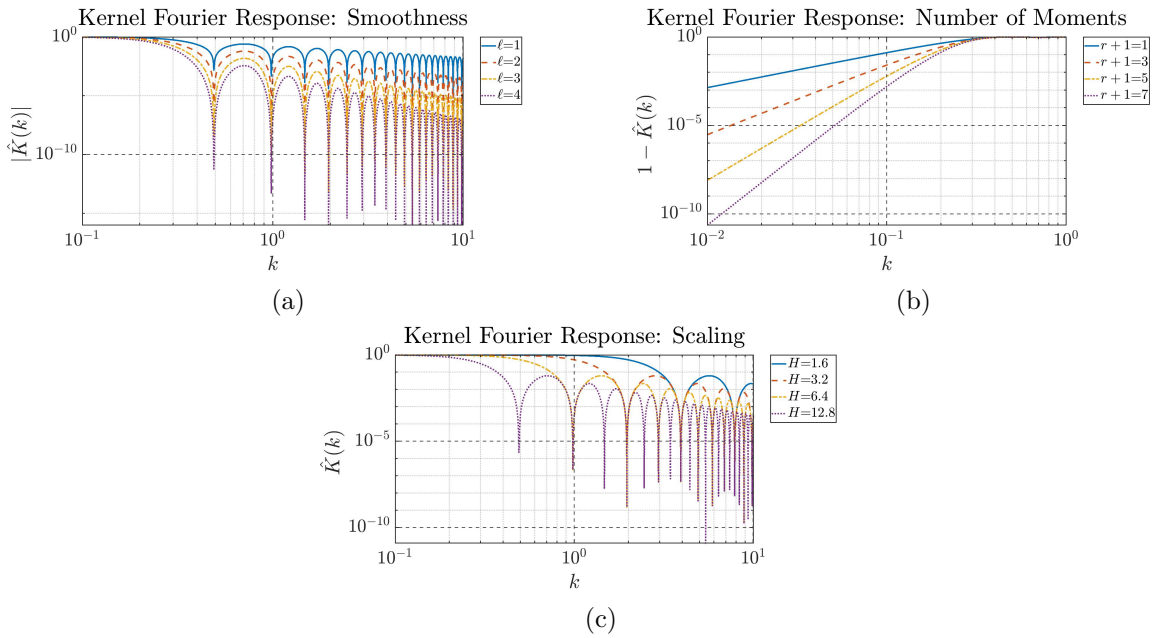


Figure 3.6 Variation of the magnitude of the Fourier representation of the $K_H^{(r+1,\ell)}$ filter with respect to (a) spline order, with higher orders corresponding to smoother and more heavily damped filtered data, (b) moment conditions, with more moment conditions allowing the capturing of more information about a point, and (c) kernel scaling, with a larger scaling resulting in more information being used in the filtering process. Note that the kernel is symmetric with respect to $k$ and that unless indicated otherwise in the legends, $H = 12.8$, $r + 1 = 3$, and $\ell = 2$.

scalings (see Section 3.4.3). The focus is on how boundary layers resolution in physical space affects the filtered data in Fourier space. It must be highlighted again that maintaining the boundary behavior of the data can be critical to obtaining accurate physics, e.g., for the Bohm criterion.

Given PIC moment data, there are three steps in the denoising procedure:

1. Construct a nodal piecewise interpolant as described in Sec. 3.2 for the given pointwise discrete data that is generated from a PIC simulation.

2. Select the kernel parameters ($r$, $\ell$, and $H$), and filter the initialization obtained in Step 1 with the corresponding SIAC kernel. The choice of parameters depends on the type of data being filtered (characteristic length scales, desired damping effects, etc.).

3. Assess the filtered data from the point of view of physics

It was determined that there is an insensitivity in the initialization step to low polynomial degrees and therefore only results for a piecewise constant initialization are included in what follows.

### 3.4.1 Assessment of Fourier Effects

For the purposes of displaying the Fourier effects of the filtering procedure on the underlying data, plots of the single-sided amplitude spectrum are given. In describing methodology for computing the spectrum, MATLAB's vector notation is used: Given a real-valued signal $\mathbf{x}$ of even length $N$

1. Compute the Discrete Fourier Transform (DFT) of the data vector: $\hat{\mathbf{x}} = \text{fft}(\mathbf{x})$.

2. Owing to the symmetry of the magnitudes in $\hat{\mathbf{x}}(n)$ only select the first $N/2$ entries: $\hat{\mathbf{x}} = \hat{\mathbf{x}}(1{:}N/2)$.

3. Double the magnitude of non-DC values: $\hat{\mathbf{x}}(2{:}\text{end}) = 2\hat{\mathbf{x}}(2{:}\text{end})$.

4. Compute the modulus of each value, normalized by the signal length: $\hat{\mathbf{x}} = \text{abs}(\hat{\mathbf{x}}/N)$

If the data is non-periodic, multiplication of the data vector by a Hanning window function is performed in the first step to minimize any spectral leakage, i.e. $\mathbf{x}(n) \to \mathbf{x}(n)\mathbf{w}(n)$, where $\mathbf{w}(n) = 1/2 - 1/2\cos(2\pi n/N)$, $n = 0, \ldots, N-1$. Presuming the data is given at a uniform spacing $\Delta x$, $\hat{\mathbf{x}}$ is defined as the vector of frequencies $\mathbf{f} = \frac{2\pi}{N\Delta x}(0{:}N/2-1)$.

### 3.4.2 Periodic boundary conditions

The first test is to measure the possible effectiveness of the SIAC filtering technique for denoising PIC data. This requires the removal of high frequency/wavenumber (with wavelengths at cell size) noises and the preservation of physical oscillations. For this purpose, PIC simulations with periodic boundary

conditions are considered. Such boundary conditions can be widely found in PIC simulations [85, 86] such as in the study of plasma instabilities [92, 93]. Here the data from the VPIC code [94] that simulates the whistler instability in magnetized plasma induced by the trapped electrons is used [95, 96]. For such periodic data, a symmetric SIAC filter with a constant kernel scaling over the whole domain can be used. Figure 3.7 shows the results of applying the filtering procedure on periodic magnetic field data where the kernel is composed of 3 B-splines of order 2 ($r+1=3$, $\ell=2$). Here the domain is $\Omega=[0,1400]$ with a uniform grid spacing of $h_{grid}=\Delta x=0.1$, in the unit of Debye length $\lambda_D$. A large kernel scaling $H=64h_{grid}$ is chosen so that the truncated wavenumber is $k_c \tilde{<} \lambda_D^{-1}$, as shown in the Fourier space plot.

We first note that the Fourier modes match very well in the low frequency domain, resolving all the physical structures we are interested in. It is even more remarkable that the theoretical model of the kernel response matches very well with the observed damped frequencies in the Fourier space (the lower-left plot). As the analysis in Section 3.3.8 suggested, the damped frequencies are primarily controlled by the kernel scaling $H$ which dilates the filter response. Therefore, by choosing the kernel scaling appropriately, it is possible to target and damp noises with high wavenumbers, while maintaining good preservation of the real whistler waves peak at $k \sim 0.1\lambda_D^{-1}$ as seen from the Fourier spectrum.

This result suggests that for periodic data with limited sharp features SIAC filtering can be applied in an informed manner. The targeting physical frequency such as the whistler wave case is typically known *a prior* and can be readily used to guide the choice of $H$ in practice.

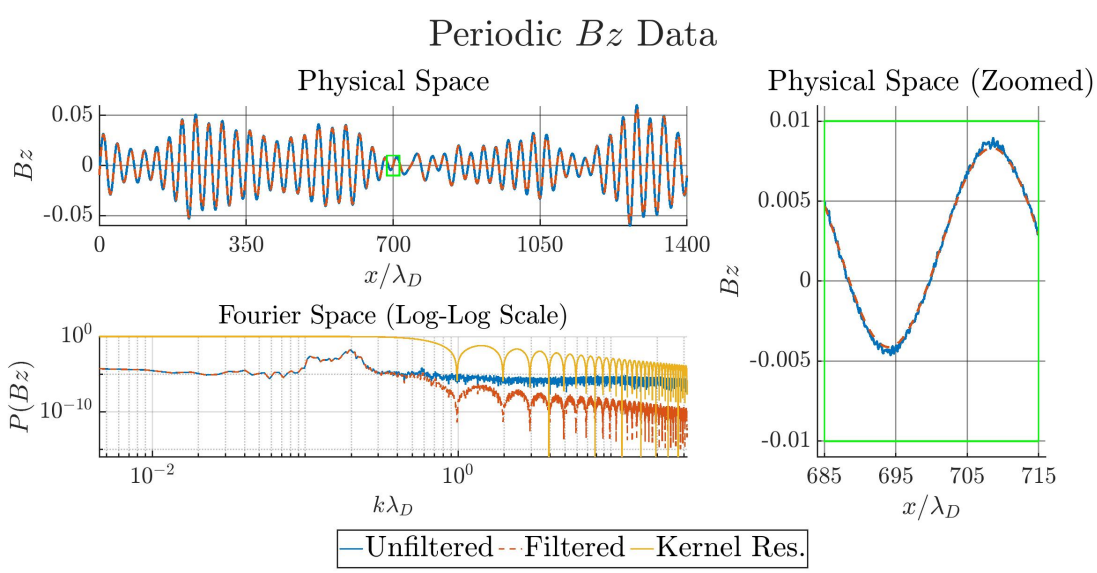


Figure 3.7 Application of SIAC filtering to periodic magnetic field for the whistler instability driven by the trapped electrons: (upper-left) filtered vs. unfiltered data over the whole domain. (right) zoomed-in physical space plot displaying spatial noise damping. (lower-left) single-sided amplitude spectrum of filtered and unfiltered data with analytic SIAC kernel response superimposed.

### 3.4.3 Data with sharp gradients near the boundary

The second test considers the capability of SIAC filtering to handle large gradients near the boundary. This can be a critical characteristic in some cases. Here the considered case is that of the non-neutral sheath (which forms near the region where a plasma intercepts a solid surface) that introduces sharp gradients to the plasma profiles; for instance see Figure 3.8 and Figure 3.10.

It is worth noting that the electron conduction heat flux, $q_n^e$, in the form of its spatial gradient as shown in (3.4), plays an important role in setting the Bohm speed [87, 97], denoising of which is extremely demanding since itself suffers higher noise pollution than the plasma density and flow.

Here, position-dependent SIAC filters that include generalized splines and adaptive kernel scalings are used to denoise the data so that the Bohm criterion can be calculated. This avoids a long-time average as proposed in [87], significantly reducing the computational cost.

The boundary behavior of the filtered data for varying kernel treatments is first examined in the physical space, shown in Figure 3.8. It can be seen that the typical position-dependent filter composed of 3 B-splines of order 2 with a scaling of $H = 32h_{grid}$ (upper-left) cannot preserve the boundary behavior of electron density. However, if a generalized spline is added to the position-dependent kernel to lend more weight to the filtering point in the weighted-average, it is possible to better maintain the boundary behavior (lower-left). However, if we keep increasing the kernel support to $H = 128h_{grid}$ while keeping all the other parameters the same, it can be observed that the generalized spline alone is not enough to maintain the boundary behavior (upper-right). This is resolved by using the proposed adaptive kernel scaling (3.9) (lower-right).

We then compare the single-sided amplitude spectrums of the filtered and unfiltered electron density data, results of which are shown in Figure 3.9. While larger scalings are shown to have greater damping capabilities, there is not much of a disparity between the different boundary treatments using the same scaling. This makes sense as, while the differences in tracking the boundary behavior is apparent in physical space, only the data in a small portion of the domain near the boundary varies between these treatments. In aggregate, neither adaptive scalings nor generalized splines seem to matter as much as the choice of interior kernel scaling, when the spectrum of the filtered data is considered. Thus, the choice of boundary treatment may not be apparent in spectral analysis. The results in the Fourier space are largely comparable to the periodic case.

Next an investigation of the same choice of kernel parameters on the electron heat flux, $q_e^n$, and its spatial gradient, $dq_e^n/dx$, is performed. Note that here in the unfiltered case $dq_e^n/dx$ is computed by a central difference stencil of $q_e^n$ in the interior of the domain, and a 1st order biased stencil at the domain

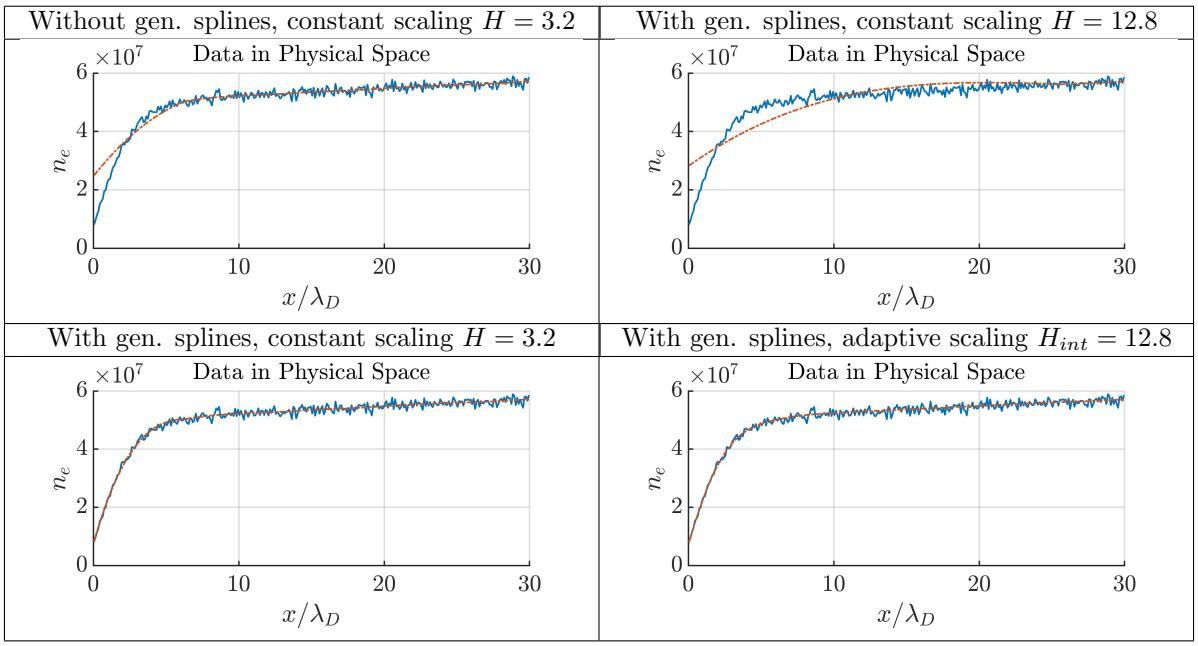


Figure 3.8 Effect of generalized splines and adaptive kernel scaling on boundary behavior preservation. The underlying data is electron density data $n_e$ with $\Omega = [0, 800]$ and $h_{grid} = \Delta x = 0.1$. Here the solid blue lines denote the unfiltered data and the dashed red lines the filtered data.

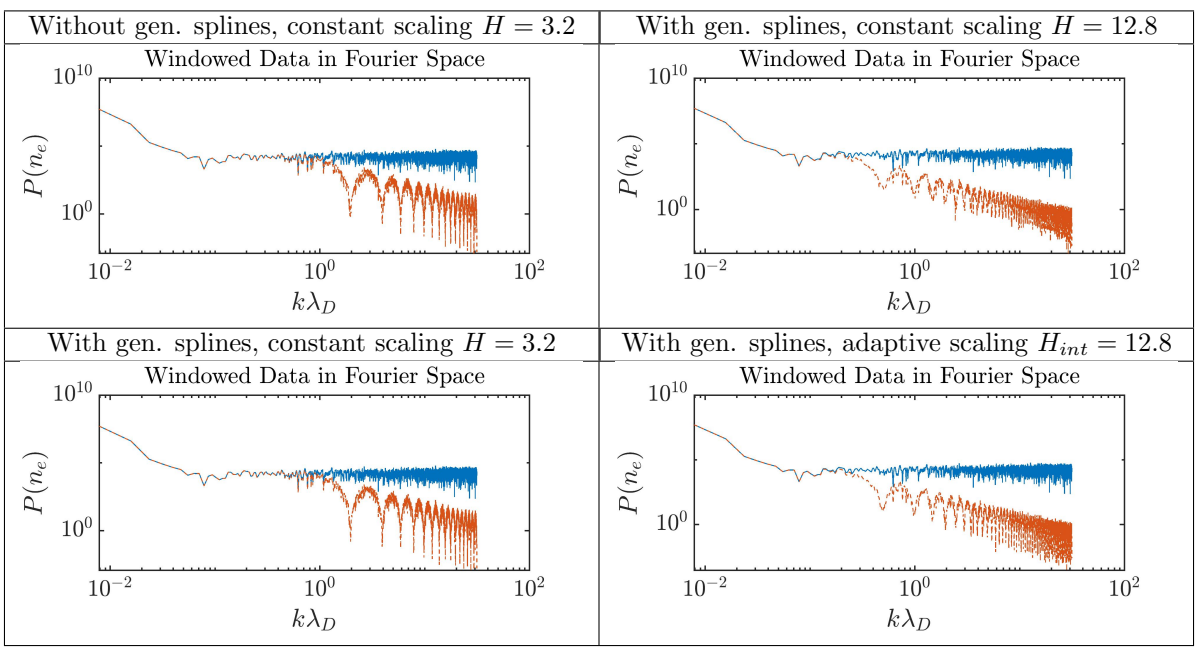


Figure 3.9 Effect of generalized splines and adaptive kernel scaling on the single-sided amplitude spectrum of electron density data $n_e$ with $\Omega = [0, 800]$ and $h_{grid} = \Delta x = 0.1$. Here the solid blue lines denote the unfiltered data and the dashed red lines the filtered data. Note that the underlying data was non-periodic so a Hanning window function was applied.

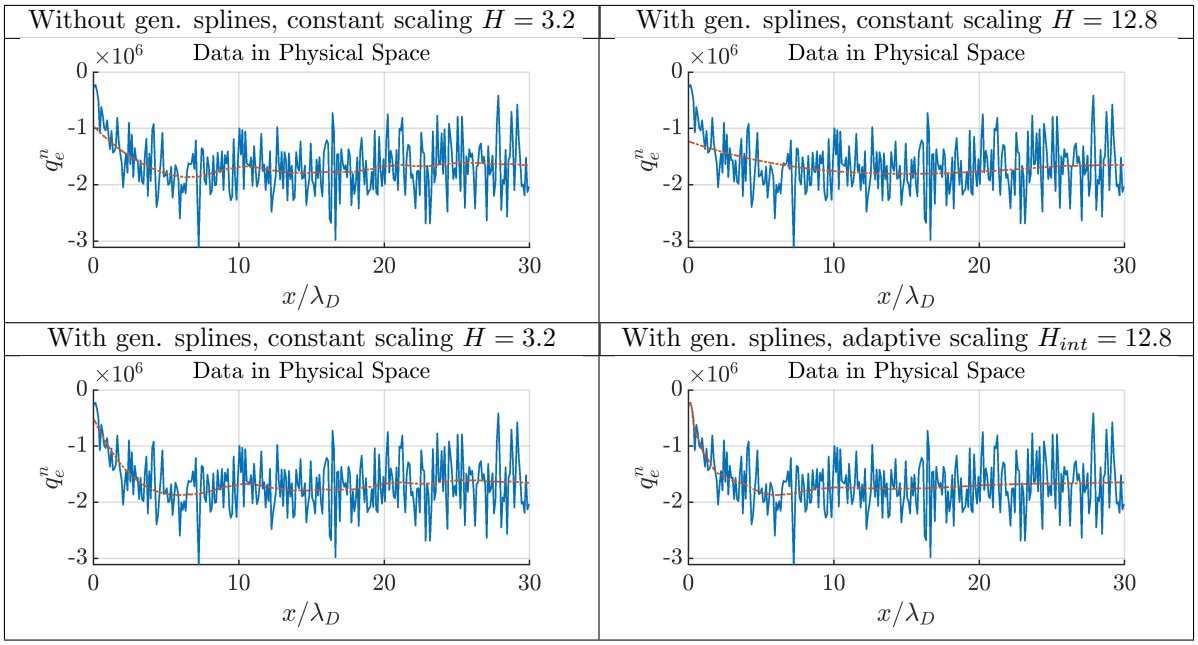


Figure 3.10 Effect of generalized splines and adaptive kernel scaling on boundary behavior preservation. The underlying data is electron heat flux data $q_e^n$ with $\Omega = [0, 800]$ and $h_{grid} = \Delta x = 0.1$. Here the solid blue lines denote the unfiltered data and the dashed red lines the filtered data.

boundaries. The filtered derivative values are obtained by applying the same differencing procedure to the filtered $q_e^n$ values. Figure 3.10 and Figure 3.12 present the effect of the filtering scheme on $q_e^n$ and $dq_e^n/dx$. The data is found to be much more oscillatory than the electron density data, though the same behavior is observed when varying filter parameterizations for $q_e^n$. Specifically, adding a generalized spline helps maintain boundary behavior for smaller $H$, but an adaptive scaling becomes necessary for larger interior scalings. Similar behavior to the $n_e$ case is also observed in the single-sided amplitude spectra depicted in Figure 3.11. With respect to the gradient of the heat flux, the differing boundary treatments are much less apparent in physical space (see Figure 3.12). The presence of noise is particularly apparent in $dq_e^n/dx$ in that it oscillates rapidly about 0, and much of the energy in the single-sided amplitude spectrum is contained in this noise manifesting in higher frequencies as depicted in Figure 3.12. The application of the filtering procedure serves to damp these higher frequencies, with the choice of interior scaling being of primary importance in controlling the magnitude of damping.

The result presented above suggest that SIAC filters, when applied to non-periodic data in an informed manner, capture information well in both physical space (e.g., boundary layers) and Fourier space. Note that while it may appear that the choice of boundary treatment has little effect on $dq_e^n/dx$, in the next test, it is shown that minor perturbations of this variable can have a large impact on computed quantities such as the Bohm speed.

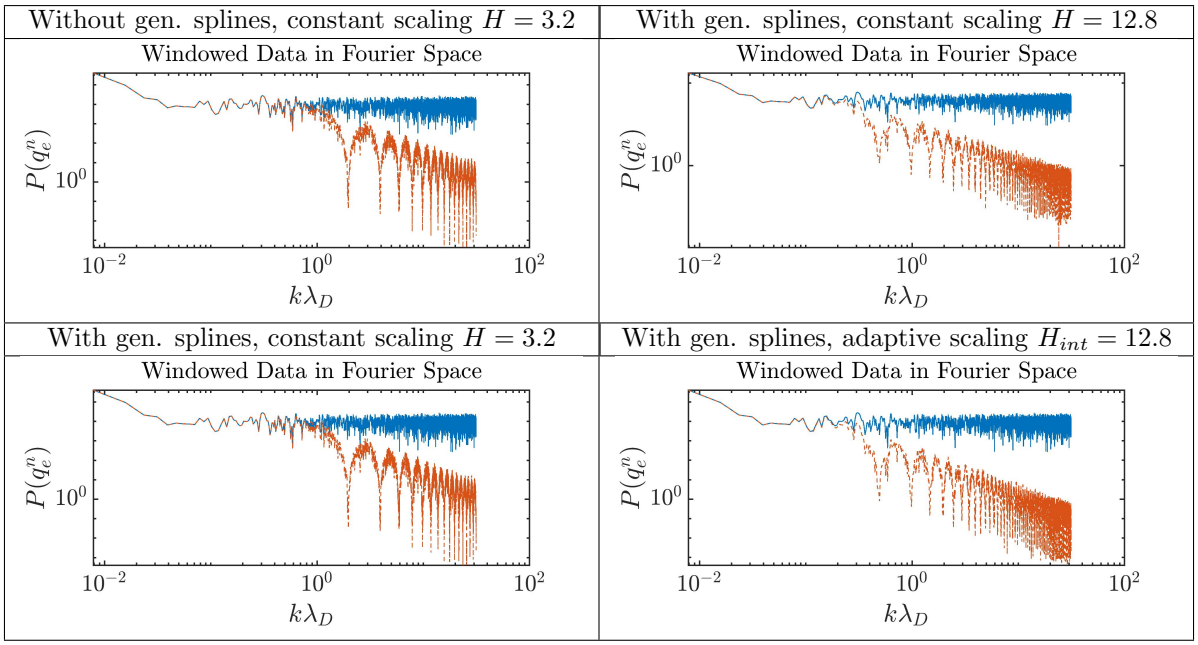


Figure 3.11 Effect of generalized splines and adaptive kernel scaling on the single-sided amplitude spectrum of electron heat flux data $q_e^n$ with $\Omega = [0, 800]$ and $h_{grid} = \Delta x = 0.1$. Here the solid blue lines denote the unfiltered data and the dashed red lines the filtered data. Note that the underlying data was non-periodic so a Hanning window function was applied.

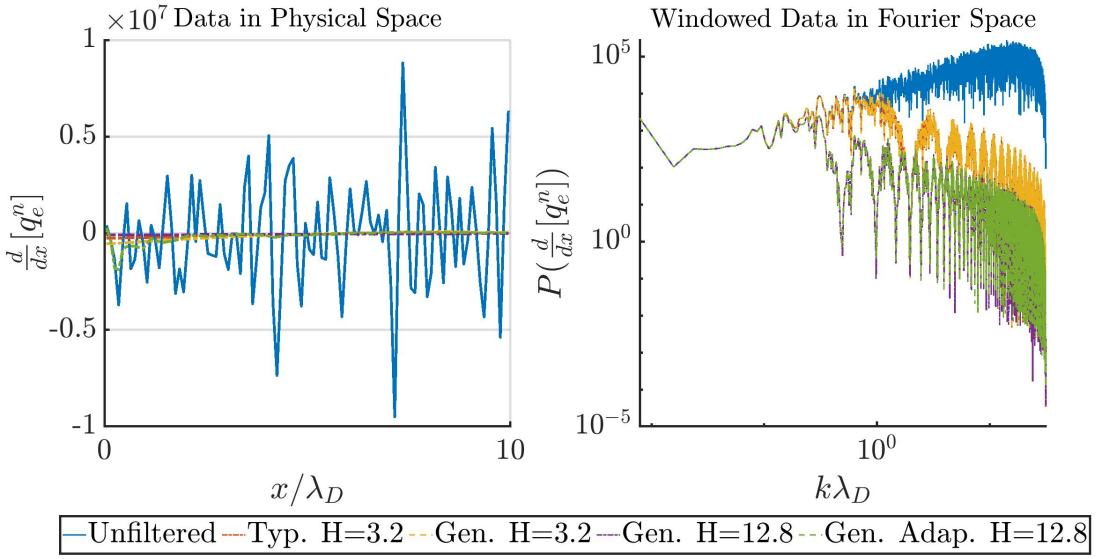


Figure 3.12 Effect of generalized splines and adaptive kernel scaling on electron heat flux gradient data $dq_e^n/dx$ in physical and frequency space. Here $\Omega = [0, 800]$, $h_{grid} = \Delta x = 0.1$, and the dashed red line (Typ.) denotes a filtering procedure with the typical kernel without generalized splines, while (Gen.) denotes a filtering with a kernel including a generalized spline. Finally, (Adap.) denotes that an adaptive kernel scaling is employed. In physical space, the effects of the differing boundary treatments are most apparent closest to the boundary, so the interior of domain is not shown. In Fourier space the interior scaling once again determines the magnitude of damping

.

### 3.4.4 Application to Bohm speed calculations

The numerical corroboration of the Bohm speed expression (3.3) is a very challenging task. For instance, the Bohm speed is very sensitive to small perturbations like PIC noise. When these perturbations in the data cause the denominator of (3.4) to flip signs or be near 0, the computed Bohm speed can be either complex or blow-up. The contribution of individual variables to such noise sensitivity of the Bohm speed are highly biased as seen from (3.4), where the electron heat flux gradient, contributes the most. Additionally, $q_n^e$ itself is very noisy due to it being a high moment calculated from the electron distribution. To mitigate such a challenge, [87] evolved to a steady state solution over an excessively long time and collected 200,000 snapshots at steady state. Their temporal averaging of this data eventually was used to corroborate the Bohm speed expression (3.3) successfully. Here we aim to show such an expensive brute-force strategy can be avoided by using a smaller number of snapshots with the proposed SIAC filters.

Table 3.1 Variable specific scalings for the 2,600 short-time averaged data set which were found to produce qualitative agreement with respect to the long-time averaged data.

| Variable | $q_n^e$ | $q_n^i$ | $Q_{ee}$ | $Q_{ei}$ | $Q_{ii}$ | $E$ | $R_T$ | $n_e$ | $n_i$ | $u_{ex}$ | $u_{ix}$ | $T_{ex}$ | $T_{ix}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $H$ | 8 | 4 | 16 | 16 | 16 | 2 | 32 | 6 | 6 | 8 | 8 | 16 | 4 |

First, it is necessary to systematically examine the filtering requirements and choose proper scalings for each variable. Our idea behind these individualized variable scalings is to filter the short-time averaged variables, obtaining filtered data which qualitatively resembles the long-time averaged variables. Procedurally, this takes the form of replacing a single long-time averaged variable in the Bohm speed computation with a noisier short-time averaged counterpart, and then varying the kernel scaling of that short-time averaged variable until good qualitative agreement is obtained with respect to the long-time averaged variable and the long-time averaged Bohm speed. The number of snapshots, 2,600, is the minimum number of time steps for which this can be achieved qualitatively for all variables. The chosen kernel scalings for each component is summarized in Table 3.1. We do note that some variables were less sensitive to filtering than others, allowing for some variability in the choice of scaling chosen. Choosing the scalings in this manner allows for the preservation of the qualitative behavior of the data and real-valued Bohm speed calculations.

Finally, when short-time averaged filtered data alone is used with the scalings chosen as indicated in Table 3.1, we obtain a desirable Bohm speed that tracks well with the long-time averaged calculations. This is shown in Figure 3.13, where a comparison of the computed Bohm speed from the filtered and unfiltered data with all variables originating from a time average over 2,600 snapshots is shown. The unfiltered long-time averaged data over 200,000 snapshots is also plotted as a baseline. The lower plot of

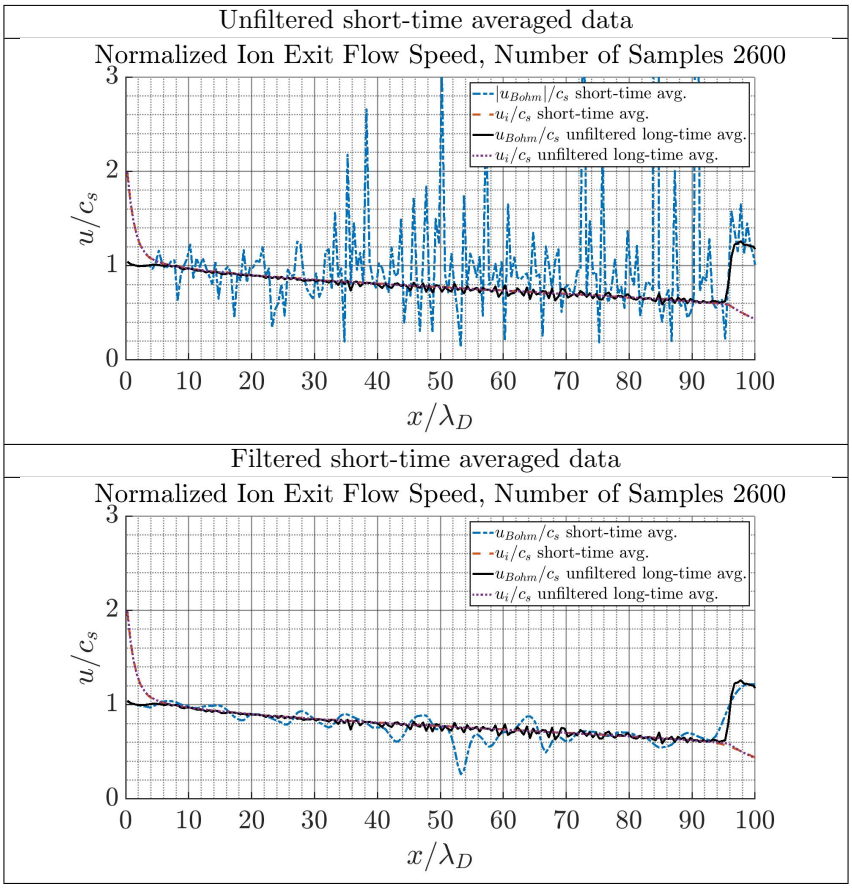


Figure 3.13 Comparison of Bohm speed and ion exit flow using filtered and unfiltered data time-averaged over 2,600 snapshots. The unfiltered long-time averaged data over 200,000 snapshots is also plotted. A $K_H^{(3,2)}$ kernel with scalings as indicated in Table 3.1 was used. Note that the modulus of $u_{Bohm}$ is plotted for the unfiltered short-time average data as the noise was sufficient to induce complex values.

Figure 3.13 shows that the filtered average of the normalized Bohm speed (blue dash line) matches well with the long-time averaged value (solid black line) from [87].

This result shows that the proposed SIAC filters are a promising data processing technique in the PIC setting, which can potentially achieve significant reductions in computational cost. For instance, the presented study of Bohm speed suggests that the previous expensive 1D3V PIC simulation in [87] can be performed in a much shorter time (1.3%) to corroborate (3.3)

### 3.5 Chapter Summary

In this chapter, Smoothness-Increasing Accuracy-Conserving filters have been extended to effectively denoise discrete PIC data while preserving underlying physics. Several aspects of the filters related to both the periodic boundary condition and a physical boundary condition that introduces the sharp gradient of plasma profiles have been investigated. Leveraging knowledge from SIAC applications in a FEM context,

the position-dependent filters have been repurposed and a new adaptive scaling methodology has been developed. The proposed filter is assessed using the challenging task of uncovering physical laws, specifically the Bohm speed formulation. The outcomes show that the computational cost in PIC can be substantially reduced by reducing the effective particle numbers or the time steps used for time-averaging. This underscores the substantial potential of the proposed filters as an effective tool for processing PIC data.

In the future, a more in-depth investigation of the sensitivity of the Bohm speed calculation to noisy variables will be investigated. Additional vital capabilities of the filter that needs to be developed are self-detection and resolution of interior structures possessing sharp gradients like the shocks. This of great importance for time-variant systems that do not attain steady states since in such cases the time-averaging approach does not apply.

### 3.6 Co-Author Contributions

Co-authors Qi Tang, Yanzeng Zhang, and Xian-Zhu Tang conceptualized the project idea, wrote and provided the physics interpretations, as well as performed editing and supervision for this project. Jennifer Ryan provided editing, supervision, and consultation for this project.

# CHAPTER 4

# (L)SIAC-MRA ON UNIFORM MESHES

This chapter contains content from the author's article [98] which is reused with publisher permissions given in Appendix D.

## 4.1 Chapter Introduction

In this chapter we introduce Line Smoothness-Increasing Accuracy-Conserving Multiresolution Analysis (LSIAC-MRA). This is a procedure for exploiting convolution kernel post-processors for obtaining error reduction under scale transition. This filtering-projection tool allows for the transition of data between different resolutions while simultaneously decreasing errors in the fine grid approximation. It implicitly allows for the computation of multiwavelet detail coefficients when translating coarse data onto finer meshes. These coefficients are not defined for the original coarse data, owing to a lack of resolution. We show how to analytically evaluate the resulting convolutions and express the filtered approximation in a new basis. This is done by combining the filtering procedure with projection operators that allow for computational implementation of this scale transition procedure. Further, this procedure can be applied to piecewise-polynomial data of any degree, including constant approximations to functions, to provide error reduction. We demonstrate the effectiveness of this technique in two and three dimensions.

The chapter outline is as follows, we begin by describing the original SIAC-MRA procedure developed by [23] for one-dimensional and tensor-product formulations. We then extend this approach to multidimensional data via LSIAC filters, and derive and implement discrete matrix operators for realizing the (L)SIAC-MRA procedure for uniform quadrilateral and hexahedral meshes. Lastly we investigate its error reduction capabilities on a suite of test problems.

## 4.2 Input Data Format

For ease of discussion, we will assume the data is given in a modal format as described in equation (2.25) and compute the the initialization via the $L^2$-projection procedure given in (2.33). As we are considering uniform quadrilateral and hexehedral meshes, we will also in this chapter and Appendices F and E, make the notational choices of $i, j, k = 1, \ldots, N$ representing element indices on a uniform mesh of $N^d$ elements. Furthermore, we will also use a separate index for each component of the modal basis function in the tensor-product basis, i.e. $\Phi_k(\mathbf{x}) = \phi_{kx}(x)\phi_{ky}(y)$. We now review the SIAC-MRA enhancement procedure in one dimension so as to better understand the two-dimensional derivations.

### 4.3 Review of One-dimensional SIAC-MRA

As developed in [23], SIAC-MRA is an enhancement procedure where coarse data is filtered and projected onto a finer mesh to construct more accurate fine mesh data. Here we review the procedure of [23] before generalizing it to multiple dimensions via LSIAC filters.

A crucial step valid for uniform meshes is that the one-dimensional convolution (2.1) can be analytically calculated and thereby allows for expressing of the filtered approximation in a new basis. As a result of the predictable ordering of mesh and kernel breaks, we have a piecewise definition for the filtered approximation itself:

$$u_h^\star(x) = \begin{cases} \sum_{m=0}^{p+1} b_m^j \chi_L^m(\zeta), & \zeta \in (-1,0) \\ \sum_{m=0}^{p+1} \tilde{b}_m^j \chi_R^m(\zeta), & \zeta \in (0,1) \end{cases},$$

for each element. Here $\zeta = \xi_j(x)$ represents a local element mapping. The filtered basis functions $\chi_L^m$ and $\chi_R^m$ used in this chapter and their associated coefficients are given in Appendix E. To construct the fine mesh representation we proceed as follows. Recalling the discussion of one-dimensional MRA discussion of Section 2.3, we assume that our initial approximation is in the approximation space $V_n^p$. Letting $N = 2^n$ be the number of elements of this approximation space, denote the fine mesh approximation by $\tilde{u}_h^{(n+1)} = \mathcal{P}^{n+1} u_h^\star$, where $\mathcal{P}^{n+1}$ represents the projection operator onto the approximation space $V_{n+1}^p$ consisting of $2N$ elements. To obtain the fine mesh modal coefficients in the ansatz $\tilde{u}_h^{(n+1)}(x)\big|_{I_j} = \sum_{k=0}^p u_k^j \phi_k^j(x)$, $j = 1, \ldots, 2N$ we require for each $j$:

$$\langle \tilde{u}_h^{(n+1)}, \phi_m^j \rangle_{I_j} = \langle u_h^\star, \phi_m^j \rangle_{I_j}, \qquad m = 0, \ldots, p.$$

Having determined our fine mesh approximation $\tilde{u}_h^{(n+1)} = \mathcal{P}^{n+1} u_h^\star$, we can isolate the multiwavelet component simply by subtraction of the coarse approximation in Equation (2.68):

$$\sum_{j=0}^{2^n-1} \sum_{k=0}^{p} \tilde{d}_{k,j}^n \psi_{k,j}^n(x) = \mathcal{P}^{n+1} u_h^\star - \mathcal{P}^n u_h^\star, \tag{4.1}$$

where $\psi_{k,j}^n$ are the basis functions of the multiwavelet space. These alternative multiwavelet coefficients provide an improvement to our approximation, and the SIAC-MRA procedure provides a definition for them. This allows for effectively transitioning the approximation onto a finer mesh. Additionally, it only requires coarse mesh data to be constructed. This differs significantly from traditional MRA where multiwavelet coefficients are intrinsically dependent upon the initial function $u$, which means that we cannot construct multiwavelet coefficients for improving our approximation if only coarse modal data is available.

Knowing the form of the filtered approximation, we can map the coefficients to the fine mesh modes simply by application of a projection operator. Furthermore, knowing the forms of the filtered

approximation allows us to discretize this procedure and construct projection matrices to perform this scale transition. This will be discussed further in the context of the multidimensional case in Section 4.5.

### 4.4 Multidimensional (L)SIAC-MRA

In this section, we introduce LSIAC-MRA and generalize SIAC-MRA to higher dimensions. These procedures allows us to obtain detail coefficients for moving information to successively finer grids. These detail coefficients are obtained for a uniform mesh by choosing $r = 2p$ and $\ell = 1$ as the filter parameters in (2.3). This choice of parameters was shown in [23] to give the best error-reduction results, ensuring enough moments in the underlying data are respected as well as that over-smoothing does not occur.

It should be noted that the expressions below for (L)SIAC-MRA are mesh-dependent in that our ability to simplify the convolution is intrinsically linked to patterns in the mesh construction. The extension to nonuniform meshes will be considered in Chapter 5. In a manner analogous to the one-dimensional case, we can take advantage of the simplicity of the kernel's B-spline components and the predictability of the element boundaries to analytically evaluate the convolutions given by (2.7) and (2.8). Following this procedure, and letting $\zeta_x = \xi_i(x)$, $\zeta_y = \xi_j(y)$, and $\zeta_z = \xi_k(z)$ denote the mapping of our filtering point to the reference element, we obtain the following expressions for our filtered approximations:

The two-dimensional tensor-product filtered approximation can be written as

$$u_h^\star(x,y) = \sum_{m=0}^{p+1}\sum_{r=0}^{p+1} a_{m,r}^{Q,i,j}\chi_Q^m(\zeta_x)\chi_Q^r(\zeta_y),$$

where $(x, y)$ belongs to quadrant $Q$ of element $(i, j)$.

Analogously, the three-dimensional tensor-product filtered approximation can be written as

$$u_h^\star(x,y,z) = \sum_{m=0}^{p+1}\sum_{r=0}^{p+1}\sum_{\ell=0}^{p+1} a_{m,r,\ell}^{O,i,j,k}\chi_O^m(\zeta_x)\chi_O^r(\zeta_y)\chi_O^\ell(\zeta_z),$$

where $(x, y, z)$ belongs to octant $O$ of element $(i, j, k)$.

The two-dimensional Line SIAC, the filtered approximation can be written as

$$u_h^\star(x,y) = \sum_{P=\{L,M,R\}}\sum_{m=0}^{p}\sum_{r=0}^{p} a_{m,r}^{S,P,i,j}\chi_{S,P}^{m,r}(\zeta_x,\zeta_y), \tag{4.2}$$

where $(x, y)$ belongs to region $S$ of element $(i, j)$. These regions are delineated in Figure 4.1. The $P$ index results from splitting up the convolution into "left","middle", and "right" components owing to element breaks. Analogously, the three-dimensional LSIAC filtered approximation can be written as

$$u_h^\star(x,y,z) = \sum_{P=\{L,LM,RM,R\}}\sum_{m=0}^{p}\sum_{r=0}^{p}\sum_{\ell=0}^{p} a_{m,r,\ell}^{S,P,i,j,k}\chi_{S,P}^{m,r,\ell}(\zeta_x,\zeta_y,\zeta_z), \tag{4.3}$$

where $(x, y, z)$ belongs to region $S$ of element $(i, j, k)$, where again the $P$ index denotes an ordering with respect to element breaks. A description of the aforementioned regions and an associated classification scheme are provided in Appendices E and F.

In two dimensions, the LSIAC-MRA implementation requires 6 quadrature domains per element for the purpose of projecting the filtered approximation onto the refined mesh. In three dimensions there are 24 quadrature domains per element for the same purpose. These compare to 4 and 8 regions for the two- and three-dimensional SIAC-MRA respectively. This is owing to the difference in orientation and dimensions of the kernel support. Note that for the LSIAC filters, the filtered basis functions are more complicated integrals of shifted Legendre Polynomials. Though the reduced dimension of the LSIAC kernel support means less information is needed to compute the $a$ coefficients, the integrand is no longer separable which leads to more complicated functions in the expansion. All the $\chi$ functions and their associated coefficients are detailed in Appendices E and F.

Much the same as in one dimension, we project these filtered approximations onto the finer mesh, and thereby construct fine mesh approximations. The only difference is that we are now projecting onto either the quadrant or octants of reference element in two and three dimensions respectively. In the LSIAC case this requires us to account for the piecewise definition of the filtered approximations and split up projections according to the associated domains of definition. We discuss this more in the next section, where discrete transition operators for realizing the enhancement procedure are constructed.

### 4.5 Transition Operator Implementation

In this section we describe a brief overview of the implementation of two-dimensional LSIAC-MRA as the same ideas extend to the tensor-product case and to higher dimensions. Additional details on the filtered approximation basis functions and a computational algorithm are included in Appendices E and F.

Given coarse mesh modal information, we wish to construct transition matrices for mapping the coarse mesh modal coefficients to their fine mesh counterparts. In our discrete setting these transition matrices are the composition of the discrete filtering and projection matrices $T = \mathcal{P}^{n+1} K_H$. In two dimensions we will have a separate transition matrix for each quadrant. It is important to note that the sequencing of mesh and kernel breaks, and the elements within the kernel's support varies from region to region (see Figure 4.1). When computing the refined approximation, we must project our filtered approximation onto the finer mesh. In the two-dimensional case we must sum the projections over the two triangular regions composing quadrants I and III, respectively. The splitting of these quadrants into two separate regions is because the elements which the filter overlap changes depending upon location of the filtering point. This changes the form of the filtered approximation, and thus the form of projection matrix. As the filtered

approximation changes in this manner, different transition matrix construction is required from quadrant-to-quadrant. In the following we outline the construction procedure and refer the reader to Figure 4.2 for a depiction of the effects of each matrix in the composition for the case $p = 0$.

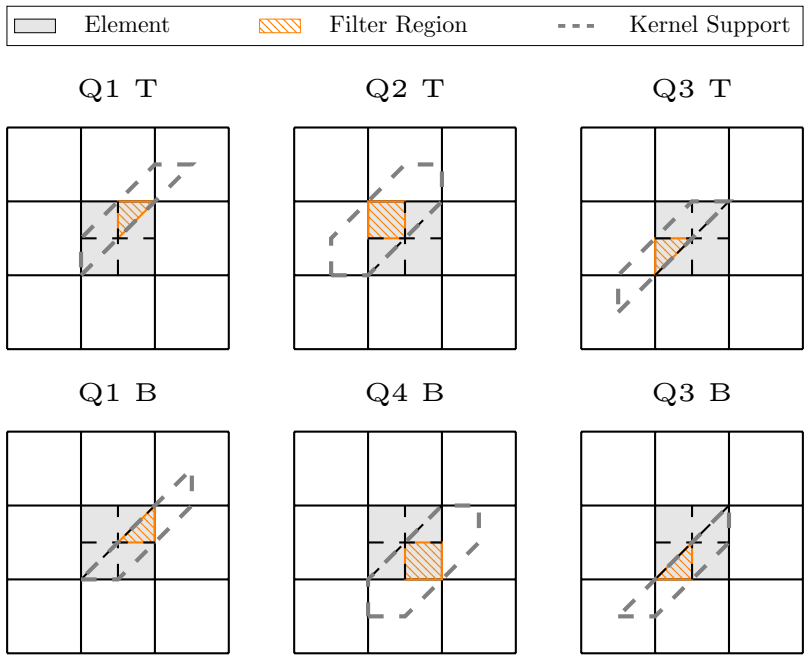


Figure 4.1 Depiction of the varying element overlap of the LSIAC kernel in two dimensions resulting in varying filtered approximation forms for $p = 0$. The element is shaded in gray, the filter region has diagonal lines, and the kernel support is encapsulated by dashed lines. The filter region (labeled by quadrant and orientation relative to the element diagonal) represents the location of possible filtering points, and the kernel support represents the overlap of the kernel support for those filtering points.

For the purposes of performing the enhancement procedure via matrix operations it is important to have a standard ordering for the modal information. As constructing the transition matrices is easier if modal coefficients are grouped by element, we denote the $(N^2(p+1))^2 \times 1$ vector of modal coefficients by

$$\vec{u} = \left[\vec{u}^1 \quad | \quad \dots \quad | \quad \vec{u}^N\right]^T,$$

where

$$\vec{u}^j = \left[\vec{u}^{1,j} \dots \vec{u}^{N,j}\right],$$

$$\vec{u}^{i,j} = \left[\vec{u}_0^{i,j} \quad | \quad \dots \quad | \quad \vec{u}_p^{i,j}\right],$$

and

$$\vec{u}_{ky}^{i,j} = \left[u_{0,ky}^{i,j} \quad | \quad \dots \quad | \quad u_{p,ky}^{i,j}\right].$$

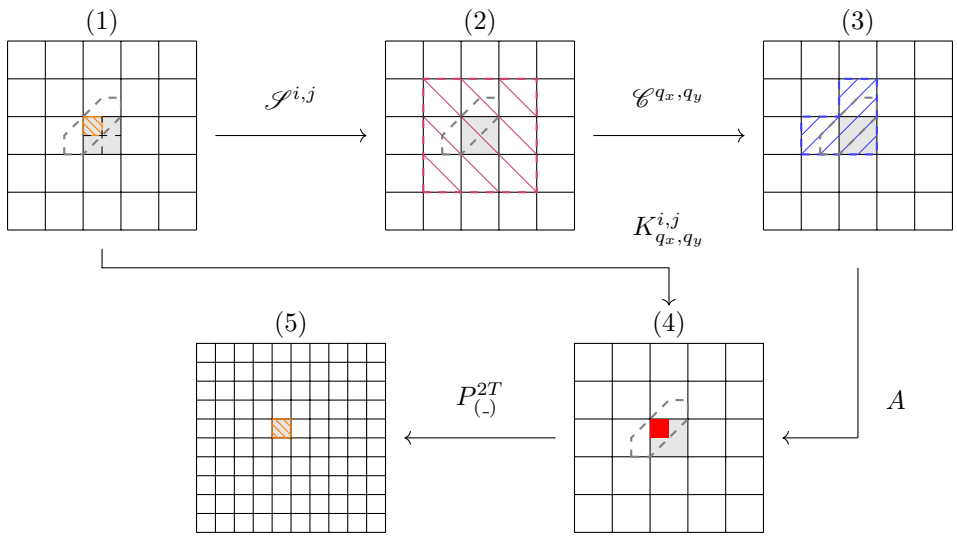


Figure 4.2 Purpose of transition matrix components during refinement procedure: (1) → (2) Select modes from elements relevant to refining element $(i, j)$, (2) → (3) Select only the modes relevant to filtering a given region, in this case $2T$, (3) → (4) Use that modal information to obtain filtered approximation coefficients, (4) → (5) Obtain fine mesh modes by projecting the filtered approximation onto the refined mesh. Notice that $K^{i,j}_{q_x,q_y} = A\mathscr{C}^{q_x,q_y}\mathscr{S}^{i,j}$.

Effectively, the modes are first concatenated along the $x$-modal index, then the $y$-modal index, then the $x$-element index, and finally the $y$-element index. Here we use a single superscript $T$ to denote the transpose of a matrix or vector. In the labelling scheme to follow we parameterize operators by labels $\{1T, 2T, 3T, 1B, 3B, 4B\}$ to denote a top or bottom orientation. To avoid confusion, note that a superscript $T$ preceded by a number does not indicate a transposition.

### 4.5.1 Sifting Matrices

All of the coefficients in (4.2) fit the form

$$a^{i,j}_{m,r}(q_x, q_y) = \sum_{\gamma=-p}^{p} c_\gamma u^{i+\gamma+q_x, j+\gamma+q_y}_{kx,ky}, \tag{4.4}$$

where $q_x, q_y \in \{-1, 0, 1\}$ are uniquely determined by the region $S$ and position $P$ as detailed in Appendices E and F. To compute these coefficients, a selection matrix, denoted $\mathscr{S}^{i,j}$, must be constructed. Assuming periodic boundary conditions, this matrix will select all modal values from the $(2p+3)^2$ elements about element $(i, j)$. We will later further pare this information to only those elements under the line filter's support. We begin here by building this selection matrix by concatenation of simpler matrices. Define the $(2p+3)N(p+1)^2 \times N^2(p+1)^2$ matrix $S^j$ by block concatenation:

$$S^j = \begin{bmatrix} \omega_j(1,1)\mathbb{I}_{N(p+1)^2\times N(p+1)^2} & \dots & \omega_j(1,N)\mathbb{I}_{N(p+1)^2\times N(p+1)^2} \\ \vdots & & \vdots \\ \omega_j(2p+3,1)\mathbb{I}_{N(p+1)^2\times N(p+1)^2} & \dots & \omega_j(2p+3,N)\mathbb{I}_{N(p+1)^2\times N(p+1)^2} \end{bmatrix}$$

where

$$\omega_j(m,r) = \begin{cases} 1, & j \neq N \;\&\; p+2-m+r = j \mod (N) \\ 1, & j = N \;\&\; p+2-m+r = j \text{ or } 0 \\ 0, & \text{else} \end{cases},$$

for $m = 1:(2p+3)$ and $r = 1:N$. Here $\mathbb{I}_{N(p+1)^2\times N(p+1)^2}$ is the $N(p+1)^2 \times N(p+1)^2$ identity matrix. This matrix selects the relevant $j$-coordinates:

$$S^j\vec{u} = [u^{j-(p+1)} \;\dots\; u^{j+(p+1)}]^T.$$

Next, construct an $(2p+3)(p+1)^2 \times N(p+1)^2$ matrix $\tilde{S}^i$ to select the relevant $i$-coordinates from within each vector. Define by block concatenation

$$\tilde{S}^i = \begin{bmatrix} \omega_i(1,1)\mathbb{I}_{(p+1)^2\times(p+1)^2} & \dots & \omega_i(1,N)\mathbb{I}_{(p+1)^2\times(p+1)^2} \\ \vdots & & \vdots \\ \omega_i(2p+3,1)\mathbb{I}_{(p+1)^2\times(p+1)^2} & \dots & \omega_i(2p+3,N)\mathbb{I}_{(p+1)^2\times(p+1)^2} \end{bmatrix}$$

for $m = 1:(2p+3)$ and $r = 1:N$. We have that

$$\tilde{S}^i u^j = [u^{i-(p+1),j} \;\dots\; u^{i+(p+1),j}]^T.$$

Hence, defining $\mathscr{S}^{i,j}$ to be the $(2p+3)^2(p+1)^2 \times N^2(p+1)^2$ matrix given by

$$\mathscr{S}^{i,j} = \begin{bmatrix} \tilde{S}^i & & 0 \\ & \ddots & \\ 0 & & \tilde{S}^i \end{bmatrix} S^j,$$

where there are $(2p+3)$ $\tilde{S}^i$ matrices on the diagonal, we have

$$\mathscr{S}^{i,j}u = \Big[[u^{i-(p+1),j-(p+1)} \;\dots u^{i+(p+1),j-(p+1)}] \;\dots\; [u^{i-(p+1),j+(p+1)} \;\dots u^{i+(p+1),j+(p+1)}]\Big]^T = \tilde{u}^{i,j}$$

which, as stated previously, only contains modal information from the $(2p+3)^2$ elements around element $(i,j)$.

Now we construct an intermediary matrix $\mathscr{C}^{q_x,q_y}$ whose function is to select the elements from $\tilde{u}^{i,j}$ which fall within the support of the line integral as affected by the shifting arguments $(q_x, q_y)$. Define the $(2p+1)(2p+3)(p+1)^2 \times (2p+3)^2(p+1)^2$ matrix $C^{q_y}$ by

$$C^{q_y} = \begin{bmatrix} \delta_{-1,q_y}\mathbb{I} & \delta_{0,q_y}\mathbb{I} & \delta_{1,q_y}\mathbb{I} & & 0 \\ & \ddots & \ddots & \ddots & \\ 0 & & \delta_{-1,q_y}\mathbb{I} & \delta_{0,q_y}\mathbb{I} & \delta_{1,q_y}\mathbb{I} \end{bmatrix},$$

where $\delta$ is the Kronecker delta and $\mathbb{I}$ is the $(2p+3)(p+1)^2 \times (2p+3)(p+1)^2$ identity matrix. This matrix has the effect of further restricting the relevant $j-$indices based off the shifting argument $q_y$. Then define the $(2p+1)(p+1)^2 \times (2p+3)(2p+1)(p+1)^2$ matrix $\mathscr{C}^{q_x}$ by

$$\mathscr{C}^{q_x} = \begin{bmatrix} \delta_{-1,q_x}\mathbb{I} & \delta_{0,q_x}\mathbb{I} & \delta_{1,q_x}\mathbb{I} & 0 \\ & & R_2 & \\ & & \vdots & \\ & & R_{2p+1} & \end{bmatrix},$$

where $\mathbb{I}$ is $(p+1)^2 \times (p+1)^2$, and $R_j$ is the first block row defined above with the non-zero entries translated $(j-1)(2p+4)(p+1)^2$ places to the right. We then have the desired $(2p+1)(p+1)^2 \times (2p+3)^2(p+1)^2$ intermediary matrix $\mathscr{C}^{q_x,q_y}$ given by the composition

$$\mathscr{C}^{q_x,q_y} = \mathscr{C}^{q_x} C^{q_y}.$$

Applying this matrix produces:

$$\mathscr{C}^{q_x,q_y}\tilde{u}^{i,j} = \tilde{u}^{i,j}_{q_x,q_y} = [u^{i-(p-q_x),j-(p-q_y)}, \ldots, u^{i+(p+q_x),j+(p+q_y)}]^T,$$

which are the modal coefficients appearing in (4.4).

### 4.5.2 Coefficient Matrices

We now describe the construction of a matrix for computing (4.4) from the sifted modal coefficients. Define the matrix $A$ to be the $(p+1)^2 \times (2p+1)(p+1)^2$ matrix given by

$$A = \frac{1}{2}\begin{bmatrix} c_{-p}\mathbb{I} & & 0 & | & \ldots & | & c_p\mathbb{I} & & 0 \\ 0 & c_{-p}\mathbb{I} & 0 & | & \ldots & | & 0 & c_p\mathbb{I} & 0 \\ & \ddots & & | & \ldots & | & & \ddots & \\ 0 & & c_{-p}\mathbb{I} & | & \ldots & | & 0 & & c_p\mathbb{I} \end{bmatrix}$$

where the identity matrices are $(p+1) \times (p+1)$. This is the matrix used to perform the summation and leads to

$$A\tilde{u}^{i,j}_{q_x,q_y} = \vec{a}^{i,j}(q_x,q_y)$$

where

$$\vec{a}^{i,j}(q_x,q_y) = [a^{i,j}_{r=0} \ \ldots \ a^{i,j}_{r=p}]^T(q_x,q_y)$$

with

$$a^{i,j}_r(q_x,q_y) = [a^{i,j}_{0,r}(q_x,q_y) \ \ldots \ a^{i,j}_{p,r}(q_x,q_y)].$$

Hence, the seven coefficient matrices $K^{i,j}_{q_x,q_y}$ are defined by the composition

$$K^{i,j}_{q_x,q_y} = A\mathscr{C}^{q_x,q_y}\mathscr{S}^{i,j}$$

and give

$$K^{i,j}_{q_x,q_y} u = \vec{a}^{i,j}(q_x, q_y).$$

Rather than concatenating these matrices at this point in the procedure, we will wait until after applying the projection in order to minimize the number of transformations performed. It is important to note that each choice of $(q_x, q_y)$ computes a different set of coefficients pertaining to a different set of basis functions, each of which is required for obtaining the enhanced approximation.

