

NUMERICAL SOLUTIONS