### 4.5.3 Projection Matrices

We now discuss the procedure necessary to perform the projections. Because of the different filtered approximation forms in each quadrant, we must perform each projection slightly differently. Quadrants II and IV are similar as are I and III owing to the number of filtered approximation forms present on that element. Here we begin with the procedure for quadrant II.

#### 4.5.3.1 Quadrant II

In quadrant II we define the projection of this approximation by $\tilde{u}_h^{(n+1)}|_{K^{2i-1,2j}}(x, y)$ where the new modal coefficients are computed by

$$\langle u_h^\star, \phi_{kx}\phi_{ky}\rangle_{QII} = \langle \tilde{u}_h^{(n+1)}, \phi_{kx}\phi_{ky}\rangle_{QII} \tag{4.5}$$

$$= u^{2i-1,2j}_{kx,ky}, \quad kx, ky = 0, \ldots, p, \;\; i, j = 1, \ldots, N. \tag{4.6}$$

Note that here and in what follows, we suppress the tilde and $(n + 1)$ superscript on the fine mesh modal values for readability. To account for the change in domain scaling define $\xi_x(\zeta_x) = \frac{\zeta_x - 1}{2}$ and $\xi_y(\zeta_y) = \frac{\zeta_y + 1}{2}$. We have

$$\begin{aligned} u^{2i-1,2j}_{kx,ky} = \sum_{m=0}^{p}\sum_{r=0}^{p}\int_{-1}^{1}\int_{-1}^{1} \Big\{ & a^{i,j}_{m,r}(-1,0)\chi^{m,r}_{2T,L}(\xi_x,\xi_y) \\ & + a^{i,j}_{m,r}(0,0)\chi^{m,r}_{2T,M}(\xi_x,\xi_y) \\ & + a^{i,j}_{m,r}(0,1)\chi^{m,r}_{2T,R}(\xi_x,\xi_y)\Big\}\phi_{kx}(\zeta_x)\phi_{ky}(\zeta_y)\, d\zeta_y\, d\zeta_x. \end{aligned}$$

This expression will be broken up into three separate integrals, one for each of the different $\chi_{(_)}$ functions where $(_) \in \{L, M, R\}$. For each $\chi_{(_)}$, define

$$P_{(_)}(kx, ky, m, r) = \int_{-1}^{1}\int_{-1}^{1} \chi^{m,r}_{(2T,_)}(\xi_x,\xi_y)\phi_{kx}(\zeta_x)\phi_{ky}(\zeta_y)\, d\zeta_y\, d\zeta_x.$$

Define the projection matrix $P^{2T}_{(_)}$ by

$$P^{2T}_{(_)} = \begin{bmatrix}\vec{P}(0,0) & \ldots & \vec{P}(p,0) & \ldots & \vec{P}(0,p) & \ldots & \vec{P}(p,p)\end{bmatrix}^T,$$

where

$$\vec{P}(kx,ky) = [P(kx,ky,0,0)\dots P(kx,ky,p,0)\dots\dots P(kx,ky,0,p)\dots P(kx,ky,p,p)]^T.$$

Hence, we have

$$u_{2i-1,2j} = P_L^{2T}\vec{a}^{i,j}(-1,0) + P_M^{2T}\vec{a}^{i,j}(0,0) + P_R^{2T}\vec{a}^{i,j}(0,1).$$

Noting the necessary choice of $K^{i,j}_{q_x,q_y}$ to obtain each of the coefficient vectors, we have

$$u_{2i-1,2j} = (P_L^{2T}K^{i,j}_{-1,0} + P_M^{2T}K^{i,j}_{0,0} + P_R^{2T}K^{i,j}_{0,1})\vec{u}.$$

Denoting this composition by $T^2_{i,j}$ we can concatenate these matrices together to obtain the global transition matrix $T^2$ defined by

$$T^2 = \begin{bmatrix} T^2_{1,1} & \dots & T^2_{N,1} & |\dots| & T^2_{1,N} & \dots T^2_{N,N} \end{bmatrix}^T.$$

We obtain the modal coefficients for the refined approximation over the second quadrants of the coarse mesh by

$$\tilde{u}_2^{(n+1)} = T^2 u.$$

#### 4.5.3.2 Quadrant IV

The projection matrices for quadrant IV are the same except that the quadrant IV basis function are used and the change of variables are instead $\xi_x(\zeta_x) = \frac{\zeta_x+1}{2}$ and $\xi_y(\zeta_y) = \frac{\zeta_y-1}{2}$.

#### 4.5.3.3 Quadrant III

The cases of quadrants I and III are similar in that a change of variables must be applied to split the integrals over the quadrant into a pair of integrals over diagonals of that quadrant. Beginning with the integral over the whole quadrant, we have the fine mesh modal coefficients

$$\begin{aligned} u^{2i-1,2j-1}_{kx,ky} &= \langle u_h^\star, \phi^i_{kx}\phi^j_{ky}\rangle_{QIII} \\ &= \int_{-1}^{1}\int_{-1}^{1} u_h^\star|_{QIII}(\xi_x(\zeta_x),\xi_y(\zeta_y))\phi^i_{kx}(\zeta_x)\phi^j_{ky}(\zeta_y)\, d\zeta_y\, d\zeta_x, \end{aligned}$$

where $\xi_x = \frac{\zeta_x-1}{2}$ and $\xi_y = \frac{\zeta_y-1}{2}$. Breaking this up into integrals over $3T$ and $3B$ we have

$$\begin{aligned} u^{2i-1,2j-1}_{kx,ky} &= \int_{-1}^{1}\int_{\zeta_x}^{1} u^\star_{3T}(\xi_x,\xi_y)\phi^i_{kx}(\zeta_x)\phi^j_{ky}(\zeta_y)\, d\zeta_y\, d\zeta_x \\ &+ \int_{-1}^{1}\int_{-1}^{\zeta_x} u^\star_{3B}(\xi_x,\xi_y)\phi^i_{kx}(\zeta_x)\phi^j_{ky}(\zeta_y)\, d\zeta_y\, d\zeta_x. \end{aligned}$$

Now to use Gauss-Legendre quadrature, we must scale the bounds of the inner integral to $[-1,1]$. To do so introduce the change of variables $\zeta_y = \frac{1-\zeta_x}{2}\alpha_T + \frac{1+\zeta_x}{2}$ for the first integral and $\zeta_y = \frac{\zeta_x-1}{2}\alpha_B + \frac{1+\zeta_x}{2}$ for the

second. This allows us to rewrite the expression as

$$
\begin{aligned}
u_{kx,ky}^{2i-1,2j-1} &= \int_{-1}^{1}\int_{-1}^{1} u_{3T}^{\star}(\xi_x,\xi_y(\zeta_y(\alpha_T))\phi_{kx}^{i}(\zeta_x)\phi_{ky}^{j}\Big(\frac{1}{2}\big[\alpha_T(1-\zeta_x)+1+\zeta_x\big]\Big)\frac{1-\zeta_x}{2}\ d\alpha_T\ d\zeta_x \\
&+ \int_{-1}^{1}\int_{-1}^{1} u_{3B}^{\star}(\xi_x,\xi_y(\zeta_y(\alpha_B))\phi_{kx}^{i}(\zeta_x)\phi_{ky}^{j}\Big(\frac{1}{2}\big[\alpha_B(\zeta_x-1)+1+\zeta_x\big]\Big)\frac{\zeta_x-1}{2}\ d\alpha_B\ d\zeta_x. \\
&= \mathscr{I}^T + \mathscr{I}^B.
\end{aligned}
$$

We now need to split the $\mathscr{I}$ terms into a sum of integrals over the basis functions and then multiply by the appropriate modal coefficients. Defining

$$
P_{(_)}^{3T}(kx,ky,m,r) = \int_{-1}^{1}\int_{-1}^{1} \chi_{3T,(_)}^{m,r}(\xi_x,\xi_y(\alpha_T))\phi_{kx}(\zeta_x)\phi_{ky}(\zeta_y(\alpha_T))\ d\zeta_x\ d\alpha_T,
$$

where $(_) \in \{L,M,R\}$, we construct the projection matrices $P_L^{3T}, P_M^{3T}, P_R^{3T}$ in the same manner as in quadrant II. Now for the second set of integrals define

$$
P_{(_)}^{3B}(kx,ky,m,r) = \int_{-1}^{1}\int_{-1}^{1} \chi_{3B,(_)}^{kx,ky}(\xi_x,\xi_y(\alpha_B))\phi_{kx}(\zeta_x)\phi_{ky}(\zeta_y(\alpha_B))\ d\zeta_x\ d\alpha_B,
$$

and analogously construct the projection matrices $P_L^{3B}, P_M^{3B}, P_R^{3B}$. We then have

$$
\begin{aligned}
u_{2i-1,2j-1} = \quad & P_L^{3T}\vec{a}^{i,j}(-1,-1) + P_M^{3T}\vec{a}^{i,j}(-1,0) + P_R^{3T}\vec{a}^{i,j}(0,0) \\
& + P_L^{3B}\vec{a}^{i,j}(-1,-1) + P_M^{3B}\vec{a}^{i,j}(0,-1) + P_R^{3B}\vec{a}^{i,j}(0,0).
\end{aligned}
$$

Now $\vec{a}^{i,j}(q_x,q_y) = K_{q_x,q_y}^{i,j}\vec{u}$, thus

$$
u_{2i-1,2j-1} = \Big[(P_L^{3T}+P_L^{3B})K_{-1,-1}^{i,j} + (P_R^{3T}+P_R^{3B})K_{0,0}^{i,j} + P_M^{3T}K_{-1,0}^{i,j} + P_M^{3B}K_{0,-1}^{i,j}\Big]\vec{u}.
$$

Denoting this composition by $T_{i,j}^3$ we can concatenate these matrices together to obtain the global transition matrix $T^3$ defined by

$$
T^3 = \begin{bmatrix} T_{1,1}^3 & \dots & T_{N,1}^3 & |\dots| & T_{1,N}^3 & \dots T_{N,N}^3 \end{bmatrix}^T,
$$

such that

$$
\tilde{u}_3^{(n+1)} = T^3 u.
$$

#### 4.5.3.4 Quadrant I

The case of quadrant I is that same as quadrant III except now $\xi_x = \frac{\zeta_x+1}{2}$, $\xi_y = \frac{\zeta_y+1}{2}$, and the $\chi$ functions used are those defined on quadrant I.

### 4.5.4 Coding Comment:

The LSIAC-MRA and SIAC-MRA procedures were implemented in MATLAB$^{TM}$ R2021a. No claim is made to the computational efficiency of the implementation but we simply state that the memory requirements are on the order of the number of modal values of the finer mesh $4N^2(p+1)^2$. An algorithm to perform the enhancement, requiring only formation of filtering and local projection matrices is provided in Algorithm 2.

## 4.6 Numerical Results

The goal of (L)SIAC-MRA is to produce a fine mesh modal approximation from coarse mesh data that has lower error than repeated $L^2$-projection alone. The measures used for comparison are the $L^2$- and $L^\infty$-errors defined respectively by

$$\|u_{exact} - u_{approx}\|_{L^2(\Omega)} = \sqrt{\int_\Omega |u_{exact} - u_{approx}|^2 \, d\Omega}\,, \qquad \|u_{exact} - u_{approx}\|_\infty = \sup_{x\in\Omega} |u_{exact} - u_{approx}|. \tag{4.7}$$

For our simulations, the $L^2$-error is discretely approximated by Gauss-Legendre quadrature at $6^d$ nodes per element. Similarly, the $L^\infty$-error is taken to be the maximum absolute error over these nodes. To ensure a consistent standard for comparison, we compute the errors on the final, most refined mesh occurring during the enhancement procedure. It is very important to use a standard mesh for computing the errors. Not doing so can lead to counter-intuitive behaviors in the error. An example of this is displayed in Figure 4.3 where repeated projections of the same polynomial approximation have varying errors, contradicting the reproduction property of the projection. The cause is the discrete approximation of the error-norms. By performing the error computation over a fixed sufficiently fine mesh, we no longer encounter these discretization artifacts.

In all the simulations considered below, periodic boundary conditions are assumed to allow the application of the filter in regions where the kernel support would surpass the domain boundaries. Before proceeding to the numerical results, a note about the error tables to appear in this section. The first two columns of the tables detail the projections errors of the $L^2$ approximation subject to scale transition by projection without filtering. This represents the standard which our filtered approximations must outperform to justify the computational expense. The third and fourth columns detail the errors of an approximation when (L)SIAC-MRA is applied only once and then $L^2$-projection is used. The final two columns detail the errors of the (L)SIAC-MRA procedure applied at each refinement. As a result, the first rows are equal. Similarly, the Enhanced Once and Enhanced Each Refinement columns will agree on the second mesh simply because only one filtering procedure has been applied at that point. These equalities

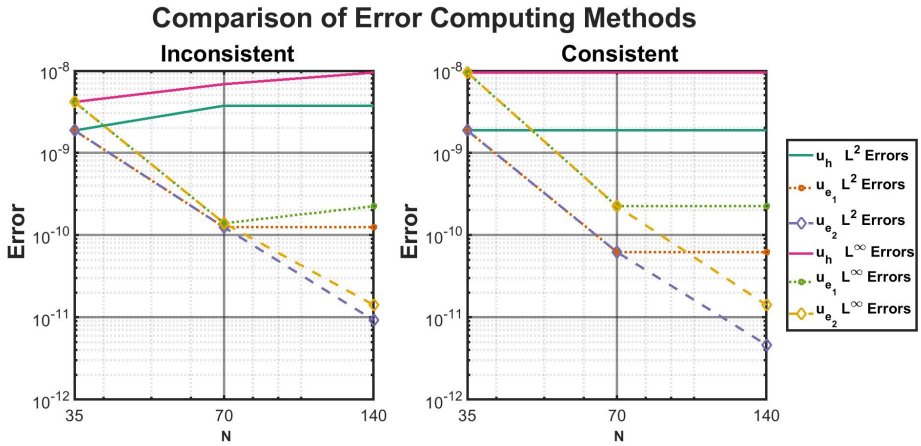


Figure 4.3 Comparison of the Log-Log plots of the $L^2$ and $L^\infty$ errors of the function $u_0(x, y) = \sin(2\pi x) + \sin(2\pi y)$ with $p = 4$ under refinement. In the left plot, the errors are computed on the mesh level of the iteration, i.e on the mesh size indicated by the data points abscissa. In the right plot, the errors are computed on the $140 \times 140$ mesh which is the finest mesh encountered during the procedure. We observe that this latter approach allows for the errors to remain constant under $L^2$-projection alone, which agrees with the intuition of the approximation being reproduced.

are emphasized by the shading of entries in the tables (gray and medium gray, respectively). We have performed many two- and three-dimensional experiments to gauge the ability of the (L)SIAC-MRA filtering-projection procedure, but below we will only focus on the most illuminating examples. Note that we only expect reduced errors, not improvement in order of convergence. We focus mainly on LSIAC-MRA but provide some results for SIAC-MRA as well.

### 4.7 Two-dimensional Test Problems

Consider the following functions on the domain $\Omega = \{(x, y) \in [0, 1]^2\}$ :

1. $u_0(x, y) = \sin(2\pi(x + y))$
2. $u_0(x, y) = \sin(10\pi x)\sin(10\pi y)$
3. $u_0(x, y) = f_1(x)f_1(y)$
4. $u_0(x, y) = f_2(x)f_2(y)$

where

$$f_1(x) = \begin{cases} 2\cos(2\pi(2x-1)), & 1/4 \le x \le 3/4 \\ \cos(4\pi(2x-1)), & Else \end{cases},$$

$$f_2(x) = \begin{cases} 2/3\sin(2\pi(2x-1)), & 1/4 \le x \le 3/4 \\ \cos(\pi(2x-1)), & Else \end{cases}.$$

These tests aid us in evaluating the effectiveness of the (L)SIAC-MRA procedure for different function types. The first function can be viewed as a combination of a product and summation of the coordinate directions. The second represents a higher frequency function. The third is discontinuous, while the fourth has discontinuous derivatives for $x$ and $y$ at $1/4$ and $3/4$. Depictions of the third and fourth functions are provided in Figure 4.4.

Performing the (L)SIAC-MRA on functions with different wave number $k$, we observe that different initial mesh resolutions are necessary for improvement. The first and second functions behave similarly, except that the first corresponds to a wavenumber $k = 1$ and has repeated improvement from (L)SIAC-MRA on a relatively coarse mesh ($35 \times 35$, see Figure 4.5 and Figure 4.6). We see clearly that the tensor-product formulation outperforms the formulation in pointwise error reduction. This makes sense as in the former case more modal information is being used in the reconstruction. The higher frequency second function $k = 5$, (see Figure 4.7 and Table 4.1) requires a much finer starting resolution of $160 \times 160$ for improvement in the LSIAC-MRA case. Even then, the resolution requirements increase with approximation order and so for coarser meshes the LSIAC-MRA procedure is more effective for lower order polynomial approximations. Note the omission of the tensor-product for the second function owing to computer memory constraints.

For the third and fourth functions we begin with their projection onto a piecewise orthonormal Legendre basis defined over a $70 \times 70$ uniform quadrilateral mesh. We observe that the refinement procedure provides error reduction for lower polynomial degrees, though performance degrades for increasing $p$. While (L)SIAC-MRA is effective in smooth regions, it is not as globally effective at error reduction as compared to the analytic examples above. This is due to the lack of smoothness of the functions under consideration.

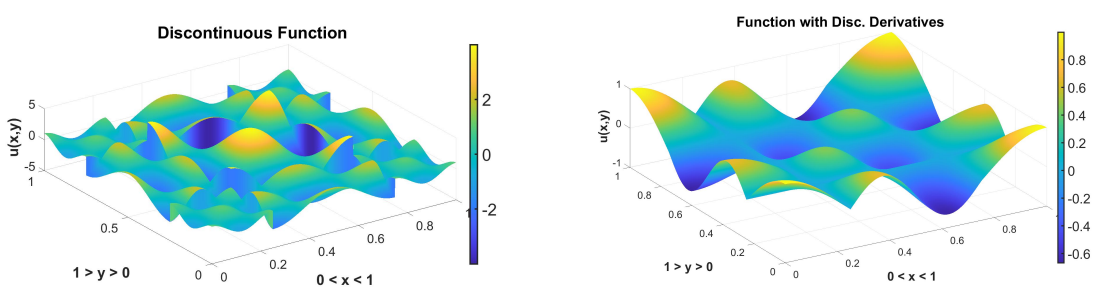


Figure 4.4 Surface plots of $u_0(x, y) = f_1(x)f_1(y)$ (Left) and $u_0(x, y) = f_2(x)f_2(y)$ (Right). The initial condition defined as a tensor-product of a discontinuous function $f_1$ represents a "strong discontinuity", while that using a tensor-product of a continuous but not globally differentiable function $f_2$ represents a continuous function with discontinuous derivatives $\frac{\partial u_0}{\partial x}$ and $\frac{\partial u_0}{\partial y}$.

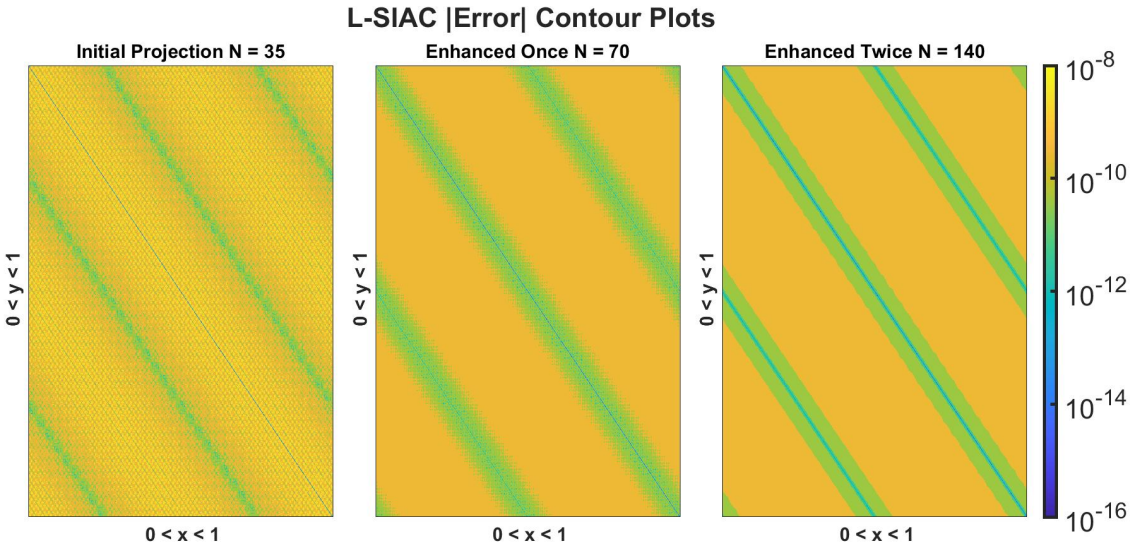


Figure 4.5 Contour plots depicting pointwise |Error| for initial condition $u_0(x, y) = \sin(2\pi(x + y))$ on a series of three meshes using **2D LSIAC-MRA** for $p = 4$. Notice that the filtering procedure regularizes and reduces the errors when it is applied at each refinement.

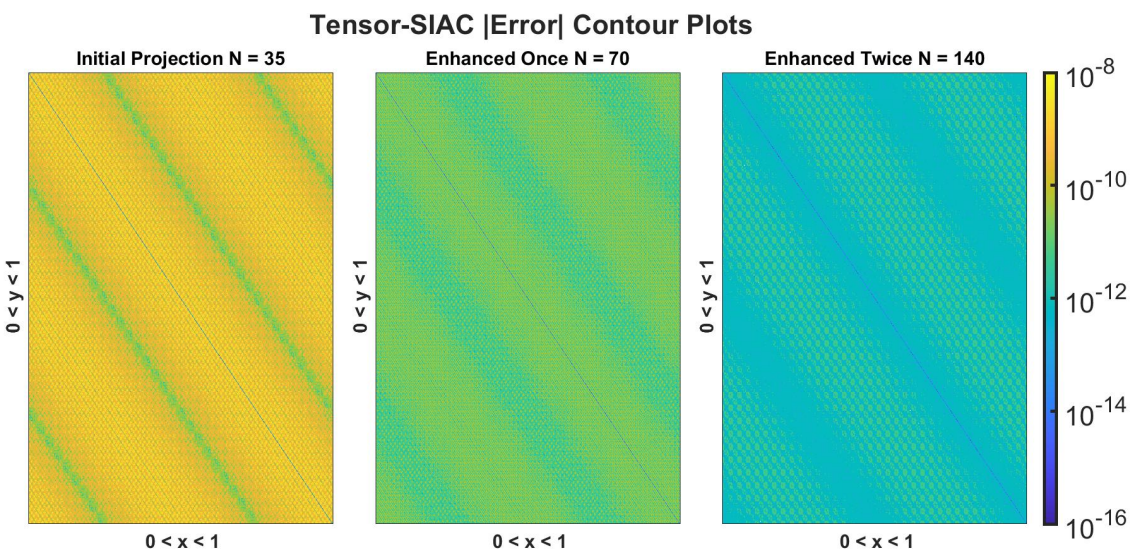


Figure 4.6 Contour plots depicting pointwise |Error| for initial condition $u_0(x, y) = \sin(2\pi(x + y))$ on a series of three meshes using **2D SIAC-MRA** for $p = 4$. Notice that the filtering procedure regularizes and reduces the errors when it is applied at each refinement.

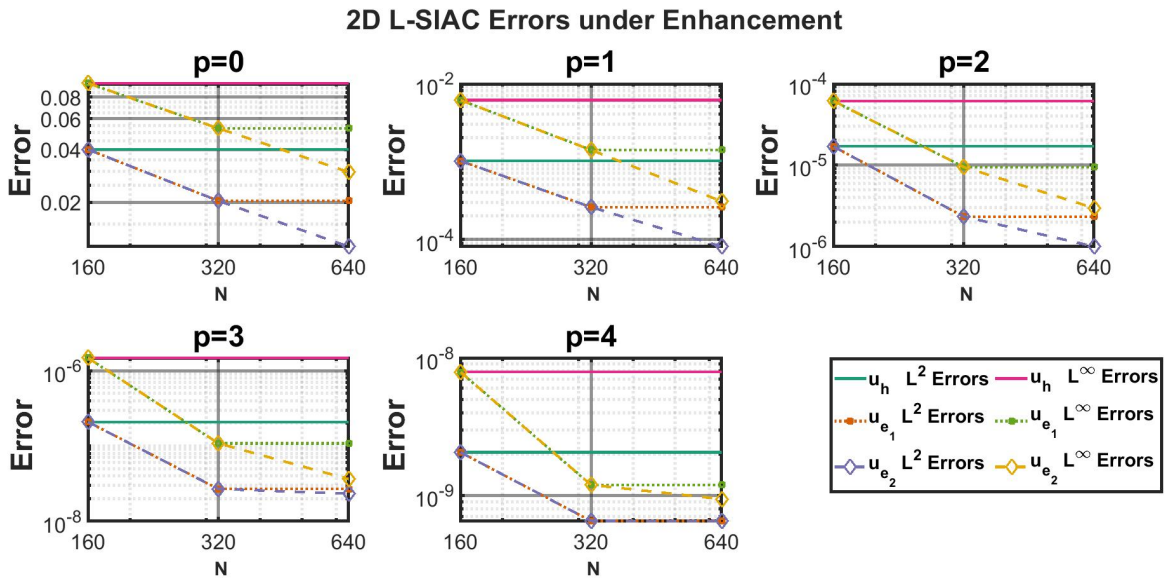


Figure 4.7 Log-Log plots of $N$ vs. Error for a high frequency function $u_0(x, y) = \sin(10\pi x)\sin(10\pi y)$ with wavenumber $k = 5$ on a series of three meshes using **2D LSIAC-MRA**. We see that the effectiveness somewhat degrades with increasing polynomial order.

Table 4.1 Table of $L^2-$ and $L^\infty-$errors for the high frequency initial condition $u(x, y) = \sin(10\pi x)\sin(10\pi y)$ on a series of three meshes using **2D LSIAC-MRA**. Because it is high frequency, we have to initialize our approximation on a more refined grid. The shaded regions of the same shade represent the same approximation. For higher wavenumbers, the enhancement procedure requires greater initial mesh resolution for error reduction.

| | **Projection Error** | | **Enhanced Once** | | **Enhanced Each Refinement** | |
|---|---|---|---|---|---|---|
| **N** | $L^2$**-Error** | $L^\infty$**-Error** | $L^2$**-Error** | $L^\infty$**-Error** | $L^2$**-Error** | $L^\infty$**-Error** |
| | | | $\mathbb{P}^0$ | | | |
| **160** | $4.01e-02$ | $9.59e-02$ | $4.01e-02$ | $9.59e-02$ | $4.01e-02$ | $9.59e-02$ |
| **320** | $4.01e-02$ | $9.59e-02$ | $2.04e-02$ | $5.30e-02$ | $2.04e-02$ | $5.30e-02$ |
| **640** | $4.01e-02$ | $9.59e-02$ | $2.04e-02$ | $5.30e-02$ | $1.11e-02$ | $2.97e-02$ |
| | | | $\mathbb{P}^1$ | | | |
| **160** | $1.02e-03$ | $6.10e-03$ | $1.02e-03$ | $6.10e-03$ | $1.02e-03$ | $6.10e-03$ |
| **320** | $1.02e-03$ | $6.10e-03$ | $2.60e-04$ | $1.40e-03$ | $2.60e-04$ | $1.40e-03$ |
| **640** | $1.02e-03$ | $6.10e-03$ | $2.60e-04$ | $1.40e-03$ | $8.11e-05$ | $3.08e-04$ |
| | | | $\mathbb{P}^2$ | | | |
| **160** | $1.69e-05$ | $6.07e-05$ | $1.69e-05$ | $6.07e-05$ | $1.69e-05$ | $6.07e-05$ |
| **320** | $1.69e-05$ | $6.07e-05$ | $2.33e-06$ | $9.44e-06$ | $2.33e-06$ | $9.44e-06$ |
| **640** | $1.69e-05$ | $6.07e-05$ | $2.33e-06$ | $9.44e-06$ | $1.00e-06$ | $2.97e-06$ |
| | | | $\mathbb{P}^3$ | | | |
| **160** | $2.08e-07$ | $1.48e-06$ | $2.08e-07$ | $1.48e-06$ | $2.08e-07$ | $1.48e-06$ |
| **320** | $2.08e-07$ | $1.48e-06$ | $2.69e-08$ | $1.08e-07$ | $2.69e-08$ | $1.08e-07$ |
| **640** | $2.08e-07$ | $1.48e-06$ | $2.69e-08$ | $1.08e-07$ | $2.32e-08$ | $3.63e-08$ |
| | | | $\mathbb{P}^4$ | | | |
| **160** | $2.06e-09$ | $7.91e-09$ | $2.06e-09$ | $7.91e-09$ | $2.06e-09$ | $7.91e-09$ |
| **320** | $2.06e-09$ | $7.91e-09$ | $6.51e-10$ | $1.19e-09$ | $6.51e-10$ | $1.19e-09$ |
| **640** | $2.06e-09$ | $7.91e-09$ | $6.51e-10$ | $1.19e-09$ | $6.47e-10$ | $9.39e-10$ |

In cases 3 and 4, application of the SIAC filter near a discontinuity will smear the discontinuity by introducing artificial smoothness, and the pollution region will increase with each application of the filter. However, for these functions with non-smooth components occurring along lines, the pollution region will reach a finite extent of $h(2p + 3/2)$ in any Cartesian direction from the point of discontinuity, where $h$ is the uniform coarse mesh size. If we look at the contour plot given in Figure 4.8 and Figure 4.9 or the zoomed in plots of Figure 4.10 and Figure 4.11, we can in fact observe that near the discontinuous derivative of the function at $x, y = 1/4, 3/4$, increased errors occur with refinement.

This error growth is lesser than compared to the discontinuous test case 3. We do not include refinement contour plots for test case 3 as the error distribution is similar to test case 4. Away from the polluted regions, significant error reduction occurs with refinement. The $L^2$- and $L^\infty$-errors of the enhancement procedure *excluding the polluted regions* for these less smooth functions are given in Table 4.2 and Table 4.3 for the LSIAC enhancement, and Table 4.4 and Table 4.5 for the tensor-product enhancement. We do not include the polluted errors because, as expected, the $L^\infty$-errors stay relatively constant while $L^2$-errors grow as the polluted region covers a larger portion of the domain. The pollution regions for tests 3 and 4 are the same, and so we use an identical procedure for determining the pollution regions and computing the errors. The procedure is relatively simple, if an element on the coarse mesh contains a non-smooth feature caused by lack of smoothness in the initial condition, we exclude that element. When we enhance, we wish to exclude any element where the filtered approximation on that element drew information from polluted regions. As the non-smooth components of these functions occur along the lines $x, y = 1/4, 3/4$ we can exclude any element within $h(p + 1/2)$ in any Cartesian direction of the non-smooth coarse elements on the refined mesh after application of (L)SIAC-MRA. Finally, following the second and final application of (L)SIAC-MRA, we exclude any element on the finest mesh within $\frac{3}{2}h(p + 1/2)$ in any Cartesian direction of the non-smooth coarse elements. We observe that application of the filtering procedure at each stage continues to provide error reductions with each refinement and outperforms only applying the filtering procedure once, except in the LSIAC-MRA $p = 4$ case for the third function. This is caused by insufficient resolution in the initial mesh, and increasing the initial resolution enables continued error reduction.

As depicted in the Log-Log plots in Figure 4.7 and Figure 4.12, (L)SIAC-MRA can indeed provide error reduction for sufficiently smooth functions; however, this technique does not guarantee an improvement in the order of the method. The contour plots in Figure 4.5, Figure 4.6, Figure 4.8, and Figure 4.9 display that repeated application of (L)SIAC-MRA will alter not only the magnitude of errors, but also their distribution throughout the domain.

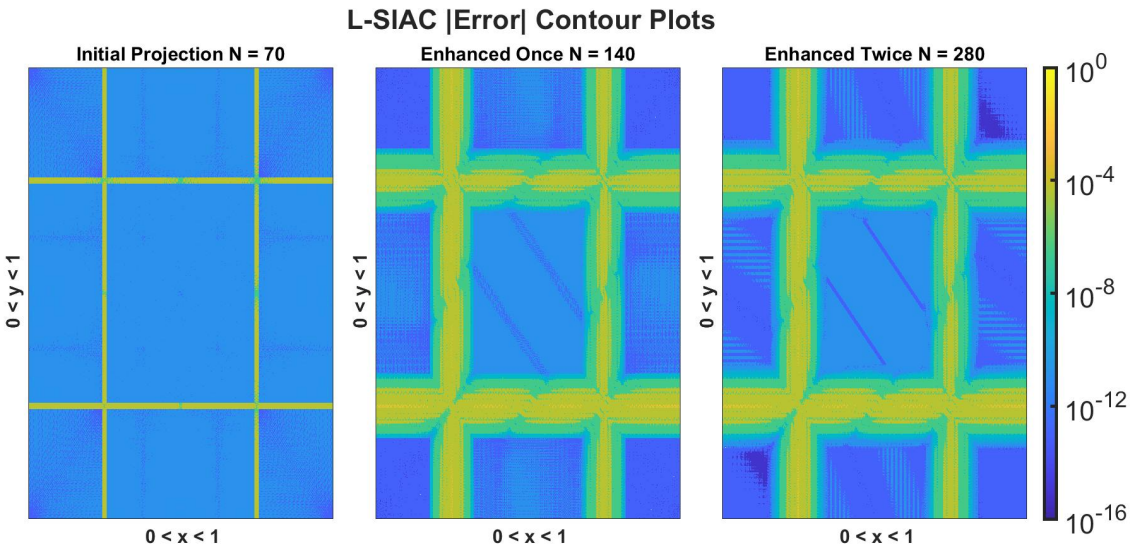


Figure 4.8 Contour plots depicting pointwise |Error| for the initial condition $u_0(x, y) = f_2(x)f_2(y)$ which contains a discontinuity in its first derivatives $\frac{\partial u_0}{\partial x}$ and $\frac{\partial u_0}{\partial y}$. 2D LSIAC-MRA was performed over a series of three meshes for $p = 4$. Notice that owing to the jump discontinuity in the derivatives at $x, y = 1/4, 3/4$, the polluted region where filtering adds artificial smoothness grows with each enhancement. Away from these locations, we still have error improvement. A zoomed in view is given in Figure Figure 4.10.

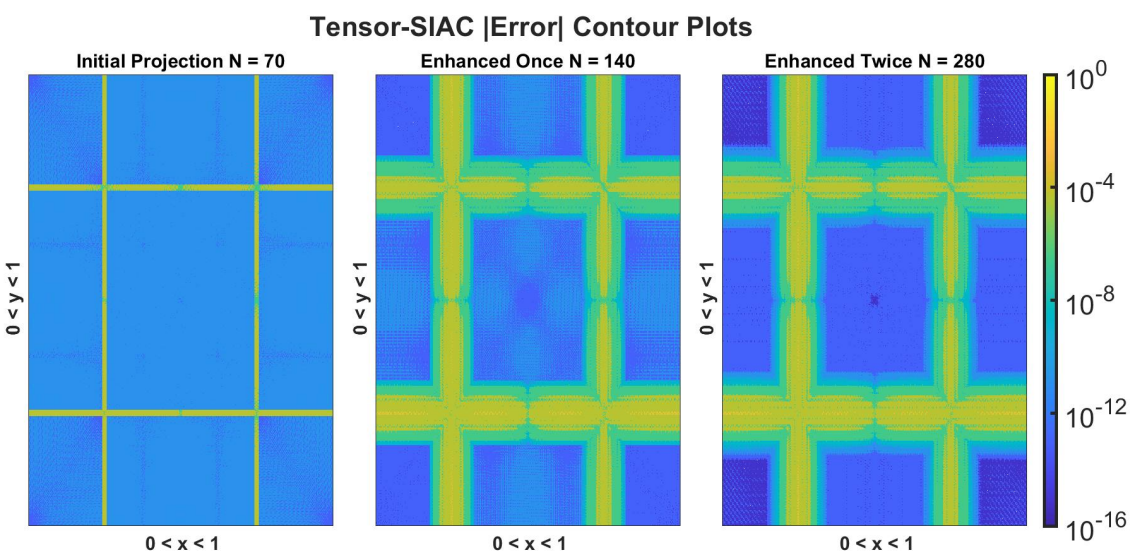


Figure 4.9 Contour plots depicting pointwise |Error| for the initial condition $u_0(x, y) = f_2(x)f_2(y)$ which contains a discontinuity in its first derivatives $\frac{\partial u_0}{\partial x}$ and $\frac{\partial u_0}{\partial y}$. 2D SIAC-MRA was performed over a series of three meshes for $p = 4$. Notice that owing to the jump discontinuity in the derivatives at $x, y = 1/4, 3/4$, the polluted region where filtering adds artificial smoothness grows with each enhancement. Away from these locations, we still have error improvement. A zoomed in view is given in Figure Figure 4.11.

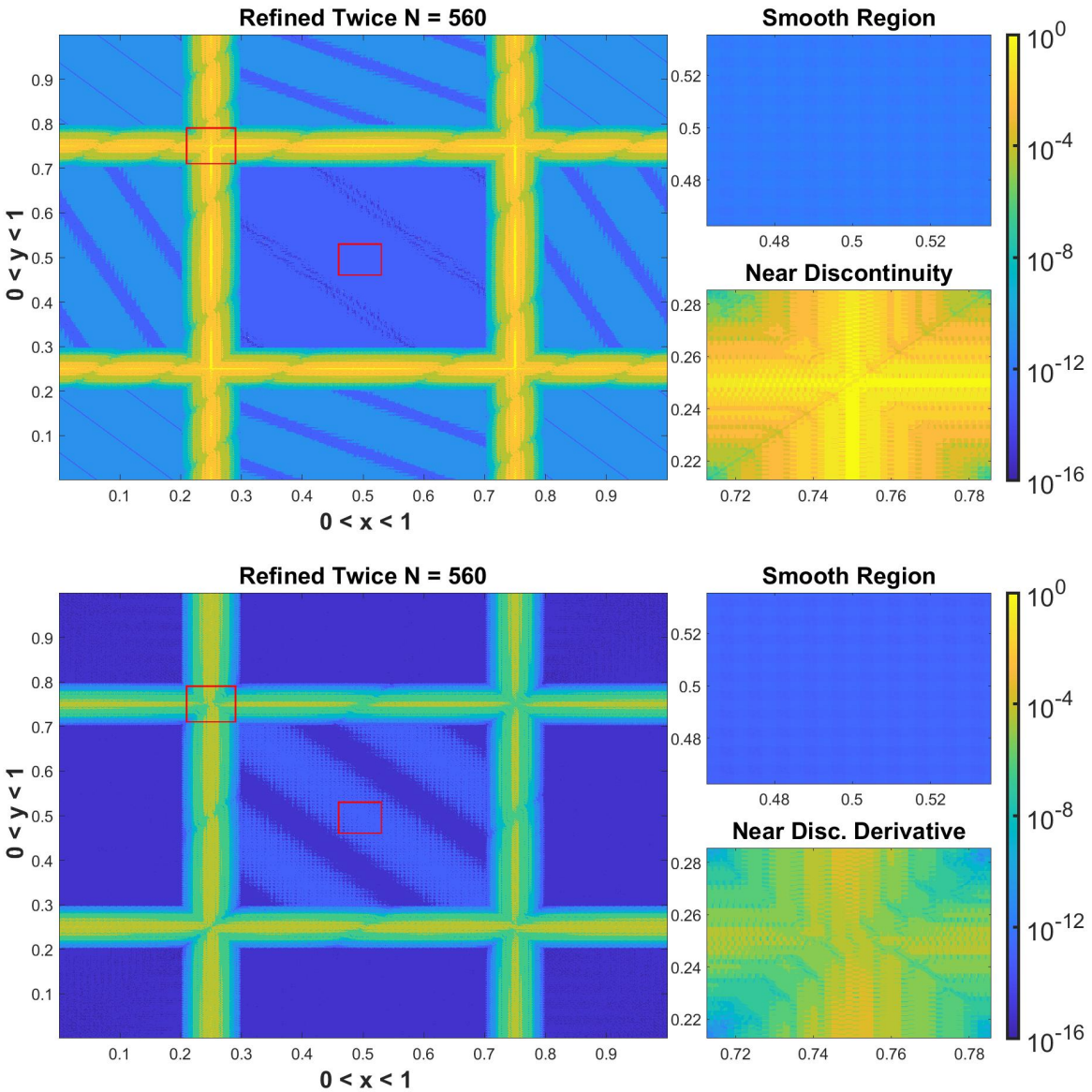


Figure 4.10 Zoomed in contour error plots for the LSIAC-MRA twice enhanced $p = 4$ approximation of a discontinuous function (Top) and a continuous function that is discontinuous in its $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$ derivatives (Bottom). Both approximations were initialized on a $140 \times 140$ mesh.

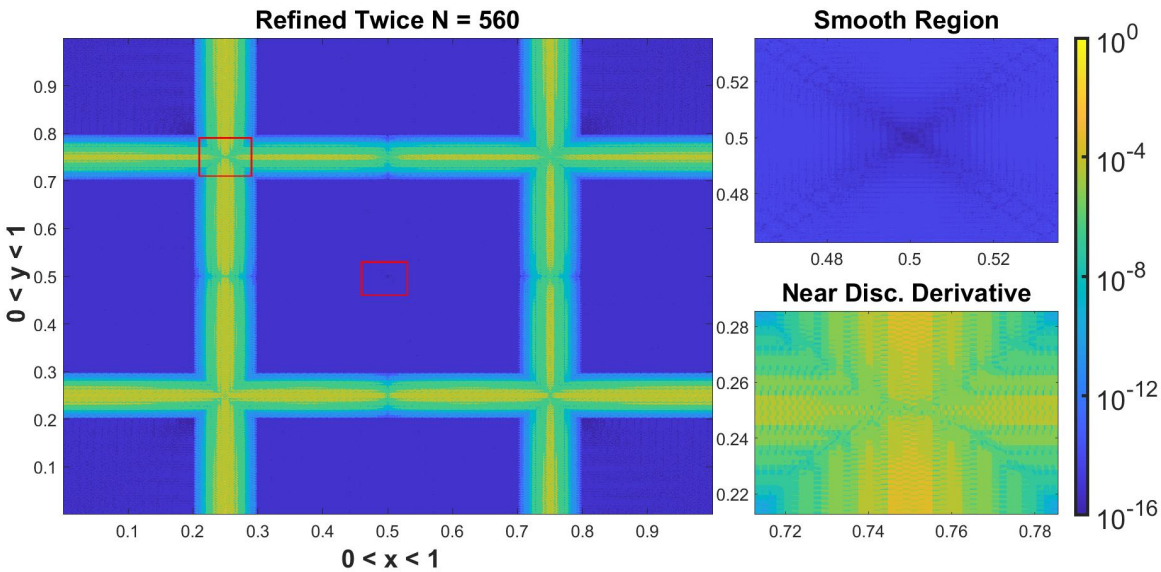


Figure 4.11 Zoomed in contour error plots for the SIAC-MRA twice enhanced $p = 4$ approximation of a continuous function that is discontinuous in its $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$ derivatives. The approximation was initialized on a $140 \times 140$ mesh.

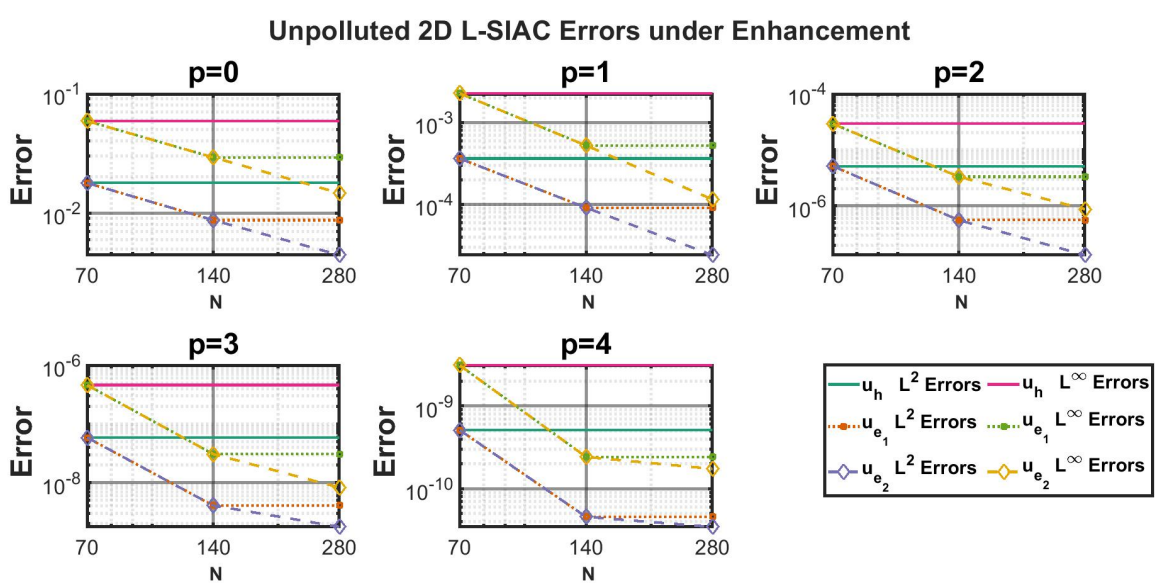


Figure 4.12 Log-Log plots of $N$ vs. Error for a continuous function that has discontinuous $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$ derivatives on a series of three meshes using **2D LSIAC-MRA**. Errors were calculated by excluding the polluted regions around the discontinuous derivatives. Notice that performance is better than for the discontinuous function.

Table 4.2 Table of $L^2$- and $L^\infty$-errors for an initial condition that has discontinuities along $x, y = 1/4, 3/4$, $u(x,y) = f_1(x)f_1(y)$. The **2D LSIAC-MRA** procedure was performed on a series of three meshes. Errors were calculated by excluding the polluted regions around the discontinuities. The LSIAC-MRA procedure reduces errors in these unpolluted regions.

| | **Projection Error** | | **Enhanced Once** | | **Enhanced Each Refinement** | |
|---|---|---|---|---|---|---|
| **N** | $L^2$**-Error** | $L^\infty$**-Error** | $L^2$**-Error** | $L^\infty$**-Error** | $L^2$**-Error** | $L^\infty$**-Error** |
| | | | $\mathbb{P}^0$ | | | |
| **70** | $1.12e-01$ | $3.52e-01$ | $1.12e-01$ | $3.52e-01$ | $1.12e-01$ | $3.52e-01$ |
| **140** | $1.12e-01$ | $3.52e-01$ | $5.58e-02$ | $1.92e-01$ | $5.58e-02$ | $1.92e-01$ |
| **280** | $1.12e-01$ | $3.52e-01$ | $5.58e-02$ | $1.92e-01$ | $3.05e-02$ | $1.07e-01$ |
| | | | $\mathbb{P}^1$ | | | |
| **70** | $4.00e-03$ | $2.55e-02$ | $4.00e-03$ | $2.55e-02$ | $4.00e-03$ | $2.55e-02$ |
| **140** | $4.00e-03$ | $2.55e-02$ | $9.08e-04$ | $5.71e-03$ | $9.08e-04$ | $5.71e-03$ |
| **280** | $4.00e-03$ | $2.55e-02$ | $9.08e-04$ | $5.71e-03$ | $3.55e-04$ | $1.35e-03$ |
| | | | $\mathbb{P}^2$ | | | |
| **70** | $1.15e-04$ | $6.95e-04$ | $1.15e-04$ | $6.95e-04$ | $1.15e-04$ | $6.95e-04$ |
| **140** | $1.15e-04$ | $6.95e-04$ | $1.94e-05$ | $1.04e-04$ | $1.94e-05$ | $1.04e-04$ |
| **280** | $1.15e-04$ | $6.95e-04$ | $1.94e-05$ | $1.04e-04$ | $1.38e-05$ | $6.15e-05$ |
| | | | $\mathbb{P}^3$ | | | |
| **70** | $2.47e-06$ | $1.76e-05$ | $2.47e-06$ | $1.76e-05$ | $2.47e-06$ | $1.76e-05$ |
| **140** | $2.47e-06$ | $1.76e-05$ | $1.09e-06$ | $4.63e-06$ | $1.09e-06$ | $4.63e-06$ |
| **280** | $2.47e-06$ | $1.76e-05$ | $1.09e-06$ | $4.63e-06$ | $9.30e-07$ | $3.86e-06$ |
| | | | $\mathbb{P}^4$ | | | |
| **70** | $4.57e-08$ | $3.00e-07$ | $4.57e-08$ | $3.00e-07$ | $4.57e-08$ | $3.00e-07$ |
| **140** | $4.57e-08$ | $3.00e-07$ | $8.91e-08$ | $3.55e-07$ | $8.91e-08$ | $3.55e-07$ |
| **280** | $4.57e-08$ | $3.00e-07$ | $8.91e-08$ | $3.55e-07$ | $7.17e-08$ | $3.49e-07$ |

Table 4.3 Shown are the $L^2$- and $L^\infty$-errors for a function $u_0(x,y) = f_2(x)f_2(y)$ that has discontinuous derivatives $\frac{\partial u_0}{\partial x}$ and $\frac{\partial u_0}{\partial y}$. The **2D LSIAC-MRA** procedure was performed on a series of three meshes. Errors were calculated by excluding the polluted regions around the discontinuous derivatives. We observe that the LSIAC-MRA is effective for error reduction in unpolluted regions.

| | **Projection Error** | | **Enhanced Once** | | **Enhanced Each Refinement** | |
|---|---|---|---|---|---|---|
| **N** | $L^2$**-Error** | $L^\infty$**-Error** | $L^2$**-Error** | $L^\infty$**-Error** | $L^2$**-Error** | $L^\infty$**-Error** |
| | | | $\mathbb{P}^0$ | | | |
| **70** | $1.78e-02$ | $5.87e-02$ | $1.78e-02$ | $5.87e-02$ | $1.78e-02$ | $5.87e-02$ |
| **140** | $1.78e-02$ | $5.87e-02$ | $8.71e-03$ | $2.92e-02$ | $8.71e-03$ | $2.92e-02$ |
| **280** | $1.78e-02$ | $5.87e-02$ | $8.71e-03$ | $2.92e-02$ | $4.44e-03$ | $1.47e-02$ |
| | | | $\mathbb{P}^1$ | | | |
| **70** | $3.63e-04$ | $2.27e-03$ | $3.63e-04$ | $2.27e-03$ | $3.63e-04$ | $2.27e-03$ |
| **140** | $3.63e-04$ | $2.27e-03$ | $9.10e-05$ | $5.23e-04$ | $9.10e-05$ | $5.23e-04$ |
| **280** | $3.63e-04$ | $2.27e-03$ | $9.10e-05$ | $5.23e-04$ | $2.40e-05$ | $1.15e-04$ |
| | | | $\mathbb{P}^2$ | | | |
| **70** | $5.10e-06$ | $2.89e-05$ | $5.10e-06$ | $2.89e-05$ | $5.10e-06$ | $2.89e-05$ |
| **140** | $5.10e-06$ | $2.89e-05$ | $5.59e-07$ | $3.30e-06$ | $5.59e-07$ | $3.30e-06$ |
| **280** | $5.10e-06$ | $2.89e-05$ | $5.59e-07$ | $3.30e-06$ | $1.34e-07$ | $8.75e-07$ |
| | | | $\mathbb{P}^3$ | | | |
| **70** | $5.86e-08$ | $4.59e-07$ | $5.86e-08$ | $4.59e-07$ | $5.86e-08$ | $4.59e-07$ |
| **140** | $5.86e-08$ | $4.59e-07$ | $4.09e-09$ | $3.07e-08$ | $4.09e-09$ | $3.07e-08$ |
| **280** | $5.86e-08$ | $4.59e-07$ | $4.09e-09$ | $3.07e-08$ | $1.77e-09$ | $8.27e-09$ |
| | | | $\mathbb{P}^4$ | | | |
| **70** | $5.12e-10$ | $3.12e-09$ | $5.12e-10$ | $3.12e-09$ | $5.12e-10$ | $3.12e-09$ |
| **140** | $5.12e-10$ | $3.12e-09$ | $4.55e-11$ | $2.42e-10$ | $4.55e-11$ | $2.42e-10$ |
| **280** | $5.12e-10$ | $3.12e-09$ | $4.55e-11$ | $2.42e-10$ | $3.43e-11$ | $1.73e-10$ |

Table 4.4 Table of $L^2$- and $L^\infty$-errors for an initial condition that has discontinuities along $x, y = 1/4, 3/4$, $u(x,y) = f_1(x)f_1(y)$. The **2D SIAC-MRA** procedure was performed on a series of three meshes. Errors were calculated by excluding the polluted regions around the discontinuities. The SIAC-MRA procedure reduces errors in unpolluted regions and does so to a greater extent than LSIAC-MRA.

| | **Projection Error** | | **Enhanced Once** | | **Enhanced Each Refinement** | |
|---|---|---|---|---|---|---|
| **N** | $L^2$**-Error** | $L^\infty$**-Error** | $L^2$**-Error** | $L^\infty$**-Error** | $L^2$**-Error** | $L^\infty$**-Error** |
| | | | $\mathbb{P}^0$ | | | |
| **70** | $1.12e-01$ | $3.52e-01$ | $1.12e-01$ | $3.52e-01$ | $1.12e-01$ | $3.52e-01$ |
| **140** | $1.12e-01$ | $3.52e-01$ | $5.55e-02$ | $1.86e-01$ | $5.55e-02$ | $1.86e-01$ |
| **280** | $1.12e-01$ | $3.52e-01$ | $5.55e-02$ | $1.86e-01$ | $2.98e-02$ | $9.99e-02$ |
| | | | $\mathbb{P}^1$ | | | |
| **70** | $4.00e-03$ | $2.55e-02$ | $4.00e-03$ | $2.55e-02$ | $4.00e-03$ | $2.55e-02$ |
| **140** | $4.00e-03$ | $2.55e-02$ | $8.62e-04$ | $5.90e-03$ | $8.62e-04$ | $5.90e-03$ |
| **280** | $4.00e-03$ | $2.55e-02$ | $8.62e-04$ | $5.90e-03$ | $2.30e-04$ | $1.33e-03$ |
| | | | $\mathbb{P}^2$ | | | |
| **70** | $1.15e-04$ | $6.95e-04$ | $1.15e-04$ | $6.95e-04$ | $1.15e-04$ | $6.95e-04$ |
| **140** | $1.15e-04$ | $6.95e-04$ | $1.22e-05$ | $8.04e-05$ | $1.22e-05$ | $8.04e-05$ |
| **280** | $1.15e-04$ | $6.95e-04$ | $1.22e-05$ | $8.04e-05$ | $2.63e-06$ | $1.73e-05$ |
| | | | $\mathbb{P}^3$ | | | |
| **70** | $2.47e-06$ | $1.76e-05$ | $2.47e-06$ | $1.76e-05$ | $2.47e-06$ | $1.76e-05$ |
| **140** | $2.47e-06$ | $1.76e-05$ | $1.27e-07$ | $9.66e-07$ | $1.27e-07$ | $9.66e-07$ |
| **280** | $2.47e-06$ | $1.76e-05$ | $1.27e-07$ | $9.66e-07$ | $2.70e-08$ | $1.48e-07$ |
| | | | $\mathbb{P}^4$ | | | |
| **70** | $4.57e-08$ | $3.00e-07$ | $4.57e-08$ | $3.00e-07$ | $4.57e-08$ | $3.00e-07$ |
| **140** | $4.57e-08$ | $3.00e-07$ | $1.27e-09$ | $9.03e-09$ | $1.27e-09$ | $9.03e-09$ |
| **280** | $4.57e-08$ | $3.00e-07$ | $1.27e-09$ | $9.03e-09$ | $4.03e-10$ | $2.31e-09$ |

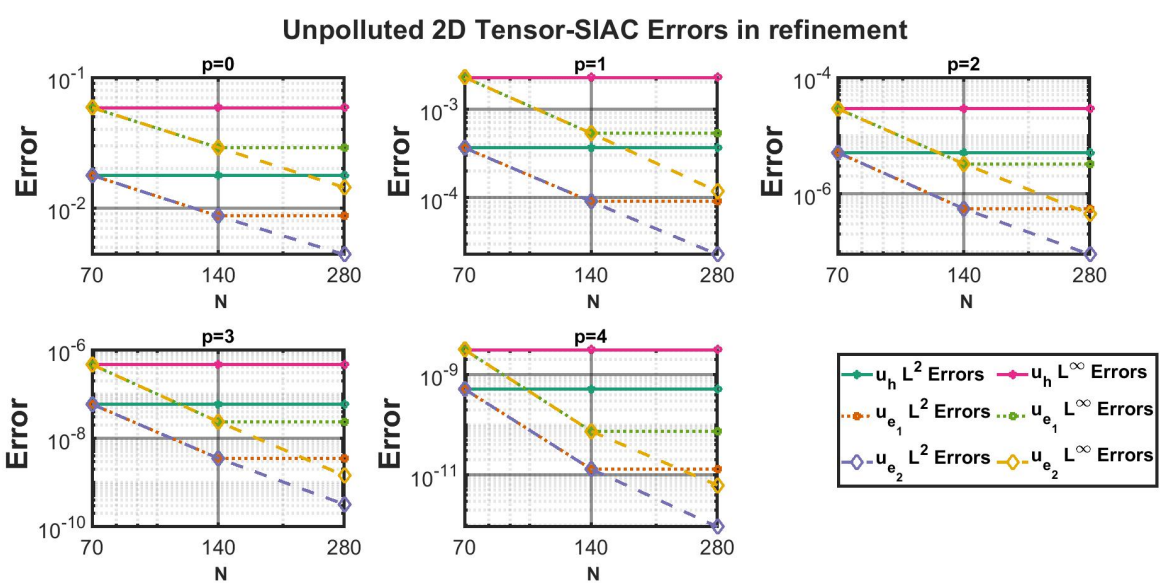


Figure 4.13 Log-Log plots of $N$ vs. Error for a continuous function that has discontinuous $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$ derivatives on a series of three meshes using **2D SIAC-MRA**. Errors were calculated by excluding the polluted regions around the discontinuous derivatives. Notice that performance is better than for the discontinuous function.

Table 4.5 Shown are the $L^2$- and $L^\infty$-errors for a function $u_0(x,y) = f_2(x)f_2(y)$ that has discontinuous derivatives $\frac{\partial u_0}{\partial x}$ and $\frac{\partial u_0}{\partial y}$. The **2D SIAC-MRA** procedure was performed on a series of three meshes. Errors were calculated by excluding the polluted regions around the discontinuous derivatives. The SIAC-MRA procedure reduces errors in unpolluted regions and does so to a greater extent than LSIAC-MRA.

| | **Projection Error** | | **Enhanced Once** | | **Enhanced Each Refinement** | |
|---|---|---|---|---|---|---|
| **N** | $L^2$**-Error** | $L^\infty$**-Error** | $L^2$**-Error** | $L^\infty$**-Error** | $L^2$**-Error** | $L^\infty$**-Error** |
| | | | $\mathbb{P}^{\mathbf{0}}$ | | | |
| **70** | $1.78e-02$ | $5.87e-02$ | $1.78e-02$ | $5.87e-02$ | $1.78e-02$ | $5.87e-02$ |
| **140** | $1.78e-02$ | $5.87e-02$ | $8.70e-03$ | $2.90e-02$ | $8.70e-03$ | $2.90e-02$ |
| **280** | $1.78e-02$ | $5.87e-02$ | $8.70e-03$ | $2.90e-02$ | $4.42e-03$ | $1.44e-02$ |
| | | | $\mathbb{P}^{\mathbf{1}}$ | | | |
| **70** | $3.63e-04$ | $2.27e-03$ | $3.63e-04$ | $2.27e-03$ | $3.63e-04$ | $2.27e-03$ |
| **140** | $3.63e-04$ | $2.27e-03$ | $9.06e-05$ | $5.33e-04$ | $9.06e-05$ | $5.33e-04$ |
| **280** | $3.63e-04$ | $2.27e-03$ | $9.06e-05$ | $5.33e-04$ | $2.28e-05$ | $1.18e-04$ |
| | | | $\mathbb{P}^{\mathbf{2}}$ | | | |
| **70** | $5.10e-06$ | $2.89e-05$ | $5.10e-06$ | $2.89e-05$ | $5.10e-06$ | $2.89e-05$ |
| **140** | $5.10e-06$ | $2.89e-05$ | $5.48e-07$ | $3.26e-06$ | $5.48e-07$ | $3.26e-06$ |
| **280** | $5.10e-06$ | $2.89e-05$ | $5.48e-07$ | $3.26e-06$ | $8.93e-08$ | $4.47e-07$ |
| | | | $\mathbb{P}^{\mathbf{3}}$ | | | |
| **70** | $5.86e-08$ | $4.59e-07$ | $5.86e-08$ | $4.59e-07$ | $5.86e-08$ | $4.59e-07$ |
| **140** | $5.86e-08$ | $4.59e-07$ | $3.53e-09$ | $2.37e-08$ | $3.53e-09$ | $2.37e-08$ |
| **280** | $5.86e-08$ | $4.59e-07$ | $3.53e-09$ | $2.37e-08$ | $3.21e-10$ | $1.43e-09$ |
| | | | $\mathbb{P}^{\mathbf{4}}$ | | | |
| **70** | $5.12e-10$ | $3.12e-09$ | $5.12e-10$ | $3.12e-09$ | $5.12e-10$ | $3.12e-09$ |
| **140** | $5.12e-10$ | $3.12e-09$ | $1.31e-11$ | $7.46e-11$ | $1.31e-11$ | $7.46e-11$ |
| **280** | $5.12e-10$ | $3.12e-09$ | $1.31e-11$ | $7.46e-11$ | $9.28e-13$ | $6.23e-12$ |

### 4.8 Three-dimensional Test Problems

We now investigate the application of (L)SIAC-MRA for three-dimensional functions. We have tested the procedure on other functions as well, but present results only for the function $u_0(x, y, z) = \sin(2\pi x) + \sin(2\pi y) + \sin(2\pi z)$ on the domain $\Omega = \{(x, y, z) \in [0, 1]^3\}$.

Beginning with a projection of this function onto a piecewise orthonormal Legendre basis on a $15 \times 15 \times 15$ uniform hexahedral mesh, the approximation is then enhanced using the (L)SIAC-MRA procedure. This procedure is applied two times, resulting in a final approximation on a $60 \times 60 \times 60$ mesh. The $L^2-$ and $L^\infty-$errors of those approximations with and without the filtering enhancement are given in Table 4.6 and Table 4.7. Notice that we have included the results for a piecewise constant approximation, $p = 0$. The theory of (L)SIAC does not generally extend to piecewise constants. However, it is evident from the results in these tables that the (L)SIAC-MRA procedure is effective for piecewise constant approximations. For piecewise linear approximations, $p = 1$, we can see that it is less effective, but still provides an error reduction. Similar to the two-dimensional case, we speculate that for higher degree polynomial approximations it is necessary to start with a higher resolution for the (L)SIAC-MRA procedure to be more effective. The Log-Log plots for the results of the three-dimensional (L)SIAC-MRA procedure are given in Figure 4.14 and Figure 4.15. Notice that the tensor-product formulation slightly outperforms the LSIAC-MRA formulation.

Table 4.6 Table of $L^2$- and $L^\infty$-errors for initial condition $u(x, y, z) = \sin(2\pi x) + \sin(2\pi y) + \sin(2\pi z)$ on a series of three meshes using **3D LSIAC-MRA**. We observe error reduction with each enhancement.

| | **Projection Error** | | **Enhanced Once** | | **Enhanced Each Refinement** | |
|---|---|---|---|---|---|---|
| **N** | **$L^2$-Error** | **$L^\infty$-Error** | **$L^2$-Error** | **$L^\infty$-Error** | **$L^2$-Error** | **$L^\infty$-Error** |
| | | | $\mathbb{P}^0$ | | | |
| **15** | $1.48e-01$ | $6.13e-01$ | $1.48e-01$ | $6.13e-01$ | $1.48e-01$ | $6.13e-01$ |
| **30** | $1.48e-01$ | $6.13e-01$ | $7.61e-02$ | $3.08e-01$ | $7.61e-02$ | $3.08e-01$ |
| **60** | $1.48e-01$ | $6.13e-01$ | $7.61e-02$ | $3.08e-01$ | $4.43e-02$ | $1.59e-01$ |
| | | | $\mathbb{P}^1$ | | | |
| **15** | $7.99e-03$ | $4.14e-02$ | $7.99e-03$ | $4.14e-02$ | $7.99e-03$ | $4.14e-02$ |
| **30** | $7.99e-03$ | $4.14e-02$ | $2.13e-03$ | $1.53e-02$ | $2.13e-03$ | $1.53e-02$ |
| **60** | $7.99e-03$ | $4.14e-02$ | $2.13e-03$ | $1.53e-02$ | $7.70e-04$ | $7.53e-03$ |

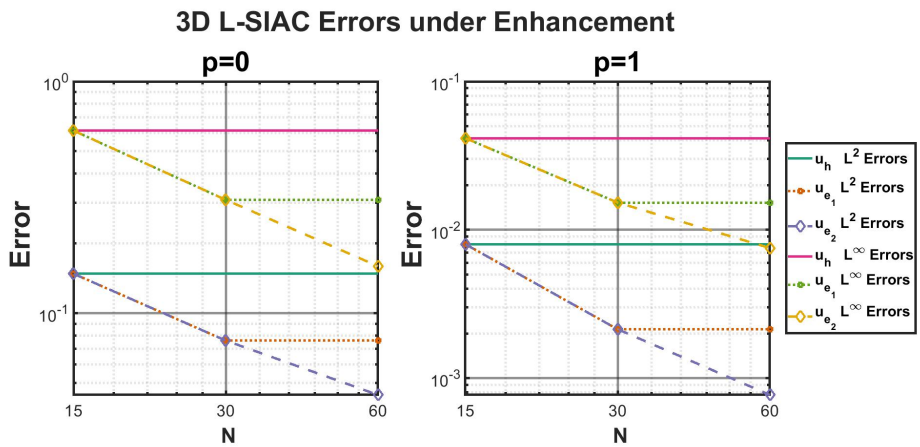


Figure 4.14 Log-Log plots of $N$ vs. Error for the function $u_0(x, y, z) = \sin(2\pi x) + \sin(2\pi y) + \sin(2\pi z)$ on a series of three meshes using **3D LSIAC-MRA**. Notice that procedure improves the approximation and provides error reduction.

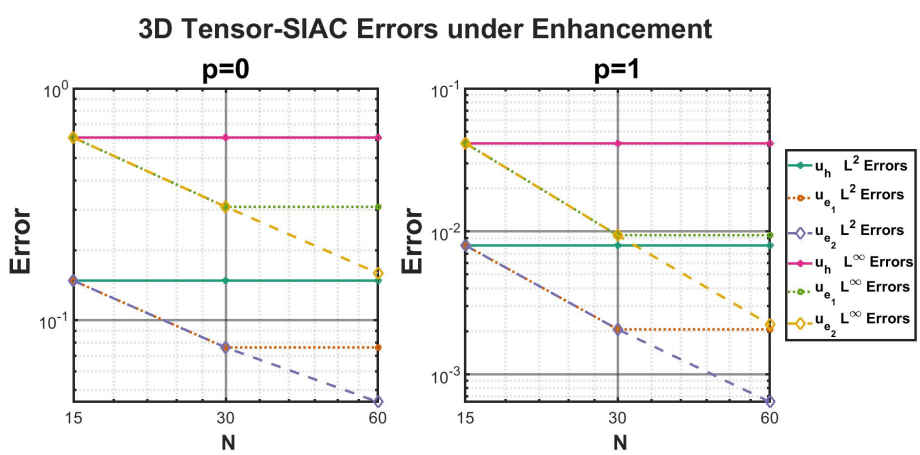


Figure 4.15 Log-Log plots of $N$ vs. Error for the function $u_0(x, y, z) = \sin(2\pi x) + \sin(2\pi y) + \sin(2\pi z)$ on a series of three meshes using **3D SIAC-MRA**. Notice that procedure improves the approximation and provides error reduction.

Table 4.7 Table of $L^2$- and $L^\infty$-errors for initial condition $u(x, y, z) = \sin(2\pi x) + \sin(2\pi y) + \sin(2\pi z)$ on a series of three meshes using **3D SIAC-MRA**. We observe error reduction with each enhancement.

| | **Projection Error** | | **Enhanced Once** | | **Enhanced Each Refinement** | |
|---|---|---|---|---|---|---|
| **N** | **$L^2$-Error** | **$L^\infty$-Error** | **$L^2$-Error** | **$L^\infty$-Error** | **$L^2$-Error** | **$L^\infty$-Error** |
| | | | $\mathbb{P}^0$ | | | |
| **15** | $1.48e-01$ | $6.13e-01$ | $1.48e-01$ | $6.13e-01$ | $1.48e-01$ | $6.13e-01$ |
| **30** | $1.48e-01$ | $6.13e-01$ | $7.61e-02$ | $3.08e-01$ | $7.61e-02$ | $3.08e-01$ |
| **60** | $1.48e-01$ | $6.13e-01$ | $7.61e-02$ | $3.08e-01$ | $4.43e-02$ | $1.59e-01$ |
| | | | $\mathbb{P}^1$ | | | |
| **15** | $7.99e-03$ | $4.14e-02$ | $7.99e-03$ | $4.14e-02$ | $7.99e-03$ | $4.14e-02$ |
| **30** | $7.99e-03$ | $4.14e-02$ | $2.07e-03$ | $9.39e-03$ | $2.07e-03$ | $9.39e-03$ |
| **60** | $7.99e-03$ | $4.14e-02$ | $2.07e-03$ | $9.39e-03$ | $6.42e-04$ | $2.23e-03$ |

## 4.9 Chapter Summary

In this chapter, we have introduced an enhancement scheme for multidimensional applications, LSIAC-MRA, and generalized to higher dimensions the SIAC-MRA established in [23]. This scheme utilizes the (L)SIAC filters which post-processes multidimensional data. In conjunction with projection methods, this allows for approximating the difference coefficients in the multiwavelet representation and allows for error reduction with mesh refinement. We have provided the underlying operational framework and demonstrated that (L)SIAC-MRA is effective for two- and three-dimensional applications. Furthermore, (L)SIAC-MRA is shown to be effective for multidimensional piecewise constant data as well. Next, in Chapter 5, we extend these techniques to more complicated mesh geometries and apply them in adaptive schemes.

## 4.10 Co-Author Contributions

Jennifer Ryan provided conceptualization, editing, and supervision for this project.

CHAPTER 5

# LSIAC-MRA ON NONUNIFORM MESHES AND APPLICATIONS TO MESH ADAPTIVE DG

Portions of this chapter are based on a paper in preparation "LSIAC-MRA on Nonuniform Meshes and applications to Mesh Adaptivity" by Matthew J. Picklo and Jennifer K. Ryan.

## 5.1 Chapter Introduction

The ability to utilize multiple levels of resolution, including subgrid scales, is useful in a variety of applications such as trouble cell detection [81, 83], multiresolution numerical methods [47, 54, 56], and error estimation for $h/p$ adaptivity [61, 62, 82]. Through application of carefully selected SIAC kernels, the authors of [23, 98] were able to demonstrate error reduction under multiple mesh transitions for a nested sequence of uniform hierarchical meshes. The error reduction was obtained for quadrilateral Cartesian meshes through the use of a SIAC kernel post-processor prior to projection onto a refined mesh. Furthermore, the authors constructed explicit transition matrices for performing the procedure efficiently. However, the approach only considered uniform meshes. Extending this procedure to nonuniform meshes and adapted meshes presents many challenges because repeating patterns in the mesh and kernel breaks do not exist. This unfortunately limits the use of the procedure for mesh adaptivity purposes. In this work, we return to the fundamental filtering and projection procedures and extend these techniques to triangular and nonuniform meshes. Furthermore, we demonstrate the ability of (L)SIAC-MRA to construct accurate fine mesh representations of coarse mesh data that can be applied within the context of adaptive Discontinuous Galerkin (DG) methods.

The (L)SIAC-MRA procedure views the application of SIAC filtering and projection within a multiresolution framework, where fine mesh data is represented as a coarse mesh projection plus additional multiwavelet details. In (L)SIAC-MRA, coarse data is convolved with a carefully chosen SIAC kernel and projected onto a refined mesh. This enhanced reconstruction represents a fine mesh approximation which is equivalent to a coarse mesh approximation with additional multiwavelet details:

$$\tilde{u}_h^{(n+1)}(\mathbf{x}) = \tilde{u}_h^{(n)}(\mathbf{x}) + \sum_{\tau \in \mathcal{T}^{(n)}} \sum_{k \in \mathcal{I}} \tilde{d}^n_{k,\tau} \psi^n_{\tau,k}(\mathbf{x}). \tag{5.1}$$

Here the $n+1$ superscript is used to indicate that the approximation is defined on a finer mesh than the coarse level-$n$ initial data. The $\tilde{d}$ multiwavelet coefficients produced by this procedure are distinct from the $d$ multiwavelet coefficients produced by the standard MRA as we do not refer back to the exact solution $u$ in their computation (compare with Section 2.3).

This representation is typically much more accurate than simply projecting the approximation on the coarse mesh alone. The procedure was originally developed for uniform meshes using tensor-product kernels in [23]. It was then later extended to higher dimensions via a Line SIAC filter in [98]. In both cases, error reduction under iterated refinement was observed numerically. The latter paper demonstrated that this procedure is also useful for finite volume type approximations. In this chapter, we introduce a refinement procedure that enables the same error reduction for unstructured meshes and applies that capability for refinement indication in adaptive DG schemes.

We note that on nonuniform meshes, LSIAC filtering can be augmented by an adaptive kernel scaling, see [19]. The adaptively scaled kernel uses a scaling choice that is based on the local average resolution of the mesh about the filtering location. Previously, the maximum mesh edge length has been chosen as the kernel scaling for nonuniform meshes. Unfortunately, when the mesh resolution varies significantly over the domain, exact evaluation of the convolution is particularly costly. By choosing the maximum edge length, the kernel support is fixed for the entire domain regardless of the local resolution. As a consequence, in high resolution regions, many separate quadrature sums are required to ensure exactness. This contrasts with the low resolution regions which would typically require fewer quadrature sums because the kernel scaling is closer to the local characteristic length scale of the mesh. In the uniform case, the choice of kernel scaling directly influences the amount of dissipation applied by the filter, and so by choosing an adaptively scaled kernel, the magnitude of attenuation of data is also localized, preventing excessive dissipation in high resolution regions. Using the adaptive scaling, the authors of [19] showed not only improved computational efficiency but also greater error reductions for unstructured meshes where the mesh resolution varied significantly. We will consider this application of this techniques later in the chapter.

One direct application of (L)SIAC-MRA is mesh and by extension approximation refinement. We show how incorporating this enhancement procedure into a multiwavelet-based framework can be useful for mesh adaptivity. Adaptive multiresolution schemes have previously been considered for finite volume methods [46, 47, 49], as well as in the DG-context [54–59, 99]. These approaches have mainly concentrated on utilizing the multiresolution analysis framework for computational efficiency, specifically the truncation of insignificant detail coefficients, which can be thought of as extraneous degrees of freedom (DOFs). The present approach is motivated by ideas similar to [61, 62]. Our (L)SIAC-MRA reconstruction is used to generate additional detail coefficients, which effectively defines a finer mesh approximation. These modified detail coefficients are then used to indicate regions where adaptivity is necessary. This work differs from those previously mentioned in that the SIAC enhancement methodologies are employed to increase resolution rather than projections of least-squares polynomials.

This chapter is organized as follows. We begin with a review of the enhancement procedure in one and two dimensions in Section 5.2. We then demonstrate the effectiveness of LSIAC-MRA for error reduction when the enhancement procedure applied to unstructured meshes in Section 5.3. As part of this, the role of an adaptive kernel scaling is investigated. Lastly, a demonstration of the usefulness of the enhancement technique in the context of mesh refinement schemes for DG approximations is provided in Section 5.4.

## 5.2 Background

In this section we describe the notation used, and the enhancement procedure for the one- and two-dimensional (L)SIAC-MRA as originally formulated in [98, 100]. We first detail the enhancement procedure, formulating the SIAC filters and projection components used in the enhancement before briefly discussing the issue of adaptive scaling.

### 5.2.1 Data format

We first note that, for the purposes of this work, it is assumed that the data $u_h^{(n)}(\mathbf{x})$ is defined on a mesh $\mathcal{T}^{(n)}$ composed of elements $\tau$ that partition some domain $\Omega$. We further assume that on each element, $\tau$, the data can be expressed in terms of some polynomial basis $\{\phi_{k,\tau}\}_{k\in\mathcal{I}}$ with $\mathcal{I}$ being an index set for the basis. Hence, we write the approximation on an element $\tau$ as:

$$u_h^{(n)}\Big|_\tau(\mathbf{x}) = \sum_{k\in\mathcal{I}} \hat{u}_\tau^k \phi_{k,\tau}(\mathbf{x}). \tag{5.2}$$

Throughout this chapter the initializations considered to obtain the $\hat{u}$ coefficients are determined by $L^2$-projection, that is by requiring that

$$\langle u, \phi_{m,\tau}\rangle_\tau = \langle u_h^{(n)}, \phi_{m,\tau}\rangle_\tau, \tag{5.3}$$

for each $m \in \mathcal{I}$ and $\tau \in \mathcal{T}^{(n)}$.

### 5.2.2 Enhancement procedure

To enhance the MRA procedure for the construction of a fine mesh approximation $\tilde{u}_h^{(n+1)}$, the coarse mesh data, $u_h^{(n)}$, is filtered by convolving it with a specially chosen kernel function, $K_H$ (see Section 2.1), before being projecting onto a refined mesh. Mathematically, we write the enhanced fine mesh approximation as

$$\tilde{u}_h^{(n+1)} = \mathcal{P}^{n+1}(K_H \star u_h^{(n)}). \tag{5.4}$$

where $\mathcal{P}^{n+1}$ represents the projection operator projecting the filtered approximation onto the fine mesh approximation space. This allows for computing the multiwavelet detail coefficients from the enhanced fine mesh approximation as shown in [98, 100]. In the following, where it does not impact clarity, we will drop

the mesh level indicator $(n)$ in the superscripts for readability. In Section 5.4 we show how fine mesh data can be leveraged in adaptive DG schemes, but first we provide a description of the projection step of the enhancement procedure and the adaptive kernel scaling of [19].

### 5.2.3 $L^2$-projection

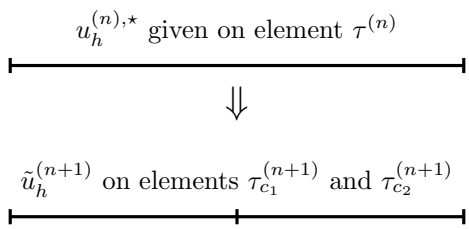


Figure 5.1 Illustration of enhancement

For the purpose of refinement in the enhancement procedure, the filtered approximation is projected onto a refined mesh. This procedure is similar to that in the initialization (5.3). Assuming $\tau^{(n)}$ is the coarse mesh element over which $u_h^{(n),\star}$ is defined and $\tau_c^{(n+1)}$ is a sub-element or "child" element of $\tau^{(n)}$ which exists on the finer mesh, the enhanced approximation on that sub-element is computed by requiring:

$$\langle u_h^{(n),\star}, \phi_{m,\tau_c^{(n+1)}} \rangle_{\tau_c^{(n+1)}} = \langle \tilde{u}_h^{(n+1)}, \phi_{m,\tau_c^{(n+1)}} \rangle_{\tau_c^{(n+1)}}, \tag{5.5}$$

for each $m \in \mathcal{I}$ (see Figure 5.1). Repeating this procedure for each $\tau^{(n)} \in \mathcal{T}^{(n)}$ and $\tau_c^{(n+1)} \subset \tau^{(n)}$ in $\mathcal{T}^{(n+1)}$ a polynomial representation is obtained on the refined mesh. Note that we drop the star superscript from the enhanced approximation as the approximation now belongs to the fine mesh approximation space over $\mathcal{T}^{(n+1)}$.

### 5.2.4 Adaptive kernel scaling

In considering meshes where elements sizes vary significantly we investigate the impact of the adaptive kernel scaling introduced by Jallepalli [19] on the enhancement procedure. The computation of this adaptive scaling consists of a pre-processing step assigning a characteristic length to each vertex in the mesh, and a post-processing step where based off the location of the filtering point in an element, a weighting of the characteristic lengths of the vertices in that element produces a characteristic length for that filtering point. We focus here on the two-dimensional case.

Let $v_i$ be the $i$-th vertex in the mesh, $\tau_j$ an element with vertex $v_i$, and $L_{\tau_j}$ the length of the edge of $\tau_j$ with maximum length. Then a characteristic length $H(v_i)$ is assigned to vertex $v_i$ via the expression:

$$H(v_i) = \frac{\sum_j L_{\tau_j} \cdot Area(\tau_j)}{\sum_j Area(\tau_j)}, \tag{5.6}$$

where $Area(\tau_j)$ is the area of element $\tau_j$. Computing these scaling for each vertex in the mesh constitutes the pre-processing step. Then, given a filtering point $x^\star$, the kernel scaling used at that location is computed via the relation

$$H(x^\star) = \sum_{\ell=0}^{N} \lambda_\ell H(v_\ell) \tag{5.7}$$

where $v_\ell$ are the vertices of the element containing $x^\star$, $N+1$ is the number of vertices of that element, and $\lambda_\ell$ are the barycentric coordinates of $x^\star$ in that element.

### 5.3 Application to Nonuniform Mesh Transition

Now that the components necessary for enhancement on an unstructured mesh have been defined, we can investigate the error-reducing capabilities of the LSIAC-MRA procedure when applied to a variety of mesh and initial condition pairings. Considered here are Delaunay triangulations and perturbed quadrilateral meshes. For the purposes of assessing the effects of adaptive scaling on the enhanced reconstruction procedure, the following two meshes (depicted in Figure 5.2) generated by using the two- and three-dimensional mesh generator application GMSH [101] are also considered. These meshes are denoted here by SVR100 and ISR2. The SVR100 mesh is smoothly varying where the ratio of the boundary to center element size is 100, while the ISR2 mesh possesses a more abrupt transition between boundary elements and interior elements with an element size ratio of 2. For ease of implementation, it is enforced in all cases that the meshes contain no hanging nodes, but it is stressed that this is not a requirement for the method itself.

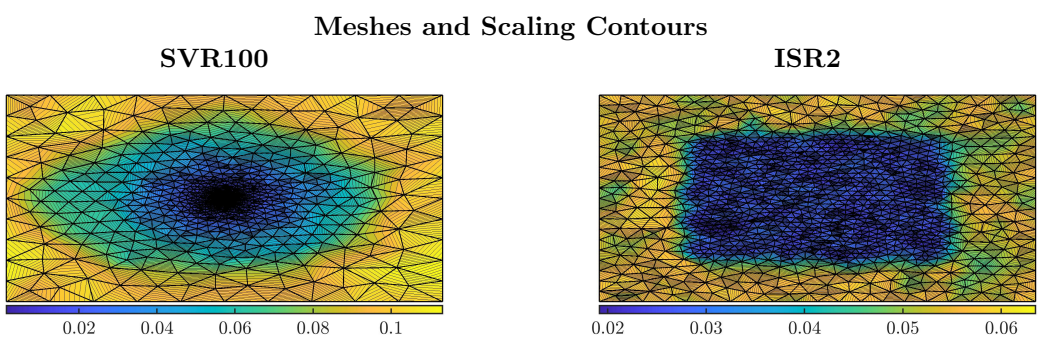


Figure 5.2 Examples of nonuniform meshes with varying ratios of boundary element to interior element sizes. The color map indicates the kernel scaling used at each point in the domain.

#### 5.3.1 Mesh Refinement

For refinement consider a uniform subdivision into $2^d$ elements. The subdivision is interpolatory in that nodes on the coarse mesh remain present after refinement. For every edge on the mesh, refine by inserting

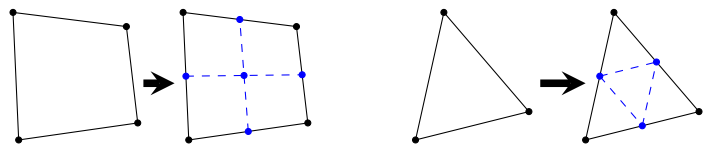

Figure 5.3 Refinement procedure for quadrilateral and triangular elements.

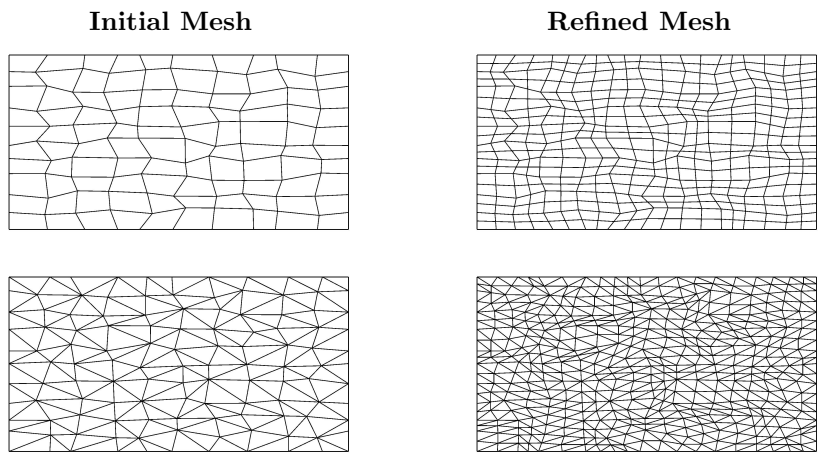


Figure 5.4 Depiction of nonuniform quadrilateral and triangular meshes (left) and the refined meshes resulting from a uniform subdivision (right).

a node at the midpoint of that edge (in the quadrilateral case a node is also inserted at the average of the midpoints of the edges of each element). These new nodes are then connected with the original nodes to form four new elements on the refined mesh. A depiction of this subdivision procedure is shown in Figure 5.3 and a comparison of initial and subdivided meshes in Figure 5.4.

### 5.3.2 Numerical tests

We consider the periodic domain $\Omega = [0,1]^2$ and meshes of $N$ elements and their refinements generated according to procedures described in Section 5.2. We note that the experiments in this section do not require the use of multiwavelet bases, which are typically expensive to compute over nonuniform meshes (see [56]). In the following we consider $\mathbb{Q}^p$ polynomial bases for the quadrilateral meshes, and $\mathbb{P}^p$ polynomial bases for the triangular meshes. We compute the $L^2$- and $L^\infty$-errors via Gauss-Legendre quadrature at $6^d$ quadrature nodes on each element relative to the finest mesh scale considered. These error expressions are given by

$$\|u - u_h\|_{L^2(\Omega)} = \sqrt{\int_\Omega |u - u_h|^2 \, d\Omega} \qquad \text{and} \qquad \|u - u_h\|_\infty = \sup_{x \in \Omega} |u - u_h|. \tag{5.8}$$

### 5.3.3 Impact of characteristic frequencies and lengths on error reduction

Below we investigate the error reduction capabilities of the enhanced reconstruction procedure for initial data of varying characteristic frequencies. As initial data, we consider the $L^2$-projections of the tensor-products of the following functions over the domain $[0,1]^2$, i.e. $u_h^{(n)} = \mathcal{P}^{(n)}u$, $u(x,y) = f_i(x)f_i(y)$ for:

- $f_1(x) = \sin(2\pi kx)$,
- $f_2(x) = \exp\left(\frac{-(x-1/2)^2}{2(kc)^2}\right)$,
- $f_3(x) = \frac{1}{2}\left[\tanh\left(\frac{x-7/20}{kw}\right) - \tanh\left(\frac{x-13/20}{kw}\right)\right]$.

The first function represents a sine wave of varying frequencies ($2\pi k$), the second a Gaussian function with varying standard deviation ($kc$), and the last a double hyperbolic tangent function with transition scaling ($kw$). Here $w$ and $c$ are chosen such that function $f_2$ and $f_3$ are less than $10^{-16}$ at the domain boundaries. The parameter $k$ is a scaling parameter that is varied to help understand the effects of initial data with different characteristic lengths on the LSIAC-MRA procedure. These test functions each represent a fundamental feature we might expect to encounter in physical simulations of fluid flow: sines representing a standard basis for the expression of waves, Gaussian representing vortices, and tanh functions possessing sharp gradients which are commonplace in PDEs with hyperbolic attributes. Furthermore, we note that these test functions are meant to be challenging functions for the enhancement procedure. Specifically, they either contain sharp gradients, or high-frequency oscillations relative to the scaling of the kernel. By increasing the frequency or the sines or the transition width of the sharp data, a better sense of the applicability of multi-element SIAC enhancements can be gained

Considering first the sine initial data, we observe in Figure 5.5 that errors are reduced for all polynomial degrees for $k = 1$. Doubling the frequency of the initial data, we obtain error reduction up through $p = 2$, with the $p = 3$ $L^2$-errors flat-lining for $k = 2$. Similarly doubling the frequency to $k = 4$ yields error reduction up through $p = 1$ and for $k = 8$ error reduction is obtained only for $p = 0$. Similar results are obtained for both perturbed quadrilateral meshes and Delaunay triangulations. Of course increasing the initial mesh resolution would increase the points per wavelength (ppw) in a single direction and thereby remedy this issue for higher orders, but that only obscures the issue which is that, based off experience in one dimension, SIAC filters damp higher frequency waves more, and to that end a tuning of kernel scalings should be considered when applying this technique and will be considered more in-depth in future work. Another interpretation here is that at least empirically the minimum resolution for improvement increases by a factor of two with each additional polynomial degree. The benefit of the procedure as an error reducing reconstruction on fine meshes must then be tempered by the expectation of high-frequency data.

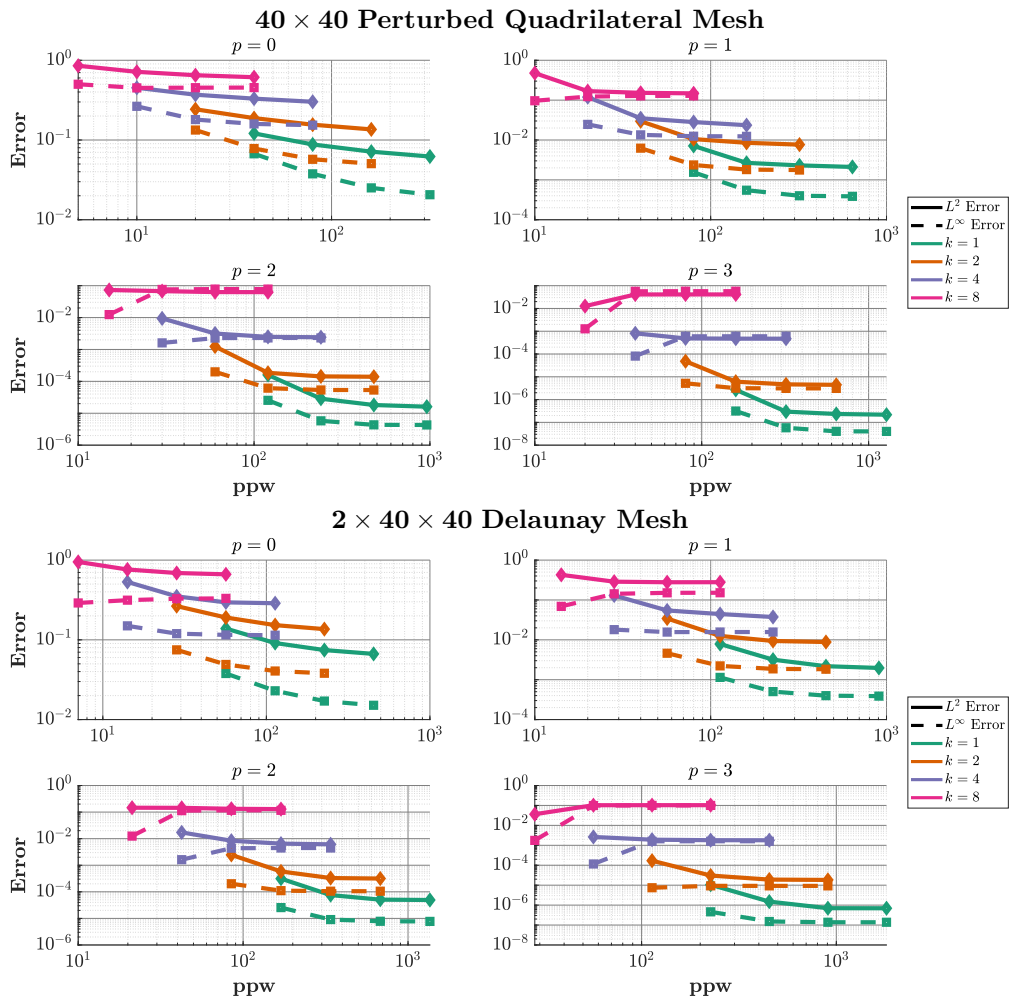


Figure 5.5 Log-Log Error plots for $u(x, y) = \sin(2\pi kx)\sin(2\pi ky)$ computed over the domain $[0, 1]^2$. On the x-axis is plotted the points per wavelength in a single coordinate direction $\sqrt{N}(p + 1) * 2^{\#enh}/k$.

Next, we consider the impact of characteristic lengths of the Gaussian and double Tanh initial conditions. Note here that as $k$ decreases, the Gaussian function becomes sharper and more closely resembles a point-source centered at $(1/2, 1/2)$, whereas the double Tanh function will more closely resemble the indicator function over the region $[7/20, 13/20]^2$. In Figure 5.6 we depict effects of repeated application of the LSIAC-MRA reconstruction where the initial data has varying characteristic scales governed by the choice of the scaling factor $k$. For the double Tanh function we observe that for small $p$ and large $k$ we see error reduction with enhancement, for $k < 1/2$ we no longer see improvement though, and for $p = 2$ no improvement is observed. This is because the sharp gradient is highly localized, and becomes sharper with decreasing $k$. A weighted average over larger and larger intervals as $p$ increases does not improve the ability of the method to capture this feature. Like in the sine case, adaptive scaling relative to the feature being captured may prove necessary.

The enhancement procedure performs better for the Gaussian initial data where we observe improvement for all the scalings depicted when $p = 0$. For $p = 1$ we observe improvement in the $L^2$-errors in all cases, but only for the first two scalings with respect to the $L^\infty$-error. When $p = 2$ we only observe improvement for the largest scaling. Similar to the hyperbolic tangent case though less pronounced, this issue would require high levels of resolution to resolve without changing the kernel formulation. In this sense, a kernel scaling implicitly informed by the underlying data seems necessary and will be a study of future work.

### 5.3.4 Adaptive versus constant scaling

We now consider the filter scaling for nonuniform meshes and how it effects the performance of LSIAC-MRA procedure. Specifically, we compare the traditional maximum edge length constant scaling versus the adaptive kernel scaling of Jallepalli [19]. As depicted in Figure 5.7 we observe that the adaptively scaled kernel performs the same or better on variable element size meshes. Of particular interest is the $p = 2$ and $p = 3$ cases for the smoothly varying meshes. In this case, improvement is observed for the adaptively scaled procedure, but not in the constantly scaled case. Though we do not pursue a computational performance study here, we note that the adaptively scaled procedure is significantly faster than the constant scaling approach. This speed up of the filtering procedure by using adaptive scalings has been investigated in [19].

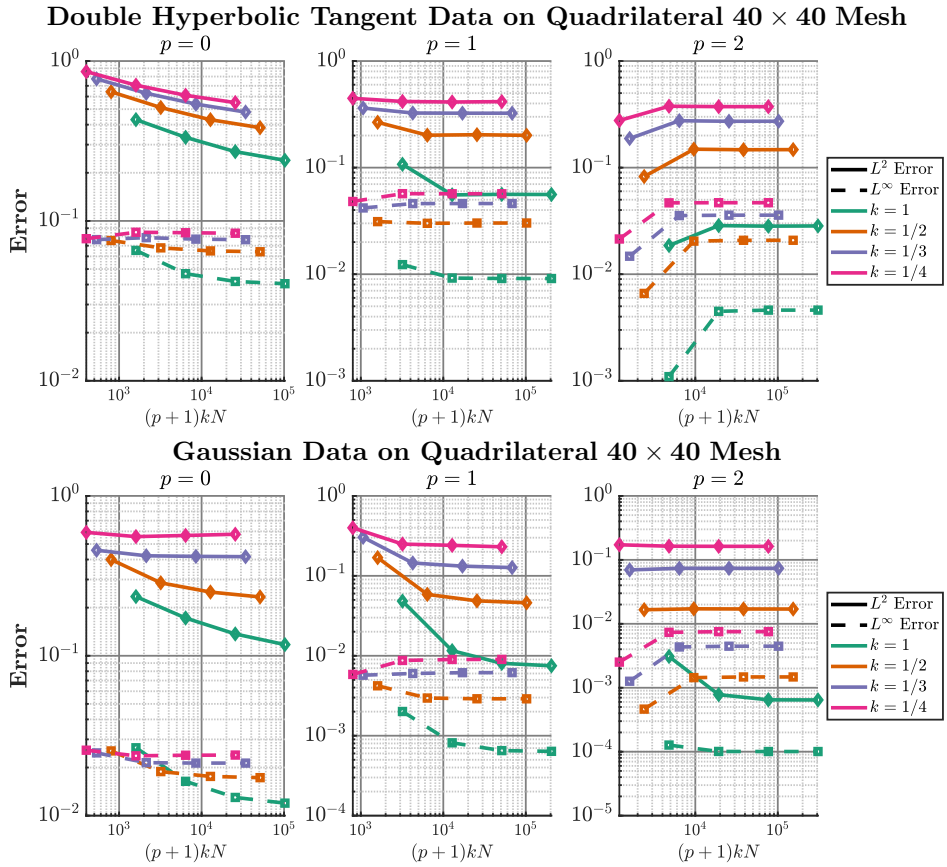


Figure 5.6 Errors under refinement of the double hyperbolic tangent and Gaussian functions for varying scaling factors $k$. Here $N$ is the number of elements in a single coordinate direction $40 * 2^{\#enh}$.

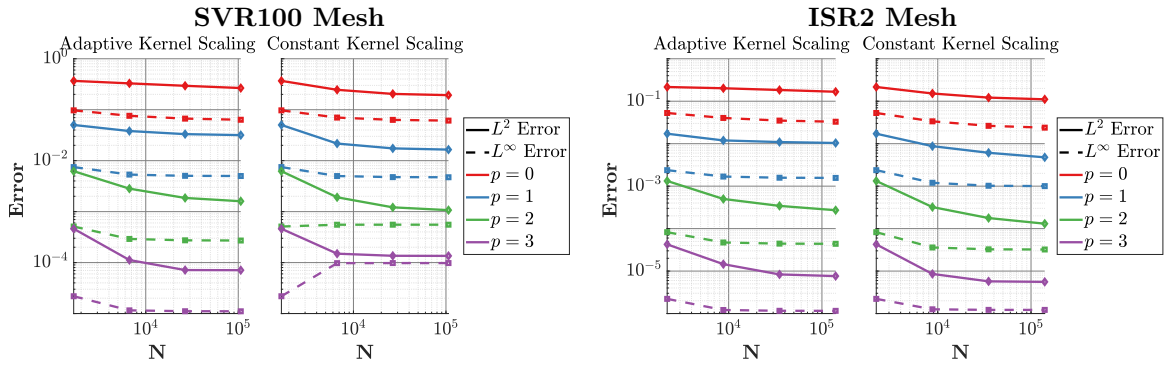


Figure 5.7 Comparison of errors when using adaptive kernel scalings versus constant max edge length kernel scalings for $u(x, y) = \sin(2\pi x)\sin(2\pi y)$ on the SVR100 and ISR2 meshes over the domain $[0, 1]^2$.

## 5.4 Application to Adaptive DG

We now focus our attention on the application of LSIAC for $h$-adaptivity using multiwavelet detail indicators. Given a discrete problem, the aim is to compute an adaptive numerical approximation to $u_h$ that satisfies a local error target, $\eta_{TOL}$. Simultaneously, we wish to reduce the number of degrees of freedom required when compared to a uniformly resolved reference approximation. To obtain such an approximation the $h$-adaptivity provided by the LSIAC-MRA methodology is employed. The work and analysis here mirrors that described in [61] where, instead of SIAC-filtering, a reconstruction based on least-squares polynomials was employed. A benefit of the LSIAC approach is that the linear systems do not need to be solved over each sub-element, instead a quadrature is performed.

### 5.4.1 Adaptivity procedure

The adaptivity procedure has the goal of computing the nonuniform mesh $\Omega_h^{target}$ for which the associated numerical solution satisfies user prescribed error targets. In general, the error of a computed numerical approximation is not known and to identify the elements that might benefit from subdivision, proxies termed error indicators are used. These are denoted here by $\eta$. To compare the quality with which a given error indicator tracks the actual error we introduce the effectivity index

$$\iota_{eff} = \frac{\eta}{e_h}. \tag{5.9}$$

Here the global error indicator $\eta$ is computed from the element-wise indicators $\eta_\tau$ by the relation $\eta = \sqrt{\sum_\tau \eta_\tau^2}$, and $e_h$ is defined to be the discrete approximation (computed as in the previous section) to the global $L^2$-norm of the error

$$e_h = \|u - u_h\|_{L^2(\Omega)}. \tag{5.10}$$

An effectivity index near 1 suggests that our chosen error indicator is serving as a good approximation of the actual error. For this work, we considered five different error indicators. Three are error indicators utilizing SIAC filtering and two are other indicators appearing in the literature. They are:

- The filtered difference indicator that simply considers the difference between the filtered solution and the approximation on a coarse grid:

$$\eta_\tau^\star = \|u_h^\star - u_h\|_{L^2(\tau)}. \tag{5.11}$$

- The reconstruction difference indicator that considers the difference between the enhanced reconstruction on the finer mesh to the approximation on the coarser grid:

$$\eta_\tau^{REC} = \left\|\mathcal{P}^{n+1}u_h^\star - u_h\right\|_{L^2(\tau)}. \tag{5.12}$$

- The spectral decay (SD) and normalized small-scale energy decay (SSED) indicators for $p > 0$ that consider the scaled effect of the highest polynomial mode, (see [62] and the references therein):

$$\eta_\tau^{SD} = \frac{\left\| \sum_{\ell=1}^{N_p} u_\tau^\ell \phi_\tau^\ell - \sum_{\ell=1}^{N_p-1} u_\tau^\ell \phi_\tau^\ell \right\|_{L^2(\tau)}}{\left\| \sum_{\ell=1}^{N_p} u_\tau^\ell \phi_\tau^\ell \right\|_{L^2(\tau)}}, \tag{5.13}$$

$$\eta_\tau^{SSED} = \frac{\left\| \sum_{\ell=1}^{N_p} u_\tau^\ell \phi_\tau^\ell - \sum_{\ell=1}^{N_p-1} u_\tau^\ell \phi_\tau^\ell \right\|_{L^2(\tau)}}{|\tau|^{1/2}}, \tag{5.14}$$

  where $\ell$ is an index ordering the basis from lowest ($\ell = 1$) to highest order modes ($\ell = N_p$) the highest order mode. Here $N_p = p + 1$ in one dimension, and $(p + 1)^2$ in two dimensions.

- Lastly, the multiwavelet detail coefficient indicator

$$\eta_\tau^W = \left\| (\mathcal{P}^{n+1} - \mathcal{P}^n) u_h^\star \right\|_{L^2(\tau)} = \left\| \sum_{k=0}^{p} \bar{d}_{k,\tau}^0 \tilde{\psi}_{k,\tau}^0 \right\|_{L^2(\tau)} = \sqrt{\sum_{k=0}^{p} (\bar{d}_{k,\tau}^0)^2}, \tag{5.15}$$

  which in two dimensions is given by

$$\eta_\tau^W = \sqrt{\sum_{k_x,k_y=0}^{p} (\bar{d}_{\mathbf{k},\tau}^{0,\alpha})^2 + (\bar{d}_{\mathbf{k},\tau}^{0,\beta})^2 + (\bar{d}_{\mathbf{k},\tau}^{0,\gamma})^2}, \tag{5.16}$$

  where $\bar{d}$ are the multiwavelet detail coefficients defined by starting a local-MRA (see Section 2.3.3) about element $\tau$.

The adapted solution error is tracked over the course of the numerical experiment, and is denoted by

$$E_h = \left\| u - u_h^{adap} \right\|_{L^2(\Omega)}.$$

The refinement procedure is given below in Algorithm 1.

We now focus our attention on the performance of these different indicators. To do so, we consider steady state test problems. In the one-dimensional case, we make use of explicit time-stepping to steady state. When looping back from STEP 4 to STEP 1, we use the projection of $u_h^{(i)}$ onto $\Omega_h^{(i+1)}$ as the initialization to the discrete problem in STEP 1. This ensures that the whole procedure only requires the coarsest mesh's resolution of the initial data for initialization, rather than a re-initialization at ever increasing mesh resolutions. For the two-dimensional tests, we consider a problem that is a priori steady state in the form of an elliptic problem. Hence, there is no modification of the refinement procedure for the two-dimensional case.

**Algorithm 1** Mesh Adaption Procedure

```
procedure ADAPT(Ω_h^(0), η_TOL)                    ▷ Given initial mesh and local error tolerance
    Compute u_h^(i) the approximate solution to the discrete problem on Ω_h^(i).
    Compute the error indicator η_τ over each element.
    for τ ∈ Ω_h^(i) do
        if η_τ ≥ η_TOL then
            Mark element τ for refinement.
        end if
    end for
    if η_τ < η_TOL for all τ then return (Ω_h^(i), u_h^(i)).
    else
        Refine marked elements and obtain the new mesh Ω_h^(i+1).
        ADAPT((Ω_h^(i+1), η_TOL))
    end if
end procedure
```

### 5.4.2 Test Problems

As a first PDE test case, consider the discrete problem of numerically approximating the solution to the forced viscous Burgers equation

$$u_t + \left(\frac{u^2}{2}\right)_x = \gamma u_{xx} + f(x,t) \tag{5.17}$$

subject to appropriate boundary and initial conditions given below. We consider a modal RKDG discretization where the viscous term is treated by central fluxes and the convective term by a local Lax-Friedrich numerical flux (See [26] Chapter 7. for the nodal version). In one dimension we consider the following test problems

1. Viscous Burgers subject to Dirichlet boundary conditions $u(\pm 1, t) = \mp 1$ with $\gamma = 0.02$, and initial data given by $u(x, 0) = -\tanh(x/2\gamma)$, with $f = 0$.

2. Viscous Burgers subject to periodic boundary conditions with $\gamma = 0.5$, and initial data given by $u(x, 0) = \sin(2\pi x)$, with a forcing term of $f(x,t) = 2\pi \sin(2\pi x)\cos(2\pi x) + 4\pi^2\gamma \sin(2\pi x)$.

The first test problem represents a function with a sharp gradient in the interior of the domain. The boundary conditions are satisfied to within an exponentially small term (see [102]), and so the initial condition is used as a proxy for the exact solution. As the data is non-periodic, the enhancement procedure is not applied in elements where the kernel support would overlap the domain boundary. The second test problem represents approximation of a smooth function, where the forcing term enforces convergence to the initial condition. In both cases, we enforce a 2:1 rule [75] requiring elements be no larger than twice the size of their neighbors. The discrete problem given by the RKDG scheme is evolved to a steady state using a 4th order 2N-storage scheme given in [103]. The time residual is computed as the $\ell_1$ difference of the

modes at successive times steps and is required to be less than $10^{-12}$.

In the two-dimensional tests, we consider an elliptic differential equation,

$$\nabla \cdot (\nabla u) = f(\mathbf{x}), \tag{5.18}$$

subject to Dirichlet boundary conditions. We consider test problems on the domain $\Omega = [0,1]^2$, where the forcing functions and boundary conditions are determined by the exact solutions given by:

1. $u(\mathbf{x}) = \sin(2\pi x)\sin(2\pi y)$,
2. $u(\mathbf{x}) = \exp(-100\,\|\mathbf{x} - (1/2, 1/2)\|_{\ell^2})$.

The first test problem represents a smooth test problem with no sharp features. For this type of function we expect the errors to not significantly differ in different parts of the domain. As such, we expect the refinement procedure to either refine everywhere, or not at all. The second problem, while possessing a smooth solution, is very localized in space and as a result has a sharper gradient in some regions as compared to others. Here we expect the refinement procedure to refine about the Gaussian spike in the interior and provide little or no refinement near the boundaries.

With respect to the DG method employed, the spatial discretization is performed by the Symmetric Interior Penalty (SIP) DG method (see [72]) with penalty parameter chosen as in [74]. The solver used again enforces a 2:1 rule which prohibits elements more than 1 level different from being neighbors (see [75]). Note that despite the Dirichlet boundary conditions, we are applying the filter in a periodic fashion which for these test problems poses no issue. In future work, position-dependent SIAC filters may be employed for finite domains.

### 5.4.3 Numerical Results: Viscous Burgers

For the purposes of adaptivity indication in the one-dimensional case, we have observed that selecting the kernel scaling to be equal to the minimum over all element lengths performs much better than the standard wisdom for obtaining superconvergence in DG data which usually elects to use the maximum element length. We employ this scaling choice to generate the results given below. For the first initial condition, an initial resolution of $2^3 = 8$ elements ($2^4 = 16$ for the $p = 3$ case) is considered. A series of seven refinements (six for the $p = 3$ case) is then performed. For this problem, $\eta_{TOL}$ has been selected to be the average element-wise $L^2$-error of the uniform $2^8 = 256$ element approximation. A table of $\eta_{TOL}$ values for varying $p$ is given in Table 5.1.

Consider first the smooth sine initial data. For this problem there is not a single feature or region of the mesh that has dramatically greater errors than any other region. As a result, the adaptivity procedure has to refine everywhere to obtain an approximation of similar quality to the uniformly refined standard. This

is displayed in Column 1 of Figure 5.8 where it is observed that the adapted approximation closely tracks the uniformly refined approximations. This does not lead to any reduction in the number of degrees of freedom, and in fact a slight over-refinement occurs for the SIAC-based indicators. However, the SD and SSED indicators over-refine much more. The SIAC-based effectivity indices track well with the actual error, though the multiwavelet indicator consistently overestimates the actual error (see Column 2 of Figure 5.8). Column 3 of Figure 5.8 depicts the final meshes achieved by the varying error indicators. All the indicators refine the mesh everywhere in the domain, though the SD and SSED indicators over-refine the mesh relative to the uniform reference.

For the hyperbolic tangent initial data, accuracy comparable to the finest mesh of the uniformly refined solution is obtained with a significant reduction in the number of degrees of freedom for the SIAC-based indicators (see Column 1 in Figure 5.9). Generally the filtered and reconstructed indicators perform similarly, as do the SD and SSED indicators. Effectivity indices are provided in Column 2 of Figure 5.9. The error indicators track the actual error well for the filter-based indicators, though the multiwavelet indicator switches between over- and underestimating the error. The SSED and SD indicators overestimate the error more and as a result cause over-refinement and a large effectivity index. Looking at the final mesh resolutions obtained by these procedures (see Column 3 Figure 5.9), in all cases the greatest resolution is obtained about the sharp gradient near $x = 0$. For $p > 0$, the multiwavelet indicator results in refinement in-between the filtered error indicators and the SD/SSED indicators.

Table 5.1 Average element-wise $L^2$-errors of the $N_{elm}$ uniform element approximations used as user supplied error tolerances $\eta_{TOL}$.

| $N_{elm}$ | IC/$\eta_{TOL}$ | $p = 0$ | $p = 1$ | $p = 2$ | $p = 3$ |
|---|---|---|---|---|---|
| 256 | 1D tanh IC | 1.1000e-03 | 5.0603e-05 | 5.2650e-07 | 5.4431e-08 |
| 256 | 1D sine IC | 9.0045e-04 | 7.3578e-05 | 2.3291e-08 | 1.6228e-09 |
| $16,384$ | 2D Gaussian IC | 7.9961e-03 | 3.0852e-04 | 2.4773e-06 | 3.5193e-10 |
| $16,384$ | 2D sine IC | 2.9829e-02 | 2.3229e-04 | 1.4290e-06 | 2.7362e-09 |

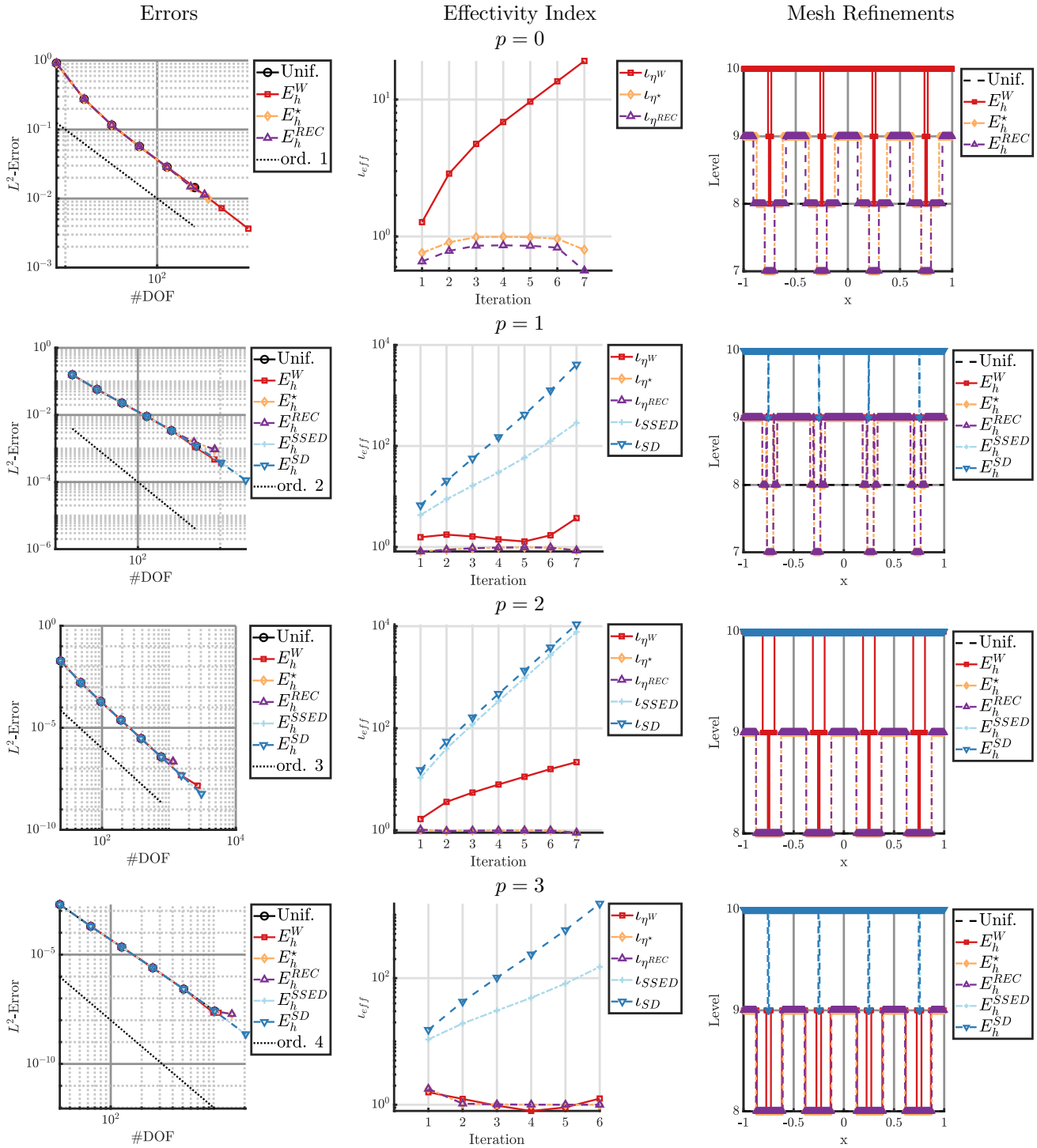


Figure 5.8 $L^2$-errors, effectivity indices, and final meshes achieved under different refinement indicators for the one-dimensional Sine IC. Here a level equal to $n$ indicates an element width of $2^{-n+1}$.

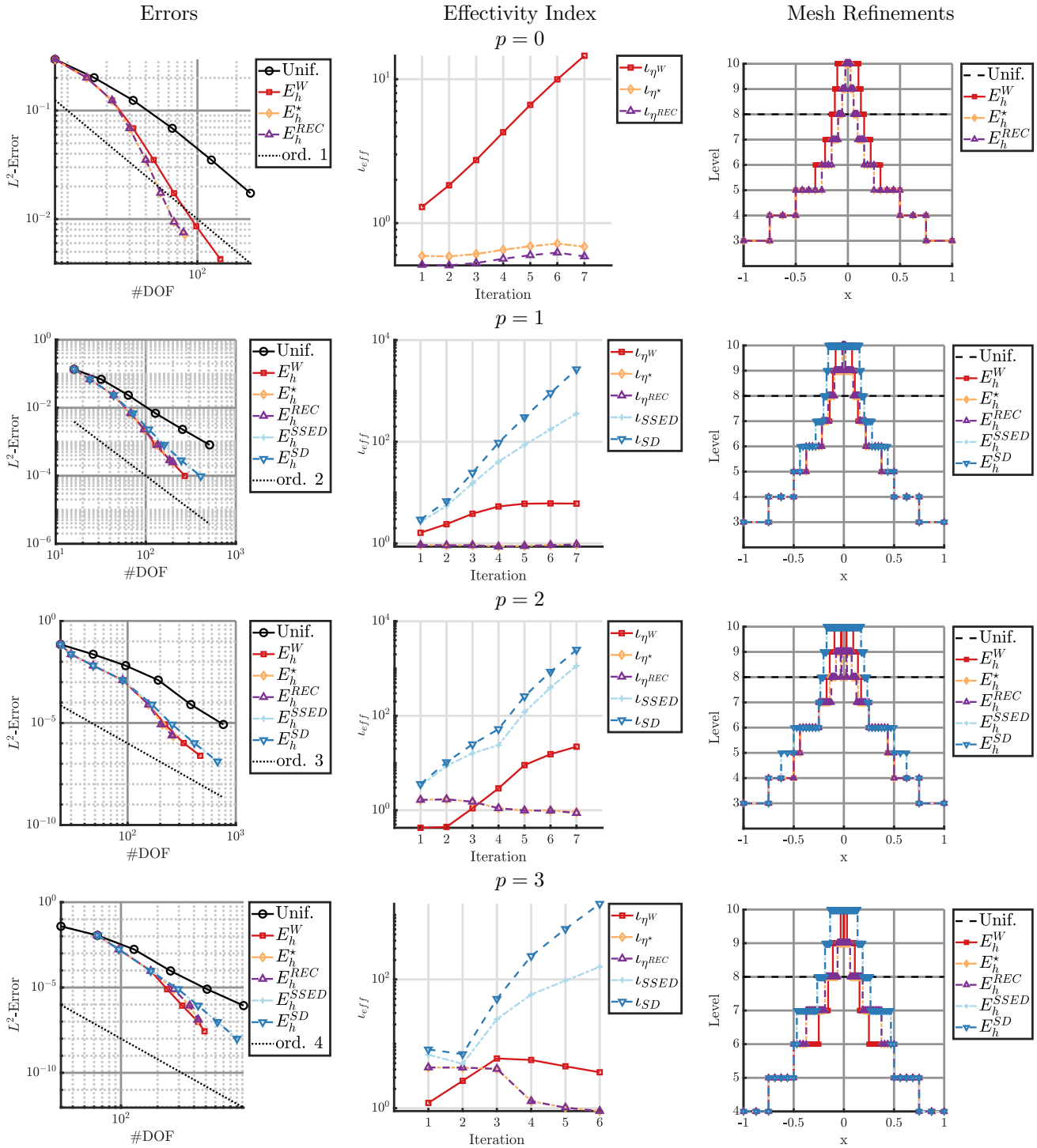


Figure 5.9 $L^2$-errors, effectivity indices, and final meshes achieved under different refinement indicators for the one-dimensional hyperbolic tangent IC. Here a level equal to $n$ indicates an element width of $2^{-n+1}$.

### 5.4.4 Numerical Results: Poisson Equation

For the two-dimensional case, we again choose the kernel scaling equal to the minimum edge length over the mesh. An initial resolution of $16 \times 16$ elements is considered, and a series of four refinements is performed. Again, $\eta_{TOL}$ has been selected as given in Table 5.1 and corresponds to the average element-wise $L^2$-error of a uniform $128 \times 128$ mesh approximation.

With respect to the Gaussian problem, we generally observe lower errors in most cases with reduced numbers of degrees of freedom (see Column 1 Figure 5.11), except for the spectral decay and multiwavelet indicators, for which over-refinement occurs. Looking at the effectivity indices (see Column 2 Figure 5.11), we observe that the multiwavelet indicator overestimates the error at a given level, though typically not as bad as the SD indicator for lower orders. The other SIAC-based indicators generally appear to track the actual error well, if slightly underestimating it, though this does lead to reduced error and reduced numbers of degrees of freedom. In total, the SSED and SD indicators do not perform well, either overestimating the error dramatically in the case of the SD indicator, or underestimating the error too much for $p > 1$ in the SSED case. This prevents a reduced error compared to the reference uniform approximation for $p > 1$.

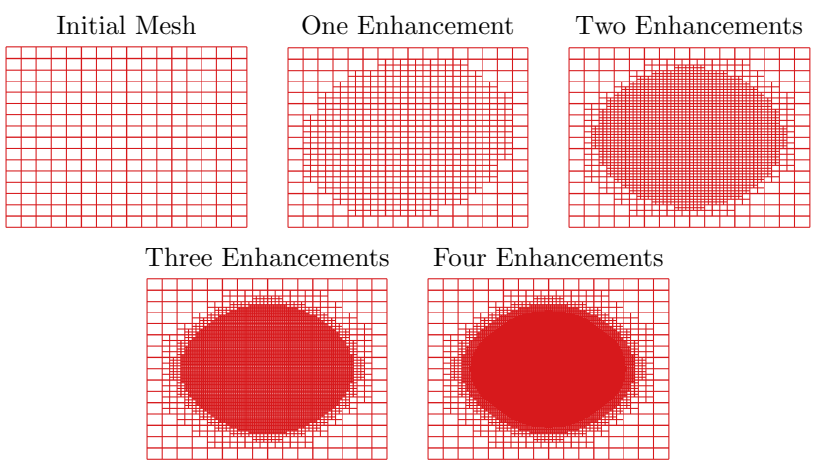


Figure 5.10 Meshes produced by the multiwavelet-based error indicator for $p = 2$. Over repeated enhancements, the resolution about the sharp Gaussian spike in the interior increases while the resolution near the boundary remains coarse.

In Figure 5.10, we plot for illustrative purposes the sequence of meshes obtained after four enhancements using the multiwavelet indicator for the $p = 2$ case. This shows that the procedure increases the refinement about the sharp gradient of the Gaussian spike, and provides less refinement elsewhere in the domain.

For comparison between different error indicators, we also depict the final meshes achieved under different indication schemes for the $p = 2$ case in Figure 5.12. Notice that all but the SD indicator refine

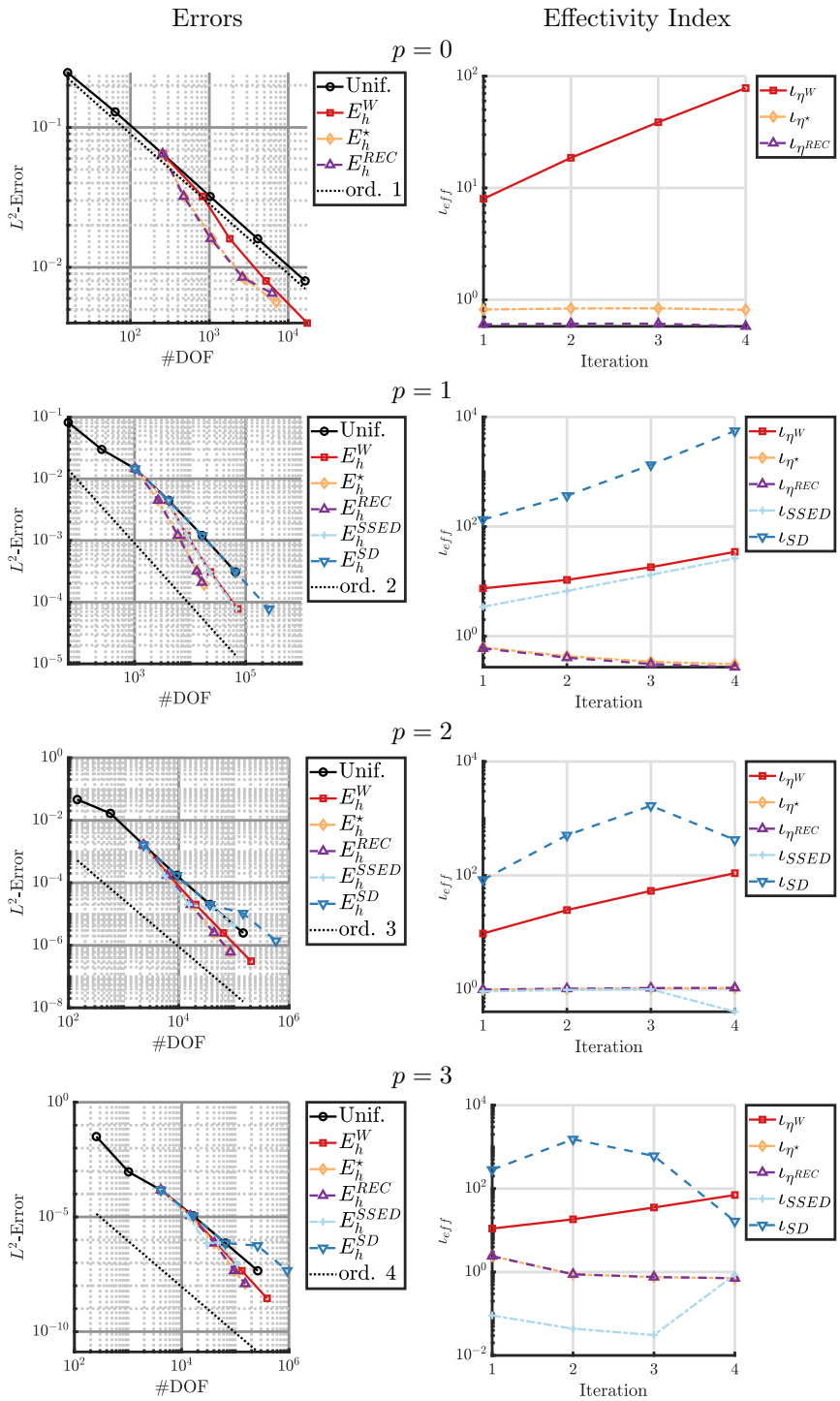


Figure 5.11 The first column depicts the $L^2$-errors and second the effectivity indices achieved under different refinement indicators for the two-dimensional Gaussian IC.

about the center of the domain, whereas the SD indicator refines practically everywhere.

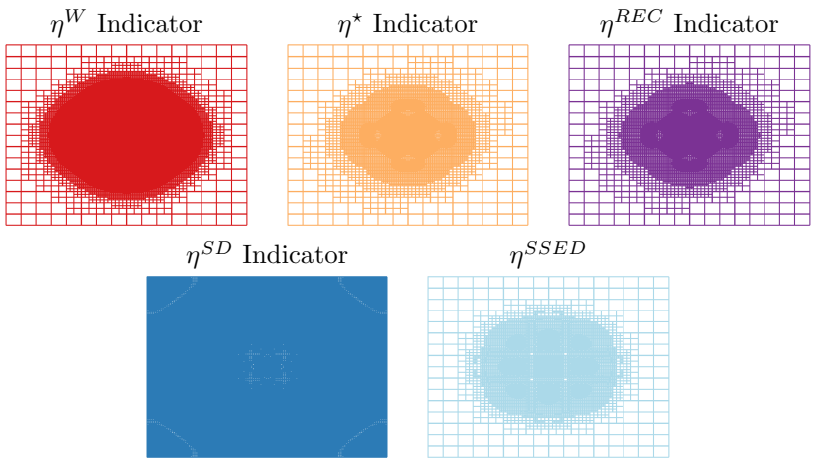


Figure 5.12 Comparison of final meshes generated after the last refinement for the Gaussian test problem with $p = 2$ for different error indicators.

Lastly, we consider the smooth sine problem (see Figure 5.13). Owing to the lack of sharp gradients, the different refinement indicators refine everywhere trying to reach the desired error tolerance. Generally the SIAC-based indicators perform well and reach a similar error level as the uniform reference, though they start to over-refine slightly for $p > 1$. The multiwavelet indicator again tends to overestimate the error and as a result over-refines. The SD and SSED indicators flip between overestimation ($p = 1$) and underestimation ($p = 2, 3$). Outside of the $p = 2$ case for the SD indicator, these indicators either over-refine and overshoot the desired error target, or refine in the wrong places and increase the number of degrees of freedom without substantially affecting the global error.

### 5.4.5 Summary of Adaptivity results

Applying the adaptivity procedure in one and two dimensions, we observe a few trends. The SIAC-based indicators $E_h^\star$ and $E_h^{REC}$ perform well in all one-dimensional cases considered, tracking the uniform error well for smooth problems, and refining about areas of higher error for problems with sharp features, resulting in a reduction of the total number of degrees of freedom. In higher dimensions they begin to overestimate the error, perhaps suggesting that the one-dimensional support of the LSIAC filter may be too restrictive. The SD and SSED do not perform as well in one dimension, where they over-refine for smooth problems, and overestimate the error for problems with sharp features. In two dimensions, they perform erratically, either underestimating, or overestimating the actual error leading to varied performance. Lastly, the multiwavelet indicator has mixed performance in one dimension, overestimating the actual error, but achieving reductions in total numbers of degrees of freedom for problems possessing

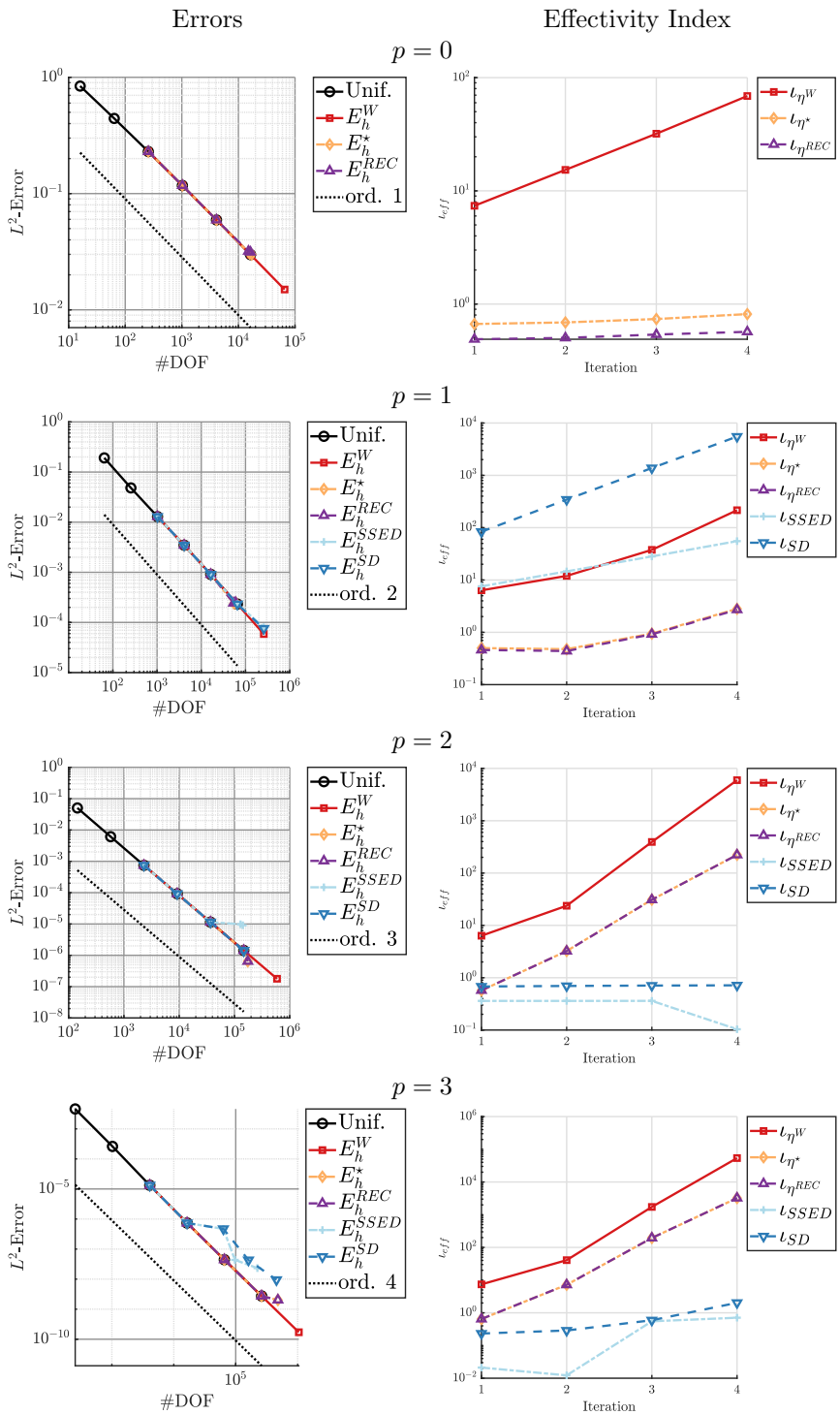


Figure 5.13 The first and second column depict the $L^2$-errors and effectivity indices achieved under different refinement indicators for the two-dimensional tensor sine IC.

sharp features. In two dimensions the multiwavelet indicator consistently overestimates the error, which while less impactful for the Gaussian problem, leads to over-refinement and consequently poor performance for the sine problem. It is possible that the line support filter is too minimal in its averaging support, and that a tensor-product style filter may be prove better in the context of the multiwavelet indicator. In total, of the methods considered, the SIAC-based indicators appear most consistent in attaining error-reductions at reduced numbers of degrees of freedom.

### 5.5 Chapter Summary

In this chapter we have demonstrated the applicability of LSIAC-MRA to nonuniform meshes including perturbed quadrilateral meshes and Delaunay triangulations. We have also confirmed that the inclusion of the adaptive kernel scaling of [19] serves to improve error reduction for meshes with widely varying mesh sizes as compared to the standard choice of the maximum element edge length. With respect to adaptivity, the enhanced reconstructions from LSIAC-MRA can be used to inform adaptation in DG schemes in a manner similar to [61]. Furthermore, adapted solutions based off SIAC-based indicators can be constructed to have fewer degrees of freedom while maintaining similar errors to the uniformly refined solutions. In the future, we anticipate that the (L)SIAC-MRA techniques will prove useful in the resolution of sub-grid dynamics within multi-scale turbulence modelling. An investigation of this application and of the effect of tensor-product filters on the multiwavelet indicator will be pursued in future work. In the next chapter we turn our attention away from $h-$adaptation and consider an alternative means of improving solution quality by SIAC-based entropy correction.

### 5.6 Co-Author Contributions

Jennifer Ryan provided conceptualization for the basis of the project, as well as editing and supervision.

CHAPTER 6

# APPLICATION OF SIAC FILTERING TO ENTROPY-CORRECTION DG SCHEMES

Portions of this chapter are based on a paper in preparation "High-Order Entropy Correction with SIAC Filters" by Matthew J. Picklo and Ayaboe K. Edoh.

## 6.1 Chapter Introduction

In this chapter we consider scalar hyperbolic equations of the form

$$\frac{d}{dt}u + \nabla \cdot \mathbf{f}(u) = 0, \tag{6.1}$$

subject to appropriate initial and boundary conditions. It is well known that even for smooth initial data, classical solutions of (6.1) may not exist for all time, as discontinuities can develop in finite time when characteristics cross [104]. As such, typically we enlarge the set of admissible solutions to include those that satisfy (6.1) in a variational, or "weak" sense. Unfortunately, these weak solutions are not unique, and furthermore when attempting to model natural phenomena, one would hope that there would be a single physically-relevant solution. A selection criterion for finding this physically-relevant weak solution is to look for weak solutions satisfying entropy conditions. Let $U(u)$ be a convex function of $u$, and let $\mathbf{F} = [F_1, F_2, \ldots, F_n]$ be corresponding entropy fluxes satisfying $U'(u)f_j'(u) = F_j'(u)$, $j = 1, \ldots, n$. We say that $u$ satisfies an entropy condition for a given entropy $U$ provided

$$\frac{d}{dt}U(u) + \nabla \cdot \mathbf{F} \leq 0 \tag{6.2}$$

is satisfied in the sense of distributions (see [104] for details). The entropy solution to the original conservation law is then defined to be the weak solution that satisfies (6.2) for every convex entropy. Note that equality holds in (6.2) when the solution $u$ is still a classical solution to (6.1).

When constructing numerical methods to solve hyperbolic conservation laws, it is desirable that these methods satisfy a discrete equivalent of (6.2). Such a relation can often be used to demonstrate the energy stability of a numerical discretization, for example see [105]. Unfortunately it is impossible for high order numerical methods to satisfy such a relation for every convex entropy, and such a restriction limits schemes to first order [106]. Generally a single convex $U$ is considered.

In [107], the authors demonstrated that the analytic Discontinuous Galerkin (DG) method, i.e. the DG method under exact integration, is entropy stable for the square entropy $U = u^2/2$. Unfortunately, exact integration of flux terms is in general either not possible or prohibitively expensive computationally. This

has motivated alternative means of ensuring a discrete entropy (in)equality such as entropy conservative (dissipative) fluxes [63].

Another approach for recovering discrete entropy consistency is the entropy correction approach of Abgrall and collaborators. Their correction procedure is useful for enforcing auxiliary dynamics to a baseline discretization, and has been utilized in compressible flows with respect to various secondary quantities such as angular momentum, kinetic energy, and entropy [108, 109]. This enforcement of additional dynamical/physical consistencies within the discretization has been shown to improve solution quality and robustness. It is this latter approach of Abgrall which we seek to improve upon in this work.

The framework provided by Abgrall allows for great flexibility in the choice of operators (filtering matrices) that are used for the correction. The key requirements are that the chosen operators must be conservative, assuming the underlying system is conservative, and that we posses a measure of the growth/decay of the secondary quantities used to enforce their target dynamics. With respect to local filtering, Ranocha et al. [110] employ element-local dissipation operators and Edoh [111] proposes block-local dissipation operators in the finite difference setting.

The choice of filter dissipation operator that is employed affects the scales at which the correction acts on the solution. To this end, the element-wise averaging relegates the forcing to a length scale associated with the element size; meanwhile general diffusion operators are able to operate at a subcell resolution.

The current work explores the use of SIAC filter operators for the correction procedure. These operators are based on a continuous convolution principle of filtering and are thus mesh agnostic and very flexible. Additionally, SIAC filters are tunable in their damping properties. When used within the finite element framework, they then provide the ability to prescribe smoothness across a range of scales that span between multiple elements and down to the subcell level.

The current work explores the performance of SIAC-based corrections relative to enforcing energy consistency for a "vanilla" Burgers DG space discretization. To make SIAC filters compatible with the procedure of Abgrall, we first introduce a volume-weighted correction that enforces conservation properties for the operator, while maintaining its nominal damping properties. Comparison with the local-element averaging procedure of Abgrall shows that the new approach performs just as well in the target auxiliary energy discretization; this is further confirmed by a proposed LS procedure that assigns a convex weighting between the correction operators. Further, the procedure is efficiently extended to two dimensions via line filtering, and fully discrete conservative properties are achieved by employing relaxation Runge-Kutta (rRK) procedures in time. Meanwhile, regularization of sharp gradients is achieved via the same correction approach, wherein the target energy decay due to shock mollification is approximated according to an artificial viscosity premise. While the inviscid Burgers equation is the PDE model considered in this work,

it is expected that SIAC entropy correction will prove useful in more complex scenarios such as under-resolved high Reynolds number compressible flows.

In Section 6.2 we describe the baseline DG method used in this study. In Section 6.3 we describe a novel adaptation of Abgrall's entropy correction scheme to include multi-element filters, introduce a least-squares procedure for minimizing subcell entropy discrepancies amongst different filter-based corrections, describe a method for shock regularization that does not require computation of an explicit diffusive term, and construct a fully discrete entropy conservative scheme by means of relaxation Runge-Kutta methods. In Section 6.4 we provide an introduction to SIAC filters and their various parameterizations and associated properties. Construction of the discrete filtering operators is also detailed. Lastly, in Section 6.5 we present numerical results, detailing the subcell entropy conservation and solution quality under regularization of different filter parameterizations and combinations of filters via the least-squares procedure. We also extend these results to two dimensions, and confirm that the higher order accuracy of the underlying discretization remains unaffected by the correction.

## 6.2 Baseline Numerical Method

In this work, we focus on the DG spectral element methods (DGSEM) using Legendre-Gauss-Lobatto (LGL) nodes which enable a summation-by-parts property to be satisfied by the spatial discretization. This has been shown by Gassner [105] to enable energy conservation, under suitable numerical flux choices, despite under-integration of volumetric terms. The scheme is formulated in one dimension as follows. Consider (6.1) with supplied initial data $u(x,0) = u_0(x)$ on some domain $\Omega$. Subdivide the domain into elements $\Omega_e = [x_{e-1/2}, x_{e+1/2}]$ s.t. $\cup_{e=1}^{N_{elm}} \Omega_e = \Omega$ and introduce the global to reference element mappings $\xi_e(x) = \frac{2}{\Delta x_e}(x - x_e)$ so that $\xi_e(\Omega_e) = [-1, 1]$.

Multiplication of (6.1) by a test function $v$ and integration over element $\Omega_e$ yields

$$\int_{\Omega_e} v\frac{d}{dt}u + v\frac{d}{dx}f(u)\, d\Omega_e = 0.$$

Integration by parts on the flux term yields

$$\int_{\Omega_e} v\frac{d}{dt}u\, d\Omega_e - \int_{\Omega_e} f(u)\frac{d}{dx}v\, d\Omega_e + [vf(u)]_{\Gamma_e} = 0. \tag{6.3}$$

On each element the solution $u$ is approximated by a polynomial of degree $p$ given by

$$u_h\Big|_{\Omega_e}(x) = \sum_{k=0}^{p} u_k^e \ell_k(\xi_e(x)),$$

where $\{\ell_k\}_{k=0}^{p}$ are Lagrange polynomials given by

$$\ell_k(\xi) = \prod_{m=0, m\neq k}^{p} \frac{\xi - \xi_m}{\xi_k - \xi_m}$$

defined at the $p+1$ point Legendre-Gauss-Lobatto nodes $-1 = \xi_0 < \ldots < \xi_p = 1$. The values $u_k^e$ are the values of the approximation at $x_k^e = \frac{\Delta x_e}{2}\xi_k + x_e$ since $\ell_i(\xi_j) = \delta_{i,j}$. Continuity of the approximation at element interfaces is not enforced in this method, so for the boundary fluxes we introduce a numerical flux term which solves the Riemann problem at the element boundary [112]. Choosing the test function $v = \ell_j(\xi_e)$ yields the form

$$\frac{d}{dt}\int_{\Omega_e} u_h \ell_j(\xi_e(x))\, d\Omega_e - \int_{\Omega_e} f(u_h)\frac{d}{dx}\ell_j(\xi_e(x))\, d\Omega_e + \ell_j(1) f_{e+1/2}^{num} - \ell_j(-1) f_{e-1/2}^{num} = 0, \quad j = 0, \ldots, p. \tag{6.4}$$

Associated with the LGL nodes is the LGL quadrature rule. Denoting the weights and nodes by $\{\omega_k, \xi_k\}_{k=0}^p$ we have

$$\int_{-1}^{1} h(\xi)\, d\xi \approx \sum_{k=0}^{p} \omega_k h(\xi_k), \tag{6.5}$$

which is exact for polynomials of degree $2p-1$.

Approximating the integral volume integrals in (6.4) using the $p+1$ node LGL quadrature results in under-integration or "Mass-lumping" of the first integral, though it produces a diagonal mass matrix. Upon substitution of the element-specific expression for $u_h$ we obtain the DGSEM scheme

$$\frac{\Delta x_e}{2}\omega_j \frac{d}{dt} u_j^e - \sum_{k=0}^{p} \omega_k f_k^e \frac{d\ell_j}{d\xi}(\xi_k) + \ell_j(1) f_{e+1/2}^{num} - \ell_j(-1) f_{e-1/2}^{num},\ j = 0, \ldots, p, \quad e = 1, \ldots, N_{elm}. \tag{6.6}$$

A matrix-vector interpretation is allowed by the scheme given above. To see this, let $\mathbf{u}_e = [u_0^e, \ldots, u_p^e]^T$, $\mathbf{f}_e = [f_0^e, \ldots, f_p^e]^T$, and $\mathbf{f}_e^{num} = [f_{e-1/2}^{num}, 0, \ldots, 0, f_{e+1/2}^{num}]^T$. We then obtain the expression

$$\frac{\Delta x_e}{2}\mathbf{M}\left(\frac{d}{dt}\mathbf{u}_e\right) = \mathbf{D}^T\mathbf{M}\mathbf{f}_e - \mathbf{B}\mathbf{f}_e^{num}, \tag{6.7}$$

where $\mathbf{M} = \mathrm{diag}([\omega_0, \ldots, \omega_p])$, $(\mathbf{D})_{ij} = \frac{d\ell_j}{d\xi}(\xi_i)$, and $\mathbf{B} = \mathrm{diag}([-1, 0, \ldots, 0, 1])$. As will become important in what follows, the local scheme can be expressed in a global format. To match conventions applied later, we choose here to include the mapping factor $\Delta x_e/2$ in the quadrature matrix i.e. $\mathbf{M}_e = \frac{\Delta x_e}{2}\mathbf{M}$. The subscript $G$ used here will denote the global appending of vectors

$$\mathbf{u}_G = [\mathbf{u}_1^T, \ldots, \mathbf{u}_{N_{elm}}^T]^T,$$

or the global block concatenation of matrices

$$\mathbf{M}_G = \begin{bmatrix} \mathbf{M}_1 & & \\ & \ddots & \\ & & \mathbf{M}_{N_{elm}} \end{bmatrix}.$$

Using this convention, the global scheme is expressible as

$$\mathbf{M}_G\left(\frac{d}{dt}\mathbf{u}_G\right) = \mathbf{D}_G^T(\mathbf{I}_{N_{elm}} \otimes \mathbf{M})\mathbf{f}_G - \mathbf{B}_G\mathbf{f}_{G,num}, \tag{6.8}$$

or directly as the semi-discrete ODE

$$\begin{aligned}\frac{d}{dt}\mathbf{u}_G &= \mathbf{M}_G^{-1}(\mathbf{D}_G^T(\mathbf{I}_{N_{elm}} \otimes \mathbf{M})\mathbf{f}_G - \mathbf{B}_G\mathbf{f}_{G,num}),\\ &= \mathbf{r}_G.\end{aligned} \tag{6.9}$$

When the scale is apparent from the context we will drop the $G$ subscript.

### 6.3 Entropy Correction

Consider now the one-dimensional inviscid Burgers equation for which $f(u) = u^2/2$. Integration of the original PDE over some domain $\Omega$ coupled with integration by parts allows an expression for the volumetric time rate of change of $u$ in the domain:

$$\frac{d}{dt}\int_\Omega u \; d\Omega = -f|_{d\Omega}. \tag{6.10}$$

Similarly, considering as an entropy the square entropy $U = u^2/2$, we can multiply both sides by the entropy variable $w = \frac{d}{du}U = u$ and perform the same procedure to obtain

$$\frac{d}{dt}\int_\Omega U \; d\Omega = -F|_{d\Omega}, \tag{6.11}$$

where $F(u) = u^3/3$. In the continuous case, we can determine the volumetric time rate of change of the conserved quantity $u$ and its energy by the fluxes at the boundary of the domain. In the following sections we are concerned with the construction of correction terms to our numerical schemes with periodic boundary conditions that will enable conservation

$$\frac{d}{dt}\int_\Omega u \; d\Omega = 0, \tag{6.12}$$

and entropy conservation/stability

$$\frac{d}{dt}\int_\Omega U \; d\Omega \leq 0, \tag{6.13}$$

of the resulting discretization in a discrete sense. If we consider the numerical methods described previously (6.9), it can be shown (see similar derivation in [113]) that on a single element

$$\mathbf{1}^T\mathbf{M}_e\left(\frac{d}{dt}\mathbf{u}_e\right) = f_{e-1/2}^{num} - f_{e+1/2}^{num}. \tag{6.14}$$

Hence, summing over all the elements yields

$$\mathbf{1}^T\mathbf{M}_G\left(\frac{d}{dt}\mathbf{u}_G\right) = f_{1/2}^{num} - f_{N_{elm}+1/2}^{num}, \tag{6.15}$$

that is to say that conservation is determined entirely by inflow/outflow at the boundary of the domain. For periodic boundary conditions, we in fact have strict conservation

$$\mathbf{1}^T \mathbf{M}_G \left( \frac{d}{dt} \mathbf{u}_G \right) = 0. \tag{6.16}$$

Stability in energy requires additional assumption on the scheme at hand. In [105], the author considered skew formulations of the Burgers DGSEM and demonstrated how to choose numerical fluxes to ensure energy conservation/stability. We instead consider the general approach of Abgrall [2] and apply a correction term to ensure energy stability. Dropping the global indicators here for readability, we have

$$\frac{1}{2}\mathbf{u}^T \mathbf{M} \left( \frac{d}{dt} \mathbf{u} \right) = \frac{1}{2}\mathbf{u}^T \mathbf{M}(\mathbf{r} + \mathbf{c}) \tag{6.17}$$

$$= \frac{1}{2}\mathbf{u}^T \mathbf{M}\mathbf{r} + \frac{1}{2}\mathbf{u}^T \mathbf{M}\mathbf{c}. \tag{6.18}$$

As will be detailed later, $\mathbf{c}$ can be chosen to cancel out the first term resulting in energy conservation. Similarly, it can be augmented with shock regularization capabilities to ensure energy dissipation.

More generally, the correction can be applied to ensure a discretization satisfies spatial entropy conservation condition. The aforementioned entropy correction formulation is as follows. Given the element-level semi-discrete ODE of (6.7) derived above (where we drop the $e$ subscripts for readability), append a correction term to ensure conservation of discrete entropy. The new equation is given by

$$\frac{d}{dt}\mathbf{u} = \mathbf{r} + \mathbf{c}, \tag{6.19}$$

where $\mathbf{c}$ is a correction to the degrees of freedom of the local element of the form

$$\mathbf{c} = \alpha \Big( \mathbf{w} - \frac{\mathbf{1}^T \mathbf{M}\mathbf{w}}{\mathbf{1}^T \mathbf{M}\mathbf{1}} \mathbf{1} \Big). \tag{6.20}$$

Here $\mathbf{w}$ is an entropy variable for the given entropy, and

$$\alpha = \frac{\phi_{loc}}{\mathbf{w}^T \mathbf{M}\mathbf{w} - \frac{(\mathbf{1}^T \mathbf{M}\mathbf{w})^2}{\mathbf{1}^T \mathbf{M}\mathbf{1}}}.$$

Notice that $\phi_{loc}$ is selected to cancel out the entropy flux on the local element. This term is defined as

$$\phi_{loc} = -\int_{\partial\Omega_e} \mathbf{n} \cdot F^{num} \, d\Omega_e - \mathbf{w}^T \mathbf{M}\mathbf{r}, \tag{6.21}$$

where $F^{num}$ is the numerical entropy flux associated with the predetermined numerical flux $f^{num}$ by the relation

$$F^{num}_{e+1/2} = \{\!\{ w_{e+1/2} \}\!\} f^{num}_{e+1/2} - \{\!\{ \psi_{e+1/2} \}\!\}, \tag{6.22}$$

where $\psi_{e+1/2} = w_{e+1/2} f_{e+1/2} - F_{e+1/2}$. The correction as written satisfies conservation

$$\mathbf{1}^T\mathbf{M}\mathbf{c} = 0, \tag{6.23}$$

and the desired correction property

$$\mathbf{w}^T\mathbf{M}\mathbf{c} = -\int_{\partial\Omega_e} \mathbf{n}\cdot F^{num}\, d\Omega_e - \mathbf{w}^T\mathbf{M}\mathbf{r}. \tag{6.24}$$

Note that typically a small constant is added to denominator of $\alpha$ to prevent division by zero in the case of a constant solution in an element. This form of correction was shown by the authors of [3] to be the solution to the minimization problem min $\|\mathbf{c}\|_{\mathbf{M}}$ subject to local the constraints of equations (6.23) and (6.24).

Having discussed this previous approach and to motivate the choice of alternative corrections, note that (6.20) requires a computation of an average of the local data. Defining the filter $\mathbf{K}_{loc.avg} = \frac{\mathbf{1}\mathbf{1}^T\mathbf{M}}{\mathbf{1}^T\mathbf{M}\mathbf{1}}$, we can rewrite (6.20) as

$$\mathbf{c} = (\phi_{loc})\frac{(\mathbf{I}-\mathbf{K}_{loc.avg})\mathbf{w}}{\mathbf{w}^T\mathbf{M}(\mathbf{I}-\mathbf{K}_{loc.avg})\mathbf{w}}. \tag{6.25}$$

That is to say, this correction employs local element-wise averaging which can be interpreted as a specific type of filter. When comparing to alternative formulations, we will refer to this element local correction approach of Abgrall as the "local" approach or the "local" method.

In this work we expand the approach to include multi-element filters such as SIAC operators previously mentioned. Below we again will use the $G$ subscript to denote a matrix or vector that is a global concatenation of single element components. Supposing a global target dynamic, define

$$\phi_{glob.} = \int_\Omega \frac{d}{dt}U_{targ.}\, d\Omega - \mathbf{w}^T\mathbf{M}_G\mathbf{r}_G. \tag{6.26}$$

Then we can construct a global correction term as

$$\mathbf{c}_G = \alpha_{glob.}(\mathbf{I}_G - \mathbf{K}_G)\mathbf{w}_G, \tag{6.27}$$

where

$$\alpha_{glob.} = \frac{\phi_{glob.}}{\mathbf{w}_G^T\mathbf{M}_G(\mathbf{I}_G-\mathbf{K}_G)\mathbf{w}_G}, \tag{6.28}$$

and $\mathbf{K}_G$ is an unspecified conservative filter ($\mathbf{1}^T\mathbf{M}_G\mathbf{K}_G\mathbf{v} = \mathbf{1}^T\mathbf{M}_G\mathbf{v}$ for arbitrary $\mathbf{v}$.) The augmented global scheme $\frac{d}{dt}\mathbf{u}_G = \mathbf{r}_G + \mathbf{c}_G$ will remain conservative: $\mathbf{1}_G^T\mathbf{M}_G\mathbf{c}_G = 0$, and will satisfy the target global dynamic: $\mathbf{w}_G^T\mathbf{M}_G(\mathbf{c}_G+\mathbf{r}_G) = \int_\Omega \frac{d}{dt}U_{targ.}\, d\Omega$. Note that while the new correction does not to the authors knowledge satisfy an optimally condition like that of the local correction, we will show in numerical results that the global correction can better reduce the relative magnitude of the correction: $\|\mathbf{c}\|_{\mathbf{M}_G} / \|\mathbf{r}\|_{\mathbf{M}_G}$. This is not a contradiction as allowing for multi-element filters affords more flexibility in allocation of the correction.

### 6.3.1 An Adaptive Filter Selection Strategy

To further improve the global correction procedure, we consider a constrained least-squares (LS) problem for selecting an optimal convex combination of two correction terms. Equivalently, this provides a means of differentiating between filtering operators and implicitly their scales of influence. Since every correction considered maintains global conservation, we need to use a subcell metric of entropy violation to differentiate between them. The metric we use comes directly from (6.2) and is given by

$$D_h(x_i^e) = w_i^e \frac{d}{dt} u_i^e + \frac{d}{dx} F(u_h(x_i^e)) = w_i^e \mathbf{r}(x_i^e) - \mathbf{rF}(x_i^e), \tag{6.29}$$

where $x_i^e$ is the $i$th node of element $e$, $\mathbf{r}$ is the DGSEM discretization of the RHS of the conservation law previously described, and $\mathbf{rF}$ is the discretization via DGSEM of the entropy flux term $F(u)_x$. The interpretation here is that if the local entropy is being well preserved at a given node, $D_h$ should be near 0. We note that while this was the choice of measure for subcell entropy conservation employed in this work, other choices could be made.

With a discrepancy metric in hand, the constrained LS problem is as follows: Given two entropy correction terms $\mathbf{c}_1$ and $\mathbf{c}_2$ resulting from conservative filters $\mathbf{K}_1$ and $\mathbf{K}_2$, choose a correction

$$\mathbf{c}_\theta = \theta \mathbf{c}_1 + (1-\theta)\mathbf{c}_2, \tag{6.30}$$
$$= \mathbf{c}_2 + \theta(\mathbf{c}_1 - \mathbf{c}_2), \tag{6.31}$$

parameterized by $\theta$, $\theta \in [0,1]$ s.t. $\|\mathbf{D}_h\|_{\mathbf{M}_G}$ is minimized. The form of the vector $\mathbf{D}_h$ in terms of $\theta$ is found by substitution of the discrete scheme into (6.29) ($\odot$ denoting element-wise multiplication):

$$\begin{aligned}
\mathbf{D}_h &= \mathbf{w} \odot \frac{d}{dt}\mathbf{u} + \frac{d}{dx}\mathbf{F}(\mathbf{u}) \\
&= \mathbf{w} \odot (\mathbf{r} + \mathbf{c}_\theta) - \mathbf{rF} \\
&= \mathbf{w} \odot (\mathbf{r} + \mathbf{c}_2) - \mathbf{rF} + \theta(\mathbf{w}) \odot (\mathbf{c}_1 - \mathbf{c}_2) \\
&= \mathbf{a} + \theta \mathbf{b},
\end{aligned} \tag{6.32}$$

where $\mathbf{a} = \mathbf{w} \odot (\mathbf{r} + \mathbf{c}_2) - \mathbf{rF}$ and $\mathbf{b} = (\mathbf{w}) \odot (\mathbf{c}_1 - \mathbf{c}_2)$. The optimization problem is then solved numerically at each stage of the time evolution of $\mathbf{u}$ to choose an appropriate correction term. Without the constraint on $\theta$ the least-squares solution is given in a straight forward manner as

$$\theta = -(\mathbf{b}^T \mathbf{M}_G \mathbf{b})^{-1} (\mathbf{b}^T \mathbf{M}_G \mathbf{a}). \tag{6.33}$$

Note that this is just the standard least-square solution procedure for $\|\mathbf{D_h}\|_2$, but the structure of $\mathbf{M}_G$ makes it easy to find the least-squares solution in its associated norm. As the quadrature matrix is positive

definite there exists a matrix $\mathbf{R}$ s.t. $\mathbf{R}^T\mathbf{R} = \mathbf{M}_G$, and so it can be shown that the 2-norm LS solution to (6.32) multiplied by $\mathbf{R}$ is the same as the $\mathbf{M}_G$-norm LS solution.

The LS approach described above does not require any numerical optimization solvers, however it unfortunately does not enforce the convex-combination constraint $\theta \in [0, 1]$. Here the authors employed the MATLAB lsqlin [114] routine to numerically solve for $\theta$ satisfying the constrained LS minimization problem, which uses an active-set quadratic programming algorithm [115], see documentation in [116] for details.

### 6.3.2 Shock Regularization based on Artificial Viscosity

Referencing the entropy inequality given in (6.2), it is expected that entropy is conserved for smooth solutions, and decreasing for non-smooth ones. Unfortunately, it is difficult to determine numerically when a discontinuity has formed in the numerical solution, and a conservative correction should be replaced with a dissipative one. This is compounded in that DGSEM already has only weak continuity at element interfaces. In any case, this is not the approach pursued here.

Additionally, high order numerical methods such as DGSEM often produce numerical instability in the form of Gibb's oscillations when trying to resolve discontinuities. As a result stabilization techniques are necessary to prevent simulations from crashing/blowing-up post shock-formation. Various methodologies have been proposed for stabilization such as filtering, special entropy dissipative fluxes, artificial dissipation, and (subcell)limiters [105, 117–119], see [120] chapter 12 for an overview of some common methods. The approach described here will build artificial dissipation into the correction technique.

The idea of artificial viscosity is motivated by an equivalent definition of an entropy solution where the entropy solution is obtained as the limiting solution as $\epsilon \to 0$ of the viscous PDE

$$\frac{d}{dt}u_\epsilon + \nabla \cdot \mathbf{f}(u_\epsilon) = \nabla \cdot (\epsilon \nabla u).$$

For this parabolic problem the solution remains smooth, with the addition of diffusion causing the sharp features of a discontinuity to be smeared. Adding artificial viscosity to a numerical scheme can then stabilize the underlying method and reduce spurious oscillations. Unfortunately, discretizing the high-order derivative terms in the parabolic equation often places more restrictive time-stepping constraints on explicit methods.

To avoid these issues, we will instead augment the corrections previously discussed to produce an entropy dissipative scheme using an artificial dissipation mechanism, whereby dissipation is ensured by augmenting the magnitude of the spatial correction term. Note we term this artificial dissipation rather than viscosity as we do not explicitly append a diffusive term to the underlying PDE.

The change appears in the definition of the magnitude term $\phi_{glob.}$. For entropy conservation in the global filter case, a spatially constant magnitude $\phi_{glob.} = -(\mathbf{w}^T\mathbf{M}\mathbf{r})$ was chosen to cancel out the global entropy production of the scheme. To ensure dissipation, we simply need to augment $\phi_{glob.}$ with a non-positive constant $\phi_{diss}$. That is replace $\phi_{glob.}$ in (6.28) with

$$\phi^{\epsilon}_{glob.} = \phi_{glob.} + \phi_{diss}. \tag{6.34}$$

Note that incorporating dissipation in this way will not affect the global conservation of the method as only the scaling of the correction changes, not its global conservation properties.

Below we discuss the choice of the constant $\phi_{diss}$ specifically for the one-dimensional Burgers equation, though we note that the procedure extends to more general conservation laws. Consider the time change of energy for the viscous Burgers equation

$$\frac{d}{dt}u + \frac{d}{dx}(u^2/2) = \frac{d}{dx}(\nu_{AV}\frac{d}{dx})u.$$

Following an approach similar to that producing (6.13), we obtain, using integration by parts,

$$\frac{d}{dt}\int_{\Omega} U\, d\Omega = -F\Big|_{d\Omega} + (\nu_{AV}\frac{d}{dx}U)\Big|_{d\Omega} - \int_{\Omega}\nu_{AV}(\frac{d}{dx}u)^2\, d\Omega. \tag{6.35}$$

Supposing $\nu_{AV} \geq 0$ makes the third term non-negative and so

$$\frac{d}{dt}\int_{\Omega} U\, d\Omega \leq -F\Big|_{d\Omega} + (\nu_{AV}\frac{d}{dx}U)\Big|_{d\Omega}. \tag{6.36}$$

To enforce entropy dissipation we use the third term integrated over a single element as an indicator of the magnitude of the dissipation:

$$\phi^e_{diss} = \frac{2}{\Delta x_e}(\mathbf{D}\mathbf{u}_e)^T\mathbf{M}[\nu^e_{AV}]\mathbf{D}\mathbf{u}_e, \tag{6.37}$$

where $\mathbf{D}$ is the differentiation operator in (6.7), and the inverse Jacobian term is resultant from switching the derivatives of the Lagrange basis functions from the reference element $[-1, 1]$ to the physical element $\Omega_e$. In what follows we will pursue an approach such that the viscosity term $\nu_{AV}$ is constant within an element. Unfortunately, the global correction term approach requires a constant dissipative term be applied globally. To that end, we will choose $\phi_{diss}$ to be the sum of each individual element's dissipative magnitude: $\phi_{diss} = \sum_e \phi^e_{diss}$. For the local approach, a piecewise constant viscosity is acceptable as changing the magnitude of the local correction term element to element does not impact conservation.

In determining $\nu_{AV}$, we consider here a variant of entropy viscosity inspired by the approach of Guermond et al. [118]. Define for each element the minimum distance between LGL nodes on that element $h_e = \frac{\Delta x_e}{2}\min_{i\neq j}|\xi_i - \xi_j|$. We make use of the entropy residual $D_h(x^e_i)$ previously defined in (6.29) to define the entropy viscosity

$$\nu_E^e = c_E(h_e^2 \max_k |D_h(x_k^e)|) / \left\|\mathbf{U} - \bar{\mathbf{U}}\right\|_\infty, \tag{6.38}$$

where for Burgers equation with the square entropy, $\bar{\mathbf{U}} = \frac{1}{2}\mathbf{u}_G^T \mathbf{M}_G \mathbf{u}_G / (\mathbf{1}^T \mathbf{M}_G \mathbf{1})$. The tuning parameter $c_E$ is user supplied to specify the amount of dissipation. The maximum viscosity $\nu_{max}$ is given by

$$\nu_{max}^e = c_{max}(\Delta x_e \max_k |f'(u_k^e)|), \tag{6.39}$$

where again $c_{max}$ is a tuning parameter. The element-wise viscosity is then selected as the minimum of the two quantities: $\nu_{AV}^e = \min\{\nu_E^e, \nu_{\max}^e\}$. The results of this technique and the problem-dependent tuning parameters will discussed in the results section. We note that the magnitude of the dissipation for this approach is determined by the error metric $D_h$ and the volume integral of the gradient of the solution squared. The spatial allocation of this dissipation is determined by the kernel filter applied.

### 6.3.3 Fully Discrete EC via relaxation Runge-Kutta time-stepping

Having described a semi-discrete numerical scheme which is entropy conserving in space, we now demonstrate how to create fully discrete energy conserving schemes by applying the conservative relaxation Runge-Kutta (rRK) time-stepping schemes of Ketcheson [121]. Note that applying this approach for conservation of more general entropies than the square entropy leads to root finding problems which we do not endeavor to face here (see [122]).

Consider the ODE IVP

$$\frac{d}{dt}u = f(u(t), t), \quad u(t_0) = u_0, \tag{6.40}$$

and suppose that the system is dissipative with respect to some inner-product:

$$\frac{d}{dt}\|u(t)\|^2 = 2\langle u, f(u,t)\rangle \leq 0. \tag{6.41}$$

The goal is to ensure that discrete time-stepping yields a solution that is also dissipative

$$\left\|u^{n+1}\right\| \leq \|u^n\|. \tag{6.42}$$

Similarly, for conservative systems ($\leq \to =$), we would like conservation in some discrete norm to hold. For our purposes, the inner-products and norms considered are those defined by global symmetric positive-definite quadrature matrices $\mathbf{M}_G$.

For time-stepping methods, Ketcheson considered Runge-Kutta methods which are given by

$$
\begin{aligned}
y_i &= u^n + \Delta t \sum_{j=1}^{s} a_{ij} f(y_j, t_n + c_j \Delta t), \\
u^{n+1} &= u^n + \Delta t \sum_{j=1}^{s} b_j f(y_j, t_n + c_j \Delta t),
\end{aligned}
\tag{6.43}
$$

where $\mathbf{A} \in \mathbb{R}^{s \times s}$, and $\mathbf{b}, \mathbf{c} \in \mathbb{R}^s$. Further, it is assumed that $c_j = \sum_{i=1}^{s} a_{ij}$. Following [121] we introduce the notational convenience

$$
f_i = f(y_i, t_n + c_i \Delta t).
$$

The change in energy over a single time step can be shown to satisfy

$$
\left\| u^{n+1} \right\|^2 - \left\| u^n \right\|^2 = 2\Delta t \sum_{j=1}^{s} b_j \langle y_j, f_j \rangle - 2(\Delta t)^2 \sum_{i,j=1}^{s} b_i a_{ij} \langle f_j, f_i \rangle + (\Delta t)^2 \sum_{i,j=1}^{s} b_i b_j \langle f_i, f_j \rangle. \tag{6.44}
$$

It is the approach of rRK methods to introduce an adaptive time step $\gamma_n \Delta t$ s.t. for

$$
u_\gamma^{n+1} = u^n + \Delta t \gamma_n \sum_{j=1}^{s} b_j f(y_j, t_n + c_j \Delta t) \tag{6.45}
$$

the change of energy over that adapted time step is 0. To see how the parameter $\gamma_n$ is determined, note that the energy change between $t_n$ and $t_n + \gamma_n \Delta t$ is given by

$$
\left\| u_{\gamma_n}^{n+1} \right\|^2 - \left\| u^n \right\|^2 = 2\Delta t \gamma_n \sum_{j=1}^{s} b_j \langle y_j, f_j \rangle - 2\Delta t^2 \gamma_n \sum_{i,j=1}^{s} b_i a_{ij} \langle f_j, f_i \rangle + (\Delta t \gamma_n)^2 \sum_{i,j=1}^{s} b_i b_j \langle f_i, f_j \rangle.
$$

In a conservative systems, the first term is zero, whereas in dissipative systems it is non-positive provided $b_j > 0$ for $j = 1, \ldots, s$. The baseline RK schemes we employ will satisfy this later requirement. The goal now is to choose $\gamma_n$ s.t. the two remaining terms cancel out. A little algebra yields a solution

$$
\gamma_n = \begin{cases} \frac{2 \sum_{i,j=1}^{s} b_i a_{ij} \langle f_i, f_j \rangle}{\sum_{i,j=1}^{s} b_i b_j \langle f_i, f_j \rangle}, & \left\| \sum_{j=1}^{s} b_j f_j \right\|^2 \neq 0, \\ 1, & \text{Else} \end{cases}. \tag{6.46}
$$

This last condition is imposed to prevent division by 0.

In relating this to the entropy-correction approach for Burgers equation with the squared entropy, we note that given a conservative/dissipative semi-discrete system, in this case

$$
\frac{d}{dt} \mathbf{u} = \mathbf{r} + \mathbf{c},
$$

which is made so by addition of the correction term, we can evolve the system in time via the rRK method and preserve the conservative/dissipative nature. In the results section this capability is put to use in a host of test problems.

### 6.4 Filter Construction and Properties

In constructing filters to be applied in the entropy correction context, we use the SIAC filters previously defined in Section 2.1. In one dimension we use the standard uniform SIAC kernels, and in two dimensions the LSIAC variant. Below we describe the construction of discrete SIAC filter matrices and some of their properties.

#### 6.4.1 Filter Matrix Construction

For applications in later sections, we detail here the construction of SIAC filtering matrices for periodic DG data in one dimension. We exclude the details for higher dimensions as they follow the same approach but with more complicated indexing. When applying the matrix form of the SIAC filter, we are effectively truncating the filtered expression so that it remains in the same polynomial space as the original data. That is, in this case we are sampling the filtered approximation at the node locations of the original Lagrange basis.

Before building a global filtering matrix, consider the evaluation of the filtered expression at a node. We have

$$\begin{aligned} u_h^\star(x_k^e) &= \int_\Omega K_H(x_k^e - y) u_h(y)\, dy \\ &= \sum_{e=1}^{N_{elm}} \sum_{k=0}^{p} u_k^e \int_{\Omega_e} K_H(x_k^e - y) L_k^e(y)\, dy \\ &= (\mathbf{k}_{e,k})^T \mathbf{u}, \end{aligned}$$

where

$$(\mathbf{k}_{e,k})_z = \int_{\Omega_j} K_H(x_k^e - y) L_i^j(y)\, dy, \quad z(i,j) = i + (j-1)(p+1), \tag{6.47}$$

for $i = 1, \ldots, p+1$, $j = 1, \ldots, N_{elm}$. Introducing an index $z(e,k) = k + (e-1)(p+1)$ for a global ordering of the row vectors, we assemble the global filtering matrix $\mathbf{K} = [\mathbf{k}_1, \ldots, \mathbf{k}_{N_{elm}(p+1)}]^T$. Multiplication of this matrix on the nodal vector returns the values of the filtered approximation at those nodes:

$$\mathbf{u}^\star = \mathbf{K}\mathbf{u}. \tag{6.48}$$

The compact support of the kernel will generally allow for a sparse filtering matrix as can be seen for the LSIAC filter in Figure 6.1 (Right). As an example of the different discrete filtering operators produced by this procedure, the local element stencils for the $K_{\Delta x}^{1,1}$ and $K_{\Delta x}^{(3,2)}$ kernels for $p = 1$ data in one dimension on a uniform grid are given by

$$\mathbf{stencil}_{\Delta x}^{(1,1)} = \frac{1}{8} \left[ \begin{array}{cc|cc|cc} 1 & 3 & \mathbf{3} & \mathbf{1} & 0 & 0 \\ \hline 0 & 0 & \mathbf{1} & \mathbf{3} & 3 & 1 \end{array} \right],$$

and

$$\mathbf{stencil}_{\Delta x}^{(3,2)} = \frac{1}{72}\left[\begin{array}{cc|cc|cc|cc|cc} -1 & -2 & 12 & 27 & \mathbf{27} & \mathbf{12} & -2 & -1 & 0 & 0 \\ \hline 0 & 0 & -1 & -2 & \mathbf{12} & \mathbf{27} & 27 & 12 & -2 & -1 \end{array}\right].$$

Here the red values denotes the location of the node being filtered by the filtering procedure, and the boldface values denote the node of the reference element. The vertical lines separate the nodal coefficients of different elements. The whole discrete filtering matrix in the periodic case is then obtained by periodic repetition of these stencils. Note that for higher polynomial degrees, while the stencils for boundary nodes are equal up to a shift, stencils for the interior nodes can vary.

In Figure 6.1 (Left) we depict the kernel support and associated integral for computing a single entry in the LSIAC $K^{(1,1)}_{\Gamma,\sqrt{2}\Delta x}$ filter matrix for $p = 5$ data. Also depicted are the LGL nodes corresponding to the degrees of freedom of the approximation.

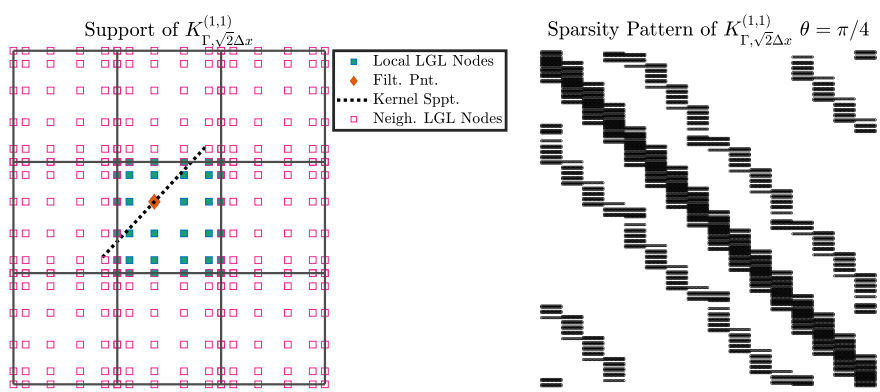


Figure 6.1 Left: Depiction of the LSIAC filter procedure for the $K^{(1,1)}_{\Gamma,\Delta x}$ filter with $\theta = \pi/4$ applied to $p = 5$. Right: Depiction of the sparsity pattern of the resultant $K^{(1,1)}_{\Gamma,\Delta x}$ filtering matrix for $p = 3$ on a $4 \times 4$ uniform mesh with $\theta = \pi/4$.

### 6.4.2 Conservation of the Filtering Procedure

An important requirement of filters being applied in the EC context is that they are globally conservative, i.e.

$$\mathbf{1}^T\mathbf{M}\mathbf{u}^\star = \mathbf{1}^T\mathbf{M}\mathbf{u}, \tag{6.49}$$

where here $\mathbf{M}$ represent the global quadrature rule accounting for Jacobians of local to reference mappings: $\mathbf{1}^T\mathbf{M}\mathbf{1} = |\Omega|$. Following the lead of Vreman [123], we describe one way of constructing a conservative correction for a non-conservative filter. For arbitrary initial data, we wish

$$\mathbf{1}^T\mathbf{M}\mathbf{u}^\star - \mathbf{1}^T\mathbf{M}\mathbf{u} = \mathbf{1}^T\mathbf{M}(\mathbf{K} - \mathbf{I})\mathbf{u} = 0.$$

Now for a non-conservative filter this last equality may not hold, hence subtracting the discrepancy in the form of a properly scaled $\mathbf{1}^T\mathbf{M}(\mathbf{K}-\mathbf{I})$ term from the original matrix should yield conservation:

$$\mathbf{K}_{corr} = \mathbf{K} - \frac{\mathbf{1}\mathbf{1}^T}{\mathbf{1}^T\mathbf{M}\mathbf{1}}\mathbf{M}(\mathbf{K}-\mathbf{I}). \tag{6.50}$$

Note that here

$$\begin{aligned}\mathbf{1}^T\mathbf{M}\mathbf{K}_{corr} &= \mathbf{1}^T\mathbf{M}(\mathbf{K} - \frac{\mathbf{1}\mathbf{1}^T}{\mathbf{1}^T\mathbf{M}\mathbf{1}}\mathbf{M}(\mathbf{K}-\mathbf{I})) \\ &= \mathbf{1}^T\mathbf{M}\mathbf{K} - \mathbf{1}^T\mathbf{M}(\mathbf{K}-\mathbf{I}) \\ &= \mathbf{1}^T\mathbf{M},\end{aligned}$$

hence

$$\mathbf{1}^T\mathbf{M}\mathbf{K}_{corr}\mathbf{u} = \mathbf{1}^T\mathbf{M}\mathbf{u}$$

for any $\mathbf{u}$. Thus conservation is achieved.

Note that there are other forms the correction could take. Such as, for example,

$$\mathbf{K}_{corr,2} = \mathbf{M}^{-1}(\mathbf{K} - \frac{\mathbf{1}\mathbf{1}^T}{\mathbf{1}^T\mathbf{1}}(\mathbf{K}-\mathbf{M})).$$

This particular choice has deleterious effects on the spectrum of the corrected operator, so we do not pursue it here. As an example of the issue this second correction causes, consider the constant mode $\mathbf{1}$. Owing to the imposed consistency condition, our filters will reproduce constants i.e. $\mathbf{K}\mathbf{1} = \mathbf{1}$, and this is a desirable property in the corrected filter. Put another way, we wish $(1, \mathbf{1})$ to be an eigenpair of $\mathbf{K}_{corr}$ when it is an eigenpair of $\mathbf{K}$. We show here that the original $\mathbf{K}_{corr}$ preserves this property while $\mathbf{K}_{corr,2}$ does not in general.

*Proof.* Supposing that $\mathbf{K}\mathbf{1} = \mathbf{1}$, note that $\mathbf{K}_{corr}$ is consistent as

$$\begin{aligned}\mathbf{K}_{corr}\mathbf{1} &= \mathbf{K}\mathbf{1} - \frac{\mathbf{1}\mathbf{1}^T}{\mathbf{1}^T\mathbf{M}\mathbf{1}}\mathbf{M}(\mathbf{K}-\mathbf{I})\mathbf{1} \\ &= \mathbf{K}\mathbf{1} - \frac{\mathbf{1}\mathbf{1}^T}{\mathbf{1}^T\mathbf{M}\mathbf{1}}\mathbf{M}(\mathbf{K}\mathbf{1}-\mathbf{1}) \\ &= \mathbf{K}\mathbf{1} - \frac{\mathbf{1}\mathbf{1}^T}{\mathbf{1}^T\mathbf{M}\mathbf{1}}\mathbf{M}(\mathbf{1}-\mathbf{1}) \\ &= \mathbf{K}\mathbf{1} = \mathbf{1}.\end{aligned}$$

Considering $\mathbf{K}_{corr,2}$ however, we have

$$
\begin{aligned}
\mathbf{K}_{corr,2}\mathbf{1} &= \mathbf{M}^{-1}(\mathbf{K}\mathbf{1} - \frac{\mathbf{1}\mathbf{1}^T}{\mathbf{1}^T\mathbf{1}}(\mathbf{K}-\mathbf{M})\mathbf{1}) \\
&= \mathbf{M}^{-1}(\mathbf{1} - \frac{\mathbf{1}\mathbf{1}^T}{\mathbf{1}^T\mathbf{1}}(\mathbf{1}-\mathbf{M}\mathbf{1})) \\
&= \mathbf{M}^{-1}\frac{\mathbf{1}\mathbf{1}^T}{\mathbf{1}^T\mathbf{1}}\mathbf{M}\mathbf{1}.
\end{aligned}
$$

Unless $\mathbf{M}$ is a constant multiple of the identity, the above expression does not equal $\mathbf{1}$.

□

The discussion above does not mean to suggest that $K_{corr,2}$ will not have eigenvalues equal to one, simply that the corresponding eigenvector will not be a constant mode as hoped. By means of numerical justification, consider the global averaging filter $\mathbf{K}_{glob.avg} = \frac{\mathbf{1}\mathbf{1}^T}{\mathbf{1}^T\mathbf{1}}$. Note that this filter is consistent ($\mathbf{K}_{glob.avg}\mathbf{1} = \mathbf{1}$), but not conservative for arbitrary quadrature rules, so a correction is appropriate. Here we consider as quadrature rule, a 20 element LGL quadrature rule with 6 points per element. The eigenvalues of the filter remain unchanged under correction, with one eigenvalue remaining 1 while the rest are 0, but the eigenvector corresponding to $\lambda = 1$ does change for the second correction see Figure 6.2. As the second correction violates the consistency requirement, it is unsuitable for present purposes.

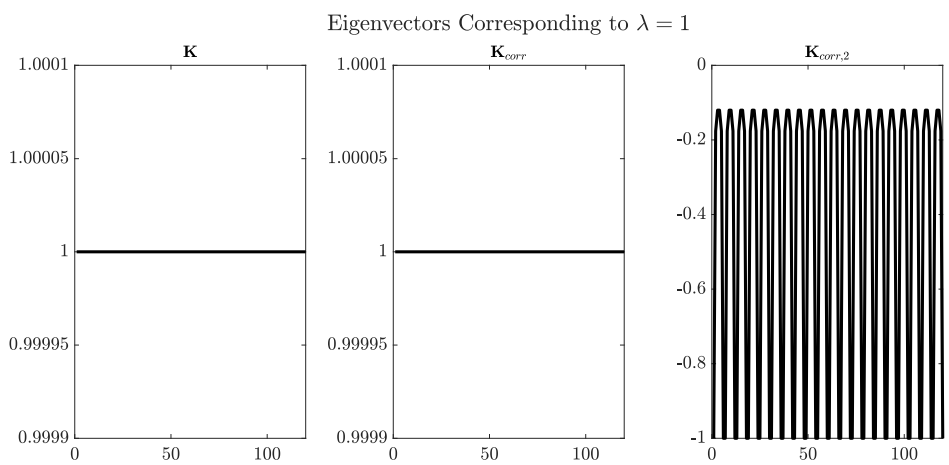


Figure 6.2 Eigenvector corresponding to $\lambda = 1$ for varying corrections of $\mathbf{K}_{glob.avg}$. Note that here a 20 element mesh using a 6 node per element LGL quadrature matrix $\mathbf{M}$ was used. The eigenvectors $\mathbf{v}$ have been normalized so that $\|\mathbf{v}\|_{\ell^\infty} = 1$.

### 6.4.3 Fourier and Spectral Response

For periodic boundary conditions, the spectral effects of the continuous SIAC filtering operation can be assessed directly from the Fourier transform of the kernel. This is given by

$$\hat{K}_H^{(r+1,\ell)}(k) = \left(\frac{\sin\left(\frac{kH}{2}\right)}{\frac{kH}{2}}\right)^{\ell}\left(c_{\lceil r/2\rceil+1} + 2\sum_{\gamma=1}^{\lceil r/2\rceil} c_\gamma \cos(\gamma k H)\right). \tag{6.51}$$

Plotted in Figure 6.3, we see the effects of various parameterizations of the SIAC kernel on its Fourier response. Specifically, varying $H$ dilates the response, while increasing the smoothness of the spline $\ell$ leads to a faster decay of the Fourier transform as the underlying kernel is smoother. Lastly, increasing the number of moments sharpens the response by making it flatter near zero, specifically the response has its first $r$ derivatives equal to 0 at $k = 0$.

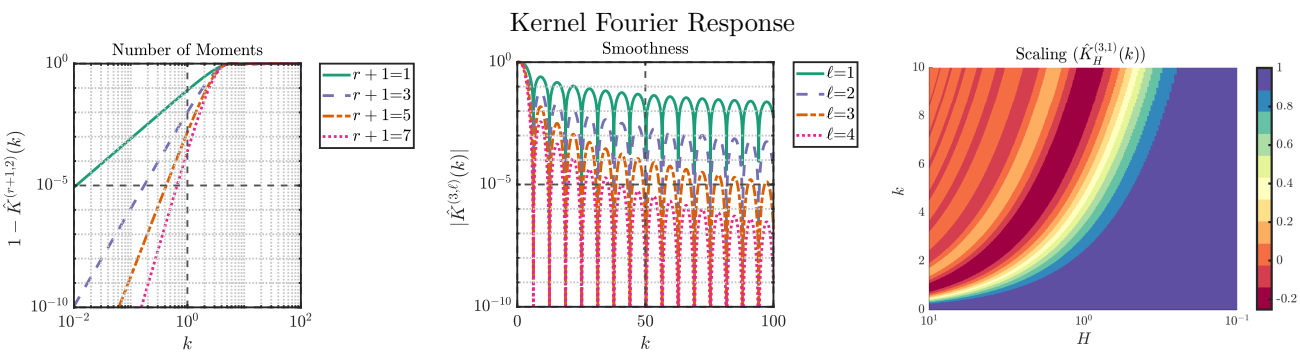


Figure 6.3 Comparison of the effect of number of moments, spline order, and kernel scaling on the Fourier response of the SIAC filter.

The truncation of the continuous filtering procedure required to produce a discrete filtering matrix may influence the spectral properties and stability of the resulting discrete filtering procedure. To that end, we consider for varying parameterizations and corrections the eigenvalues of the filtering matrices. As these matrices are non-symmetric, they may still exhibit temporary transient growth even when their eigenvalues are bounded above in magnitude by unity [124]. To address this, we also consider the singular values of the resulting filtering matrices which are plotted for varying moments, spline orders, and scalings in Figure 6.4. In all cases the maximum singular values are greater than 1 so transient growth may occur if the filter were applied in an iterative fashion, but that is not a concern in the correction procedure. Similar to the continuous case, increasing the number of moments, i.e. polynomials reproduced by the filter, results in eigenvalues moving closer to 1. For periodic BCs, only the "periodic" constant polynomial is reproduced exactly. Increasing the spline order and thus the smoothness of the kernel increases the damping capabilities of the filter and causes eigenvalues to gravitate towards 0. Similarly, increasing the scaling also

increases the damping capabilities of the filter and moves eigenvalues away from 1. All that is said for the eigenvalues also applies to the singular values, except that they are strictly non-negative and not bounded above by unity.

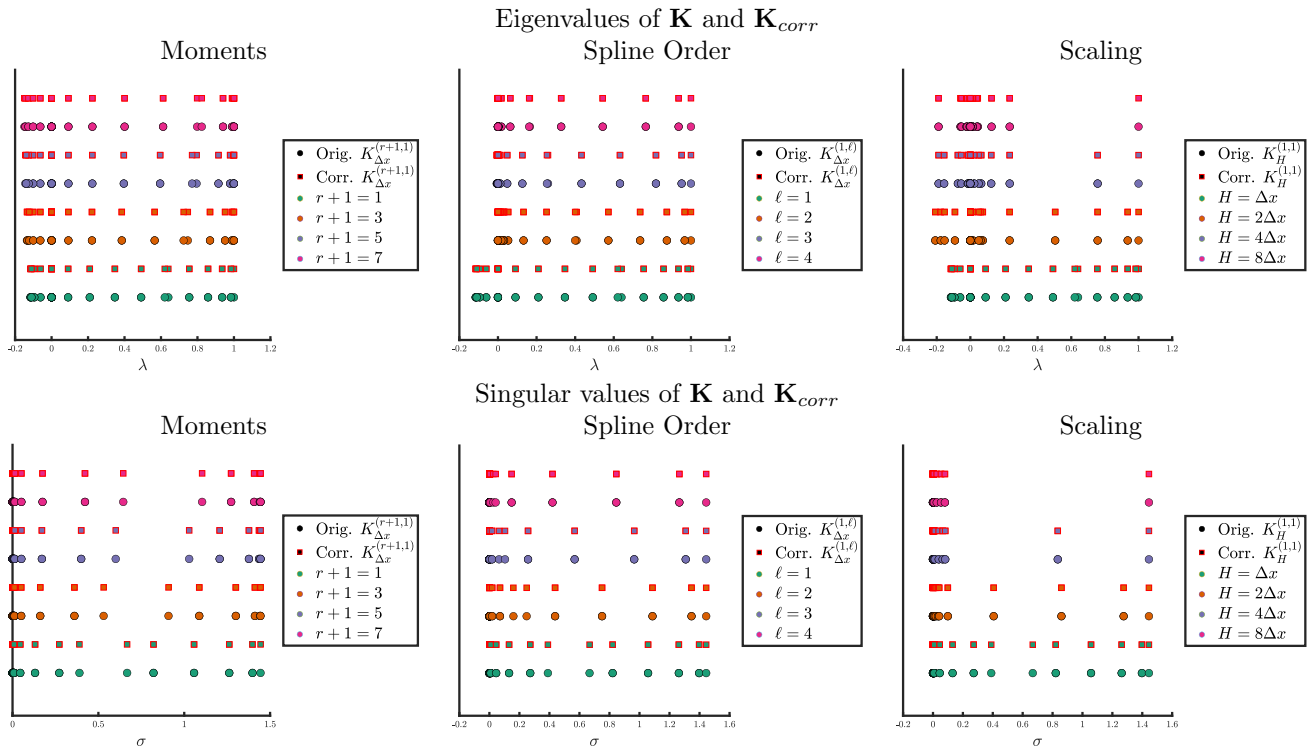


Figure 6.4 Comparison of the effect of number of moments, spline order, and kernel scaling on the spectrum and singular values of the discrete filtering matrix for a $p = 3$ approximation on a $N_{elm} = 10$ mesh. Note only real values of the eigenvalues are plotted here as the imaginary components are, if not 0, on the order of machine precision.

## 6.5 Results

In this section we demonstrate the applicability of SIAC-based entropy-correction terms for energy conservation, as well as solution regularization. We also investigate the impact of different SIAC correction parameterizations. Additionally, we show experimentally that SIAC corrections preserve the original order of accuracy of the underlying method.

### 6.5.1 One-dimensional entropy conservation

We compare the performance of the different correction methods applied to the inviscid Burgers problem $u_t + (u^2/2)_x = 0$ on the periodic domain $\Omega = [0, 2]$ subject to initial data $u(x, 0) = \sin(\pi x) + 0.01$. For this problem, the characteristics of the PDE cross and a shock forms at $x = 1$ at $t = 1/\pi$. We begin by addressing the choice of numerical flux in the method. We compare between a non-dissipative central flux

$f^{num}_{cent.,\, e+1/2} = \frac{1}{2}(f(u^e_p) + f(u^{e+1}_0))$ and slightly-more stabilizing Local Lax-Friedrichs (LLF) formulation $f^{num}_{LLF,\, e+1/2} = \frac{1}{2}(f(u^e_p) + f(u^{e+1}_0)) + \frac{1}{2}\alpha_{LLF}(u^e_p - u^{e+1}_0)$. Results are depicted in Figure 6.5 for a $p = 5$ approximation over a $N_{elm} = 21$ element mesh. In terms of solution quality we see that the approximations using the central flux are less desirable, and also less stable. In fact, the uncorrected central flux scheme crashes before reaching $t = 1.5/\pi$, though the uncorrected LLF scheme does not reach the final time of $t = 2/\pi$ either. In terms of the subcell entropy error metric, the LLF approaches perform better than their central flux counterparts. In both cases, the correction approach making use of SIAC filters is necessary to stabilize the scheme, and since the LLF does not appear to introduce numerical stiffness into the rRK-LS procedure, we will exclusively use the LLF DGSEM scheme in the experiments to follow. We note that the results are applicable to alternative baseline flux choices.

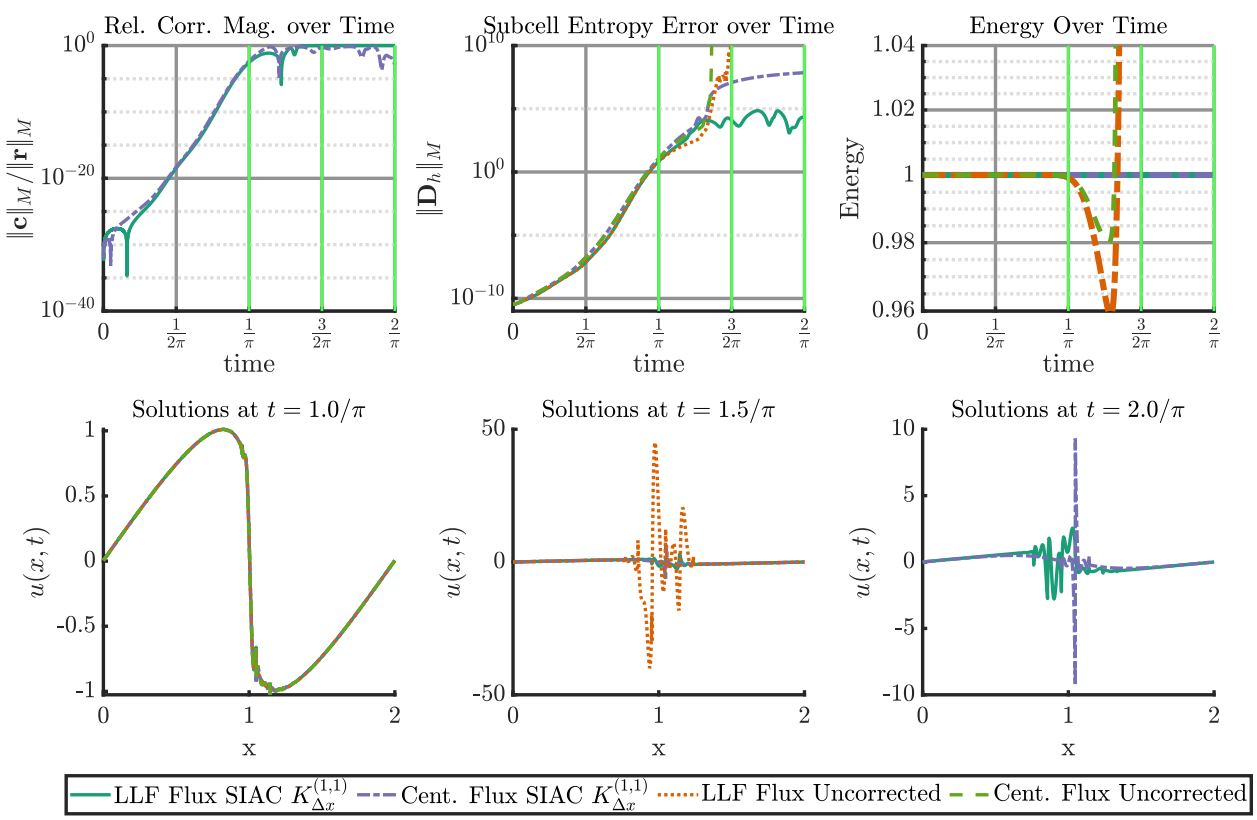


Figure 6.5 Comparison of SIAC corrected and uncorrected approaches using central fluxes and LLF fluxes. First row: Relative magnitude of the correction to the RHS, subcell entropy metric, and energy over time of the method. Second Row: Solutions at time of shock formation $t = \frac{1}{\pi}$ and then afterwards at $t = \frac{3}{2\pi}$ (the unplotted uncorrected central flux solution has crashed) and $t = \frac{2}{\pi}$ (Only corrected solutions remain).

Having fully specified the baseline scheme we now consider the effects of different correction formulations. In Figure 6.6 we depict the performance of the $K^{(1,1)}_{\Delta x}$ filter correction against the Local approach and the various LS combinations for a $p = 5$ and $N_{elm} = 21$ element approximation. These

approaches are using a LLF flux in the DGSEM method. We also compare with the Skew-Symmetric Entropy-Conserving method of Gassner [105]. As depicted, similar performance is observed with respect to the subcell entropy metric. Only after shock formation do any of the methods deviate. As expected, after this time the uncorrected scheme begins to blow up. We note that significant flexibility in the choice of subcell entropy metric exists, and other options are certainly possible. For all tests here the same subcell metric was applied to enable comparison of performance.

With respect to the relative correction magnitude, the SIAC approach has a lower relative magnitude then the other formulation prior to shock formation. The unconstrained LS approach has a much larger magnitude owing to $\theta$ being unconstrained, increasing the magnitude of $\mathbf{c}$. Looking at the change in $\theta$ over time, we see that initially neither correction is favored in reducing $\|\mathbf{D}_h\|_{\mathbf{M}}$, but eventually the SIAC correction becomes favorable just prior to shock formation and remains so after shock formation until eventually oscillations between the two corrections occur. In terms of the solution quality, we see that at shock formation the different approximations performs similarly, but afterwards the conservation of energy is no longer mathematically required and so this lack of decay introduces oscillations into the solution. These oscillations develop on the right side of the shock before eventually spreading throughout the profile. The energy-conserving Skew-Symmetric formulation produces oscillations on both sides of the shock profile, which over time also disperse through the profile. We wish to note that the uncorrected DGSEM method using the LLF flux (not plotted) begins to blow up following shock formation and crashes prior to the final time of $t = 2/\pi$, further justifying the need for augmentation of the baseline scheme.

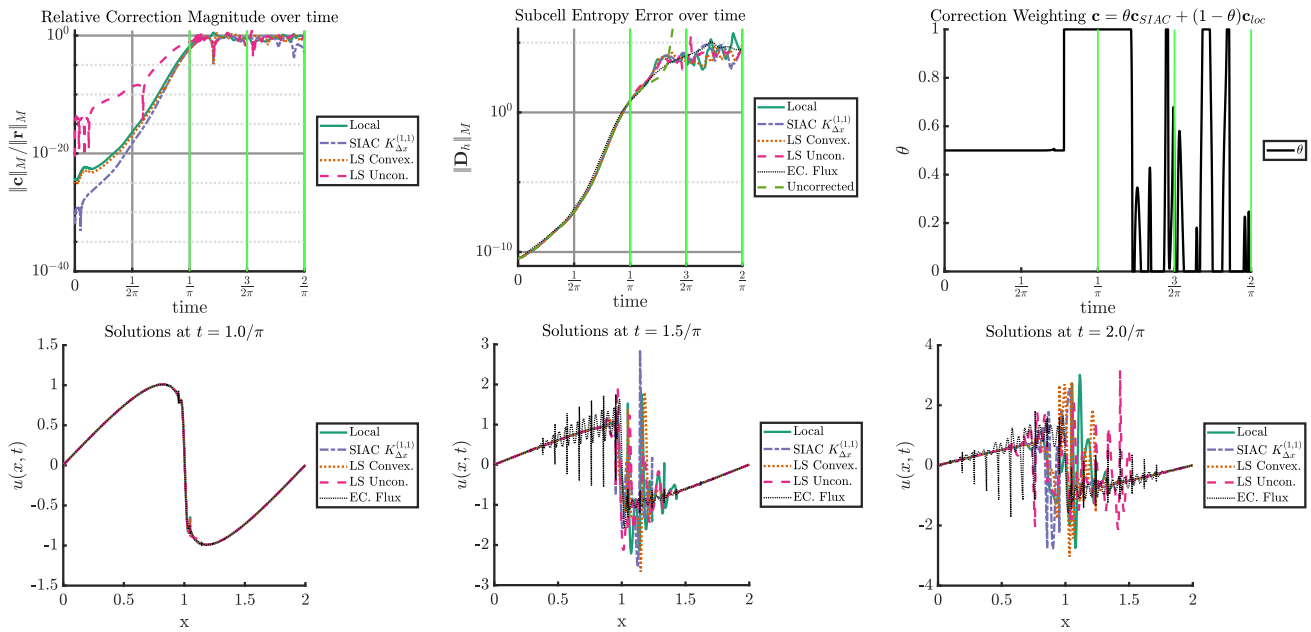


Figure 6.6 Comparison of SIAC and Local EC approaches, and their LS combinations with the EC Skew-Symmetric DGSEM formulation. First row: Relative magnitude of the correction to the RHS, subcell entropy metric, and weighing of constrained LS combination of corrections with $\theta = 1$ being only a SIAC correction and $\theta = 0$ being only a local correction). Second Row: Solutions at time of shock formation $t = \frac{1}{\pi}$ and then afterwards at $t = \frac{3}{2\pi}$ and $t = \frac{2}{\pi}$.

### 6.5.2 One-dimensional regularization

Taking the one-dimensional shock problem and adding a dissipative term to the correction as describe in section 6.3.2, we investigate the performance of different filter parameterizations, and compare the local correction approach to SIAC-based corrections.

For all regularization tests, a $p = 5$ approximation on a $N_{elm} = 21$ element mesh was evolved past shock formation to a final time of $t = 5/\pi$. In comparing the qualitative performance of different SIAC-correction parameterizations the tuning parameters were held constant at $c_E = 10$ and $c_{max} = 1$. In Figure 6.7 (Left) we observe that increasing the number of moments which is the number of splines, leads to degradation of solution quality at sharp gradients. Increasing the spline order, which increases the smoothness of the kernel, slightly increases the energy dissipation of the regularization, which can be observe in Figure 6.8 (Center). The kernel scaling has the greatest effect on dissipation though, with greater scalings leading to more energy dissipation as depicted in the solution profiles Figure 6.7 (Right) and the normalized energy plots Figure 6.8 (Right). Lastly, we note that all parameterizations conserve the domain integral of $u$ up to floating point arithmetic accuracy Figure 6.8 (Row 1).

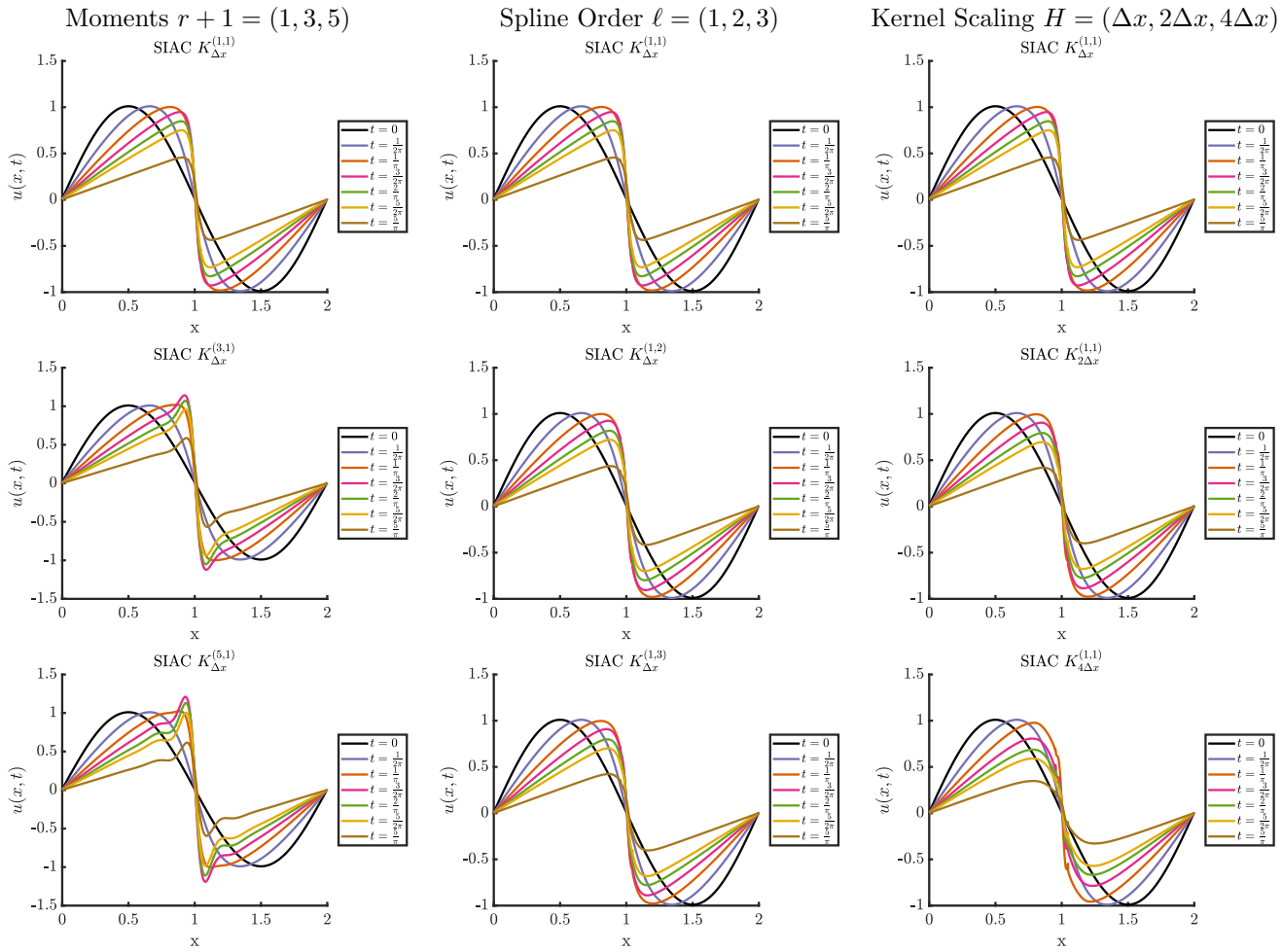


Figure 6.7 Comparison of SIAC correction regularization results for varying kernel parameterizations: number of moments/splines (Left), spline order/smoothness (Center), and kernel scaling (Right). Here $K_{\Delta x}^{(1,1)}$ is the baseline kernel with changed parameters as indicated. For all tests $c_e = 10$ and $c_{max} = 1$ was used. Increasing moments degrades solution quality when the approximation loses smoothness. Increasing the order and scaling makes the regularization more dissipative.

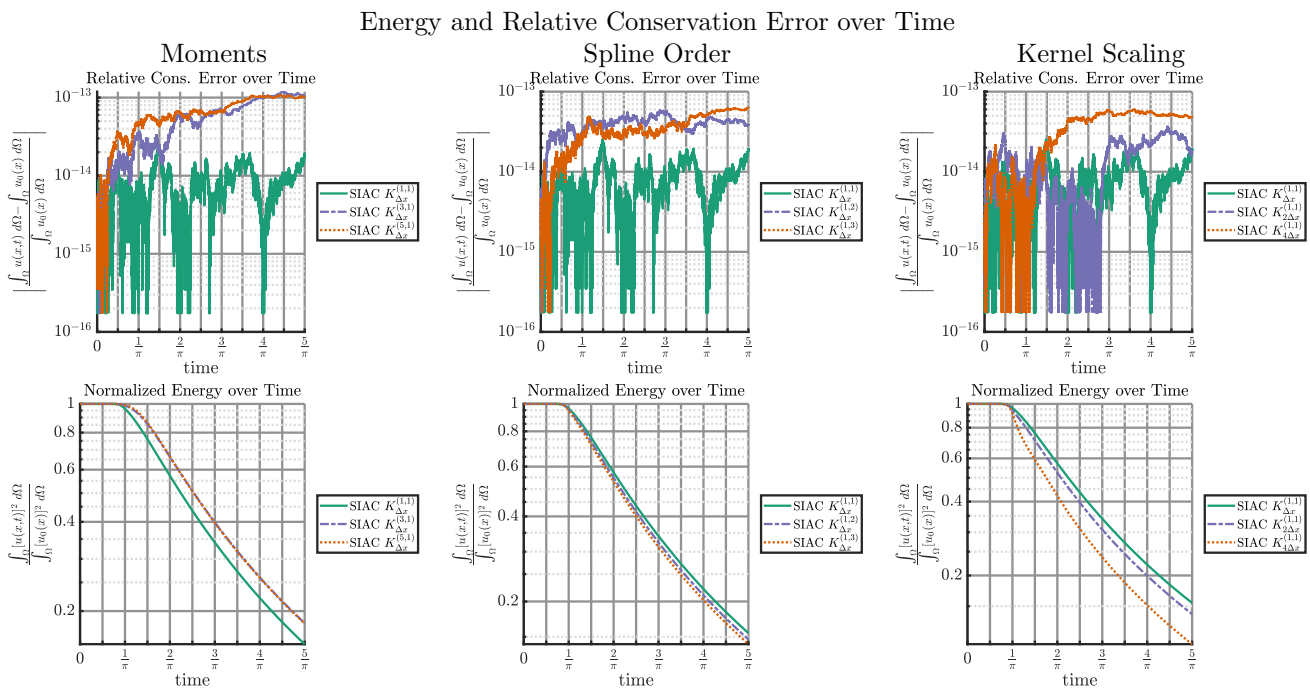


Figure 6.8 Comparison of the relative conservation error (Row 1) and energy dissipation (Row 2) of the SIAC correction regularization for varying kernel parameterizations. Here $K_{\Delta x}^{(1,1)}$ is the baseline kernel with changed parameters as indicated. For all tests $c_e = 10$ and $c_{max} = 1$ was used.

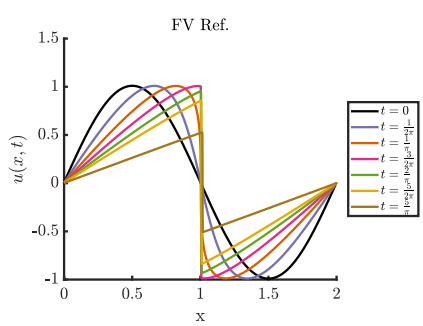


Figure 6.9 Reference finite volume solution to Burgers equation with $N_{elm} = 2000$ elements and a Local Lax-Friedrichs flux.

Comparing the local approach of Abgrall to a SIAC-based correction, we investigate the solution quality as well as dissipation magnitudes for different scalings. Here the tuning parameters were chosen so as to ensure qualitatively good solution profiles as compared to a $N_{elm} = 2000$ fine mesh finite volume reference solution depicted in Figure 6.9. Owing to the difficulty in finding quantitative measures of qualitative agreement we have elected to distinguish the performance of the two approaches by applying the same tuning parameters to each. Here we tune each correction type for "best" qualitative performance and then apply those same tuning parameters to the other correction and compare the numerical solutions.

In tuning the local correction, it was found that $c_E = 4$ and $c_{max} = 0.15$ produced the best early time profiles. These dissipation parameters were then applied to both the SIAC and local corrections and metrics collected. Note that in terms of solution quality, parameter combinations that damped oscillations while preventing eventual flattening of the solution in the element containing the shock were elusive for the local approximation. The solution profiles in Figure 6.10 (Left) depict this, with the SIAC correction performing better in damping oscillations and preventing their spread into other elements near the shock. The local approach did not fair as well. As discussed previously, the solution quality degrades eventually as the solution in the element containing the shock flattens and oscillations appear in its neighbors. The gradient-based sensor performs well in both cases. Figure 6.10 (Center) shows how $\phi_{diss}$ varies spatially. Noting that the global approach requires a single dissipation magnitude for all elements, we see that the magnitude of dissipation increases when the shock begins to form at $t = 1/\pi$, and in the local case, the greatest dissipation occurs in the elements containing the shock, and its neighbors. Both methods conserve mass, and have similar magnitudes of energy dissipation over time Figure 6.10 (Right), with both being more dissipative then the finite volume reference solution.

The SIAC correction is much less sensitive to the choice of tuning parameters and after minimal tuning, $c_E = 10$ and $c_{max} = 1$ were selected to produce solution profiles with minimal oscillations. Applying these dissipation parameters to both correction schemes, the solution profiles in Figure 6.11 (Left) were obtained. The SIAC approach performs better, with an oscillation free profile over time. The local approach eventually devolves into a stair-stepping approximation, with flattened local approximation spreading out from the center element over time. This is caused by the increasing dissipation applied to those elements as shown in Figure 6.11 (Center). Comparatively, the dissipation magnitudes of the SIAC correction remain similar to the local-tuned case. While both methods still conserve mass, the accuracy to which the local approach does so is inhibited, perhaps by floating point arithmetic effects. Similar magnitudes of energy dissipation over time are obtained, see Figure 6.11 (Right). The SIAC approach is more dissipative in later times than the local approach, and both remain more dissipative than the finite volume reference.

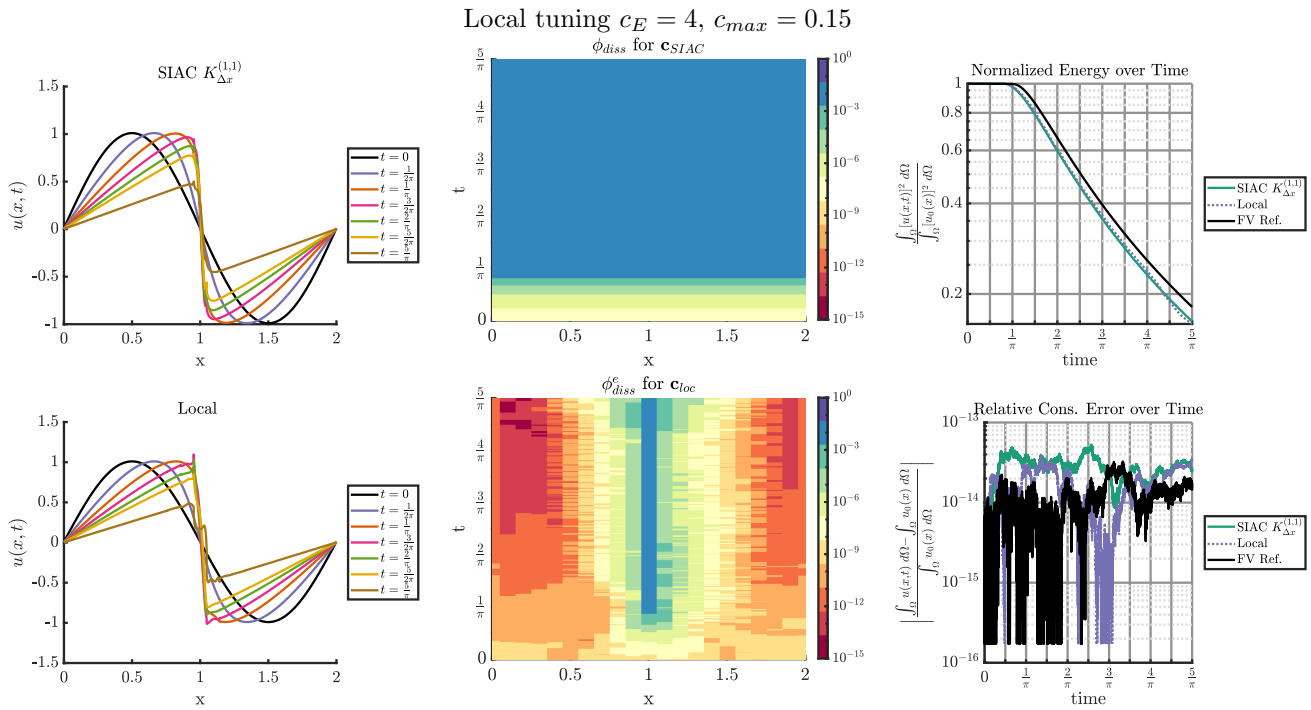


Figure 6.10 Comparison of Local and SIAC correction approaches to regularization. Here the tuning parameters used were those that qualitatively gave the best performance for the local correction.

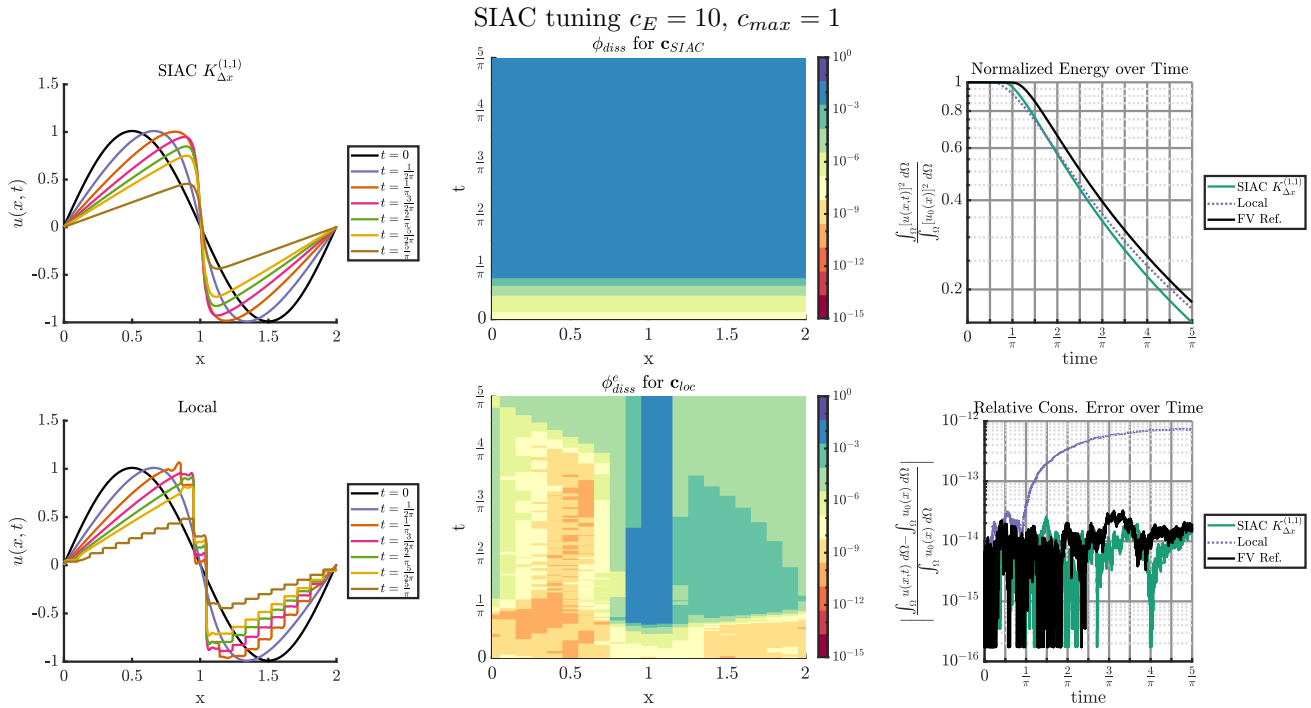


Figure 6.11 Comparison of Local and SIAC correction approaches to regularization. Here the tuning parameters used were those that qualitatively gave the best performance for the SIAC correction.

### 6.5.3 Two-dimensional regularization

We apply the same methodology to two-dimensional inviscid Burgers problem:

$$\frac{d}{dt}u + \nabla \cdot \left(\frac{u^2}{2}(1,1)\right) = 0,$$

on the domain $\Omega = [0,1]^2$ subject to periodic boundary conditions and initial data $u(x,y,0) = \sin(2\pi(x+y)) + 0.01$. This problem develops a shock in finite time at $t = \frac{1}{2\pi}$ along the lines $x + y = \frac{1}{4}, \frac{3}{4}$ which introduce spurious oscillations in high-order polynomial approximations.

In solving this problem, we wish to stress that the correction methodology does not necessitate a specific formulation of the DG method. As such, here we discretize the PDE in space by a modal DG-style approximation [34] using a Local Lax-Friedrichs flux, though time-stepping is still handled by the SSPRK(3,3) rRK method.

In Figure 6.12 we depict the solutions at the time of shock formation $t = \frac{1}{2\pi}$ and then at a later time $t = \frac{1}{\pi}$ using a $N_{elm} = 8 \times 8 = 64$ element mesh and a $p = 5$ $\mathbb{Q}_k$ tensor-product basis. For the regularized correction approach we consider the $K^{(1,1)}_{\Gamma,\sqrt{2}\Delta x}$ filter and chose dissipation parameters $c_E = 0.1$ and $c_{max} = 0.15$. Here the goal is to demonstrate improved solution quality under the regularization and so improved qualitative performance can perhaps be obtained by a more exhaustive searching of the tuning parameter space. In any case, the solutions with the energy-dissipative correction are of higher quality with oscillations of reduced magnitude as compared to the uncorrected case. This is most readily apparent in the $y = x$ cross-section plots in Figure 6.13. Note that both approximations conserve mass, and the energy-dissipative correction causes greater dissipation of energy (Figure 6.12 (Right)). Dissipation of energy observed for the uncorrected scheme could be a result of the exact integration of volume terms in the modal DG scheme, compared to under-integration in the DGSEM case.

### 6.5.4 Confirmation of high-order accuracy

In [2] it was shown that the local correction procedure would maintain the order of accuracy of the underlying spatial discretization method provided sufficiently high-order quadratures were applied. Here we verify the spatial accuracy of the one-dimensional DGSEM method for the inviscid Burgers equation under the correction. We also demonstrate the retention of the order of accuracy for corrected simulation of two-dimensional linear advection. Also in one-dimension we demonstrate numerically the energy decay caused by non-rRK time-stepping methods under the correction.

Focusing on the original one-dimensional test problem first, we consider the same initial data, but this time evolve the solution to a final time of $t = 1/(2\pi)$ which is prior to shock formation and so the solution is still smooth. We compute the $L^2-$ and $L^\infty-$errors for the approximation at 17 Gauss-Legendre points

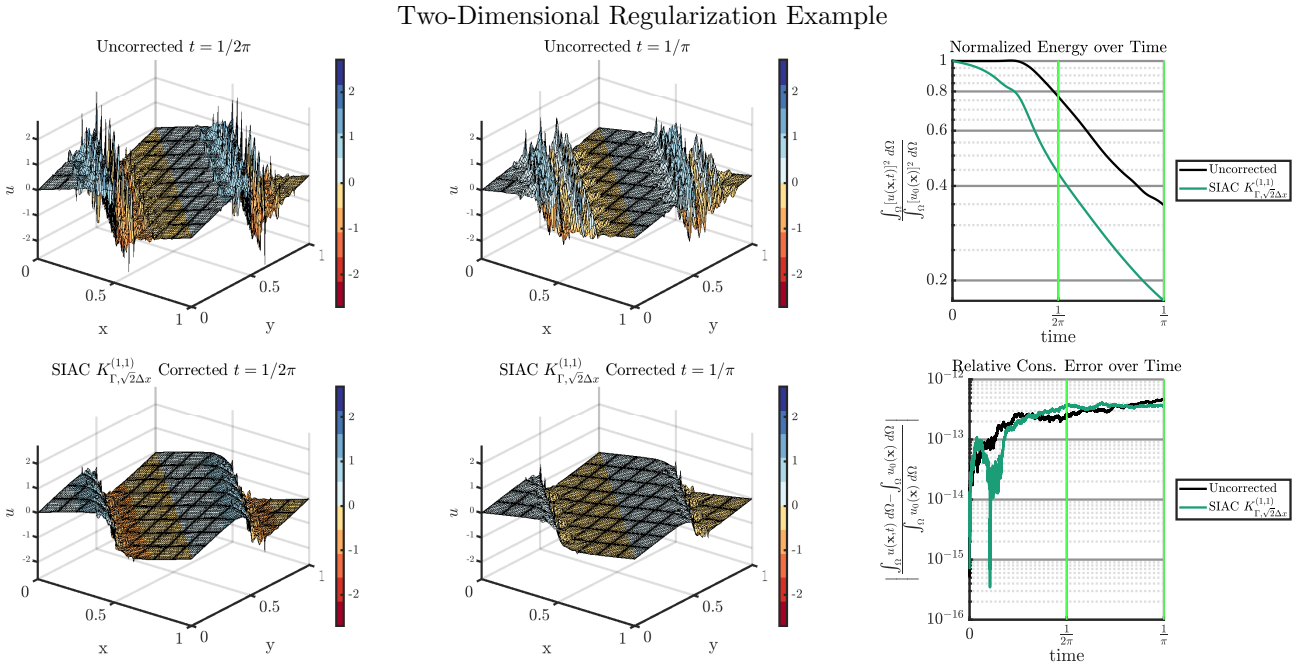


Figure 6.12 Comparison of uncorrected and corrected 2D Modal DG approximations of the periodic inviscid Burgers equation with $u_0(x, y) = \sin(2\pi(x + y)) + 0.01$ on $[0, 1]^2$ with $p = 5$ and $N_{elm} = 8 \times 8 = 64$ elements. For regularization we used $c_E = 0.1$ and $c_{max} = 0.15$.

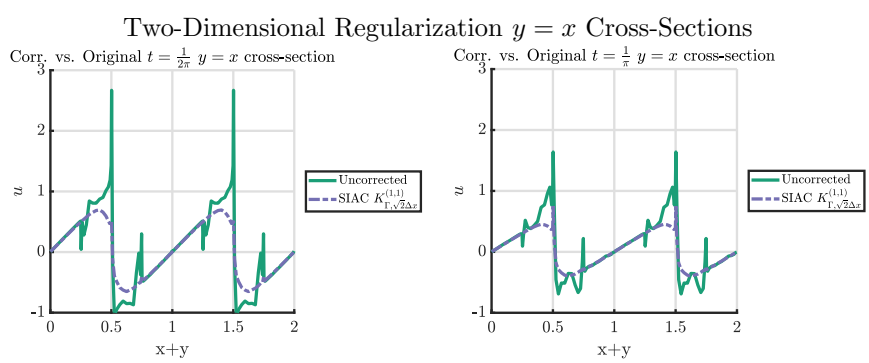


Figure 6.13 Cross-sections ($y = x$) of uncorrected and corrected 2D Modal DG approximations of the periodic inviscid Burgers equation with $u_0(x, y) = \sin(2\pi(x + y)) + 0.01$ on $[0, 1]^2$ with $p = 5$ and $N_{elm} = 8 \times 8 = 64$ elements. Solutions depicted at shock formation $t = \frac{1}{2\pi}$ and afterwards at $t = \frac{1}{\pi}$.

per element, where the exact solution is computed via Newton's method applied to the implicit solution $u(x,t) = u_0(x - ut)$. In Figure 6.14 and Table 6.1 we compare the order of accuracy of the uncorrected scheme against the local and SIAC corrected schemes. Only the orders of accuracy of the $L^2$-errors are provided as the $L^\infty$-error orders are similar. In general the uncorrected scheme exhibits order of accuracy between $h^p$ and $h^{p+1}$. The SIAC corrected schemes maintain this order of accuracy. The local approach performs similarly except for the $p = 1$ case when the method does not appear to converge even at order $h$. Furthermore, the SIAC correction does not necessitate a low moment and order formulation as the $K^{(3,2)}_{\Delta x}$ filter performs equally well as the $K^{(1,1)}_{\Delta x}$ filter.

Table 6.1 $L^2-$ orders of convergence for DGSEM solution to Burgers equation with initial data $\sin(\pi x) + 0.01$ on $[0, 2]$. Errors computed at a final time of $t = 1/2\pi$.

| Uncorrected $L^2-$Error Order | | | | |
|---|---|---|---|---|
| $N_0 \to 2N_0$ | $p = 1$ | $p = 2$ | $p = 3$ | $p = 4$ |
| $N_0 = 20$ | 1.3664 | 2.4936 | 2.9906 | 4.5180 |
| $N_0 = 40$ | 1.4108 | 2.5765 | 3.2344 | 4.4176 |
| $N_0 = 80$ | 1.4557 | 2.6864 | 3.2991 | 4.5704 |
| $N_0 = 160$ | 1.4749 | 2.7806 | 3.3551 | 4.7000 |
| Local Correction $L^2-$Error Order | | | | |
| $N_0 \to 2N_0$ | $p = 1$ | $p = 2$ | $p = 3$ | $p = 4$ |
| $N_0 = 20$ | 0.8521 | 2.4927 | 2.9913 | 4.5180 |
| $N_0 = 40$ | 0.4958 | 2.5745 | 3.2343 | 4.4176 |
| $N_0 = 80$ | 0.2446 | 2.6851 | 3.2990 | 4.5704 |
| $N_0 = 160$ | 0.1321 | 2.7798 | 3.3551 | 4.6996 |
| $K^{(3,2)}_{\Delta x}$ $L^2-$Error Order | | | | |
| $N_0 \to 2N_0$ | $p = 1$ | $p = 2$ | $p = 3$ | $p = 4$ |
| $N_0 = 20$ | 1.3289 | 2.4992 | 2.9907 | 4.5180 |
| $N_0 = 40$ | 1.3901 | 2.5847 | 3.2344 | 4.4176 |
| $N_0 = 80$ | 1.4481 | 2.6938 | 3.2991 | 4.5704 |
| $N_0 = 160$ | 1.4743 | 2.7861 | 3.3551 | 4.7000 |
| $K^{(1,1)}_{\Delta x}$ $L^2-$Error Order | | | | |
| $N_0 \to 2N_0$ | $p = 1$ | $p = 2$ | $p = 3$ | $p = 4$ |
| $N_0 = 20$ | 1.3389 | 2.4948 | 2.9906 | 4.5180 |
| $N_0 = 40$ | 1.3941 | 2.5768 | 3.2344 | 4.4176 |
| $N_0 = 80$ | 1.4485 | 2.6864 | 3.2991 | 4.5704 |
| $N_0 = 160$ | 1.4729 | 2.7806 | 3.3551 | 4.7000 |

In two dimensions we consider the a linear advection problem of $u(x, 0) = (\sin(\pi x) + 0.01)(\sin(\pi y) + 0.01)$ on the periodic domain $\Omega = [0, 2]^2$ with constant velocity field $(1, 1)$. Here we compute the errors from the exact solution $u(x,t) = u_0(x - t, y - t)$ at 36 Gauss-Legendre nodes per element. As depicted in Figure 6.15 the order of accuracy of the scheme is again maintained, while energy (not depicted) under the $K^{(1,1)}_{\Gamma,\sqrt{2}\Delta x}$ correction is conserved in a manner analogous to the one-dimensional case.

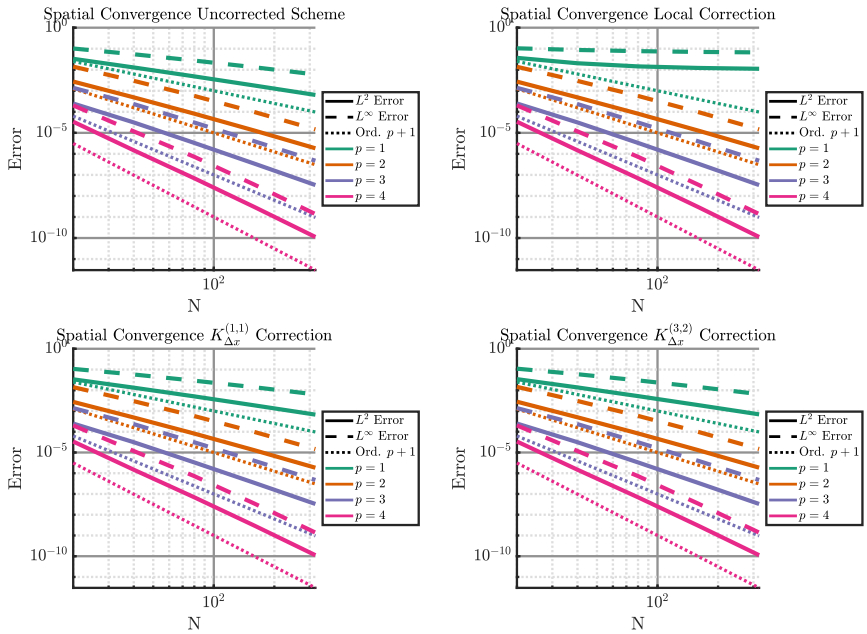


Figure 6.14 Spatial order of convergence results for the periodic one-dimensional Burgers problem with $u(x,0) = \sin(\pi x)$ on $\Omega = [0,2]$ prior to shock formation. Exact solution obtained by Newton iteration.

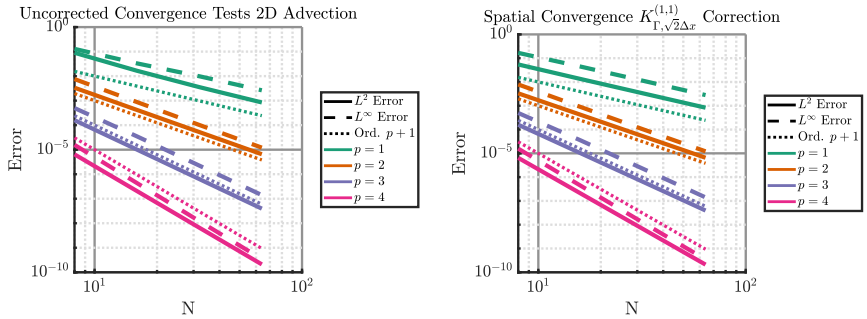


Figure 6.15 Spatial order of convergence results for the periodic two-dimensional constant advection problem. Here $N$ is the number of elements in a given Cartesian direction.

Consider now the effects of the time-stepping discretization on the relative energy error. As the rRK time-stepping method ensures energy conservation, we instead focus on the time-stepping errors incurred by non-energy conserving baseline RK methods. We consider here Forward Euler, the optimal 2- and 3-stage Strong Stability Preserving Runge-Kutta (SSPRK) methods, SSPRK(2,2) and SSPRK(3,3) respectively, and the standard 4-stage 4th order Runge-Kutta method RK(4,4) (see [125] and Appendix B for explicit formulations). In Figure 6.16 we plot the relative energy conservation errors with respect to the time step size $\Delta t$. We observe that under the uncorrected method, energy is not conserved by either the spatial or temporal discretization, and so of course no convergence is obtained under reductions in $\Delta t$. With respect to the spatially energy conserving discretizations, we observe convergence in relative energy error at the order of the time discretization, except for the SSPRK(2,2) method which exhibits 3rd order error. This phenomena was previously observed in [126].

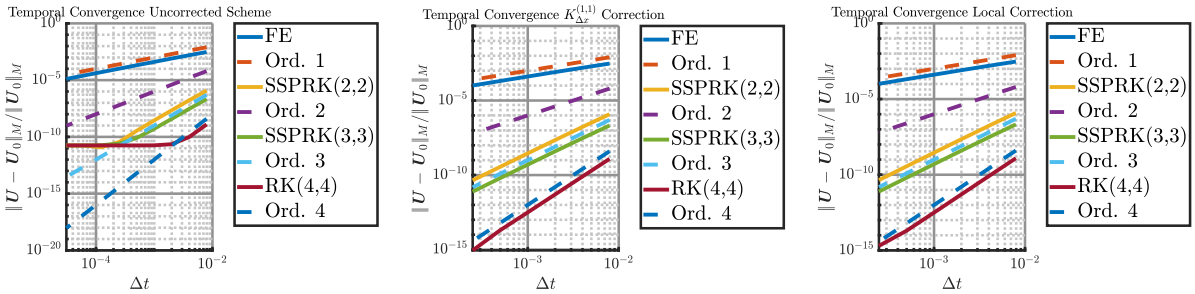


Figure 6.16 Relative energy change over time using energy non-conserving time-stepping methods.

## 6.6 Chapter Summary

We have demonstrated in this work that the local entropy correction approach of Abgrall can be viewed from the perspective of a filtering procedure, and generalized to include arbitrary conservative filters. The significant flexibility in the choice of filter allowed us to apply multi-element (L)SIAC filters for energy conservation of one- and two-dimensional inviscid Burgers equation where energy conservative time-stepping was guaranteed by application of relaxation Runge-Kutta methods. The SIAC-based correction was shown to have a smaller magnitude correction relative to the scheme residual then the original single-element correction approach, and was shown to have similar performance in terms of a subcell energy conservation metric. Furthermore, with addition of gradient based regularization term for shock capture, the SIAC approach was shown to produce qualitatively better solutions. Error of accuracy experiments demonstrate that the SIAC-based correction preserves the original order of accuracy of the scheme, even in the $p = 1$ case where the original approach did not. In the future we expect that SIAC-style correction procedures will prove useful in more complex scenarios such as hyperbolic systems,

enforcement of multiple target dynamics, and alternative entropies, as well for stabilization of meshless methods such as radial basis function schemes.

### 6.7 Co-Author Contributions

Ayaboe Edoh contributed to the introduction and provided conceptualization, editing, and supervision to this project.

# CHAPTER 7
# SUMMARY AND FUTURE WORK

In this thesis we have demonstrated the effectiveness of SIAC filtering methods in a wide variety of applications. For purposes of denoising, we have applied SIAC filters to PIC data in one dimension and been able to obtain substantial reduction in the quantity of information needed in computing Bohm speed estimate (see Chapter 3). While the complete integration of SIAC filters inside the PIC method is outside the scope of this project, it represents a promising future research direction as PIC methods have many similarities to finite element methods, such as approximation forms expressible in terms of low-order splines. In the future we wish to embed the SIAC filter denoisers insider the actual PIC simulation to denoise data on a time step to time step level. We would also like to investigate high order polynomial data variants of the dissipative-difference summation-by-parts filters [127] whose component difference operators innately possess stable boundary treatments.

In the context of adaptive approximation (Chapters 4 and 5), we have shown that SIAC filters can be used to enable error reduction under mesh refinement. Furthermore, we have demonstrated in Chapter 5 that the enhancement procedure generalizes to nonuniform meshes and can be applied for adaptive mesh refinement in the numerical solution of PDEs. In the future we would like to combine the enhancement procedure with truly adaptive numerical DG schemes which allow for coarsening and refinement of numerical approximation during discrete time evolution of hyperbolic equations. We would also like to better investigate the role of a data-informed adaptive kernel scaling. In filtering functions with sharp features, it is apparent that smearing of those features by the enhancement procedure results in poor approximations, and perhaps a filter width informed by the local data behavior would prove more effective.

Lastly, in Chapter 6 we developed a SIAC-based global filter for entropy correction of DG data. Of the projects in this thesis, here we were able to embed SIAC filters directly into the numerical scheme, rather than as a post-processor afterwards. Using SIAC corrections, we were able to develop energy conserving DG methods that made use of relaxation RK methods for fully discrete conservation under time evolution. We demonstrated experimentally that the correction improved solution robustness and maintained the order of accuracy of the underlying scheme. Demonstrating the flexibility of the approach, we also applied it to two-dimensional modal RKDG methods and obtained similar results using LSIAC filters. In the future we wish to expand this approach to more complicated scenarios such as hyperbolic systems and multiple target dynamics, to investigate alternative subcell measures of entropy error, and to apply these techniques for non-linear stabilization in meshless methods such as radial basis function schemes.

REFERENCES


[1] Jennifer K. Ryan, Xiaozhou Li, Robert M. Kirby, and Kees Vuik. One-sided position-dependent smoothness-increasing accuracy-conserving (SIAC) filtering over uniform and non-uniform meshes. *Journal of Scientific Computing*, 64:773–817, 9 2015. ISSN 0885-7474. doi: 10.1007/s10915-014-9946-6.

[2] Rémi Abgrall. A general framework to construct schemes satisfying additional conservation relations. application to entropy conservative and entropy dissipative schemes. *Journal of Computational Physics*, 372:640–666, 11 2018. ISSN 00219991. doi: 10.1016/j.jcp.2018.06.031.

[3] Rémi Abgrall, Philipp Öffner, and Hendrik Ranocha. Reinterpretation and extension of entropy correction terms for residual distribution and discontinuous Galerkin schemes: Application to structure preserving discretization. *Journal of Computational Physics*, 453:110955, 3 2022. ISSN 00219991. doi: 10.1016/j.jcp.2022.110955.

[4] Michael S. Mock and Peter D. Lax. The computation of discontinuous solutions of linear hyperbolic equations. *Communications on Pure and Applied Mathematics*, 31:423–430, 7 1978. ISSN 00103640. doi: 10.1002/cpa.3160310403.

[5] J. H. Bramble and A. H. Schatz. Higher order local accuracy by averaging in the finite element method. *Mathematics of Computation*, 31:94–111, 1977. ISSN 0025-5718. doi: 10.1090/S0025-5718-1977-0431744-9.

[6] Bernardo Cockburn, Mitchell Luskin, Chi-Wang Shu, and Endre Süli. Enhanced accuracy by post-processing for finite element methods for hyperbolic equations. *Mathematics of Computation*, 72:577–606, 11 2002. ISSN 0025-5718. doi: 10.1090/S0025-5718-02-01464-3.

[7] David Walfisch, Jennifer K. Ryan, Robert M. Kirby, and Robert Haimes. One-sided smoothness-increasing accuracy-conserving filtering for enhanced streamline integration through discontinuous fields. *Journal of Scientific Computing*, 38:164–184, 2 2009. ISSN 0885-7474. doi: 10.1007/s10915-008-9230-8.

[8] Paulien van Slingerland, Jennifer K. Ryan, and C. Vuik. Position-dependent smoothness-increasing accuracy-conserving (SIAC) filtering for improving discontinuous galerkin solutions. *SIAM Journal on Scientific Computing*, 33:802–825, 1 2011. ISSN 1064-8275. doi: 10.1137/100782188.

[9] X. Li, J.K. Ryan, R.M. Kirby, and C. Vuik. Smoothness-increasing accuracy-conserving (SIAC) filters for derivative approximations of discontinuous Galerkin (DG) solutions over nonuniform meshes and near boundaries. *Journal of Computational and Applied Mathematics*, 294:275–296, 3 2016. ISSN 03770427. doi: 10.1016/j.cam.2015.08.011.

[10] Jennifer Ryan and Chi-Wang Shu. On a one-sided post-processing technique for the discontinuous Galerkin methods. *Methods and Applications of Analysis*, 10:295–308, 2003. ISSN 10732772. doi: 10.4310/MAA.2003.v10.n2.a8.

[11] Liangyue Ji, Paulien van Slingerland, Jennifer K. Ryan, and Kees Vuik. Superconvergent error estimates for position-dependent smoothness-increasing accuracy-conserving (SIAC) post-processing of discontinuous Galerkin solutions. *Mathematics of Computation*, 83:2239–2262, 4 2014. ISSN 0025-5718. doi: 10.1090/S0025-5718-2014-02835-4.

[12] Hanieh Mirzaee, Liangyue Ji, Jennifer K. Ryan, and Robert M. Kirby. Smoothness-increasing accuracy-conserving (SIAC) postprocessing for discontinuous Galerkin solutions over structured triangular meshes. *SIAM Journal on Numerical Analysis*, 49:1899–1920, 1 2011. ISSN 0036-1429. doi: 10.1137/110830678.

[13] Hanieh Mirzaee, James King, Jennifer K. Ryan, and Robert M. Kirby. Smoothness-increasing accuracy-conserving filters for discontinuous Galerkin solutions over unstructured triangular meshes. *SIAM Journal on Scientific Computing*, 35:A212–A230, 1 2013. ISSN 1064-8275. doi: 10.1137/120874059.

[14] Xiaozhou Li, Jennifer K. Ryan, Robert M. Kirby, and Kees Vuik. Smoothness-increasing accuracy-conserving (SIAC) filtering for discontinuous Galerkin solutions over nonuniform meshes: Superconvergence and optimal accuracy. *Journal of Scientific Computing*, 81:1150–1180, 12 2019. ISSN 0885-7474. doi: 10.1007/s10915-019-00920-7.

[15] Mahsa Mirzargar, Ashok Jallepalli, Jennifer K. Ryan, and Robert M. Kirby. Hexagonal smoothness-increasing accuracy-conserving filtering. *Journal of Scientific Computing*, 73:1072–1093, 12 2017. ISSN 0885-7474. doi: 10.1007/s10915-017-0517-5.

[16] Xiaozhou Li. How to design a generic accuracy-enhancing filter for discontinuous Galerkin methods, 2020. URL https://arxiv.org/abs/2008.13170.

[17] Mahsa Mirzargar, Jennifer K. Ryan, and Robert M. Kirby. Smoothness-increasing accuracy-conserving (SIAC) filtering and quasi-interpolation: A unified view. *Journal of Scientific Computing*, 67:237–261, 4 2016. ISSN 0885-7474. doi: 10.1007/s10915-015-0081-9.

[18] Ashok Jallepalli and Robert M. Kirby. Efficient algorithms for the line-SIAC filter. *Journal of Scientific Computing*, 80:743–761, 8 2019. ISSN 0885-7474. doi: 10.1007/s10915-019-00954-x.

[19] Ashok Jallepalli, Robert Haimes, and Robert M. Kirby. Adaptive characteristic length for L-SIAC filtering of fem data. *Journal of Scientific Computing*, 79:542–563, 4 2019. ISSN 0885-7474. doi: 10.1007/s10915-018-0868-6.

[20] Jennifer K. Ryan and Julia Docampo-Sanchez. One-dimensional line SIAC filtering for multi-dimensions: Applications to streamline visualization. In *Numerical Methods for Flows: FEF 2017 Selected Contributions*, pages 145–154. Springer International Publishing, 2020. doi: 10.1007/978-3-030-30705-9_13.

[21] B. W. Wissink, G. B. Jacobs, J. K. Ryan, W. S. Don, and E. T. A. van der Weide. Shock regularization with smoothness-increasing accuracy-conserving dirac-delta polynomial kernels. *Journal of Scientific Computing*, 77:579–596, 10 2018. ISSN 0885-7474. doi: 10.1007/s10915-018-0719-5.

[22] Marvin Bohm, Sven Schermeng, Andrew R. Winters, Gregor J. Gassner, and Gustaaf B. Jacobs. Multi-element SIAC filter for shock capturing applied to high-order discontinuous Galerkin spectral element methods. *Journal of Scientific Computing*, 81:820–844, 11 2019. ISSN 0885-7474. doi: 10.1007/s10915-019-01036-8.

[23] Jennifer K. Ryan. Capitalizing on superconvergence for more accurate multi-resolution discontinuous Galerkin methods. *Communications on Applied Mathematics and Computation*, 7 2021. ISSN 2096-6385. doi: 10.1007/s42967-021-00121-w.

[24] Chi-Wang Shu. Discontinuous Galerkin method for time-dependent problems: Survey and recent developments, 2014.

[25] Daniele Antonio Di Pietro and Alexandre Ern. *Mathematical Aspects of Discontinuous Galerkin Methods*, volume 69. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-22979-4. doi: 10.1007/978-3-642-22980-0.

[26] Jan S. Hesthaven and Tim Warburton. *Nodal Discontinuous Galerkin Methods*, volume 54. Springer New York, 2008. ISBN 978-0-387-72065-4. doi: 10.1007/978-0-387-72067-8.

[27] Bernardo Cockburn. *An Introduction to the Discontinuous Galerkin Method for Convection-Dominated Problems*, pages 150–268. Springer Berlin Heidelberg, 1998. doi: 10.1007/BFb0096353.

[28] Jan S. Hesthaven. *Numerical Methods for Conservation Laws*. Society for Industrial and Applied Mathematics, 12 2018. ISBN 978-1-61197-509-3. doi: 10.1137/1.9781611975109.

[29] W. H. Reed and T. R. Hill. Triangular mesh methods for the neutron transport equation. Technical report, Los Alamos Scientific Lab., 1973.

[30] P. LASAINT and P.A. RAVIART. On a finite element method for solving the neutron transport equation. In Carl de Boor, editor, *Mathematical Aspects of Finite Elements in Partial Differential Equations*, pages 89–123. Academic Press, 1974. ISBN 978-0-12-208350-1. doi: https://doi.org/10.1016/B978-0-12-208350-1.50008-X. URL https://www.sciencedirect.com/science/article/pii/B978012208350150008X.

[31] Bernardo Cockburn and Chi-Wang Shu. The Runge-Kutta local projection P1-discontinuous-Galerkin finite element method for scalar conservation laws. In *1st National Fluid Dynamics Conference*. American Institute of Aeronautics and Astronautics, 1988. doi: 10.2514/6.1988-3797.

[32] Bernardo Cockburn and Chi-Wang Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. II. general framework. *Mathematics of computation*, 52: 411–435, 1989.

[33] Bernardo Cockburn, San-Yih Lin, and Chi-Wang Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: One-dimensional systems. *Journal of Computational Physics*, 84:90–113, 9 1989. ISSN 00219991. doi: 10.1016/0021-9991(89)90183-6.

[34] Bernardo Cockburn, Suchung Hou, and Chi-Wang Shu. The Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. IV. the multidimensional case. *Mathematics of Computation*, 54:545–581, 1990. ISSN 0025-5718. doi: 10.1090/S0025-5718-1990-1010597-0.

[35] Bernardo Cockburn and Chi-Wang Shu. The Runge–Kutta discontinuous Galerkin method for conservation laws V. *Journal of Computational Physics*, 141:199–224, 4 1998. ISSN 00219991. doi: 10.1006/jcph.1998.5892.

[36] Chi-Wang Shu. Discontinuous Galerkin methods: General approach and stability, 2008.

[37] Slimane Adjerid, Karen D. Devine, Joseph E. Flaherty, and Lilia Krivodonova. A posteriori error estimation for discontinuous Galerkin solutions of hyperbolic problems. *Computer Methods in Applied Mechanics and Engineering*, 191:1097–1112, 1 2002. ISSN 00457825. doi: 10.1016/S0045-7825(01)00318-8.

[38] Slimane Adjerid and Thomas C. Massey. Superconvergence of discontinuous Galerkin solutions for a nonlinear scalar hyperbolic problem. *Computer Methods in Applied Mechanics and Engineering*, 195: 3331–3346, 5 2006. ISSN 00457825. doi: 10.1016/j.cma.2005.06.017.

[39] Yang Yang and Chi-Wang Shu. Analysis of optimal superconvergence of discontinuous Galerkin method for linear hyperbolic equations. *SIAM Journal on Numerical Analysis*, 50:3110–3133, 1 2012. ISSN 0036-1429. doi: 10.1137/110857647.

[40] S.G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:674–693, 7 1989. ISSN 01628828. doi: 10.1109/34.192463.

[41] Yves Meyer. Ondelettes et fonctions splines. *Séminaire Équations aux dérivées partielles (Polytechnique) dit aussi "Séminaire Goulaouic-Schwartz"*, 1986.

[42] Bradley K. Alpert. A class of bases in $L^2$ for the sparse representation of integral operators. *SIAM Journal on Mathematical Analysis*, 24:246–262, 1 1993. ISSN 0036-1410. doi: 10.1137/0524016.

[43] Kai Schneider and Oleg V. Vasilyev. Wavelet methods in computational fluid dynamics. *Annual Review of Fluid Mechanics*, 42:473–503, 1 2010. ISSN 0066-4189. doi: 10.1146/annurev-fluid-121108-145637.

[44] Albert Cohen. *Numerical Analysis of Wavelet Methods*. Elsevier Science, 2003. ISBN 9780444511249. URL https://books.google.com/books?id=Dz9RnDItrAYC.

[45] Ingrid Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, 1 1992. ISBN 978-0-89871-274-2. doi: 10.1137/1.9781611970104.

[46] Ami Harten. Multiresolution algorithms for the numerical solution of hyperbolic conservation laws. *Communications on Pure and Applied Mathematics*, 48:1305–1342, 9 1993. ISSN 00103640. doi: 10.1002/cpa.3160481201.

[47] Siegfried Müller. *Adaptive Multiscale Schemes for Conservation Laws*, volume 27. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-44325-4. doi: 10.1007/978-3-642-18164-1.

[48] Albert Cohen, Sidi Mahmoud Kaber, Siegfried Müller, and Marie Postel. Fully adaptive multiresolution finite volume schemes for conservation laws. *Mathematics of Computation*, 72: 183–226, 12 2001. ISSN 0025-5718. doi: 10.1090/S0025-5718-01-01391-6.

[49] Ami Harten. Adaptive multiresolution schemes for shock computations. *Journal of Computational Physics*, 115:319–338, 12 1994. ISSN 00219991. doi: 10.1006/jcph.1994.1199.

[50] Remi Abgrall and Ami Harten. Multiresolution representation in unstructured meshes. *SIAM journal on Numerical Analysis*, 35:2128–2126, 12 1998.

[51] Gregory Beylkin and James M. Keiser. On the adaptive numerical solution of nonlinear partial differential equations in wavelet bases. *Journal of Computational Physics*, 132:233–259, 4 1997. ISSN 00219991. doi: 10.1006/jcph.1996.5562.

[52] B. Alpert, G. Beylkin, D. Gines, and L. Vozovoi. Adaptive solution of partial differential equations in multiwavelet bases. *Journal of Computational Physics*, 182:149–190, 10 2002. ISSN 00219991. doi: 10.1006/jcph.2002.7160.

[53] Rick Archibald, George Fann, and William Shelton. Adaptive discontinuous Galerkin methods in multiwavelets bases. *Applied Numerical Mathematics*, 61:879–890, 7 2011. ISSN 01689274. doi: 10.1016/j.apnum.2011.02.005.

[54] Nune Hovhannisyan, Siegfried Müller, and Roland Schäfer. Adaptive multiresolution discontinuous Galerkin schemes for conservation laws. *Mathematics of Computation*, 83:113–151, 1 2014.

[55] Nils Gerhard. *An Adaptive Multiresolution Discontinuous Galerkin Scheme for Conservation Laws*. PhD thesis, RWTH Aachen University, 2017.

[56] Nils Gerhard, Siegfried Müller, and Aleksey Sikstel. A wavelet-free approach for multiresolution-based grid adaptation for conservation laws. *Communications on Applied Mathematics and Computation*, 4: 108–142, 3 2022. ISSN 2096-6385. doi: 10.1007/s42967-020-00101-6.

[57] Andrew B. Shelton. *A Multi-Resolution Discontinuous Galerkin Method for Unsteady Compressible Flows*. PhD thesis, Georgia Institute of Technology, 8 2008.

[58] Nils Gerhard, Daniel Caviedes-Voullième, Siegfried Müller, and Georges Kesserwani. Multiwavelet-based grid adaptation with discontinuous Galerkin schemes for shallow water equations. *Journal of Computational Physics*, 301:265–288, 11 2015. ISSN 00219991. doi: 10.1016/j.jcp.2015.08.030.

[59] Nils Gerhard and Siegfried Müller. Adaptive multiresolution discontinuous Galerkin schemes for conservation laws: multi-dimensional case. *Computational and Applied Mathematics*, 35:321–349, 7 2016. ISSN 0101-8205. doi: 10.1007/s40314-014-0134-y.

[60] Vít Dolejší and Pavel Solin. hp-discontinuous Galerkin method based on local higher order reconstruction. *Applied Mathematics and Computation*, 279:219–235, 4 2016. ISSN 00963003. doi: 10.1016/j.amc.2016.01.024.

[61] Javier García Bautista, Marta de la Llave Plata, Vincent Couaillier, Michel Visonneau, and Kai Schneider. h-adaptation for high-order discontinuous Galerkin schemes built on local multiwavelet analysis. *Computers and Fluids*, 256:105844, 2023. ISSN 0045-7930. doi: https://doi.org/10.1016/j.compfluid.2023.105844.

[62] Javier Bautista. *Multiwavelet-based hp-adaptation for discontinuous Galerkin methods*. PhD thesis, École Centrale de Nantes, 2022.

[63] Eitan Tadmor. Entropy stability theory for difference approximations of nonlinear conservation laws and related time-dependent problems. *Acta Numerica*, 12:451–512, 5 2003. ISSN 0962-4929. doi: 10.1017/S0962492902000156.

[64] J. K. Ryan, C.-W. Shu, and H. Atkins. Extension of a post-processing technique for the discontinuous Galerkin method for hyperbolic equations with application to an aeroacoustic problem. *SIAM J. Sci. Comput.*, 26:821–843, 2005.

[65] Jennifer K. Ryan and Bernardo Cockburn. Local derivative post-processing for the discontinuous Galerkin method. *Journal of Computational Physics*, 228:8642–8664, 12 2009. ISSN 00219991. doi: 10.1016/j.jcp.2009.08.017.

[66] L. Ji, Y. Xu, and J. K. Ryan. Negative-order norm estimates for nonlinear hyperbolic conservation laws. *J. Sci. Comput.*, 54:269–310, 2013.

[67] Julia Docampo-Sánchez, Jennifer K. Ryan, Mahsa Mirzargar, and Robert M. Kirby. Multi-dimensional filtering: Reducing the dimension through rotation. *SIAM Journal on Scientific Computing*, 39:A2179–A2200, 1 2017. ISSN 1064-8275. doi: 10.1137/16M1097845.

[68] Carl de Boor. *A Practical Guide to Splines*. Springer-Verlag, revised edition edition, 1978. ISBN 978-0-387-95366-3.

[69] Larry Schumaker. *Spline Functions: Basic Theory*. Cambridge University Press, 2007. ISBN 9780511618994. doi: 10.1017/CBO9780511618994.

[70] Hanieh Mirzaee. *Smoothness-Increasing Accuracy-Conserving Filter (SIAC) for Discontinuous Galerkin Solutions*. PhD thesis, The University of Utah, 2013.

[71] Francis X. Giraldo. *An Introduction to Element-Based Galerkin Methods on Tensor-Product Bases*. Springer Cham, 1 edition, 2021. doi: https://doi.org/10.1007/978-3-030-55069-1.

[72] Douglas N. Arnold, Franco Brezzi, Bernardo Cockburn, and L. Donatella Marini. Unified analysis of discontinuous Galerkin methods for elliptic problems. *SIAM Journal on Numerical Analysis*, 39(5): 1749–1779, 2002. ISSN 00361429. URL http://www.jstor.org/stable/4101034.

[73] F. Bassi and S. Rebay. A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations. *Journal of Computational Physics*, 131(2): 267–279, 1997. ISSN 0021-9991. doi: https://doi.org/10.1006/jcph.1996.5572.

[74] Mohammad Alhawwary and Zhijian Wang. A combined-mode fourier analysis of DG methods for linear parabolic problems. *SIAM Journal on Scientific Computing*, 42(6):A3825–A3858, jan 2020. doi: 10.1137/20m1316962. URL https://doi.org/10.1137%2F20m1316962.

[75] Michal A. Kopera and Francis X. Giraldo. Analysis of adaptive mesh refinement for IMEX discontinuous Galerkin solutions of the compressible Euler equations with application to atmospheric simulations. *Journal of Computational Physics*, 275:92–117, 2014. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2014.06.026.

[76] Philip J. Davis and Philip Rabinowitz. *Methods of Numerical Integration*. Academic Press, second edition edition, 1984.

[77] Bernardo Cockburn and Chi-Wang Shu. Runge–Kutta discontinuous Galerkin methods for convection-dominated problems. *J. Sci. Comput.*, 16(3):173–261, jan 2002. ISSN 0885-7474. doi: 10.1023/A:1012873910884.

[78] Thomas P.Y. Yu, Krasmir Kolarov, and William Lynch. Barysymmetric multiwavelets on triangle. Technical report, Interval Research Corporation, 1997.

[79] Albert Cohen. *Numerical Analysis of Wavelet Methods*. Elsevier, Amsterdam, 2003.

[80] Fritz Keinert. *Wavelets and Multiwavelets*. Chapman & Hall/CRC Press, 2004.

[81] Thea Vuik. *Multiwavelets and outlier detection for troubled-cell indication in discontinuous Galerkin methods*. PhD thesis, Delft University of Technology, 2017.

[82] Javier Garcia Bautista, Marta de la Llave Plata, Vincent Couaillier, Michel Visonneau, and Kai Schneider. Local multiwavelet-based adaptation within a discontinuous Galerkin framework. In *AIAA Scitech 2021 Forum*. American Institute of Aeronautics and Astronautics, 2021. ISBN 978-1-62410-609-5. doi: 10.2514/6.2021-1780.

[83] Mathea J. Vuik and Jennifer K. Ryan. Multiwavelet troubled-cell indicator for discontinuity detection of discontinuous Galerkin schemes. *Journal of Computational Physics*, 270:138–160, 8 2014. ISSN 00219991. doi: 10.1016/j.jcp.2014.03.047.

[84] Nicholas A. Krall and Alvin W. Trivelpiece. *Principles of Plasma Physics*. McGraw-Hill Book Company, 1973.

[85] R. W. Hockney and J. W. Eastwood. *Computer Simulation Using Particles*. Taylor & Francis, 1988.

[86] Charles K. Birdsall and A. Bruce Langdon. *Plasma Physics via Computer Simulation*. McGraw-Hill Book Company, 1985.

[87] Y. Li, B. Srinivasan, Y. Zhang, and X.Z. Tang. Bohm criterion of plasma sheaths away from asymptotic limits. *Physical Review Letters*, 128:085002, 2 2022. doi: https://doi.org/10.1103/PhysRevLett.128.085002.

[88] David Bohm. The characteristics of electrical discharges in magnetic fields. *Qualitative Description of the Arc Plasma in a Magnetic Field*, 1949.

[89] Jörg Peters. General spline filters for discontinuous Galerkin solutions. *Computers & Mathematics with Applications*, 70(5):1046–1050, 2015. ISSN 0898-1221. doi: https://doi.org/10.1016/j.camwa.2015.06.034. URL https://www.sciencedirect.com/science/article/pii/S089812211500320X.

[90] Dang-Manh Nguyen and Jörg Peters. Nonuniform discontinuous Galerkin filters via shift and scale. *SIAM Journal on Numerical Analysis*, 54(3):1401–1422, 2016. doi: 10.1137/15M1053220.

[91] J. Docampo-Sánchez, G.B. Jacobs, X. Li, and J.K. Ryan. Enhancing accuracy with a convolution filter: What works and why! *Computers & Fluids*, 213:104727, 12 2020. ISSN 00457930. doi: 10.1016/j.compfluid.2020.104727.

[92] Philip L Pritchett. Particle-in-cell simulations of magnetosphere electrodynamics. *IEEE transactions on plasma science*, 28(6):1976–1990, 2000.

[93] John P Verboncoeur. Particle simulation of plasmas: review and advances. *Plasma Physics and Controlled Fusion*, 47(5A):A231, 2005.

[94] Kevin J Bowers, BJ Albright, L Yin, B Bergen, and TJT Kwan. Ultrahigh performance three-dimensional electromagnetic relativistic kinetic plasma simulation. *Physics of Plasmas*, 15(5): 055703, 2008.

[95] Zehua Guo and Xian-Zhu Tang. Ambipolar transport via trapped-electron whistler instability along open magnetic field lines. *Physical Review Letters*, 109(13):135005, 2012.

[96] Yanzeng Zhang and Xian-Zhu Tang. On the collisional damping of plasma velocity space instabilities. *Physics of Plasmas*, 30(3):030701, 2023.

[97] Xian-Zhu Tang and Zehua Guo. Critical role of electron heat flux on Bohm criterion. *Physics of Plasmas*, 23(12):120701, 2016. doi: 10.1063/1.4971808. URL https://doi.org/10.1063/1.4971808.

[98] Matthew J. Picklo and Jennifer K. Ryan. Enhanced multiresolution analysis for multidimensional data utilizing line filtering techniques. *SIAM Journal on Scientific Computing*, 44(4):A2628–A2650, 2022. doi: 10.1137/21M144013X.

[99] Jorge L. Díaz Calle, Philippe R. B. Devloo, and Sônia M. Gomes. Wavelets and adaptive grids for the discontinuous Galerkin method. *Numerical Algorithms*, 39:143–154, 7 2005. ISSN 1017-1398. doi: 10.1007/s11075-004-3626-9.

[100] Jennifer K. Ryan. Capitalizing on superconvergence for more accurate multi-resolution discontinuous Galerkin methods. *Communications in Applied Mathematics and Computation*, 2021.

[101] C. Geuzaine and J.-F. Remacle. Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11): 1309–1331, 2009. doi: https://doi.org/10.1002/nme.2579.

[102] Luis G. Reyna and Michael J. Ward. On the exponentially slow motion of a viscous shock. *Communications on Pure and Applied Mathematics*, 48(2):79–120, 1995. doi: https://doi.org/10.1002/cpa.3160480202.

[103] Mark H. Carpenter and Christopher A. Kennedy. Fourth-order 2N-storage Runge-Kutta schemes. Technical report, NASA, 1994.

[104] Randall J. LeVeque. *Numerical Methods for Conservation Laws*. Springer Basel AG, 1993. ISBN 9783764327231.

[105] Gregor J. Gassner. A skew-symmetric discontinuous Galerkin spectral element discretization and its relation to SBP-SAT finite difference methods. *SIAM Journal on Scientific Computing*, 35(3): A1233–A1253, 2013. doi: 10.1137/120890144. URL https://doi.org/10.1137/120890144.

[106] A. Harten, J. M. Hyman, P. D. Lax, and B. Keyfitz. On finite-difference approximations and entropy conditions for shocks. *Communications on Pure and Applied Mathematics*, 29(3):297–322, 1976. doi: https://doi.org/10.1002/cpa.3160290305.

[107] Guang Shan Jiang and Chi-Wang Shu. On a cell entropy inequality for discontinuous Galerkin methods. *Mathematics of Computation*, 62:531–538, 1994. ISSN 0025-5718. doi: 10.1090/S0025-5718-1994-1223232-7.

[108] Rémi Abgrall and Fatemeh Nassajian Mojarrad. Conservative scheme compatible with some other conservation laws: Conservation of the local angular momentum. *Computers & Fluids*, 247:105663, 2022. ISSN 0045-7930. doi: https://doi.org/10.1016/j.compfluid.2022.105663.

[109] Rémi Abgrall, Saray Busto, and Michael Dumbser. A simple and general framework for the construction of thermodynamically compatible schemes for computational fluid and solid mechanics. *Applied Mathematics and Computation*, 440:127629, 2023. ISSN 0096-3003. doi: https://doi.org/10.1016/j.amc.2022.127629.

[110] Hendrik Ranocha, Jan Glaubitz, Philipp Öffner, and Thomas Sonar. Stability of artificial dissipation and modal filtering for flux reconstruction schemes using summation-by-parts operators. *Applied Numerical Mathematics*, 128:1–23, 2018. ISSN 0168-9274. doi: https://doi.org/10.1016/j.apnum.2018.01.019.

[111] Ayaboe Edoh. Conservative correction procedures utilizing artificial dissipation operators. Pre-Print, 2023.

[112] Randall J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2002. doi: 10.1017/CBO9780511791253.

[113] Philipp Öffner. *Approximation and stability properties of numerical methods for hyperbolic conservation laws*. Springer Nature, 2023.

[114] Mathworks. MATLAB lsqlin documentation. https://www.mathworks.com/help/optim/ug/lsqlin.html, 2023. Accessed: 10-16-2023.

[115] Nick Gould and Philippe L. Toint. Preprocessing for quadratic programming. *Math. Program.*, 100 (1):95–132, may 2004. ISSN 0025-5610.

[116] Mathworks. MATLAB quadratic programming algorithms documentation. https://www.mathworks.com/help/optim/ug/quadratic-programming-algorithms.html, 2023. Accessed: 10-16-2023.

[117] J. Glaubitz, A. C. Nogueira, J. L. S. Almeida, R. F. Cantão, and C. A. C. Silva. Smooth and compactly supported viscous sub-cell shock capturing for discontinuous Galerkin methods. *Journal of Scientific Computing*, 79:249–272, 2019. ISSN 1. doi: 10.1007/s10915-018-0850-3.

[118] Jean-Luc Guermond, Richard Pasquetti, and Bojan Popov. Entropy viscosity method for nonlinear conservation laws. *Journal of Computational Physics*, 230(11):4248–4267, 2011. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2010.11.043. Special issue High Order Methods for CFD Problems.

[119] Andrés M. Rueda-Ramírez, Will Pazner, and Gregor J. Gassner. Subcell limiting strategies for discontinuous Galerkin spectral element methods. *Computers & Fluids*, 247:105627, 2022. ISSN 0045-7930. doi: https://doi.org/10.1016/j.compfluid.2022.105627.

[120] Jan S. Hesthaven. *Numerical Methods for Conservation Laws*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2018. doi: 10.1137/1.9781611975109. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611975109.

[121] David I. Ketcheson. Relaxation Runge–Kutta methods: Conservation and stability for inner-product norms. *SIAM Journal on Numerical Analysis*, 57(6):2850–2870, 2019. doi: 10.1137/19M1263662. URL https://doi.org/10.1137/19M1263662.

[122] Hendrik Ranocha, Mohammed Sayyari, Lisandro Dalcin, Matteo Parsani, and David I. Ketcheson. Relaxation Runge–Kutta methods: Fully discrete explicit entropy-stable schemes for the compressible Euler and Navier–Stokes equations. *SIAM Journal on Scientific Computing*, 42(2):A612–A638, 2020. doi: 10.1137/19M1263480.

[123] A. W. Vreman. The adjoint filter operator in large-eddy simulation of turbulent flow. *Physics of Fluids*, 16(6):2012–2022, 05 2004. ISSN 1070-6631. doi: 10.1063/1.1710479. URL https://doi.org/10.1063/1.1710479.

[124] L.N. Trefethen and M. Embree. *Spectra and Pseudospectra: The Behavior of Nonnormal Matrices and Operators*. Princeton University Press, 2005. ISBN 9780691119465. URL https://books.google.com/books?id=7gIbT-Y7-AIC.

[125] Jan S. Hesthaven, Sigal Gottlieb, and David Gottlieb. *Spectral Methods for Time-Dependent Problems*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2007. doi: 10.1017/CBO9780511618352.

[126] F. Capuano, G. Coppola, L. Rández, and L. de Luca. Explicit Runge–Kutta schemes for incompressible flow with improved energy-conservation properties. *Journal of Computational Physics*, 328:86–94, 2017. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2016.10.040.

[127] Ayaboe Edoh and Venke Sankaran. Boundary prescriptions for spectrally-tunable discrete filters, 2019.

[128] M. Abramowitz and I.A. Stegun. *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables*. Dover Publications, 1965. ISBN 9780486612720. URL https://books.google.com/books?id=MtU8uP7XMvoC.

# APPENDIX A

# LEGENDRE POLYNOMIALS

The Legendre polynomial are a special case of the Jacobi polynomials orthogonal over the interval $[-1, 1]$ with respect to the weight function 1. They can be defined by the Rodrigues' formula [128]:

$$P_k(x) = \frac{1}{2^k k!} \frac{d^k (x^2 - 1)^k}{dx^k}.$$

The usefulness of these polynomials is realized by their orthogonality over the interval $[-1, 1]$:

$$\int_{-1}^{1} P_k(x) P_r(x)\, dx = \frac{2}{2k+1} \delta_{kr}$$

where $\delta_{mr}$ is the Kronecker delta equaling 0 for $m \neq r$ and 1 if $m = r$. The usefulness of these polynomials extends beyond the interval $[-1, 1]$ as one can define a change of variables $\xi = \frac{2}{b-a}\left(x - \frac{a+b}{2}\right)$ which will map $[a, b] \to [-1, 1]$. Defining the shifted Legendre polynomials by $\tilde{P}_k(x) = P_k(\xi(x))$ yields an orthogonal polynomial basis for $[a, b]$:

$$\int_{a}^{b} \tilde{P}_k(x) \tilde{P}_r(x)\, dx = \frac{b-a}{2} \int_{-1}^{1} P_k(\xi) P_r(\xi)\, d\xi \tag{A.1}$$

$$= \frac{b-a}{2k+1} \delta_{kr}. \tag{A.2}$$

# APPENDIX B

# RUNGE-KUTTA SCHEMES

In this appendix we provide the Butcher Tableau's of the RK schemes used in this work, as well as the formulation of the 5-stage 4th order low storage RK method. The Butcher Tableau's are given in the form

$$\begin{array}{c|ccc} c_1 & a_{11} & \dots & a_{1s} \\ \vdots & \vdots & & \vdots \\ c_s & a_{s1} & \dots & a_{ss} \\ \hline & b_1 & \dots & b_s \end{array}.$$

- Forward Euler

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}.$$

- SSPRK(2,2) also known as Heun's method

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}.$$

- SSPRK(3,3)

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{4} & 0 \\ \hline & \frac{1}{6} & \frac{1}{6} & \frac{2}{3} \end{array}.$$

- RK(4,4)

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}.$$

- The low storage 5-stage 4th order RK method (see [26, 103]) is given by:

$$
\begin{aligned}
&p^{(0)} = u^n, \\
&\qquad \text{For } i = 1, \ldots, 5 \quad : \quad \begin{cases} k_i & = a_i k_{i-1} + \Delta t f(p^{(i-1)}, t^n + c_i \Delta t) \\ p^{(i)} & = p^{(i-1)} + b_i k_i \end{cases}, \\
&u^{n+1} = p^{(5)},
\end{aligned}
$$

where the coefficients are given in Table B.1.

Table B.1 Coefficients for the low storage 5-stage 4th order RK method.

| $i$ | $a_i$ | $b_i$ | $c_i$ |
|---|---|---|---|
| 1 | 0 | $\frac{1432997174477}{9575080441755}$ | 0 |
| 2 | $-\frac{567301805773}{1357537059087}$ | $\frac{5161836677717}{13612068292357}$ | $\frac{1432997174477}{9575080441755}$ |
| 3 | $-\frac{2404267990393}{2016746695238}$ | $\frac{1720146321549}{2090206949498}$ | $\frac{2526269341429}{6820363962896}$ |
| 4 | $-\frac{3550918686646}{2091501179385}$ | $\frac{3134564353537}{4481467310338}$ | $\frac{2006345519317}{3224310063776}$ |
| 5 | $-\frac{1275806237668}{842570457699}$ | $\frac{2277821191437}{14882151754819}$ | $\frac{2802321613138}{2924317926251}$ |

# APPENDIX C

# CO-AUTHOR PERMISSIONS

Permission to include Chapter 3 as part of this thesis has been obtained from co-authors who are not on the thesis committee; emails containing the permission statements are given below.

10/19/23, 8:37 AM Mail - Matthew Picklo (Student) - Outlook

[EXTERNAL] Re: Approval for inclusion of PIC-SIAC work in Thesis

Zhang, Yanzeng <yzengzhang@lanl.gov>
Thu 10/12/2023 10:10 AM
To:Matthew Picklo (Student) <mpicklo@mines.edu>

**CAUTION:** This email originated from outside of the Colorado School of Mines organization. Do not click on links or open attachments unless you recognize the sender and know the content is safe.

Hi Matt,

You have my approval to use the materials in the paper "Denoising Particle-In-Cell Data via Smoothness-Increasing".

Congratulations and good luck with your defense!

Yanzeng

---

**From:** Matthew Picklo (Student) <mpicklo@mines.edu>
**Sent:** Thursday, October 12, 2023 10:02 AM
**To:** Tang, Qi <qtang@lanl.gov>; Zhang, Yanzeng <yzengzhang@lanl.gov>; Tang, Xianzhu <xtang@lanl.gov>; jryan@kth.se <jryan@kth.se>
**Subject:** [EXTERNAL] Approval for inclusion of PIC-SIAC work in Thesis

Good morning everyone,

I hope you all are doing well.

I am emailing, because I am writing up my thesis, which I will defend in early November.
I would like to include some of the work I did in the PIC-SIAC paper in my thesis.

Since we submitted it to a journal, the university here requires that I get approval from co-authors to include any materials from it in my thesis.

If you are okay with me using the work I did in my thesis, you just need to reply to this email saying I have approval to do so, and I will then include that approval in the Appendices.

I will make sure to send you a copy of the final version of the thesis as well.

Thank you so much for your time
Matt

https://outlook.office.com/mail/id/AAQkAGJkYWUyY2RjLTg2YzItNGQyYi1hMWI2LTdiNDJkNTQzMGM2YgAQAFuWUwiKbBVJm5RqLs8WbmQ%3D 1/1

10/19/23, 8:38 AM Mail - Matthew Picklo (Student) - Outlook

[EXTERNAL] Re: Approval for inclusion of PIC-SIAC work in Thesis

Tang, Qi <qtang@lanl.gov>
Thu 10/12/2023 10:52 AM
To:Matthew Picklo (Student) <mpicklo@mines.edu>;Zhang, Yanzeng <yzengzhang@lanl.gov>;Tang, Xianzhu <xtang@lanl.gov>; jryan@kth.se <jryan@kth.se>

**CAUTION:** This email originated from outside of the Colorado School of Mines organization. Do not click on links or open attachments unless you recognize the sender and know the content is safe.

Sure, feel free to do that. Good luck with your thesis and defense. I am sure it will be excellent.

Qi

---

**From:** Matthew Picklo (Student) <mpicklo@mines.edu>
**Sent:** Thursday, October 12, 2023 10:02 AM
**To:** Tang, Qi <qtang@lanl.gov>; Zhang, Yanzeng <yzengzhang@lanl.gov>; Tang, Xianzhu <xtang@lanl.gov>; jryan@kth.se <jryan@kth.se>
**Subject:** [EXTERNAL] Approval for inclusion of PIC-SIAC work in Thesis

Good morning everyone,

I hope you all are doing well.

I am emailing, because I am writing up my thesis, which I will defend in early November.
I would like to include some of the work I did in the PIC-SIAC paper in my thesis.

Since we submitted it to a journal, the university here requires that I get approval from co-authors to include any materials from it in my thesis.

If you are okay with me using the work I did in my thesis, you just need to reply to this email saying I have approval to do so, and I will then include that approval in the Appendices.

I will make sure to send you a copy of the final version of the thesis as well.

Thank you so much for your time
Matt

https://outlook.office.com/mail/id/AAQkAGJkYWUyY2RjLTg2YzItNGQyYi1hMWI2LTdiNDJkNTQzMGM2YgAQAFuWUwiKbBVJm5RqLs8WbmQ%3D 1/1

10/19/23, 8:38 AM Mail - Matthew Picklo (Student) - Outlook

[EXTERNAL] Re: Approval for inclusion of PIC-SIAC work in Thesis

Tang, Xianzhu <xtang@lanl.gov>
Thu 10/12/2023 10:20 AM
To:Matthew Picklo (Student) <mpicklo@mines.edu>;Tang, Qi <qtang@lanl.gov>;Zhang, Yanzeng <yzengzhang@lanl.gov>; jryan@kth.se <jryan@kth.se>

**CAUTION:** This email originated from outside of the Colorado School of Mines organization. Do not click on links or open attachments unless you recognize the sender and know the content is safe.

Sure, you have my consent to include the PIC-SAIC work. Good luck with your thesis writing and oral defense. --Xianzhu

Get Outlook for iOS

---

**From:** Matthew Picklo (Student) <mpicklo@mines.edu>
**Sent:** Thursday, October 12, 2023 10:02:36 AM
**To:** Tang, Qi <qtang@lanl.gov>; Zhang, Yanzeng <yzengzhang@lanl.gov>; Tang, Xianzhu <xtang@lanl.gov>; jryan@kth.se <jryan@kth.se>
**Subject:** [EXTERNAL] Approval for inclusion of PIC-SIAC work in Thesis

Good morning everyone,

I hope you all are doing well.

I am emailing, because I am writing up my thesis, which I will defend in early November.
I would like to include some of the work I did in the PIC-SIAC paper in my thesis.

Since we submitted it to a journal, the university here requires that I get approval from co-authors to include any materials from it in my thesis.

If you are okay with me using the work I did in my thesis, you just need to reply to this email saying I have approval to do so, and I will then include that approval in the Appendices.

I will make sure to send you a copy of the final version of the thesis as well.

Thank you so much for your time
Matt

https://outlook.office.com/mail/id/AAQkAGJkYWUyY2RjLTg2YzItNGQyYi1hMWI2LTdiNDJkNTQzMGM2YgAQAFuWUwiKbBVJm5RqLs8WbmQ%3D 1/1

Figure C.1 Co-author approvals for inclusion of Chapter 3.

# APPENDIX D

# COPYRIGHT PERMISSIONS

## D.1 Journal Article Permissions

Permission for use from the SIAM Journal on Scientific Computing, published by the Society for Industrial and Applied Mathematics (SIAM), is given below for the content in Chapter 4 containing materials in an altered form from the article:

M. J. Picklo and J. K. Ryan, Enhanced Multiresolution Analysis for Multidimensional Data utilizing Line Filtering Techniques, *SIAM Journal on Scientific Computing*, 44 (2022), pp. A2628–A2650, https://doi.org/10.1137/21M144013X.

## [EXTERNAL] Permission

Kelly Thomas <Thomas@siam.org>

Fri 10/20/2023 9:48 AM

To:Matthew Picklo (Student) <mpicklo@mines.edu>

**CAUTION:** This email originated from outside of the Colorado School of Mines organization. Do not click on links or open attachments unless you recognize the sender and know the content is safe.

Dear Mr. Picklo:

You may reuse material from your article "Enhanced Multiresolution Analysis for Multidimensional Data Utilizing Line Filtering Techniques" in your thesis. Please acknowledge the original publication, using the complete bibliographic information.

Sincerely,

**Kelly Thomas**
Managing Editor
Society for Industrial and Applied Mathematics
3600 Market Street - 6th Floor
Philadelphia, PA 19104
thomas@siam.org / (267) 350-6387

Did you know? SIAM regular members can nominate up to two students per year for free membership. Nominate now and remember to renew your 2024 membership if you haven't yet!



Figure D.1 Reproduction approval from SIAM.

APPENDIX E

SIAC-MRA TRANSITION MATRICES

## E.1 SIAC-MRA: Filtered Basis Functions, Coefficients, and Transition Matrices:

In this appendix we provide the SIAC-MRA filtered basis functions and associated coefficient derived in [23] and in [98]. We also detail the construction of transition matrices for performing the enhancement procedure over uniform meshes. Figures are provided delineating the different domains of definition for these functions. Denote by $\zeta$ the mapping of the filtering point to the reference element $[-1, 1]$, i.e. $\zeta_x = \frac{2}{h}(x - x_i)$, $\zeta_y = \frac{2}{h}(y - y_j)$, $\zeta_z = \frac{2}{h}(z - z_k)$. Here $u^f$ will sometimes be used to denote the fine mesh modal coefficients computed by the enhancement procedure.

## E.2 One-dimensional SIAC-MRA

We first recall that the filtered approximation given in Ryan, (2021) is of the form

$$u_h^\star(x) = \begin{cases} \sum_{m=0}^{p+1} b_m^j \chi_L^m(\zeta_x), & \zeta_x \in (-1, 0) \\ \sum_{m=0}^{p+1} \tilde{b}_m^j \chi_R^m(\zeta_x), & \zeta_x \in (0, 1) \end{cases},$$

with the $\chi$ functions and coefficients given in Table E.1.

Table E.1 Table of basis functions and associated coefficients resulting from the analytical evaluation of the convolution in $u_h^\star(x)$. Here $P_k$ denote Legendre polynomials.

| $\mathbf{k}$ | $\mathbf{b_k^j}$ | $\mathbf{\tilde{b}_k^j}$ |
|---|---|---|
| $\mathbf{0}$ | $\frac{\sqrt{2}}{2}\left\{\sum_{\gamma=-p}^{p} c_\gamma u_0^{j+\gamma}\right\}$ | $\frac{\sqrt{2}}{2}\left\{\sum_{\gamma=-p}^{p} c_\gamma u_0^{j+\gamma}\right\}$ |
| $\mathbf{1}$ | $\frac{\sqrt{2}}{4}\left\{\sum_{\gamma=-p}^{p} c_\gamma (u_0^{j+\gamma} - u_0^{j+\gamma-1})\right\}$ | $\frac{\sqrt{2}}{4}\left\{\sum_{\gamma=-p}^{p} c_\gamma (u_0^{j+\gamma+1} - u_0^{j+\gamma})\right\}$ |
| $\mathbf{k > 1}$ | $\frac{1}{2}\left\{\sum_{\gamma=-p}^{p} c_\gamma (u_k^{j+\gamma} - u_k^{j+\gamma-1})\right\}$ | $\frac{1}{2}\left\{\sum_{\gamma=-p}^{p} c_\gamma (u_k^{j+\gamma+1} - u_k^{j+\gamma})\right\}$ |
| | $\chi_{\mathbf{L}}^{\mathbf{k}}(\zeta)$ | $\chi_{\mathbf{R}}^{\mathbf{k}}(\zeta)$ |
| $\mathbf{0}$. | $1$ | $1$ |
| $\mathbf{1}$. | $\zeta$ | $\zeta$ |
| $\mathbf{k > 1}$. | $\frac{\sqrt{k-1/2}}{2k-1}\Big[P_k(1+\zeta) - P_{k-2}(1+\zeta)\Big]$ | $\frac{\sqrt{k-1/2}}{2k-1}\Big[P_k(\zeta-1) - P_{k-2}(\zeta-1)\Big]$ |

### E.2.1 One-dimensional Transition Matrices

Suppose $\bar{x} \in [x_{j-1/2}, x_j)$, $u_h(x)|_{I_j} = \sum_{kx=0}^{p} u_{kx}^j \phi_{kx}(x)$, and note that as previously established

$$u_h^*(\bar{x}) = \sum_{m=0}^{p+1} b_j^{(m)} \chi_L^{(m)}(\zeta)$$

where

$$b_j^{(0)} = \frac{\sqrt{2}}{2}\left\{ \sum_{\gamma=-p}^{p} c_\gamma u_0^{i+\gamma} \right\},$$
$$b_j^{(1)} = \frac{\sqrt{2}}{4}\left\{ \sum_{\gamma=-p}^{p} c_\gamma \Big( - u_0^{j+\gamma-1} + u_0^{j+\gamma} \Big) \right\},$$

and for $m \geq 1$

$$b_j^{(m+1)} = \frac{1}{2}\Big[ - c_{-p} u_m^{j-(p+1)} + \sum_{\gamma=-p}^{p-1} u_m^{j+\gamma}\Big(c_\gamma - c_{\gamma+1}\Big) + c_p u_m^{j+p}\Big],$$

with corresponding basis functions

$$\chi_L^{(0)}(\zeta) = 1,$$
$$\chi_L^{(1)}(\zeta) = \zeta,$$

and for $m > 1$

$$\chi_L^{(m)}(\zeta) = \frac{\sqrt{m-1/2}}{2m-1}\Big[P_m(1+\zeta) - P_{m-2}(1+\zeta)\Big]$$

Define the vector of modal coefficients over the mesh by

$$\vec{u} = \begin{bmatrix} u_0^1 & \dots & u_p^1 & | & \dots & | & u_0^N & \dots & u_p^N \end{bmatrix}^T.$$

For filtering in the left half of the $j_{th}$ element we only need $2p+2$ elements worth of modal values, with the supposition that periodic boundary conditions guarantee their existence. For generalization to the right-sided case, lets order the vector containing the necessary modes by

$$\tilde{u} = \begin{bmatrix} u_0^{j-(p+1)} & \dots & u_p^{j-(p+1)} & | & \dots & | & u_0^{j+(p+1)} & \dots & u_p^{j+(p+1)} \end{bmatrix}^T.$$

To obtain this vector, we multiply $u$ by the $(2p+3)(p+1) \times N(p+1)$ matrix

$$S^j = \begin{bmatrix} \omega_j(1,1)\mathbb{I}_{(p+1)\times(p+1)} & \dots & \omega_j(1,N)\mathbb{I}_{(p+1)\times(p+1)} \\ \vdots & & \vdots \\ \omega_j(2p+3,1)\mathbb{I}_{(p+1)\times(p+1)} & \dots & \omega_j(2p+3,N)\mathbb{I}_{(p+1)\times(p+1)} \end{bmatrix}^T,$$

where

$$\omega_j(m,r) = \begin{cases} 1, & j \neq N \ \& \ p+2-m+r = j \ mod(N) \\ 1, & j = N \ \& \ p+2-m+r = j \text{ or } 0 \\ 0, & Else \end{cases},$$

for $m = 1 : (2p+3)$ and $r = 1 : N$. Since $\tilde{u} = S^j u$, we now need to use the definitions of $b_j^m$ to obtain a matrix $A_L$ s.t. $\vec{b}_j = A_L \tilde{u}$, for

$$\vec{b}_j = \begin{bmatrix} b_j^0 & \dots & b_j^{p+1} \end{bmatrix}^T.$$

This matrix is given by

$$A_L = \begin{bmatrix} \vec{\ell}_0 \\ \vec{\ell}_1 \\ \vdots \\ \vec{\ell}_{p+1} \end{bmatrix}^T,$$

for

$$\vec{\ell}_0 = \frac{\sqrt{2}}{2} \begin{bmatrix} \vec{0} & | & c_{-p} & 0\dots0 & | & \dots & | & c_p & 0\dots0 & | & \vec{0} \end{bmatrix}^T,$$

$$\vec{\ell}_1 = \frac{\sqrt{2}}{4} \begin{bmatrix} -c_{-p} & 0\dots0 & | & -c_{-p+1}+c_{-p} & 0\dots0 & | & \dots & | & -c_p+c_{p-1} & 0\dots0 & | & c_p & 0\dots0 & | & \vec{0} \end{bmatrix}^T,$$

$$\vec{\ell}_m = \vec{\ell}_1 \text{ but with the nonzero entries appearing } m-1 \text{ more positions to the right.}$$

The matrix $A_R$, for performing this action if our filter point was in the right half of the element is similarly defined, but with a vector of zeros appended to the beginning of $\vec{\ell}_m$ $m > 1$ and the last vector of zeros deleted.

We now have $\vec{b}_j = A_L \tilde{u} = A_L S^j \vec{u}$. Now to compute the new modal coefficients on element $I_{2j-1}$ we only need to multiply $\vec{b}_j$ by the projection matrix defined by

$$P_L(m,r) = \langle X_L^r, \phi_m \rangle = \int_{-1}^{1} \chi_L^r\Big(\frac{\xi-1}{2}\Big)\phi_m(\xi)\, d\xi, \quad r = 0,\dots,p+1,\ m = 0,\dots,p.$$

We now have $\vec{u}^{2j-1} = P_L \vec{b}^j = P_L A_L S^i \vec{u}$, and thus can compute all the left-sided elements on the finer mesh. Similarly we have $\vec{u}^{2j} = P_R \vec{b}^j = P_R A_R S^j \vec{u}$ where $P_R$ is the matrix defined by

$$P_R(m,r) = \langle \chi_R^r, \phi_m \rangle = \int_{-1}^{1} X_R^r\Big(\frac{\xi+1}{2}\Big)\phi_m(\xi)\, d\xi, \quad r = 0,\dots,p+1,\ m = 0,\dots,p.$$

To define global transition matrices for computing fine mesh modes over the whole mesh we can simply concatenate the composition of these matrices. Define

$$T_L = \begin{bmatrix} P_L A_L S^1 \\ \vdots \\ P_L A_L S^N \end{bmatrix},$$

and notice that the vector modal coefficients on the left-handed elements of the finer mesh is given by $\vec{u}^f_{left} = T_L \vec{u}$. Similarly, we have

$$T_R = \begin{bmatrix} P_R A_R S^1 \\ \vdots \\ P_R A_R S^N \end{bmatrix},$$

and $\vec{u}^f_{right} = T_R \vec{u}$.

Thus in one dimension we have constructed global transition matrices for moving information between scales while applying one-dimensional SIAC filtering.

### E.3 Two-dimensional SIAC-MRA

Extending the one-dimensional approximation given in [23] to two dimensions yields the tensor product filtered approximation:

$$\overset{\star}{h}(x,y) = \sum_{m=0}^{p+1} \sum_{r=0}^{p+1} a^{Q,i,j}_{m,r} \chi^m_Q(\zeta_x) \chi^r_Q(\zeta_y),$$

where $(x, y)$ belongs to quadrant $Q$ of element $(i, j)$.

The $\chi$ functions are the same as in the one-dimensional case with the exception that now their choice depends upon whether a given quadrant is left or right of the origin with respect to a given Cartesian axis. Hence, to select the correct functions for the filtered approximation we identify a given quadrant with an ordered pair $\{a, b\}$ for $a, b \in \{L, R\}$ and choose the appropriate $\chi$ functions as $\chi^m_a(\zeta_x)\chi^r_b(\zeta_y)$. The coefficients $a^{Q,i,j}_{m,r} = a^{i,j}_{m,r}(q_x, q_y)$ also depend upon the quadrant, with the parameters $(q_x, q_y)$ serving to shift the kernel support in combining modal values. The values of all these components are indicated in Table E.3 and the coefficients are given in Table E.2.

#### E.3.1 Two-dimensional SIAC-MRA Transition Matrices

For the two-dimensional case we will only consider the circumstances of the filter point belonging in the third quadrant of whatever quadrilateral element it belongs. The reason for so doing is that only a few of the matrices depend upon the location of the filter point, and those that do only change slightly if the filter

Table E.2 Two-dimensional tensor-product filter coefficients. Note $m, r = 0, \ldots, p+1$.

| $\mathbf{r}$ | $\mathbf{a^{i,j}_{m,r}(q_x, q_y)}$ | $\mathbf{m}$ | $\mathbf{b^{i,j}_{m,r}}$ |
|---|---|---|---|
| **0** | $\frac{\sqrt{2}}{2}\sum_{\gamma=-p}^{p} c_\gamma b^{i,j+\gamma}_{m,0}$ | **0** | $\frac{\sqrt{2}}{2}\sum_{\gamma=-p}^{p} c_\gamma u^{i+\gamma,j}_{0,r}$ |
| **1** | $\frac{\sqrt{2}}{4}\sum_{\gamma=-p}^{p} c_\gamma\big[-b^{i,j+\gamma+q_y}_{m,0} + b^{i,j+\gamma+q_y+1}_{m,0}\big]$ | **1** | $\frac{\sqrt{2}}{4}\sum_{\gamma=-p}^{p} c_\gamma\big[-u^{i+\gamma+q_x,j}_{0,r} + u^{i+\gamma+q_x+1,j}_{0,r}\big]$ |
| $\mathbf{r > 1}$ | $\frac{1}{2}\sum_{\gamma=-p}^{p} c_\gamma\big[-b^{i,j+\gamma+q_y}_{m,r-1} + b^{i,j+\gamma+q_y+1}_{m,r-1}\big]$ | $\mathbf{m > 1}$ | $\frac{1}{2}\sum_{\gamma=-p}^{p} c_\gamma\big[-u^{i+\gamma+q_x,j}_{m-1,r} + u^{i+\gamma+q_x+1,j}_{m-1,r}\big]$ |

Table E.3 Orientation of quadrants in the numbering scheme with respect to the origin. The orientation indicates which $\chi$ functions to use with respect to which variable by quadrant. The coefficient parameters indicate how coefficients adjust by quadrant.

| **Quadrant** | $\chi$ **function orientation** $(a, b)$ | $a^{Q,i,j}_{m,r}$ **parameters** $(q_x, q_y)$ |
|---|---|---|
| **1** | (R,R) | (0,0) |
| **2** | (L,R) | (-1,0) |
| **3** | (L,L) | (-1,-1) |
| **4** | (R,L) | (0,-1) |

point belongs in another quadrant. These alternate formulations will be denoted with an $R$ instead of an $L$ subscript. Note that just as in the one-dimensional case, the alternate matrices will be provided. Suppose that $u_h(x,y)|_{Q_{i,j}} = \sum_{kx=0}^{p}\sum_{ky=0}^{p} u^{i,j}_{kx,ky}\phi_{kx}(\xi_x)\phi_{ky}(\xi_y)$ with

$$u^\star_h(\bar{x},\bar{y}) = \sum_{m=0}^{p+1}\sum_{r=0}^{p+1} a^{Q3,i,j}_{(m,k)}\chi^{(m)}_L(\zeta_x)\chi^{(r)}_L(\zeta_y),$$

for

$$a^{Q3,i,j}_{m,0} = \frac{\sqrt{2}}{2}\sum_{\gamma=-p}^{p} c_\gamma b^{i,j+\gamma}_{(m,0)},$$
$$a^{Q3,i,j}_{m,1} = \frac{\sqrt{2}}{4}\sum_{\gamma=-p}^{p} c_\gamma\Big[-b^{i,j+\gamma-1}_{(m,0)} + b^{i,j+\gamma}_{(m,0)}\Big]$$

and for $r \geq 1$

$$a^{Q3,i,j}_{m,r+1} = \frac{1}{2}\Big[-c_{-p}b^{i,j-(p+1)}_{(m,k)} + \sum_{\gamma=-p}^{p-1} b^{i,j+\gamma}_{(m,k)}\big(c_\gamma - c_{\gamma+1}\big) + c_p b^{i,j+p}_{(m,k)}\Big],$$

with

$$b^{i,j}_{(0,r)} = \frac{\sqrt{2}}{2}\Bigg\{\sum_{\gamma=-p}^{p} c_\gamma u^{i+\gamma,j}_{0,r}\Bigg\},$$
$$b^{i,j}_{(1,r)} = \frac{\sqrt{2}}{4}\Bigg\{\sum_{\gamma=-p}^{p} c_\gamma\Big(-u^{i+\gamma-1,j}_{0,r} + u^{i+\gamma,j}_{0,r}\Big)\Bigg\},$$

and for $m \geq 1$

$$b^{i,j}_{(m+1,r)} = \frac{1}{2}\Big[ - c_{-p} u^{i-(p+1),j}_{m,r} + \sum_{\gamma=-p}^{p-1} u^{i+\gamma,j}_{m,r} \Big( c_\gamma - c_{\gamma+1} \Big) + c_p u^{i+p,j}_{m,r} \Big].$$

The two-dimensional cases necessitates some arbitrary ordering of modal coefficients into a vector. Denote the $(N(p+1))^2 \times 1$ vector of modal coefficients by

$$\vec{u} = \begin{bmatrix} \vec{u}^1 & | & \dots & | & \vec{u}^N \end{bmatrix}^T,$$

where

$$\vec{u}^j = \begin{bmatrix} \vec{u}^j_0 \dots \vec{u}^j_p \end{bmatrix}^T,$$

$$\vec{u}^j_{ky} = \begin{bmatrix} \vec{u}^{1,j}_{ky} & | & \dots & | & \vec{u}^{N,j}_{ky} \end{bmatrix}^T,$$

and

$$\vec{u}^{i,j}_{ky} = \begin{bmatrix} u^{i,j}_{0,ky} & | & \dots & | & u^{i,j}_{p,ky} \end{bmatrix}^T.$$

We will first construct matrices to compute $\vec{b}$, a vector which contain all the $b$ coefficients. We will then construct matrices to transform $\vec{b}$ into a vector containing the $a^{Q3}$ coefficients, whose corresponding basis functions we can then project. Define the $(p+2) \times 1$ vector

$$\vec{b}^{i,j}_{ky} = \begin{bmatrix} b^{i,j}_{0,ky} \dots b^{i,j}_{p+1,ky} \end{bmatrix}^T.$$

To compute this vector, we only need information from the vector $\vec{u}^j_{ky}$. First we need to sort the vector appropriately, and then perform the summations. We have

$$\vec{b}^{i,j}_{ky} = A_L S^i \vec{u}^j_{ky},$$

where $S^i$ and $A_L$ are the same as in the one-dimensional case. Note that if we were in quadrant I or IV we would use $A_R$ instead. We can now compute every $b$ coefficient by iterating over $i, j$ and $ky$. We can more generally state

$$\vec{b}^{i,j} = \begin{bmatrix} A_L S^i & & 0 \\ & \ddots & \\ 0 & & A_L S^i \end{bmatrix} \vec{u}^j = K^i_L \vec{u}^j,$$

where

$$\vec{b}^{i,j} = \left[\vec{b}_0^{i,j} \dots \vec{b}_p^{i,j}\right]^b.$$

Hence,

$$\vec{b}^j = \left[\vec{b}^{1,j} \dots \vec{b}^{N,j}\right]^T = \begin{bmatrix} K_L^1 \\ \vdots \\ K_L^N \end{bmatrix} \vec{u}^j = \mathbb{K}_L \vec{u}^j,$$

and so

$$\vec{b} = \left[\vec{b}^1 \dots \vec{b}^N\right]^T = \begin{bmatrix} \mathbb{K}_L & & 0 \\ & \ddots & \\ 0 & & \mathbb{K}_L \end{bmatrix} \vec{u} = \mathscr{K}_L \vec{u}.$$

### E.3.2 Construction of a permutation matrix

We now wish to construct a vector $\vec{b}$ containing all the $b$ coefficients so that we can apply the same methodology to compute the $a^{Q3}$ coefficients. We run into the issue that where we fixed $ky, j$ and ordered $\vec{u}_{ky}^j$ by the index $kx$, we require something similarly ordered in $b$ but with $kx, i$ fixed instead. Here is how we solve this issue. Define

$$\tilde{b} = \left[\tilde{b}^1 \quad | \quad \dots \quad | \quad \tilde{b}^N\right]^T,$$

where

$$\tilde{b}^i = \left[\tilde{b}_0^i \dots \tilde{b}_{p+1}^i\right]^T,$$

$$\tilde{b}_{kx}^i = \left[\tilde{b}_{kx}^{i,1} \quad | \quad \dots \quad | \quad \tilde{b}_{kx}^{i,N}\right]^T,$$

and

$$\tilde{b}_{kx}^{i,j} = \left[b_{kx,0}^{i,j} \quad | \quad \dots \quad | \quad b_{kx,p}^{i,j}\right]^T.$$

Note that $kx$ is able to run $0 : p+1$. These $\tilde{\cdot}$ denoted vectors are just a reordering of the entries in $\vec{b}$. To compute this permutation, note that

$$\begin{bmatrix} \tilde{b}_{kx=0}^{i,j} \\ \vdots \\ \tilde{b}_{kx=p+1}^{i,j} \end{bmatrix} = \begin{bmatrix} \Delta_{1,1} & \Delta_{2,1} & \dots & \Delta_{p+1,1} \\ \Delta_{1,2} & \Delta_{2,2} & \dots & \Delta_{p+1,2} \\ \vdots & \vdots & \vdots & \vdots \\ \Delta_{1,p+2} & \Delta_{2,1} & \dots & \Delta_{p+1,p+2} \end{bmatrix} \begin{bmatrix} \vec{b}_{ky=0}^{i,j} \\ \vdots \\ \vec{b}_{ky=p}^{i,j} \end{bmatrix} = P\vec{b}^{i,j},$$

where $\Delta_{i,j}$ is the $(p+1)\times(p+2)$ matrix with 1 in entry $(i,j)$ and $0's$ elsewhere. It is natural then to say that $\tilde{b}^{i,j}_{kx} = P\big|_{kx+1}\vec{b}^{i,j}$, where $P\big|_{kx+1} = \begin{bmatrix}\Delta_{1,kx+1} & \Delta_{2,kx+1} & \dots & \Delta_{p+1,kx+1}\end{bmatrix}$ is $(p+1)\times(p+2)(p+1)$. Now to compute the necessary $\tilde{b}$ vector we simply need to append these block matrices. Define the block matrix

$$Q^{i,j}_{kx} = \begin{bmatrix}\delta_{1j}\chi_{i,kx}\dots\delta_{Nj}\chi_{i,kx}\end{bmatrix},$$

where $\delta_{mj}$ is the Kronecker delta function, and $\chi_{i,kx} = [\delta_{i,1}P\big|_{kx+1}\dots\delta_{i,N}P\big|_{kx+1}]$. We now can permute our $b$ coefficients into a usable form by the relation

$$\tilde{b}^i_{kx} = \begin{bmatrix}Q^{i1}_{kx}\\ \vdots \\ Q^{iN}_{kx}\end{bmatrix}\vec{b} = Q^i_{kx}\vec{b}.$$

We can then observe that

$$\tilde{b}^i = \begin{bmatrix}Q^i_0\\ \vdots \\ Q^i_{p+1}\end{bmatrix}\vec{b} = Q^i\vec{b},$$

and so

$$\tilde{b} = \begin{bmatrix}Q^1\\ \vdots \\ Q^N\end{bmatrix}\vec{b} = Q\vec{b}.$$

Finally we have obtained the desired permutation matrix.

Defining

$$\vec{a}^{Q3,i,j}_m = \begin{bmatrix}a^{Q3,i,j}_{m,0}\dots a^{Q3,i,j}_{m,p+1}\end{bmatrix}^T,$$

we have $\vec{a}^{Q3,i,j}_m = \tilde{A}_L\tilde{S}^j\vec{d}^i_m$ where $\tilde{A}_L$ and $\tilde{S}^j$ are as defined previously. Note that were we in quadrant I or II we would use $\tilde{A}_R$ instead. We now wish to append these vectors into a form amicable to multiplication by a projection matrix. Let

$$\vec{a}^{Q3} = \begin{bmatrix}\vec{a}^{Q3,i=1}\dots\vec{a}^{Q3,i=N}\end{bmatrix}^T,$$

$$\vec{a}^{Q3,i} = \begin{bmatrix}\vec{a}^{Q3,i,1}\dots\vec{a}^{Q3,i,N}\end{bmatrix}^T,$$

$$\vec{a}^{Q3,i,j}_m = \begin{bmatrix}\vec{a}^{Q3,i,j}_{m=0}\dots\vec{a}^{Q3,i,j}_{m=p+1}\end{bmatrix}^T,$$

$$\vec{a}_m^{Q3,i,j} = \begin{bmatrix} \vec{a}_{m,r=0}^{Q3,i,j} \dots \vec{a}_{m,r=p+1}^{Q3,i,j} \end{bmatrix}^T.$$

These vectors have been constructed so that we can follow the previous arguments used in computing $\vec{b}$. We have that

$$\vec{a}_m^{Q3,i,j} = \begin{bmatrix} \tilde{A}_L \tilde{S}^j & & 0 \\ & \ddots & \\ 0 & & \tilde{A}_L \tilde{S}^j \end{bmatrix} \tilde{b}^i = \tilde{K}_L^j \tilde{b}^i.$$

$$\vec{a}^{Q3,0} i = \begin{bmatrix} \tilde{K}_L^1 \\ \vdots \\ \tilde{K}_L^N \end{bmatrix} \tilde{b} = \tilde{\mathbb{K}}_L \tilde{b}.$$

Hence,

$$\vec{a}^{Q3} = \begin{bmatrix} \tilde{\mathbb{K}}_L^1 & & 0 \\ & \ddots & \\ 0 & & \tilde{\mathbb{K}}_L^N \end{bmatrix} \tilde{b} = \tilde{\mathscr{K}}_L \tilde{b}.$$

So

$$\vec{a}^{Q3} = \tilde{\mathscr{K}}_L Q \vec{b} = \tilde{\mathscr{K}}_L Q \mathscr{K}_L \vec{u}.$$

Now to perform the projections. With $u_h^\star(x,y)|_{K_{i,j}} = \sum_{m=0}^{p+1} \sum_{r=0}^{p+1} a_{m,r}^{Q3,i,j} \chi_L^m(\xi_x) \chi_L^r(\xi_y)$, we define the projection of this approximation by $u_h^f|_{K^{2i-1,2j-1}}(x,y)$ where the new modal coefficients are computed by

$$\begin{aligned} \langle u_h^\star, \phi_{kx}\phi_{ky} \rangle_{LL} &= \langle u_h^f, \phi_{kx}\phi_{ky} \rangle && \text{(E.1)} \\ &= u_{kx,ky}^{2i-1.2j-1}, \quad kx, ky = 0, \dots, p. && \text{(E.2)} \end{aligned}$$

Define $H^{kx,m}$ to be the $(p+1) \times (p+2)$ matrix defined by

$$H_{ky,r}^{kx,m} = \int_{-1}^{1} \int_{-1}^{1} X_L^m \Big( \frac{\xi_x - 1}{2} \Big) X_L^r \Big( \frac{\xi_y - 1}{2} \Big) \phi_{kx}(\xi_x) \phi_{ky}(\xi_y) \, d\xi_x \, d\xi_y.$$

We only need information from the element $i,j$ components of $\vec{a}^{Q3}$. To do so we first multiply $\vec{a}^{Q3}$ by the $(N(p+2)^2) \times (N^2(p+2)^2)$ block matrix $C_i$ defined by

$$C_i = \begin{bmatrix} \delta_{1i} \mathbb{I}^{N(p+2)^2 \times (N(p+2)^2)} & | & \dots & | & \delta_{Ni} \mathbb{I}^{N(p+2)^2 \times (N(p+2)^2)} \end{bmatrix}.$$

This selects the $i$-coordinate. Then we multiply by the $(p+2)^2 \times (N(p+2)^2)$ block matrix $D_j$ defined by

$$D_j = \begin{bmatrix} \delta_{1j}\mathbb{I}^{(p+2)^2\times(p+2)^2} & | & \dots & | & \delta_{Nj}\mathbb{I}^{(p+2)^2\times(p+2)^2} \end{bmatrix}.$$

This selects the $j$-coordinate. Note that here $\delta$ represents the Kronecker-delta functions. Defining $\vec{a}_m^{Q3,i,j} = D_j C_i \vec{a}^{Q3}$ and $\tilde{u}^{i,j} = \begin{bmatrix} u_{0,0}^{i,j} \dots u_{0,p}^{i,j} & | & \dots & | & u_{p,0}^{i,j} \dots u_{p,p}^{i,j} \end{bmatrix}$, we have

$$\tilde{u}^{2i-1,2j-1} = \begin{bmatrix} H^{0,0} & \dots & H^{0,p+1} \\ \vdots & H^{kx,m} & \vdots \\ H^{p,0} & \dots & H^{p,p+1} \end{bmatrix} \vec{a}_m^{Q3,i,j} = H_{LL}\vec{a}_m^{Q3,i,j}.$$

So we have almost come full circle, given the coefficients globally over a coarse level, we can compute locally all the coefficients on the finer mesh. We are not done however, as we would like something like $\vec{u}_{LL}^f = T_{LL}\vec{u}$. To construct this transition matrix we simply need permute our coefficient vector $\vec{u}^{2i-1,2j-1}$ and stack the permutations together. Consider the $P$ matrix defined when permuting the $b$ coefficients. If we make all the block matrices $(p+1)\times(p+1)$ and only have $(p+1)\times(p+1)$ of them, it follows that

$$\begin{bmatrix} u_{0,0}^{i,j} \\ \vdots \\ u_{p,0}^{i,j} \\ \vdots \\ u_{0,p}^{i,j} \\ \vdots \\ u_{p,p}^{i,j} \end{bmatrix} = \begin{bmatrix} \Delta_{1,1} & \Delta_{2,1} & \dots & \Delta_{p+1,1} \\ \Delta_{1,2} & \Delta_{2,2} & \dots & \Delta_{p+1,2} \\ \vdots & \vdots & \vdots & \vdots \\ \Delta_{1,p+1} & \Delta_{2,1} & \dots & \Delta_{p+1,p+1} \end{bmatrix} \begin{bmatrix} u_{0,0}^{i,j} \\ \vdots \\ u_{0,p}^{i,j} \\ \vdots \\ u_{p,0}^{i,j} \\ \vdots \\ u_{p,p}^{i,j} \end{bmatrix} = P\tilde{u}^{i,j}.$$

Hence,

$$\vec{u}^{i,j} = P\Big|_{ky} \tilde{u}^{i,j},$$

where $P\Big|_{ky} = P\Big(1 + ky(p+1) : (ky+1)(p+1), :\Big)$. Defining $\tilde{u}^j = [\tilde{u}^{1j} \dots \tilde{u}^{Nj}]^T$, we have

$$\vec{u}_{ky}^j = \begin{bmatrix} P\Big|_{ky} & & 0 \\ & \ddots & \\ 0 & & P\Big|_{ky} \end{bmatrix} \tilde{u}^j = P_{ky}\tilde{u}^j.$$

Continuing we have

$$\vec{u}^j = \begin{bmatrix} P_0 \\ \vdots \\ P_p \end{bmatrix} \tilde{u}^j = \mathbb{P}\tilde{u}^j.$$

Lastly, defining $\tilde{u} = [\tilde{u}^1 \dots \tilde{u}^N]^T$, we have

$$\vec{u} = \begin{bmatrix} \mathbb{P} & & 0 \\ & \ddots & \\ 0 & & \mathbb{P} \end{bmatrix} \tilde{u} = \mathscr{P}\tilde{u}.$$

So

$$\vec{u}^f_{LL} = \mathscr{P}\tilde{u}.$$

Now to compute $\tilde{u}$ from our known vector $\vec{a}^{Q3}$, define

$$M^j_{LL} = \begin{bmatrix} H_{LL}D_jC_1 \\ \vdots \\ H_{LL}D_jC_N \end{bmatrix},$$

and note that then

$$\tilde{u} = \begin{bmatrix} M^1_{LL} \\ \vdots \\ M^N_{LL} \end{bmatrix} \vec{a}^{Q3} = \mathscr{M}_{LL}\vec{a}^{Q3}.$$

Hence,

$$\vec{u}^f_{LL} = \mathscr{P}\tilde{u} = \mathscr{P}\mathscr{M}_{LL}\vec{a}^{Q3} = \mathscr{P}\mathscr{M}_{LL}\tilde{\mathscr{K}}_L Q\vec{d} = \mathscr{P}\mathscr{M}_{LL}\tilde{\mathscr{K}}_L Q\mathscr{K}_L\vec{u}.$$

If we wanted to compute the coefficients in any of the other quadrants, simply choose the change the $\tilde{\mathscr{K}}_L$, $\mathscr{K}_L$, or the $\mathscr{M}_{LL}$ matrices to their right-sided counterparts. Defining $T_{(..)}$ to be the appropriate compositions, we have

$$\begin{aligned} \vec{u}^f_{LL} &= T_{LL}\vec{u}, \\ \vec{u}^f_{RL} &= T_{RL}\vec{u}, \\ \vec{u}^f_{LR} &= T_{LR}\vec{u}, \\ \vec{u}^f_{RR} &= T_{RR}\vec{u}. \end{aligned}$$

### E.4 Three-dimensional SIAC-MRA

In three dimensions, the tensor product filtered approximation is of the form

$$u_h^\star(x,y,z) = \sum_{m=0}^{p+1}\sum_{r=0}^{p+1}\sum_{\ell=0}^{p+1} a_{m,r,\ell}^{O,i,j,k}\chi_O^m(\zeta_x)\chi_O^r(\zeta_y)\chi_O^\ell(\zeta_z),$$

where $(x,y,z)$ belongs to octant $O$ of element $(i,j,k)$. Just as in the two-dimensional case, our filtered approximation functions are just tensor-products of one-dimensional functions. Hence, to the select the correct functions for our filtered approximation form we identify a given octant with an ordered triple $\{a,b,c\}$ for $a,b,c \in \{L,R\}$ and choose the appropriate $\chi$ functions as $\chi_a^m(\zeta_x)\chi_b^r(\zeta_y)\chi_c^\ell(\zeta_z)$. The coefficients $a_{m,r,\ell}^{Q,i,j,k} = a_{m,r,\ell}^{i,j}(q_x,q_y,q_z)$ also depend upon the quadrant, with the parameters $(q_x,q_y,q_z)$ serving to shift the kernel support in combining modal values. Table E.4 and Table E.5 detail these ordered triples and the resulting values of the coefficients.

Table E.4 Orientation of octants in the numbering scheme with respect to the origin. The orientation indicates which $\chi$ functions to use with respect to which variable by octant. The shifting parameters for the coefficients are also determined by the octant and correspond to the shifting of the kernel support depending on the filtering point location. The octant numbering scheme is given in Figure F.1.

| **Octant** | $\chi$ **function orientation** $(a,b,c)$ | $a_{m,r,\ell}^{Q,i,j,k}$ **parameters** $(q_x,q_y,q_z)$ |
|---|---|---|
| **1** | (R,R,R) | (0,0,0) |
| **2** | (R,R,L) | (0,0,-1) |
| **3** | (L,R,R) | (-1,0,0) |
| **4** | (L,R,L) | (-1,0,-1) |
| **5** | (R,L,R) | (0,-1,0) |
| **6** | (R,L,L) | (0,-1,-1) |
| **7** | (R,L,L) | (0,-1,-1) |
| **8** | (L,L,L) | (-1,-1,-1) |

#### E.4.1 Three-dimensional SIAC-MRA Transition Matrices

For brevity we eschew providing the 3D construction as it is analogous to the 2D construction but much longer.

Table E.5 Three-dimensional tensor-product filter coefficients. Note $m, r, \ell = 0, \ldots, p+1$.

| $\ell$ | $\mathbf{a^{i,j,k}_{m,r,\ell}(q_x, q_y, q_z)}$ |
|---|---|
| **0** | $\frac{\sqrt{2}}{2} \sum_{\gamma=-p}^{p} c_\gamma b^{i,j,k+\gamma}_{(m,r,0)}$ |
| **1** | $\frac{\sqrt{2}}{4} \sum_{\gamma=-p}^{p} c_\gamma \Big[ -b^{i,j,k+\gamma+q_z}_{(m,r,0)} + b^{i,j,k+\gamma+q_z+1}_{(m,r,0)} \Big]$ |
| $\ell > \mathbf{1}$ | $\frac{1}{2} \sum_{\gamma=-p}^{p} c_\gamma \Big[ -b^{i,j,k+\gamma+q_z}_{(m,r,\ell-1)} + b^{i,j,k+\gamma+q_z+1}_{(m,r,\ell-1)} \Big]$ |
| **r** | $\mathbf{b^{i,j,k}_{m,r,\ell}(q_x, q_y, q_z)}$ |
| **0** | $\frac{\sqrt{2}}{2} \sum_{\gamma=-p}^{p} c_\gamma b^{i,j+\gamma,k}_{m,0,\ell}$ |
| **1** | $\frac{\sqrt{2}}{4} \sum_{\gamma=-p}^{p} c_\gamma \big[ -b^{i,j+\gamma+q_y,k}_{m,0,\ell} + b^{i,j+\gamma+q_y+1,k}_{m,0,\ell} \big]$ |
| $\mathbf{r} > \mathbf{1}$ | $\frac{1}{2} \sum_{\gamma=-p}^{p} c_\gamma \big[ -b^{i,j+\gamma+q_y,k}_{m,r-1,\ell} + b^{i,j+\gamma+q_y+1,k}_{m,r-1,\ell} \big]$ |
| **m** | $\mathbf{b^{i,j,k}_{m,r,\ell}}$ |
| **0** | $\frac{\sqrt{2}}{2} \sum_{\gamma=-p}^{p} c_\gamma u^{i+\gamma,j,k}_{0,r,\ell}$ |
| **1** | $\frac{\sqrt{2}}{4} \sum_{\gamma=-p}^{p} c_\gamma \big[ -u^{i+\gamma+q_x,j,k}_{0,r,\ell} + u^{i+\gamma+q_x+1,j,k}_{0,r,\ell} \big]$ |
| $\mathbf{m} > \mathbf{1}$ | $\frac{1}{2} \sum_{\gamma=-p}^{p} c_\gamma \big[ -u^{i+\gamma+q_x,j,k}_{m-1,r,\ell} + u^{i+\gamma+q_x+1,j,k}_{m-1,r,\ell} \big]$ |

APPENDIX F

LSIAC-MRA TRANSITION MATRICES

## F.1 Two-dimensional LSIAC-MRA

There are seven different forms for the coefficients included in the two-dimensional LSIAC case with the basis given in Table F.1. These coefficients are of the form

$$a_{m,r}^{S,P} = a_{m,r}(q_x, q_y) = \sum_{\gamma=-p}^{p} c_\gamma u_{m,r}^{i+\gamma+qx, j+\gamma+qy},$$

where the arguments $(q_x, q_y)$ are determined by the region $S$ and position $P$ of the filter point. A table of these dependencies by region and position is provided in Table F.2. The regions themselves are depicted in Figure F.1. An algorithm for the LSIAC-MRA which differs from the algorithm for 2D SIAC-MRA in the number of filtering regions is included in Algorithm 2.

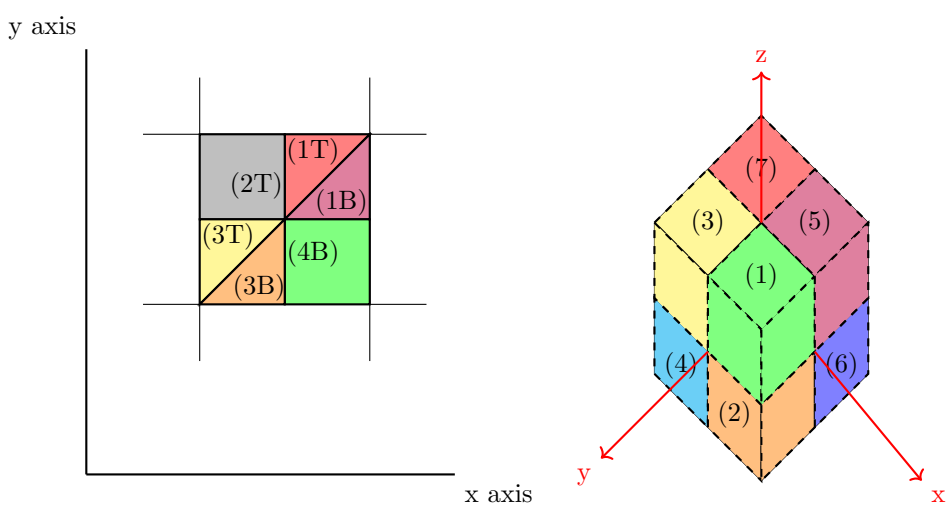


Figure F.1 Partition of domain for both 2D LSIAC (left) and the octant numbering scheme in 3D (right).

Table F.1 Two-dimensional LSIAC basis functions. For notational convenience suppose $\mathbf{x}$ belongs to element $(i, j)$.

| **Region $S$** | **Basis Function** | **Definition** |
|---|---|---|
| **1T** | $\chi^{m,r}_{1T,L}(\zeta_x,\zeta_y)$ | $\frac{2-\zeta_y}{2}\int_{-1}^{1}\phi_{kx}\Big(\zeta_x-\frac{\zeta_y}{2}+\big(1-\frac{\zeta_y}{2}\big)\eta\Big)\phi_{ky}\Big(\big(1-\frac{\zeta_y}{2}\big)\eta+\frac{\zeta_y}{2}\Big)\,d\eta$ |
| | $\chi^{m,r}_{1T,M}(\zeta_x,\zeta_y)$ | $\frac{\zeta_y-\zeta_x}{2}\int_{-1}^{1}\phi_{kx}\Big(\frac{\zeta_y-\zeta_x}{2}\eta+\frac{\zeta_x-\zeta_y}{2}+1\Big)\phi_{ky}\Big(\frac{\zeta_y-\zeta_x}{2}\eta+\frac{\zeta_y-\zeta_x}{2}-1\Big)\,d\eta$ |
| | $\chi^{m,r}_{1T,R}(\zeta_x,\zeta_y)$ | $\frac{\zeta_x}{2}\int_{-1}^{1}\phi_{kx}\Big(\frac{\zeta_x}{2}\eta+\frac{\zeta_x}{2}-1\Big)\phi_{ky}\Big(\frac{\zeta_x}{2}\eta+\zeta_y-1-\frac{\zeta_x}{2}\Big)\,d\eta$ |
| **1B** | $\chi^{m,r}_{1B,L}(\zeta_x,\zeta_y)$ | $\frac{2-\zeta_x}{2}\int_{-1}^{1}\phi_{kx}\Big(\big(1-\frac{\zeta_x}{2}\big)\eta+\frac{\zeta_x}{2}\Big)\phi_{ky}\Big(\zeta_y-\frac{\zeta_x}{2}+\big(1-\frac{\zeta_x}{2}\big)\eta\Big)\,d\eta$ |
| | $\chi^{m,r}_{1B,M}(\zeta_x,\zeta_y)$ | $\frac{\zeta_x-\zeta_y}{2}\int_{-1}^{1}\phi_{kx}\Big(\frac{\zeta_x-\zeta_y}{2}\eta+\frac{\zeta_x-\zeta_y}{2}-1\Big)\phi_{ky}\Big(\frac{\zeta_x-\zeta_y}{2}\eta+\frac{\zeta_y-\zeta_x}{2}+1\Big)\,d\eta$ |
| | $\chi^{m,r}_{1B,R}(\zeta_x,\zeta_y)$ | $\frac{\zeta_y}{2}\int_{-1}^{1}\phi_{kx}\Big(\frac{\zeta_y}{2}\eta+\zeta_x-\frac{\zeta_y}{2}-1\Big)\phi_{ky}\Big(\frac{\zeta_y}{2}\eta-1+\frac{\zeta_y}{2}\Big)\,d\eta$ |
| **2T** | $\chi^{m,r}_{2T,L}(\zeta_x,\zeta_y)$ | $-\frac{\zeta_x}{2}\int_{-1}^{1}\phi_{kx}\Big(-\frac{\zeta_x}{2}\eta+\frac{\zeta_x}{2}+1\Big)\phi_{ky}\Big(-\frac{\zeta_x}{2}\eta+\zeta_y-1-\frac{\zeta_x}{2}\Big)\,d\eta$ |
| | $\chi^{m,r}_{2T,M}(\zeta_x,\zeta_y)$ | $\frac{2+\zeta_x-\zeta_y}{2}\int_{-1}^{1}\phi_{kx}\Big(\frac{2+\zeta_x-\zeta_y}{2}\eta+\frac{\zeta_x-\zeta_y}{2}\Big)\phi_{ky}\Big(\frac{2+\zeta_x-\zeta_y}{2}\eta+\frac{\zeta_y-\zeta_x}{2}\Big)\,d\eta$ |
| | $\chi^{m,r}_{2T,R}(\zeta_x,\zeta_y)$ | $\frac{\zeta_y}{2}\int_{-1}^{1}\phi_{kx}\Big(\frac{\zeta_y}{2}\eta+\zeta_x-\frac{\zeta_y}{2}+1\Big)\phi_{ky}\Big(\frac{\zeta_y}{2}\eta+\frac{\zeta_y}{2}-1\Big)\,d\eta$ |
| **3T** | $\chi^{m,r}_{3T,L}(\zeta_x,\zeta_y)$ | $-\frac{\zeta_y}{2}\int_{-1}^{1}\phi_{kx}\Big(-\frac{\zeta_y}{2}\eta-\frac{\zeta_y}{2}+\zeta_x+1\Big)\phi_{ky}\Big(-\frac{\zeta_y}{2}\eta+1+\frac{\zeta_y}{2}\Big)\,d\eta$ |
| | $\chi^{m,r}_{3T,M}(\zeta_x,\zeta_y)$ | $\frac{\zeta_y-\zeta_x}{2}\int_{-1}^{1}\phi_{kx}\Big(\frac{\zeta_y-\zeta_x}{2}\eta+\frac{\zeta_x-\zeta_y}{2}+1\Big)\phi_{ky}\Big(\frac{\zeta_y-\zeta_x}{2}\eta+\frac{\zeta_y-\zeta_x}{2}-1\Big)\,d\eta$ |
| | $\chi^{m,r}_{3T,R}(\zeta_x,\zeta_y)$ | $\frac{2+\zeta_x}{2}\int_{-1}^{1}\phi_{kx}\Big(\big(\frac{2+\zeta_x}{2}\big)\eta+\frac{\zeta_x}{2}\Big)\phi_{ky}\Big(\big(\frac{2+\zeta_x}{2}\big)\eta-\frac{\zeta_x}{2}+\zeta_y\Big)\,d\eta$ |
| **3B** | $\chi^{m,r}_{3B,L}(\zeta_x,\zeta_y)$ | $-\frac{\zeta_x}{2}\int_{-1}^{1}\phi_{kx}\Big(-\frac{\zeta_x}{2}\eta+\frac{\zeta_x}{2}+1\Big)\phi_{ky}\Big(-\frac{\zeta_x}{2}\eta+1+\zeta_y-\frac{\zeta_x}{2}\Big)\,d\eta$ |
| | $\chi^{m,r}_{3B,M}(\zeta_x,\zeta_y)$ | $\frac{\zeta_x-\zeta_y}{2}\int_{-1}^{1}\phi_{kx}\Big(\frac{\zeta_x-\zeta_y}{2}\eta+\frac{\zeta_x-\zeta_y}{2}-1\Big)\phi_{ky}\Big(\frac{\zeta_x-\zeta_y}{2}\eta+\frac{\zeta_y-\zeta_x}{2}+1\Big)\,d\eta$ |
| | $\chi^{m,r}_{3B,R}(\zeta_x,\zeta_y)$ | $\frac{2+\zeta_y}{2}\int_{-1}^{1}\phi_{kx}\Big(\big(\frac{2+\zeta_y}{2}\big)\eta+\zeta_x-\frac{\zeta_y}{2}\Big)\phi_{ky}\Big(\big(\frac{2+\zeta_y}{2}\big)\eta+\frac{\zeta_y}{2}\Big)\,d\eta$ |
| **4B** | $\chi^{m,r}_{4B,L}(\zeta_x,\zeta_y)$ | $-\frac{\zeta_y}{2}\int_{-1}^{1}\phi_{kx}\Big(-\frac{\zeta_y}{2}\eta-\frac{\zeta_y}{2}+\zeta_x-1\Big)\phi_{ky}\Big(-\frac{\zeta_y}{2}\eta+1+\frac{\zeta_y}{2}\Big)\,d\eta$ |
| | $\chi^{m,r}_{4B,M}(\zeta_x,\zeta_y)$ | $\frac{\zeta_y-\zeta_x+2}{2}\int_{-1}^{1}\phi_{kx}\Big(\big(1+\frac{\zeta_y-\zeta_x}{2}\big)\eta+\frac{\zeta_x-\zeta_y}{2}\Big)\phi_{ky}\Big(\big(1+\frac{\zeta_y-\zeta_x}{2}\big)\eta+\frac{\zeta_y-\zeta_x}{2}\Big)\,d\eta$ |
| | $\chi^{m,r}_{4B,R}(\zeta_x,\zeta_y)$ | $\frac{\zeta_x}{2}\int_{-1}^{1}\phi_{kx}\Big(\frac{\zeta_x}{2}\eta+\frac{\zeta_x}{2}-1\Big)\phi_{ky}\Big(\frac{\zeta_x}{2}\eta+1+\zeta_y-\frac{\zeta_x}{2}\Big)\,d\eta$ |

Table F.2 Table describing how the shifting of the arguments of the $a$ coefficients vary by region.

| **Region $S$** \ **Arguments of $a_\alpha(q_x,q_y)$** | **P = L** | **P = M** | **P = R** |
|---|---|---|---|
| **1T** | $(0,0)$ | $(0,1)$ | $(1,1)$ |
| **1B** | $(0,0)$ | $(1,0)$ | $(1,1)$ |
| **2T** | $(-1,0)$ | $(0,0)$ | $(0,1)$ |
| **3T** | $(-1,-1)$ | $(-1,0)$ | $(0,0)$ |
| **3B** | $(-1,-1)$ | $(0,-1)$ | $(0,0)$ |
| **4B** | $(0,-1)$ | $(0,0)$ | $(1,0)$ |

### F.1.1 Two-dimensional LSIAC-MRA Transition Matrices

See Section 4.5 in the text for the matrix construction.

## F.2 Three-dimensional LSIAC-MRA

In three-dimensions, the filtered expansion functions are of the form

$$\chi^{m,r,\ell}_{S,P} = C\int_{-1}^{1}\phi_m(\arg_x)\phi_r(\arg_y)\phi_\ell(\arg_z)$$

**Algorithm 2** Calculate refined modal coefficients $u^f$ from $u$ using the LSIAC filter

**Input:** Number of elements in one direction $N$, and polynomial degree $p$.
**Output:** Fine mesh modal coefficients $u^f$

% Compute modal projection on coarse mesh
$u = \text{Initialize}(N, p, \Omega, u_0)$

% Compute projection matrices by region
$[P1TL, P1TM, P1TR] = \text{ProjMatrixL}(N, p, Q1T)$
$[P1BL, P1BM, P1BR] = \text{ProjMatrixL}(N, p, Q1B)$
$[P2TL, P2TM, P2TR] = \text{ProjMatrixL}(N, p, Q2T)$
$[P3TL, P3TM, P3TR] = \text{ProjMatrixL}(N, p, Q3T)$
$[P3BL, P3BM, P3BR] = \text{ProjMatrixL}(N, p, Q3B)$
$[P4BL, P4BM, P4BR] = \text{ProjMatrixL}(N, p, Q4B)$

% Loop through mesh elements
**for** $i \leftarrow 1, N$ **do**
  **for** $j \leftarrow 1, N$ **do**

    % Construct filtering/sifting matrices
      $K11 = \text{FiltMatrixL}(p, 1, 1, i, j)$
      $K01 = \text{FiltMatrixL}(p, 0, 1, i, j)$
      $K10 = \text{FiltMatrixL}(p, 1, 0, i, j)$
      $K00 = \text{FiltMatrixL}(p, 0, 0, i, j)$
      $Kn10 = \text{FiltMatrixL}(p, -1, 0, i, j)$
      $K0n1 = \text{FiltMatrixL}(p, 0, -1, i, j)$
      $Kn1n1 = \text{FiltMatrixL}(p, -1, -1, i, j)$

    % Compute local fine mesh coefficients
    $u^f_{tensor}(:, 2i, 2j) =$
      $((P1TL + P1BL) * K00 +$
      $(P1TR + P1BR) * K11 +$
      $(P1TM * K01 + P1BM * K10) * u$
    $u^f_{tensor}(:, 2i-1, 2j) =$
      $(P2L * Kn10 + P2M * K00 + P2R * K01) * u$
    $u^f_{tensor}(:, 2i-1, 2j-1) =$
      $((P3TL + P3BL) * Kn1n1 +$
      $(P3TR + P3BR) * K00 +$
      $P3TM * Kn10 + P3BM * K0n1) * u$
    $u^f_{tensor}(:, 2i, 2j-1) =$
      $(P4L * K0n1 + P4M * K00 + P4R * K10) * u$
  **end for**
**end for**

% Reshape fine modal array into original array form
$u^f = \text{ReshapeArray}(p, N, u^f_{tensor})$

where the row of the table determines the Region $S$ and Position $P = \{L, LM, RM, R\}$. The regions are described in Table F.3 and Table F.4 with visual depictions in Figure F.1 and Figure F.2. The coefficient $C$ is given in the coefficient column, and the arguments of the Legendre polynomials are given in the appropriate argument column. These can be found in Table F.5. There are fifteen different forms for the coefficients included in the 3D LSIAC case. These coefficients are of the form

$$a_{m,r,\ell}^{S,P} = a_{m,r,\ell}(q_x, q_y, q_z) = \sum_{\gamma=-p}^{p} c_\gamma u_{m,r,\ell}^{i+\gamma+qx, j+\gamma+qy, k+\gamma+qz},$$

where the coefficients $(q_x, q_y, q_z)$ are determined by the region $S$ and position $P$ of the filter point. A table of these dependencies by region and position is provided in Table F.9.

Table F.3 Regions present by octant.

| **Octant** | **Regions Present** |
|---|---|
| **1** | 1A, 1B, 1C, 1D, 1E, 1F |
| **2** | 2E, 2F |
| **3** | 3A, 3B |
| **4** | 4B, 4F |
| **5** | 5C, 5D |
| **6** | 6D, 6E |
| **7** | 7A, 7C |
| **8** | 8A, 8B, 8C, 8D, 8E, 8F |

Table F.4 Classification of regions.

| **Class** | **Filtering Coordinate Inequality** |
|---|---|
| **(A)** | $\zeta_x \leq \zeta_y \leq \zeta_z$ |
| **(B)** | $\zeta_x \leq \zeta_z < \zeta_y$ |
| **(C)** | $\zeta_y < \zeta_x \leq \zeta_z$ |
| **(D)** | $\zeta_y \leq \zeta_z < \zeta_x$ |
| **(E)** | $\zeta_z < \zeta_y \leq \zeta_x$ |
| **(F)** | $\zeta_z < \zeta_x < \zeta_y$ |

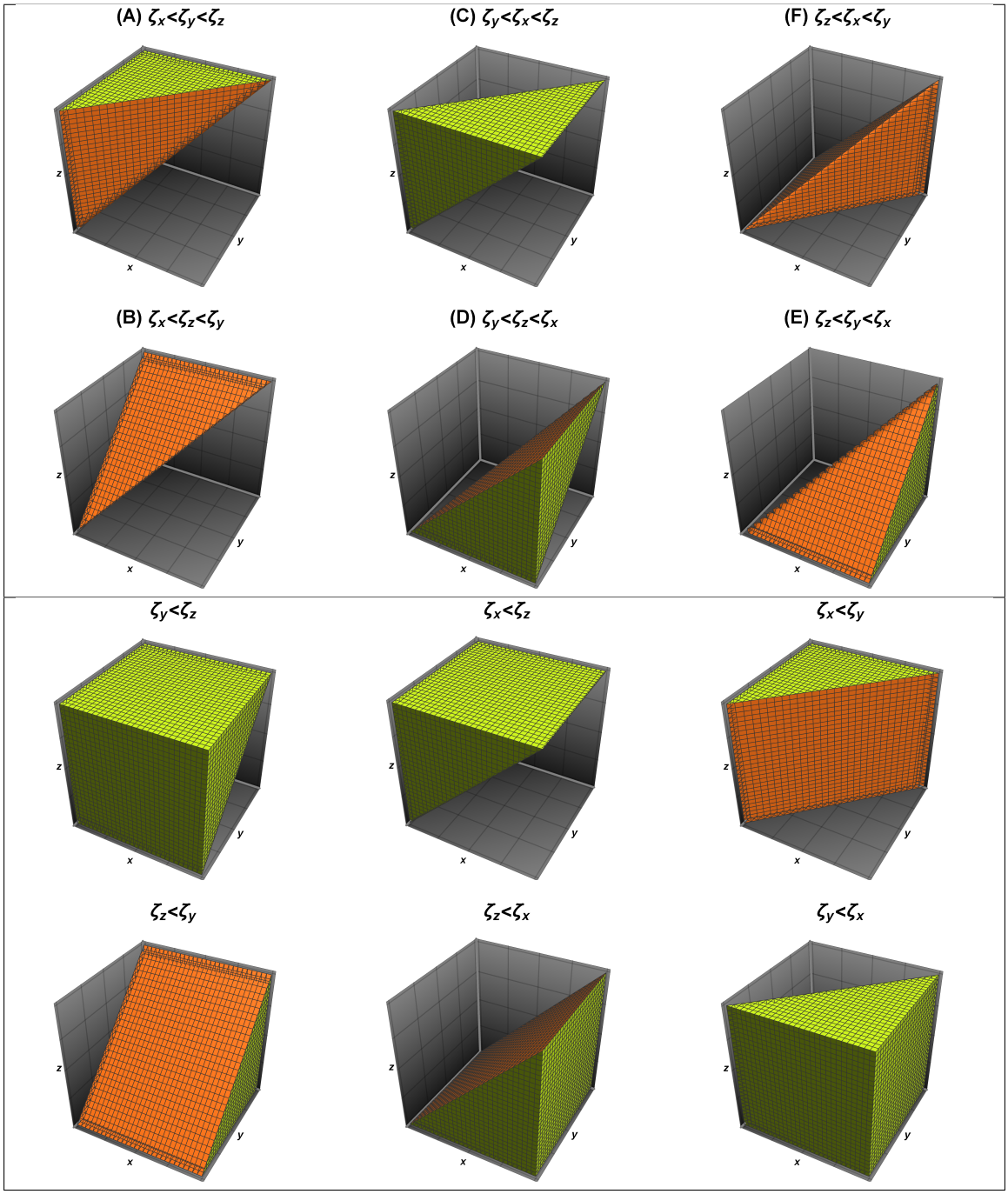


Figure F.2 Depiction of how the domain of dependence of the filtered approximation form varies from octant to octant. Octants 1 and 8 have six regions (Top), while octants 2 through 7 have only a pair of regions each (Bottom). The form of these regions is a consequence of the filtering coordinate inequalities given in Table F.4 coupled with the relative location of each octant in the reference element.

Table F.5 Three-dimensional LSIAC basis functions for octant 1. For notational convenience suppose that $\mathbf{x}$ belong to element $(i, j, k)$ and that the region $S$ and position $P$ of the function are as indicated in the table rows.

| **Region** $S$ | **Basis Function Parameters** | | | | |
|---|---|---|---|---|---|
| | **Basis Function** | **Coefficient** | $\mathbf{arg}_x$ | $\mathbf{arg}_y$ | $\mathbf{arg}_z$ |
| **1A** | $\chi^{S,L}$ | $(2-\zeta_z)/4$ | $w+\zeta_x-\zeta_z/2-(w*\zeta_z)/2$ | $w+\zeta_y-\zeta_z/2-(w*\zeta_z)/2$ | $w-((-1+w)*\zeta_z)/2$ |
| **1A** | $\chi^{S,LM}$ | $(-\zeta_y+\zeta_z)/4$ | $(2+2*\zeta_x-(1+w)*\zeta_y-\zeta_z+w*\zeta_z)/2$ | $(2+\zeta_y-w*\zeta_y+(-1+w)*\zeta_z)/2$ | $(-2-(1+w)*\zeta_y+\zeta_z+w*\zeta_z)/2$ |
| **1A** | $\chi^{S,RM}$ | $(-\zeta_x+\zeta_y)/4$ | $(2+\zeta_x-w*\zeta_x+(-1+w)*\zeta_y)/2$ | $(-2-(1+w)*\zeta_x+\zeta_y+w*\zeta_y)/2$ | $-1-((1+w)*\zeta_x)/2+((-1+w)*\zeta_y)/2+\zeta_z$ |
| **1A** | $\chi^{S,R}$ | $\zeta_x/4$ | $(-2+\zeta_x+w*\zeta_x)/2$ | $-1+((-1+w)*\zeta_x)/2+\zeta_y$ | $-1+((-1+w)*\zeta_x)/2+\zeta_z$ |
| **Region** | **Basis Function Parameters** | | | | |
| | **Basis Function** | **Coefficient** | $\mathbf{arg}_x$ | $\mathbf{arg}_y$ | $\mathbf{arg}_z$ |
| **1B** | $\chi^{S,L}$ | $(2-\zeta_y)/4$ | $w+\zeta_x-\zeta_y/2-(w*\zeta_y)/2$ | $(-(w*(-2+\zeta_y))+\zeta_y)/2$ | $w-\zeta_y/2-(w*\zeta_y)/2+\zeta_z$ |
| **1B** | $\chi^{S,LM}$ | $(\zeta_y-\zeta_z)/4$ | $(2+2*\zeta_x+(-1+w)*\zeta_y-\zeta_z-w*\zeta_z)/2$ | $(-2+(1+w)*\zeta_y-(1+w)*\zeta_z)/2$ | $(2+(-1+w)*\zeta_y+\zeta_z-w*\zeta_z)/2$ |
| **1B** | $\chi^{S,RM}$ | $(-\zeta_x+\zeta_z)/4$ | $(2+\zeta_x-w*\zeta_x+(-1+w)*\zeta_z)/2$ | $(-2-(1+w)*\zeta_x+2*\zeta_y-\zeta_z+w*\zeta_z)/2$ | $(-2-(1+w)*\zeta_x+\zeta_z+w*\zeta_z)/2$ |
| **1B** | $\chi^{S,R}$ | $\zeta_x/4$ | $(-2+\zeta_x+w*\zeta_x)/2$ | $-1+((-1+w)*\zeta_x)/2+\zeta_y$ | $-1+((-1+w)*\zeta_x)/2+\zeta_z$ |
| **Region** | **Basis Function Parameters** | | | | |
| | **Basis Function** | **Coefficient** | $\mathbf{arg}_x$ | $\mathbf{arg}_y$ | $\mathbf{arg}_z$ |
| **1C** | $\chi^{S,L}$ | $(2-\zeta_z)/4$ | $w+\zeta_x-\zeta_z/2-(w*\zeta_z)/2$ | $w+\zeta_y-\zeta_z/2-(w*\zeta_z)/2$ | $w-((-1+w)*\zeta_z)/2$ |
| **1C** | $\chi^{S,LM}$ | $(-\zeta_x+\zeta_z)/4$ | $(2+\zeta_x-w*\zeta_x+(-1+w)*\zeta_z)/2$ | $(2-(1+w)*\zeta_x+2*\zeta_y-\zeta_z+w*\zeta_z)/2$ | $(-2-(1+w)*\zeta_x+\zeta_z+w*\zeta_z)/2$ |
| **1C** | $\chi^{S,RM}$ | $(\zeta_x-\zeta_y)/4$ | $(-2+(1+w)*\zeta_x-(1+w)*\zeta_y)/2$ | $(2+(-1+w)*\zeta_x+\zeta_y-w*\zeta_y)/2$ | $-1+((-1+w)*\zeta_x)/2-((1+w)*\zeta_y)/2+\zeta_z$ |
| **1C** | $\chi^{S,R}$ | $\zeta_y/4$ | $-1+\zeta_x+((-1+w)*\zeta_y)/2$ | $(-2+\zeta_y+w*\zeta_y)/2$ | $-1+((-1+w)*\zeta_y)/2+\zeta_z$ |
| **Region** | **Basis Function Parameters** | | | | |
| | **Basis Function** | **Coefficient** | $\mathbf{arg}_x$ | $\mathbf{arg}_y$ | $\mathbf{arg}_z$ |
| **1D** | $\chi^{S,L}$ | $(2-\zeta_x)/4$ | $(-(w*(-2+\zeta_x))+\zeta_x)/2$ | $w-\zeta_x/2-(w*\zeta_x)/2+\zeta_y$ | $w-\zeta_x/2-(w*\zeta_x)/2+\zeta_z$ |
| **1D** | $\chi^{S,LM}$ | $(\zeta_x-\zeta_z)/4$ | $(-2+(1+w)*\zeta_x-(1+w)*\zeta_z)/2$ | $(2+(-1+w)*\zeta_x+2*\zeta_y-\zeta_z-w*\zeta_z)/2$ | $(2+(-1+w)*\zeta_x+\zeta_z-w*\zeta_z)/2$ |
| **1D** | $\chi^{S,RM}$ | $(-\zeta_y+\zeta_z)/4$ | $(-2+2*\zeta_x-(1+w)*\zeta_y-\zeta_z+w*\zeta_z)/2$ | $(2+\zeta_y-w*\zeta_y+(-1+w)*\zeta_z)/2$ | $(-2-(1+w)*\zeta_y+\zeta_z+w*\zeta_z)/2$ |
| **1D** | $\chi^{S,R}$ | $\zeta_y/4$ | $-1+\zeta_x+((-1+w)*\zeta_y)/2$ | $(-2+\zeta_y+w*\zeta_y)/2$ | $-1+((-1+w)*\zeta_y)/2+\zeta_z$ |
| **Region** | **Basis Function Parameters** | | | | |
| | **Basis Function** | **Coefficient** | $\mathbf{arg}_x$ | $\mathbf{arg}_y$ | $\mathbf{arg}_z$ |
| **1E** | $\chi^{S,L}$ | $(2-\zeta_x)/4$ | $(-(w*(-2+\zeta_x))+\zeta_x)/2$ | $w-\zeta_x/2-(w*\zeta_x)/2+\zeta_y$ | $w-\zeta_x/2-(w*\zeta_x)/2+\zeta_z$ |
| **1E** | $\chi^{S,LM}$ | $(\zeta_x-\zeta_y)/4$ | $(-2+(1+w)*\zeta_x-(1+w)*\zeta_y)/2$ | $(2+(-1+w)*\zeta_x+\zeta_y-w*\zeta_y)/2$ | $1+((-1+w)*\zeta_x)/2-((1+w)*\zeta_y)/2+\zeta_z$ |
| **1E** | $\chi^{S,RM}$ | $(\zeta_y-\zeta_z)/4$ | $(-2+2*\zeta_x+(-1+w)*\zeta_y-\zeta_z-w*\zeta_z)/2$ | $(-2+(1+w)*\zeta_y-(1+w)*\zeta_z)/2$ | $(2+(-1+w)*\zeta_y+\zeta_z-w*\zeta_z)/2$ |
| **1E** | $\chi^{S,R}$ | $\zeta_z/4$ | $-1+\zeta_x+((-1+w)*\zeta_z)/2$ | $-1+\zeta_y+((-1+w)*\zeta_z)/2$ | $(-2+\zeta_z+w*\zeta_z)/2$ |
| **Region** | **Basis Function Parameters** | | | | |
| | **Basis Function** | **Coefficient** | $\mathbf{arg}_x$ | $\mathbf{arg}_y$ | $\mathbf{arg}_z$ |
| **1F** | $\chi^{S,L}$ | $(2-\zeta_y)/4$ | $w+\zeta_x-\zeta_y/2-(w*\zeta_y)/2$ | $(-(w*(-2+\zeta_y))+\zeta_y)/2$ | $w-\zeta_y/2-(w*\zeta_y)/2+\zeta_z$ |
| **1F** | $\chi^{S,LM}$ | $(-\zeta_x+\zeta_y)/4$ | $(2+\zeta_x-w*\zeta_x+(-1+w)*\zeta_y)/2$ | $(-2-(1+w)*\zeta_x+\zeta_y+w*\zeta_y)/2$ | $1-((1+w)*\zeta_x)/2+((-1+w)*\zeta_y)/2+\zeta_z$ |
| **1F** | $\chi^{S,RM}$ | $(\zeta_x-\zeta_z)/4$ | $(-2+(1+w)*\zeta_x-(1+w)*\zeta_z)/2$ | $(-2+(-1+w)*\zeta_x+2*\zeta_y-\zeta_z-w*\zeta_z)/2$ | $(2+(-1+w)*\zeta_x+\zeta_z-w*\zeta_z)/2$ |
| **1F** | $\chi^{S,R}$ | $\zeta_z/4$ | $-1+\zeta_x+((-1+w)*\zeta_z)/2$ | $-1+\zeta_y+((-1+w)*\zeta_z)/2$ | $(-2+\zeta_z+w*\zeta_z)/2$ |

Table F.6 Three-dimensional LSIAC basis functions for octants 2 through 4. For notational convenience suppose that $\mathbf{x}$ belong to element $(i, j, k)$ and that the region $S$ and position $P$ of the function are as indicated in the table rows.

| **Region** | **Basis Function Parameters** | | | | |
|---|---|---|---|---|---|
| | **Basis Function** | **Coefficient** | $\mathbf{arg}_x$ | $\mathbf{arg}_y$ | $\mathbf{arg}_z$ |
| **2E** | $\chi^{S,L}$ | $-\zeta_z/4$ | $-1+\zeta_x-((1+w)*\zeta_z)/2$ | $-1+\zeta_y-((1+w)*\zeta_z)/2$ | $(2+\zeta_z-w*\zeta_z)/2$ |
| **2E** | $\chi^{S,LM}$ | $(2-\zeta_x+\zeta_z)/4$ | $(\zeta_x-\zeta_z+w*(2-\zeta_x+\zeta_z))/2$ | $(-\zeta_x+2*\zeta_y-\zeta_z+w*(2-\zeta_x+\zeta_z))/2$ | $(-\zeta_x+\zeta_z+w*(2-\zeta_x+\zeta_z))/2$ |
| **2E** | $\chi^{S,RM}$ | $(\zeta_x-\zeta_y)/4$ | $(-2+(1+w)*\zeta_x-(1+w)*\zeta_y)/2$ | $(2+(-1+w)*\zeta_x+\zeta_y-w*\zeta_y)/2$ | $1+((-1+w)*\zeta_x)/2-((1+w)*\zeta_y)/2+\zeta_z$ |
| **2E** | $\chi^{S,R}$ | $\zeta_y/4$ | $-1+\zeta_x+((-1+w)*\zeta_y)/2$ | $(-2+\zeta_y+w*\zeta_y)/2$ | $1+((-1+w)*\zeta_y)/2+\zeta_z$ |
| **Region** | **Basis Function Parameters** | | | | |
| | **Basis Function** | **Coefficient** | $\mathbf{arg}_x$ | $\mathbf{arg}_y$ | $\mathbf{arg}_z$ |
| **2F** | $\chi^{S,L}$ | $-\zeta_z/4$ | $-1+\zeta_x-((1+w)*\zeta_z)/2$ | $-1+\zeta_y-((1+w)*\zeta_z)/2$ | $(2+\zeta_z-w*\zeta_z)/2$ |
| **2F** | $\chi^{S,LM}$ | $(2-\zeta_y+\zeta_z)/4$ | $(2*\zeta_x-\zeta_y-\zeta_z+w*(2-\zeta_y+\zeta_z))/2$ | $(\zeta_y-\zeta_z+w*(2-\zeta_y+\zeta_z))/2$ | $(-\zeta_y+\zeta_z+w*(2-\zeta_y+\zeta_z))/2$ |
| **2F** | $\chi^{S,RM}$ | $(-\zeta_x+\zeta_y)/4$ | $(2+\zeta_x-w*\zeta_x+(-1+w)*\zeta_y)/2$ | $(-2-(1+w)*\zeta_x+\zeta_y+w*\zeta_y)/2$ | $1-((1+w)*\zeta_x)/2+((-1+w)*\zeta_y)/2+\zeta_z$ |
| **2F** | $\chi^{S,R}$ | $\zeta_x/4$ | $(-2+\zeta_x+w*\zeta_x)/2$ | $-1+((-1+w)*\zeta_x)/2+\zeta_y$ | $1+((-1+w)*\zeta_x)/2+\zeta_z$ |
| **Region** | **Basis Function Parameters** | | | | |
| | **Basis Function** | **Coefficient** | $\mathbf{arg}_x$ | $\mathbf{arg}_y$ | $\mathbf{arg}_z$ |
| **3A** | $\chi^{S,L}$ | $-\zeta_x/4$ | $(2+\zeta_x-w*\zeta_x)/2$ | $-1-((1+w)*\zeta_x)/2+\zeta_y$ | $-1-((1+w)*\zeta_x)/2+\zeta_z$ |
| **3A** | $\chi^{S,LM}$ | $(2+\zeta_x-\zeta_z)/4$ | $(\zeta_x+w*(2+\zeta_x-\zeta_z)-\zeta_z)/2$ | $(-\zeta_x+2*\zeta_y+w*(2+\zeta_x-\zeta_z)-\zeta_z)/2$ | $(-\zeta_x+w*(2+\zeta_x-\zeta_z)+\zeta_z)/2$ |
| **3A** | $\chi^{S,RM}$ | $(-\zeta_y+\zeta_z)/4$ | $(2+2*\zeta_x-(1+w)*\zeta_y-\zeta_z+w*\zeta_z)/2$ | $(2+\zeta_y-w*\zeta_y+(-1+w)*\zeta_z)/2$ | $(-2-(1+w)*\zeta_y+\zeta_z+w*\zeta_z)/2$ |
| **3A** | $\chi^{S,R}$ | $\zeta_y/4$ | $1+\zeta_x+((-1+w)*\zeta_y)/2$ | $(-2+\zeta_y+w*\zeta_y)/2$ | $-1+((-1+w)*\zeta_y)/2+\zeta_z$ |
| **Region** | **Basis Function Parameters** | | | | |
| | **Basis Function** | **Coefficient** | $\mathbf{arg}_x$ | $\mathbf{arg}_y$ | $\mathbf{arg}_z$ |
| **3B** | $\chi^{S,L}$ | $-\zeta_x/4$ | $(2+\zeta_x-w*\zeta_x)/2$ | $-1-((1+w)*\zeta_x)/2+\zeta_y$ | $-1-((1+w)*\zeta_x)/2+\zeta_z$ |
| **3B** | $\chi^{S,LM}$ | $(2+\zeta_x-\zeta_y)/4$ | $(\zeta_x+w*(2+\zeta_x-\zeta_y)-\zeta_y)/2$ | $(-\zeta_x+w*(2+\zeta_x-\zeta_y)+\zeta_y)/2$ | $(-\zeta_x+w*(2+\zeta_x-\zeta_y)-\zeta_y+2*\zeta_z)/2$ |
| **3B** | $\chi^{S,RM}$ | $(\zeta_y-\zeta_z)/4$ | $(2+2*\zeta_x+(-1+w)*\zeta_y-\zeta_z-w*\zeta_z)/2$ | $(-2+(1+w)*\zeta_y-(1+w)*\zeta_z)/2$ | $(2+(-1+w)*\zeta_y+\zeta_z-w*\zeta_z)/2$ |
| **3B** | $\chi^{S,R}$ | $\zeta_z/4$ | $1+\zeta_x+((-1+w)*\zeta_z)/2$ | $-1+\zeta_y+((-1+w)*\zeta_z)/2$ | $(-2+\zeta_z+w*\zeta_z)/2$ |
| **Region** | **Basis Function Parameters** | | | | |
| | **Basis Function** | **Coefficient** | $\mathbf{arg}_x$ | $\mathbf{arg}_y$ | $\mathbf{arg}_z$ |
| **4B** | $\chi^{S,L}$ | $-\zeta_z/4$ | $1+\zeta_x-((1+w)*\zeta_z)/2$ | $-1+\zeta_y-((1+w)*\zeta_z)/2$ | $(2+\zeta_z-w*\zeta_z)/2$ |
| **4B** | $\chi^{S,LM}$ | $(-\zeta_x+\zeta_z)/4$ | $(2+\zeta_x-w*\zeta_x+(-1+w)*\zeta_z)/2$ | $(-2-(1+w)*\zeta_x+2*\zeta_y-\zeta_z+w*\zeta_z)/2$ | $(-2-(1+w)*\zeta_x+\zeta_z+w*\zeta_z)/2$ |
| **4B** | $\chi^{S,RM}$ | $(2+\zeta_x-\zeta_y)/4$ | $(\zeta_x+w*(2+\zeta_x-\zeta_y)-\zeta_y)/2$ | $(-\zeta_x+w*(2+\zeta_x-\zeta_y)+\zeta_y)/2$ | $(-\zeta_x+w*(2+\zeta_x-\zeta_y)-\zeta_y+2*\zeta_z)/2$ |
| **4B** | $\chi^{S,R}$ | $\zeta_y/4$ | $1+\zeta_x+((-1+w)*\zeta_y)/2$ | $(-2+\zeta_y+w*\zeta_y)/2$ | $1+((-1+w)*\zeta_y)/2+\zeta_z$ |
| **Region** | **Basis Function Parameters** | | | | |
| | **Basis Function** | **Coefficient** | $\mathbf{arg}_x$ | $\mathbf{arg}_y$ | $\mathbf{arg}_z$ |
| **4F** | $\chi^{S,L}$ | $-\zeta_x/4$ | $(2+\zeta_x-w*\zeta_x)/2$ | $-1-((1+w)*\zeta_x)/2+\zeta_y$ | $1-((1+w)*\zeta_x)/2+\zeta_z$ |
| **4F** | $\chi^{S,LM}$ | $(\zeta_x-\zeta_z)/4$ | $(-2+(1+w)*\zeta_x-(1+w)*\zeta_z)/2$ | $(-2+(-1+w)*\zeta_x+2*\zeta_y-\zeta_z-w*\zeta_z)/2$ | $(2+(-1+w)*\zeta_x+\zeta_z-w*\zeta_z)/2$ |
| **4F** | $\chi^{S,RM}$ | $(2-\zeta_y+\zeta_z)/4$ | $(2*\zeta_x-\zeta_y-\zeta_z+w*(2-\zeta_y+\zeta_z))/2$ | $(\zeta_y-\zeta_z+w*(2-\zeta_y+\zeta_z))/2$ | $(-\zeta_y+\zeta_z+w*(2-\zeta_y+\zeta_z))/2$ |
| **4F** | $\chi^{S,R}$ | $\zeta_y/4$ | $1+\zeta_x+((-1+w)*\zeta_y)/2$ | $(-2+\zeta_y+w*\zeta_y)/2$ | $1+((-1+w)*\zeta_y)/2+\zeta_z$ |

Table F.7 Three-dimensional LSIAC basis functions for octants 5 through 7. For notational convenience suppose that $\mathbf{x}$ belong to element $(i, j, k)$ and that the region $S$ and position $P$ of the function are as indicated in the table rows.

| **Region** | **Basis Function Parameters** | | | | |
|---|---|---|---|---|---|
| | **Basis Function** | **Coefficient** | $\mathbf{arg}_x$ | $\mathbf{arg}_y$ | $\mathbf{arg}_z$ |
| **5C** | $\chi^{S,L}$ | $-\zeta_y/4$ | $-1+\zeta_x-((1+w)*\zeta_y)/2$ | $(2+\zeta_y-w*\zeta_y)/2$ | $-1-((1+w)*\zeta_y)/2+\zeta_z$ |
| **5C** | $\chi^{S,LM}$ | $(2+\zeta_y-\zeta_z)/4$ | $(2*\zeta_x-\zeta_y+w*(2+\zeta_y-\zeta_z)-\zeta_z)/2$ | $(\zeta_y+w*(2+\zeta_y-\zeta_z)-\zeta_z)/2$ | $(-\zeta_y+w*(2+\zeta_y-\zeta_z)+\zeta_z)/2$ |
| **5C** | $\chi^{S,RM}$ | $(-\zeta_x+\zeta_z)/4$ | $(2+\zeta_x-w*\zeta_x+(-1+w)*\zeta_z)/2$ | $(2-(1+w)*\zeta_x+2*\zeta_y-\zeta_z+w*\zeta_z)/2$ | $(-2-(1+w)*\zeta_x+\zeta_z+w*\zeta_z)/2$ |
| **5C** | $\chi^{S,R}$ | $\zeta_x/4$ | $(-2+\zeta_x+w*\zeta_x)/2$ | $1+((-1+w)*\zeta_x)/2+\zeta_y$ | $-1+((-1+w)*\zeta_x)/2+\zeta_z$ |
| **Region** | **Basis Function Parameters** | | | | |
| | **Basis Function** | **Coefficient** | $\mathbf{arg}_x$ | $\mathbf{arg}_y$ | $\mathbf{arg}_z$ |
| **5D** | $\chi^{S,L}$ | $-\zeta_y/4$ | $-1+\zeta_x-((1+w)*\zeta_y)/2$ | $(2+\zeta_y-w*\zeta_y)/2$ | $-1-((1+w)*\zeta_y)/2+\zeta_z$ |
| **5D** | $\chi^{S,LM}$ | $(2-\zeta_x+\zeta_y)/4$ | $(\zeta_x-\zeta_y+w*(2-\zeta_x+\zeta_y))/2$ | $(-\zeta_x+\zeta_y+w*(2-\zeta_x+\zeta_y))/2$ | $(-\zeta_x-\zeta_y+w*(2-\zeta_x+\zeta_y)+2*\zeta_z)/2$ |
| **5D** | $\chi^{S,RM}$ | $(\zeta_x-\zeta_z)/4$ | $(-2+(1+w)*\zeta_x-(1+w)*\zeta_z)/2$ | $(2+(-1+w)*\zeta_x+2*\zeta_y-\zeta_z-w*\zeta_z)/2$ | $(2+(-1+w)*\zeta_x+\zeta_z-w*\zeta_z)/2$ |
| **5D** | $\chi^{S,R}$ | $\zeta_z/4$ | $-1+\zeta_x+((-1+w)*\zeta_z)/2$ | $1+\zeta_y+((-1+w)*\zeta_z)/2$ | $(-2+\zeta_z+w*\zeta_z)/2$ |
| **Region** | **Basis Function Parameters** | | | | |
| | **Basis Function** | **Coefficient** | $\mathbf{arg}_x$ | $\mathbf{arg}_y$ | $\mathbf{arg}_z$ |
| **6D** | $\chi^{S,L}$ | $-\zeta_z/4$ | $-1+\zeta_x-((1+w)*\zeta_z)/2$ | $1+\zeta_y-((1+w)*\zeta_z)/2$ | $(2+\zeta_z-w*\zeta_z)/2$ |
| **6D** | $\chi^{S,LM}$ | $(-\zeta_y+\zeta_z)/4$ | $(-2+2*\zeta_x-(1+w)*\zeta_y-\zeta_z+w*\zeta_z)/2$ | $(2+\zeta_y-w*\zeta_y+(-1+w)*\zeta_z)/2$ | $(-2-(1+w)*\zeta_y+\zeta_z+w*\zeta_z)/2$ |
| **6D** | $\chi^{S,RM}$ | $(2-\zeta_x+\zeta_y)/4$ | $(\zeta_x-\zeta_y+w*(2-\zeta_x+\zeta_y))/2$ | $(-\zeta_x+\zeta_y+w*(2-\zeta_x+\zeta_y))/2$ | $(-\zeta_x-\zeta_y+w*(2-\zeta_x+\zeta_y)+2*\zeta_z)/2$ |
| **6D** | $\chi^{S,R}$ | $\zeta_x/4$ | $(-2+\zeta_x+w*\zeta_x)/2$ | $1+((-1+w)*\zeta_x)/2+\zeta_y$ | $1+((-1+w)*\zeta_x)/2+\zeta_z$ |
| **Region** | **Basis Function Parameters** | | | | |
| | **Basis Function** | **Coefficient** | $\mathbf{arg}_x$ | $\mathbf{arg}_y$ | $\mathbf{arg}_z$ |
| **6E** | $\chi^{S,L}$ | $-\zeta_y/4$ | $-1+\zeta_x-((1+w)*\zeta_y)/2$ | $(2+\zeta_y-w*\zeta_y)/2$ | $1-((1+w)*\zeta_y)/2+\zeta_z$ |
| **6E** | $\chi^{S,LM}$ | $(\zeta_y-\zeta_z)/4$ | $(-2+2*\zeta_x+(-1+w)*\zeta_y-\zeta_z-w*\zeta_z)/2$ | $(-2+(1+w)*\zeta_y-(1+w)*\zeta_z)/2$ | $(2+(-1+w)*\zeta_y+\zeta_z-w*\zeta_z)/2$ |
| **6E** | $\chi^{S,RM}$ | $(2-\zeta_x+\zeta_z)/4$ | $(\zeta_x-\zeta_z+w*(2-\zeta_x+\zeta_z))/2$ | $(-\zeta_x+2*\zeta_y-\zeta_z+w*(2-\zeta_x+\zeta_z))/2$ | $(-\zeta_x+\zeta_z+w*(2-\zeta_x+\zeta_z))/2$ |
| **6E** | $\chi^{S,R}$ | $\zeta_x/4$ | $(-2+\zeta_x+w*\zeta_x)/2$ | $1+((-1+w)*\zeta_x)/2+\zeta_y$ | $1+((-1+w)*\zeta_x)/2+\zeta_z$ |
| **Region** | **Basis Function Parameters** | | | | |
| | **Basis Function** | **Coefficient** | $\mathbf{arg}_x$ | $\mathbf{arg}_y$ | $\mathbf{arg}_z$ |
| **7A** | $\chi^{S,L}$ | $-\zeta_y/4$ | $1+\zeta_x-((1+w)*\zeta_y)/2$ | $(2+\zeta_y-w*\zeta_y)/2$ | $-1-((1+w)*\zeta_y)/2+\zeta_z$ |
| **7A** | $\chi^{S,LM}$ | $(-\zeta_x+\zeta_y)/4$ | $(2+\zeta_x-w*\zeta_x+(-1+w)*\zeta_y)/2$ | $(-2-(1+w)*\zeta_x+\zeta_y+w*\zeta_y)/2$ | $-1-((1+w)*\zeta_x)/2+((-1+w)*\zeta_y)/2+\zeta_z$ |
| **7A** | $\chi^{S,RM}$ | $(2+\zeta_x-\zeta_z)/4$ | $(\zeta_x+w*(2+\zeta_x-\zeta_z)-\zeta_z)/2$ | $(-\zeta_x+2*\zeta_y+w*(2+\zeta_x-\zeta_z)-\zeta_z)/2$ | $(-\zeta_x+w*(2+\zeta_x-\zeta_z)+\zeta_z)/2$ |
| **7A** | $\chi^{S,R}$ | $\zeta_z/4$ | $1+\zeta_x+((-1+w)*\zeta_z)/2$ | $1+\zeta_y+((-1+w)*\zeta_z)/2$ | $(-2+\zeta_z+w*\zeta_z)/2$ |
| **Region** | **Basis Function Parameters** | | | | |
| | **Basis Function** | **Coefficient** | $\mathbf{arg}_x$ | $\mathbf{arg}_y$ | $\mathbf{arg}_z$ |
| **7C** | $\chi^{S,L}$ | $-\zeta_x/4$ | $(2+\zeta_x-w*\zeta_x)/2$ | $1-((1+w)*\zeta_x)/2+\zeta_y$ | $-1-((1+w)*\zeta_x)/2+\zeta_z$ |
| **7C** | $\chi^{S,LM}$ | $(\zeta_x-\zeta_y)/4$ | $(-2+(1+w)*\zeta_x-(1+w)*\zeta_y)/2$ | $(2+(-1+w)*\zeta_x+\zeta_y-w*\zeta_y)/2$ | $-1+((-1+w)*\zeta_x)/2-((1+w)*\zeta_y)/2+\zeta_z$ |
| **7C** | $\chi^{S,RM}$ | $(2+\zeta_y-\zeta_z)/4$ | $(2*\zeta_x-\zeta_y+w*(2+\zeta_y-\zeta_z)-\zeta_z)/2$ | $(\zeta_y+w*(2+\zeta_y-\zeta_z)-\zeta_z)/2$ | $(-\zeta_y+w*(2+\zeta_y-\zeta_z)+\zeta_z)/2$ |
| **7C** | $\chi^{S,R}$ | $\zeta_z/4$ | $1+\zeta_x+((-1+w)*\zeta_z)/2$ | $1+\zeta_y+((-1+w)*\zeta_z)/2$ | $(-2+\zeta_z+w*\zeta_z)/2$ |

Table F.8 Three-dimensional LSIAC basis functions for octant 8. For notational convenience suppose that $\mathbf{x}$ belong to element $(i, j, k)$ and that the region $S$ and position $P$ of the function are as indicated in the table rows.

| **Region** | **Basis Function Parameters** | | | | |
|---|---|---|---|---|---|
| | **Basis Function** | **Coefficient** | $\mathbf{arg}_x$ | $\mathbf{arg}_y$ | $\mathbf{arg}_z$ |
| **8A** | $\chi^{S,L}$ | $-\zeta_z/4$ | $1+\zeta_x-((1+w)*\zeta_z)/2$ | $1+\zeta_y-((1+w)*\zeta_z)/2$ | $(2+\zeta_z-w*\zeta_z)/2$ |
| **8A** | $\chi^{S,LM}$ | $(-\zeta_y+\zeta_z)/4$ | $(2+2*\zeta_x-(1+w)*\zeta_y-\zeta_z+w*\zeta_z)/2$ | $(2+\zeta_y-w*\zeta_y+(-1+w)*\zeta_z)/2$ | $(-2-(1+w)*\zeta_y+\zeta_z+w*\zeta_z)/2$ |
| **8A** | $\chi^{S,RM}$ | $(-\zeta_x+\zeta_y)/4$ | $(2+\zeta_x-w*\zeta_x+(-1+w)*\zeta_y)/2$ | $(-2-(1+w)*\zeta_x+\zeta_y+w*\zeta_y)/2$ | $-1-((1+w)*\zeta_x)/2+((-1+w)*\zeta_y)/2+\zeta_z$ |
| **8A** | $\chi^{S,R}$ | $(2+\zeta_x)/4$ | $(\zeta_x+w*(2+\zeta_x))/2$ | $w-\zeta_x/2+(w*\zeta_x)/2+\zeta_y$ | $w-\zeta_x/2+(w*\zeta_x)/2+\zeta_z$ |
| **Region** | **Basis Function Parameters** | | | | |
| | **Basis Function** | **Coefficient** | $\mathbf{arg}_x$ | $\mathbf{arg}_y$ | $\mathbf{arg}_z$ |
| **8B** | $\chi^{S,L}$ | $-\zeta_y/4$ | $1+\zeta_x-((1+w)*\zeta_y)/2$ | $(2+\zeta_y-w*\zeta_y)/2$ | $1-((1+w)*\zeta_y)/2+\zeta_z$ |
| **8B** | $\chi^{S,LM}$ | $(\zeta_y-\zeta_z)/4$ | $(2+2*\zeta_x+(-1+w)*\zeta_y-\zeta_z-w*\zeta_z)/2$ | $(-2+(1+w)*\zeta_y-(1+w)*\zeta_z)/2$ | $(2+(-1+w)*\zeta_y+\zeta_z-w*\zeta_z)/2$ |
| **8B** | $\chi^{S,RM}$ | $(-\zeta_x+\zeta_z)/4$ | $(2+\zeta_x-w*\zeta_x+(-1+w)*\zeta_z)/2$ | $(-2-(1+w)*\zeta_x+2*\zeta_y-\zeta_z+w*\zeta_z)/2$ | $(-2-(1+w)*\zeta_x+\zeta_z+w*\zeta_z)/2$ |
| **8B** | $\chi^{S,R}$ | $(2+\zeta_x)/4$ | $(\zeta_x+w*(2+\zeta_x))/2$ | $w-\zeta_x/2+(w*\zeta_x)/2+\zeta_y$ | $w-\zeta_x/2+(w*\zeta_x)/2+\zeta_z$ |
| **Region** | **Basis Function Parameters** | | | | |
| | **Basis Function** | **Coefficient** | $\mathbf{arg}_x$ | $\mathbf{arg}_y$ | $\mathbf{arg}_z$ |
| **8C** | $\chi^{S,L}$ | $-\zeta_z/4$ | $1+\zeta_x-((1+w)*\zeta_z)/2$ | $1+\zeta_y-((1+w)*\zeta_z)/2$ | $(2+\zeta_z-w*\zeta_z)/2$ |
| **8C** | $\chi^{S,LM}$ | $(-\zeta_x+\zeta_z)/4$ | $(2+\zeta_x-w*\zeta_x+(-1+w)*\zeta_z)/2$ | $(2-(1+w)*\zeta_x+2*\zeta_y-\zeta_z+w*\zeta_z)/2$ | $(-2-(1+w)*\zeta_x+\zeta_z+w*\zeta_z)/2$ |
| **8C** | $\chi^{S,RM}$ | $(\zeta_x-\zeta_y)/4$ | $(-2+(1+w)*\zeta_x-(1+w)*\zeta_y)/2$ | $(2+(-1+w)*\zeta_x+\zeta_y-w*\zeta_y)/2$ | $-1+((-1+w)*\zeta_x)/2-((1+w)*\zeta_y)/2+\zeta_z$ |
| **8C** | $\chi^{S,R}$ | $(2+\zeta_y)/4$ | $w+\zeta_x-\zeta_y/2+(w*\zeta_y)/2$ | $(\zeta_y+w*(2+\zeta_y))/2$ | $w-\zeta_y/2+(w*\zeta_y)/2+\zeta_z$ |
| **Region** | **Basis Function Parameters** | | | | |
| | **Basis Function** | **Coefficient** | $\mathbf{arg}_x$ | $\mathbf{arg}_y$ | $\mathbf{arg}_z$ |
| **8D** | $\chi^{S,L}$ | $-\zeta_x/4$ | $(2+\zeta_x-w*\zeta_x)/2$ | $1-((1+w)*\zeta_x)/2+\zeta_y$ | $1-((1+w)*\zeta_x)/2+\zeta_z$ |
| **8D** | $\chi^{S,LM}$ | $(\zeta_x-\zeta_z)/4$ | $(-2+(1+w)*\zeta_x-(1+w)*\zeta_z)/2$ | $(2+(-1+w)*\zeta_x+2*\zeta_y-\zeta_z-w*\zeta_z)/2$ | $(2+(-1+w)*\zeta_x+\zeta_z-w*\zeta_z)/2$ |
| **8D** | $\chi^{S,RM}$ | $(-\zeta_y+\zeta_z)/4$ | $(-2+2*\zeta_x-(1+w)*\zeta_y-\zeta_z+w*\zeta_z)/2$ | $(2+\zeta_y-w*\zeta_y+(-1+w)*\zeta_z)/2$ | $(-2-(1+w)*\zeta_y+\zeta_z+w*\zeta_z)/2$ |
| **8D** | $\chi^{S,R}$ | $(2+\zeta_y)/4$ | $w+\zeta_x-\zeta_y/2+(w*\zeta_y)/2$ | $(\zeta_y+w*(2+\zeta_y))/2$ | $w-\zeta_y/2+(w*\zeta_y)/2+\zeta_z$ |
| **Region** | **Basis Function Parameters** | | | | |
| | **Basis Function** | **Coefficient** | $\mathbf{arg}_x$ | $\mathbf{arg}_y$ | $\mathbf{arg}_z$ |
| **8E** | $\chi^{S,L}$ | $-\zeta_x/4$ | $(2+\zeta_x-w*\zeta_x)/2$ | $1-((1+w)*\zeta_x)/2+\zeta_y$ | $1-((1+w)*\zeta_x)/2+\zeta_z$ |
| **8E** | $\chi^{S,LM}$ | $(\zeta_x-\zeta_y)/4$ | $(-2+(1+w)*\zeta_x-(1+w)*\zeta_y)/2$ | $(2+(-1+w)*\zeta_x+\zeta_y-w*\zeta_y)/2$ | $1+((-1+w)*\zeta_x)/2-((1+w)*\zeta_y)/2+\zeta_z$ |
| **8E** | $\chi^{S,RM}$ | $(\zeta_y-\zeta_z)/4$ | $(-2+2*\zeta_x+(-1+w)*\zeta_y-\zeta_z-w*\zeta_z)/2$ | $(-2+(1+w)*\zeta_y-(1+w)*\zeta_z)/2$ | $(2+(-1+w)*\zeta_y+\zeta_z-w*\zeta_z)/2$ |
| **8E** | $\chi^{S,R}$ | $(2+\zeta_z)/4$ | $w+\zeta_x-\zeta_z/2+(w*\zeta_z)/2$ | $w+\zeta_y-\zeta_z/2+(w*\zeta_z)/2$ | $w+((1+w)*\zeta_z)/2$ |
| **Region** | **Basis Function Parameters** | | | | |
| | **Basis Function** | **Coefficient** | $\mathbf{arg}_x$ | $\mathbf{arg}_y$ | $\mathbf{arg}_z$ |
| **8F** | $\chi^{S,L}$ | $-\zeta_y/4$ | $1+\zeta_x-((1+w)*\zeta_y)/2$ | $(2+\zeta_y-w*\zeta_y)/2$ | $1-((1+w)*\zeta_y)/2+\zeta_z$ |
| **8F** | $\chi^{S,LM}$ | $(-\zeta_x+\zeta_y)/4$ | $(2+\zeta_x-w*\zeta_x+(-1+w)*\zeta_y)/2$ | $(-2-(1+w)*\zeta_x+\zeta_y+w*\zeta_y)/2$ | $1-((1+w)*\zeta_x)/2+((-1+w)*\zeta_y)/2+\zeta_z$ |
| **8F** | $\chi^{S,RM}$ | $(\zeta_x-\zeta_z)/4$ | $(-2+(1+w)*\zeta_x-(1+w)*\zeta_z)/2$ | $(-2+(-1+w)*\zeta_x+2*\zeta_y-\zeta_z-w*\zeta_z)/2$ | $(2+(-1+w)*\zeta_x+\zeta_z-w*\zeta_z)/2$ |
| **8F** | $\chi^{S,R}$ | $(2+\zeta_z)/4$ | $w+\zeta_x-\zeta_z/2+(w*\zeta_z)/2$ | $w+\zeta_y-\zeta_z/2+(w*\zeta_z)/2$ | $w+((1+w)*\zeta_z)/2$ |

Table F.9 Table describing how the shifting arguments $(q_x, q_y, q_z)$ of the $a$ coefficients vary by region.

| **Region S** \ **Arguments of** $\mathbf{a}_\alpha(\mathbf{q_x}, \mathbf{q_y}, \mathbf{q_z})$ | $\mathbf{P = L}$ | $\mathbf{P = LM}$ | $\mathbf{P = RM}$ | $\mathbf{P = R}$ |
|---|---|---|---|---|
| **1A** | $(0,0,0)$ | $(0,0,1)$ | $(0,1,1)$ | $(1,1,1)$ |
| **1B** | $(0,0,0)$ | $(0,1,0)$ | $(0,1,1)$ | $(1,1,1)$ |
| **1C** | $(0,0,0)$ | $(0,0,1)$ | $(1,0,1)$ | $(1,1,1)$ |
| **1D** | $(0,0,0)$ | $(1,0,0)$ | $(1,0,1)$ | $(1,1,1)$ |
| **1E** | $(0,0,0)$ | $(1,0,0)$ | $(1,1,0)$ | $(1,1,1)$ |
| **1F** | $(0,0,0)$ | $(0,1,0)$ | $(1,1,0)$ | $(1,1,1)$ |
| **2E** | $(0,0,-1)$ | $(0,0,0)$ | $(1,0,0)$ | $(1,1,0)$ |
| **2F** | $(0,0,-1)$ | $(0,0,0)$ | $(0,1,0)$ | $(1,1,0)$ |
| **3A** | $(-1,0,0)$ | $(0,0,0)$ | $(0,0,1)$ | $(0,1,1)$ |
| **3B** | $(-1,0,0)$ | $(0,0,0)$ | $(0,1,0)$ | $(0,1,1)$ |
| **4B** | $(-1,0,-1)$ | $(-1,0,0)$ | $(0,0,0)$ | $(0,1,0)$ |
| **4F** | $(-1,0,-1)$ | $(0,0,-1)$ | $(0,0,0)$ | $(0,1,0)$ |
| **5C** | $(0,-1,0)$ | $(0,0,0)$ | $(0,0,1)$ | $(1,0,1)$ |
| **5D** | $(0,-1,0)$ | $(0,0,0)$ | $(1,0,0)$ | $(1,0,1)$ |
| **6D** | $(0,-1,-1)$ | $(0,-1,0)$ | $(0,0,0)$ | $(1,0,0)$ |
| **6E** | $(0,-1,-1)$ | $(0,0,-1)$ | $(0,0,0)$ | $(1,0,0)$ |
| **7A** | $(-1,-1,0)$ | $(-1,0,0)$ | $(0,0,0)$ | $(0,0,1)$ |
| **7C** | $(-1,-1,0)$ | $(0,-1,0)$ | $(0,0,0)$ | $(0,0,1)$ |
| **8A** | $(-1,-1,-1)$ | $(-1,-1,0)$ | $(-1,0,0)$ | $(0,0,0)$ |
| **8B** | $(-1,-1,-1)$ | $(-1,0,-1)$ | $(-1,0,0)$ | $(0,0,0)$ |
| **8C** | $(-1,-1,-1)$ | $(-1,-1,0)$ | $(0,-1,0)$ | $(0,0,0)$ |
| **8D** | $(-1,-1,-1)$ | $(0,-1,-1)$ | $(0,-1,0)$ | $(0,0,0)$ |
| **8E** | $(-1,-1,-1)$ | $(0,-1,-1)$ | $(0,0,-1)$ | $(0,0,0)$ |
| **8F** | $(-1,-1,-1)$ | $(-1,0,-1)$ | $(0,0,-1)$ | $(0,0,0)$ |

### F.2.1 Three-dimensional LSIAC-MRA Transition Matrices

Unlike in the tensor-product case, constructing these matrices is easier if we group modal coefficients by element. Denote the $N^3(p+1)^3 \times 1$ vector of modal coefficients by

$$\vec{u} = \begin{bmatrix} \vec{u}^1 & | & \dots & | & \vec{u}^N \end{bmatrix}^T,$$

where

$$\vec{u}^k = \begin{bmatrix} \vec{u}^{1,k} \dots \vec{u}^{N,k} \end{bmatrix}^T,$$

$$\vec{u}^{j,k} = \begin{bmatrix} \vec{u}^{1,j,k} & \dots & \vec{u}^{N,j,k} \end{bmatrix}^T,$$

$$\vec{u}^{i,j,k} = \begin{bmatrix} \vec{u}_0^{i,j,k} & \dots & \vec{u}_p^{i,j,k} \end{bmatrix}^T,$$

$$\vec{u}_{kz}^{i,j,k} = \begin{bmatrix} \vec{u}_{0,kz}^{i,j,k} & \dots & \vec{u}_{p,kz}^{i,j,k} \end{bmatrix}^T,$$

and

$$\vec{u}_{ky,kz}^{i,j,k} = \begin{bmatrix} \vec{u}_{0,ky,kz}^{i,j,k} & \dots & \vec{u}_{p,ky,kz}^{i,j,k} \end{bmatrix}^T.$$

### F.2.2 Sifting Matrices

All the coefficients fit the form $a_{kx,ky,kz}^{i,j,k}(q_x, q_y, q_z) = \sum_{\gamma=-p}^{p} c_\gamma u_{kx,ky,kz}^{i+\gamma+q_x, j+\gamma+q_y, k+\gamma+q_z}$ for $q_x, q_y, q_z \in \{-1, 0, 1\}$. To compute these coefficients I will first construct a $(2p+3)^3(p+1)^3 \times N^3(p+1)^3$ selection matrix $\mathscr{S}^{i,j,k}$ which presuming periodic boundary conditions will select all modes relevant to that element. This matrix will be constructed by composition. Define the $(2p+3)N^2(p+1)^3 \times N^3(p+1)^3$ matrix $S^k$ by

$$S^k = \begin{bmatrix} \delta_k(1,1)\mathbb{I}_{N^2(p+1)^3 \times N^2(p+1)^3} & \dots & \delta_k(1,N)\mathbb{I}_{N^2(p+1)^3 \times N^2(p+1)^3} \\ \vdots & & \vdots \\ \delta_k(2p+3,1)\mathbb{I}_{N^2(p+1)^3 \times N^2(p+1)^3} & \dots & \delta_k(2p+3,N)\mathbb{I}_{N^2(p+1)^3 \times N^2(p+1)^3} \end{bmatrix}$$

where

$$\delta_k(m,r) = \begin{cases} 1, & k \neq N \ \& \ p+2-m+r = k \ mod(N) \\ 1, & k = N \ \& \ p+2-m+r = k \text{ or } 0 \\ 0, & Else \end{cases},$$

for $m = 1 : (2p+3)$ and $r = 1 : N$. This matrix selects the relevant $j$-coordinates:

$$S^k \vec{u} = [u^{k-(p+1)} \ \dots \ u^{k+(p+1)}]^T.$$

Next I construct an $N(2p+3)(p+1)^3 \times N^2(p+1)^3$ matrix $\tilde{S}^j$ to select the relevant $j-$coordinates from within each vector. Define

$$\tilde{S}^j = \begin{bmatrix} \delta_j(1,1)\mathbb{I}_{N(p+1)^3 \times N(p+1)^3} & \dots & \delta_j(1,N)\mathbb{I}_{N(p+1)^3 \times N(p+1)^3} \\ \vdots & & \vdots \\ \delta_j(2p+3,1)\mathbb{I}_{N(p+1)^3 \times N(p+1)^3} & \dots & \delta_j(2p+3,N)\mathbb{I}_{N(p+1)^3 \times N(p+1)^3} \end{bmatrix}$$

for $m = 1 : (2p+3)$ and $r = 1 : N$.

Next I construct an $(2p+3)(p+1)^3 \times N(p+1)^3$ matrix $\hat{S}^i$ to select the relevant $i-$coordinates from within each vector. Define

$$\hat{S}^i = \begin{bmatrix} \delta_i(1,1)\mathbb{I}_{(p+1)^3 \times (p+1)^3} & \dots & \delta_i(1,N)\mathbb{I}_{(p+1)^3 \times (p+1)^3} \\ \vdots & & \vdots \\ \delta_i(2p+3,1)\mathbb{I}_{(p+1)^3 \times (p+1)^3} & \dots & \delta_i(2p+3,N)\mathbb{I}_{(p+1)^3 \times (p+1)^3} \end{bmatrix}$$

for $m = 1 : (2p+3)$ and $r = 1 : N$.

Hence, defining $\mathscr{S}^{i,j,k}$ to be the $(2p+3)^3(p+1)^3 \times N^3(p+1)^3$ matrix given by

$$\mathscr{S}^{i,j,k} = \begin{bmatrix} \hat{S}^i & & 0 \\ & \ddots & \\ 0 & & \hat{S}^i \end{bmatrix} \begin{bmatrix} \tilde{S}^j & & 0 \\ & \ddots & \\ 0 & & \tilde{S}^j \end{bmatrix} S^k,$$

where there are $(2p+3)^2$ $\hat{S}^i$ matrices and $(2p+3)$ $\tilde{S}^j$ matrices on their respective diagonals. We have

$$\mathscr{S}^{i,j,k} u = \Big[ [u^{i-(p+1),j-(p+1),k-(p+1)} \ \dots u^{i+(p+1),j-(p+1),k-(p+1)}] \ \dots \ [u^{i-(p+1),j+(p+1),k+(p+1)} \ \dots u^{i+(p+1),j+(p+1),k+(p+1)}] \Big] =$$

which contains only those modal coefficients relevant to computing $a^{i,j,k}_{kx,ky,kz}(q_x, q_y, q_z)$ on element $(i,j,k)$.

### F.2.3 Coefficient Matrices

Now to construct matrices for computing the appropriate coefficients as determined by the $(q_x, q_y, q_z)$ index. First I construct an intermediary matrix $C^{q_x,q_y,q_z}$ whose function is to from the select the elements from $\tilde{u}^{i,j,k}$ which the line integral overlaps. Define the $(2p+1)(2p+3)^2(p+1)^3 \times (2p+3)^3(p+1)^3$ matrix

$C^{q_z}$ by

$$C^{q_z} = \begin{bmatrix} \delta_{-1,q_z}\mathbb{I} & \delta_{0,q_z}\mathbb{I} & \delta_{1,q_z}\mathbb{I} & & 0 \\ & \ddots & \ddots & \ddots & \\ 0 & & \delta_{-1,q_z}\mathbb{I} & \delta_{0,q_z}\mathbb{I} & \delta_{1,q_z}\mathbb{I} \end{bmatrix},$$

where $\delta$ is the Kronecker delta and and $\mathbb{I}$ is the $(2p+3)^2(p+1)^3 \times (2p+3)^2(p+1)^3$ identity matrix. This matrix has the effect of selecting the relevant $k-$coordinates. Then define the $(2p+1)(2p+3)(p+1)^3 \times (2p+1)(2p+3)^2(p+1)^3$ matrix $\hat{C}^{q_y}$ by

$$\hat{C}^{q_y} = \begin{bmatrix} \delta_{-1,q_y}\mathbb{I} & \delta_{0,q_y}\mathbb{I} & \delta_{1,q_y}\mathbb{I} & 0 \\ & & R_2 & \\ & & \vdots & \\ & & R_{2p+1} & \end{bmatrix},$$

where $\mathbb{I}$ is $(2p+3)(p+1)^3 \times (2p+3)(p+1)^3$, and $R_j$ is the 1st block row defined above where the non-zero entries are translated $(j-1)(2p+4)(p+1)^3(2p+3)$ places to the right. This matrix pairs the $j$ and $k$ coordinates.

Then define the $(2p+1)(p+1)^3 \times (2p+3)(2p+1)(p+1)^3$ matrix $\tilde{C}^{q_x}$ by

$$\tilde{C}^{q_x} = \begin{bmatrix} \delta_{-1,q_x}\mathbb{I} & \delta_{0,q_x}\mathbb{I} & \delta_{1,q_x}\mathbb{I} & 0 \\ & & \tilde{R}_2 & \\ & & \vdots & \\ & & \tilde{R}_{2p+1} & \end{bmatrix},$$

where $\mathbb{I}$ is $(p+1)^3 \times (p+1)^3$, and $\tilde{R}_j$ is the 1st block row defined above where the non-zero entries are translated $(j-1)(2p+4)(p+1)^3$ places to the right. This matrix completes the sorting procedure.

Now define the $(2p+1)(p+1)^3 \times (2p+3)^3(p+1)^3$ matrix $C^{q_x,q_y,q_z}$ as the composition

$$C^{q_x q_y,q_z} = \tilde{C}^{q_x}\mathscr{C}^{q_y}C^{q_z}$$

This concatenated matrix has the effect:

$$C^{q_x,q_y,q_z}\tilde{u}^{i,j,k} = \tilde{u}^{i,j,k}_{q_y,q_z} = [u^{i-(p-q_x),j-(p-q_y),k-(p-q_z)}, \ldots, u^{i+(p+q_x),j+(p+q_y),k+(p+1_z)}]^T,$$

so it selects exactly the modes used in computing the sum $a^{i,j,k}_{kx,ky,kz}(q_x,q_y,q_z) = \sum_{\gamma=-p}^{p} c_\gamma u^{i+\gamma+q_x,j+\gamma+q_y,k+\gamma+q_z}_{kx,ky,kz}$. Now for the actual matrix which performs the

summation. Define $A$ to be the $(p+1)^3 \times (2p+1)(p+1)^3$ matrix given by

$$A = \begin{bmatrix} c_{-p}\mathbb{I} & & & 0 & | & \dots & | & c_p\mathbb{I} & & 0 \\ 0 & c_{-p}\mathbb{I} & & 0 & | & \dots & | & 0 & c_p\mathbb{I} & 0 \\ & & \ddots & & | & \dots & | & & \ddots & \\ 0 & & & c_{-p}\mathbb{I} & | & \dots & | & 0 & & c_p\mathbb{I} \end{bmatrix}$$

where the identity matrices are $(p+1)^3$. This matrix has the effect that

$$A\tilde{u}^{i,j,k}_{q_x,q_y,q_z} = \vec{a}^{i,j,k}(q_x, q_y, q_z)$$

where

$$\vec{a}(q_x, q_y, q_z) = \begin{bmatrix} \vec{a}^{i,j,k}_{kz=0} & \dots & \vec{a}^{i,j,k}_{kz=p} \end{bmatrix}^T,$$

$$\vec{a}^{i,j,k}_{kz} = \begin{bmatrix} \vec{a}^{i,j,k}_{ky=0,kz} & \dots & \vec{a}^{i,j,k}_{ky=p,kz} \end{bmatrix}^T,$$

with

$$\vec{a}^{i,j,k}_{ky,kz} = \begin{bmatrix} \vec{a}^{i,j,k}_{kx=0,ky,kz} & \dots & \vec{a}^{i,j,k}_{kx=p,ky,kz} \end{bmatrix}^T.$$

Hence, the 27 coefficient matrices (of which only 15 are employed) denoted by $K^{i,j,k}_{q_x,q_y,q_z}$ are defined by the composition

$$K^{i,j,k}_{q_x q_y,q_z} = AC^{q_x,q_y,q_z}\mathscr{S}^{i,j,k}$$

and have the effect

$$K^{i,j,k}_{q_x,q_y,q_z} u = \vec{a}^{i,j,k}(q_x, q_y, q_z).$$

### F.2.4 Global Filtered Coefficient Matrices

To aid in testing the filtered solution formulations, we can concatenate these filtered coefficient matrices to produce the filtered coefficients over the whole mesh. Define $\vec{a}(q_x, q_y, q_z)$ by

$$\vec{a}(q_x, q_y, q_z) = [a^{1,1,1}(q_x, q_y, q_z) \dots | \dots | \dots a^{N,N,N}(q_x, q_y, q_z)]^T,$$

where the concatenation scheme being followed is the same as established previously.

We have $K^{i,j,k}_{q_x,q_y,q_z} u = a^{i,j,k}(q_x, q_y, q_z)$, so the global coefficient matrix is given by

$$K_{q_x,q_y,q_z} = [K^{1,1,1}_{q_x,q_y,q_z} \dots | \dots | \dots K^{N,N,N}_{q_x,q_y,q_z}]^T.$$

### F.2.5 Projection Matrices

We now wish to project our filtered approximation onto the finer mesh. Given $u^\star_h(x)$, we compute our filtered approximation on an element $O$ of the finer mesh $u^f_h(x) = \sum_{\beta \le p} \hat{u}_\beta \phi_\beta(x)$ for $\beta = (m, r, \ell)$ by projection:

$$\hat{u}_\beta = \langle u^\star_h, \phi_\beta(x) \rangle_O.$$

Now depending upon the octant, this inner product will be computed as a sum of projections over separate regions which will have to be computed on a case by case basis. Let us ignore that for now and just consider the case of projection over a single domain of definition:

$$\begin{aligned} u^O_\beta = \sum_{\alpha \le p}^{p} \Big\{ & \langle a^{i,j,k}_\alpha(q^O_L) \chi^O_{L,\alpha} \phi_\beta \rangle_O + \langle a^{i,j,k}_\alpha(q^O_{LM}) \chi^O_{LM,\alpha} \phi_\beta \rangle_O \\ & + \langle a^{i,j,k}_\alpha(q^O_{RM}) \chi^O_{RM,\alpha} \phi_\beta \rangle_O + \langle a^{i,j,k}_\alpha(q^O_R) \chi^O_{R,\alpha} \phi_\beta \rangle_O \Big\}. \end{aligned}$$

This expression will be broken up into four separate integrals, one for each of the different $\chi_{(_)}$ functions. For each $\chi_{(_)}$ define $P^O_{(_)}(kx, ky, kz, m, r, \ell) = \langle \chi^O_{_,\alpha} \phi_\beta \rangle_O$. Define the projection matrix $P_{(_)}$ by

$$P^O_{(_)} = \begin{bmatrix} \vec{P}(\ell = 0) \\ \vdots \\ \vec{P}(\ell = p) \end{bmatrix},$$

where

$$\vec{P}(\ell = p) = \begin{bmatrix} \vec{P}(0,0,\ell) \\ \vdots \\ \vec{P}(p,0,\ell) \\ \\ \vdots \\ \\ \vec{P}(0,p,\ell) \\ \vdots \\ \vec{P}(p,p,\ell) \end{bmatrix},$$

$$\vec{P}(m,r,\ell) = [P(kz=0,m,r,\ell)\dots P(kz=p,m,r,\ell)],$$

and

$$\vec{P}(kz,m,r,\ell) = [P(0,0,kz,m,r,\ell)\dots P(p,0,kz,m,r,\ell)\dots\dots P(0,p,kz,m,r,\ell)\dots P(p,p,kz,m,r,\ell)].$$

This complicated concatenation gives us

$$u_O^f = P_L^O \vec{a}_L + P_{LM}^O \vec{a}_{LM} + P_{RM}^O \vec{a}_{RM} + P_R^O \vec{a}_R.$$

Noting the choice of $K_{q_x,q_y,q_z}^{i,j,k}$ to obtain each of the coefficient vectors, we simply have to select the appropriate $K$ matrix s.t. $\vec{a} = K_q^{i,j,k}\vec{u}$, we can then substitute these in and obtain our fine mesh modal coefficients as a function of the course mesh coefficients $u_O^f = T^O u$. I will perform this octant by octant below, but thereafter we can concatenate these matrices by element superscripts and obtain a global transition matrix:

$$T_O^g = \begin{bmatrix} T_{1,1,1}^O & \dots & T_{N,N,1}^O & |\dots| & T_{1,1,N}^{2T} & \dots T_{N,N,N}^{2T} \end{bmatrix}^T,$$

such that

$$u_{O,g}^f = T_O^g u.$$

We want to use Gauss-Legendre quadrature to compute the projections of our filtered approximations onto a finer mesh. It turns out that as we are going to break all projection on a single octant into a sum of projections over the subregions of that octant, we can just compute the necessary changes of variables for

each of those subregions, keeping track of the new arguments for each function in the integrand, and then just substitute in the appropriate basis function. Table F.10 indicates what the arguments are for each function in the integrand, the order of integration, and the coefficients in front of each integral.

Denote by $g, h$, and $f$ the change of variables applied to $\underline{\zeta} = (\zeta_x, \zeta_y, \zeta_z)$ i.e. the argument of $\phi_\beta$ $(\zeta_x(g), \zeta_y(h), \zeta_z(f))$. We have the table on the following page. Note that octant 8 is omitted as it is the same octant 1 but with $\underline{\xi} = \frac{\underline{\zeta}-1}{2}$ instead.

Table F.10 Projection integrals and transformations by region for 3D LSIAC-MRA.

| Octant 1 | $\xi_x = (\zeta_x + 1)/2,\ \xi_y = (\zeta_y + 1)/2,\ \xi_z = (\zeta_z + 1)/2$ | | | |
|---|---|---|---|---|
| Region | Integral | $g$ | $h$ | $f$ |
| 1A | $\int_{-1}^{1} \frac{\zeta_z+1}{2} \int_{-1}^{1} \frac{\zeta_y+1}{2} \int_{-1}^{1} u_h^\star(\underline{\xi})\phi_\beta(\underline{\zeta})\ dg\ dh\ d\zeta_z$ | $\frac{2}{\zeta_y+1}(\zeta_x - \frac{\zeta_y-1}{2})$ | $\frac{2}{\zeta_z+1}(\zeta_y - \frac{\zeta_z-1}{2})$ | $\zeta_z$ |
| 1B | $\int_{-1}^{1} \frac{\zeta_y+1}{2} \int_{-1}^{1} \frac{\zeta_z+1}{2} \int_{-1}^{1} u_h^\star(\underline{\xi})\phi_\beta(\underline{\zeta})\ dg\ df\ d\zeta_y$ | $\frac{2}{\zeta_z+1}(\zeta_x - \frac{\zeta_z-1}{2})$ | $\zeta_y$ | $\frac{2}{\zeta_y+1}(\zeta_z - \frac{\zeta_y-1}{2})$ |
| 1C | $\int_{-1}^{1} \frac{\zeta_z+1}{2} \int_{-1}^{1} \frac{\zeta_x+1}{2} \int_{-1}^{1} u_h^\star(\underline{\xi})\phi_\beta(\underline{\zeta})\ dh\ dg\ d\zeta_z$ | $\frac{2}{\zeta_z+1}(\zeta_x - \frac{\zeta_z-1}{2})$ | $\frac{2}{\zeta_x+1}(\zeta_y - \frac{\zeta_x-1}{2})$ | $\zeta_z$ |
| 1D | $\int_{-1}^{1} \frac{\zeta_x+1}{2} \int_{-1}^{1} \frac{\zeta_z+1}{2} \int_{-1}^{1} u_h^\star(\underline{\xi})\phi_\beta(\underline{\zeta})\ dh\ df\ d\zeta_x$ | $\zeta_x$ | $\frac{2}{\zeta_z+1}(\zeta_y - \frac{\zeta_z-1}{2})$ | $\frac{2}{\zeta_x+1}(\zeta_z - \frac{\zeta_x-1}{2})$ |
| 1E | $\int_{-1}^{1} \frac{\zeta_x+1}{2} \int_{-1}^{1} \frac{\zeta_y+1}{2} \int_{-1}^{1} u_h^\star(\underline{\xi})\phi_\beta(\underline{\zeta})\ df\ dh\ d\zeta_x$ | $\zeta_x$ | $\frac{2}{\zeta_x+1}(\zeta_y - \frac{\zeta_x-1}{2})$ | $\frac{2}{\zeta_y+1}(\zeta_z - \frac{\zeta_y-1}{2})$ |
| 1F | $\int_{-1}^{1} \frac{\zeta_y+1}{2} \int_{-1}^{1} \frac{\zeta_x+1}{2} \int_{-1}^{1} u_h^\star(\underline{\xi})\phi_\beta(\underline{\zeta})\ df\ dg\ d\zeta_y$ | $\frac{2}{\zeta_y+1}(\zeta_x - \frac{\zeta_y-1}{2})$ | $\zeta_y$ | $\frac{2}{\zeta_x+1}(\zeta_z - \frac{\zeta_x-1}{2})$ |
| Octant 2 | $\xi_x = (\zeta_x + 1)/2,\ \xi_y = (\zeta_y + 1)/2,\ \xi_z = (\zeta_z - 1)/2$ | | | |
| Region | Integral | $g$ | $h$ | $f$ |
| 2E | $\int_{-1}^{1} \int_{-1}^{1} \frac{\zeta_x+1}{2} \int_{-1}^{1} u_h^\star(\underline{\xi})\phi_\beta(\underline{\zeta})\ dh\ d\zeta_x\ d\zeta_z$ | $\zeta_x$ | $\frac{2}{\zeta_x+1}(\zeta_y - \frac{\zeta_x-1}{2})$ | $\zeta_z$ |
| 2F | $\int_{-1}^{1} \int_{-1}^{1} \frac{\zeta_y+1}{2} \int_{-1}^{1} u_h^\star(\underline{\xi})\phi_\beta(\underline{\zeta})\ dg\ d\zeta_y\ d\zeta_z$ | $\frac{2}{\zeta_y+1}(\zeta_x - \frac{\zeta_y-1}{2})$ | $\zeta_y$ | $\zeta_z$ |
| Octant 3 | $\xi_x = (\zeta_x - 1)/2,\ \xi_y = (\zeta_y + 1)/2,\ \xi_z = (\zeta_z + 1)/2$ | | | |
| Region | Integral | $g$ | $h$ | $f$ |
| 3A | $\int_{-1}^{1} \int_{-1}^{1} \frac{\zeta_z+1}{2} \int_{-1}^{1} u_h^\star(\underline{\xi})\phi_\beta(\underline{\zeta})\ dh\ d\zeta_z\ d\zeta_x$ | $\zeta_x$ | $\frac{2}{\zeta_z+1}(\zeta_y - \frac{\zeta_z-1}{2})$ | $\zeta_z$ |
| 3B | $\int_{-1}^{1} \int_{-1}^{1} \frac{\zeta_y+1}{2} \int_{-1}^{1} u_h^\star(\underline{\xi})\phi_\beta(\underline{\zeta})\ df\ d\zeta_y\ d\zeta_x$ | $\zeta_x$ | $\zeta_y$ | $\frac{2}{\zeta_y+1}(\zeta_z - \frac{\zeta_y-1}{2})$ |
| Octant 4 | $\xi_x = (\zeta_x - 1)/2,\ \xi_y = (\zeta_y + 1)/2,\ \xi_z = (\zeta_z - 1)/2$ | | | |
| Region | Integral | $g$ | $h$ | $f$ |
| 4B | $\int_{-1}^{1} \int_{-1}^{1} \frac{\zeta_z+1}{2} \int_{-1}^{1} u_h^\star(\underline{\xi})\phi_\beta(\underline{\zeta})\ dg\ d\zeta_z\ d\zeta_y$ | $\frac{2}{\zeta_z+1}(\zeta_x - \frac{\zeta_z-1}{2})$ | $\zeta_y$ | $\zeta_z$ |
| 4F | $\int_{-1}^{1} \int_{-1}^{1} \frac{\zeta_x+1}{2} \int_{-1}^{1} u_h^\star(\underline{\xi})\phi_\beta(\underline{\zeta})\ df\ d\zeta_x\ d\zeta_y$ | $\zeta_x$ | $\zeta_y$ | $\frac{2}{\zeta_x+1}(\zeta_z - \frac{\zeta_x-1}{2})$ |
| Octant 5 | $\xi_x = (\zeta_x + 1)/2,\ \xi_y = (\zeta_y - 1)/2,\ \xi_z = (\zeta_z + 1)/2$ | | | |
| Region | Integral | $g$ | $h$ | $f$ |
| 5C | $\int_{-1}^{1} \int_{-1}^{1} \frac{\zeta_z+1}{2} \int_{-1}^{1} u_h^\star(\underline{\xi})\phi_\beta(\underline{\zeta})\ dg\ d\zeta_z\ d\zeta_y$ | $\frac{2}{\zeta_z+1}(\zeta_x - \frac{\zeta_z-1}{2})$ | $\zeta_y$ | $\zeta_z$ |
| 5D | $\int_{-1}^{1} \int_{-1}^{1} \frac{\zeta_x+1}{2} \int_{-1}^{1} u_h^\star(\underline{\xi})\phi_\beta(\underline{\zeta})\ df\ d\zeta_x\ d\zeta_y$ | $\zeta_x$ | $\zeta_y$ | $\frac{2}{\zeta_x+1}(\zeta_z - \frac{\zeta_x-1}{2})$ |
| Octant 6 | $\xi_x = (\zeta_x + 1)/2,\ \xi_y = (\zeta_y - 1)/2,\ \xi_z = (\zeta_z - 1)/2$ | | | |
| Region | Integral | $g$ | $h$ | $f$ |
| 6D | $\int_{-1}^{1} \int_{-1}^{1} \frac{\zeta_z+1}{2} \int_{-1}^{1} u_h^\star(\underline{\xi})\phi_\beta(\underline{\zeta})\ dh\ d\zeta_z\ d\zeta_x$ | $\zeta_x$ | $\frac{2}{\zeta_z+1}(\zeta_y - \frac{\zeta_z-1}{2})$ | $\zeta_z$ |
| 6E | $\int_{-1}^{1} \int_{-1}^{1} \frac{\zeta_y+1}{2} \int_{-1}^{1} u_h^\star(\underline{\xi})\phi_\beta(\underline{\zeta})\ df\ d\zeta_y\ d\zeta_x$ | $\zeta_x$ | $\zeta_y$ | $\frac{2}{\zeta_y+1}(\zeta_z - \frac{\zeta_y-1}{2})$ |
| Octant 7 | $\xi_x = (\zeta_x - 1)/2,\ \xi_y = (\zeta_y - 1)/2,\ \xi_z = (\zeta_z + 1)/2$ | | | |
| Region | Integral | $g$ | $h$ | $f$ |
| 7A | $\int_{-1}^{1} \int_{-1}^{1} \frac{\zeta_y+1}{2} \int_{-1}^{1} u_h^\star(\underline{\xi})\phi_\beta(\underline{\zeta})\ dg\ d\zeta_y\ d\zeta_z$ | $\frac{2}{\zeta_y+1}(\zeta_x - \frac{\zeta_y-1}{2})$ | $\zeta_y$ | $\zeta_z$ |
| 7C | $\int_{-1}^{1} \int_{-1}^{1} \frac{\zeta_x+1}{2} \int_{-1}^{1} u_h^\star(\underline{\xi})\phi_\beta(\underline{\zeta})\ dh\ d\zeta_x\ d\zeta_z$ | $\zeta_x$ | $\frac{2}{\zeta_x+1}(\zeta_y - \frac{\zeta_x-1}{2})$ | $\zeta_z$ |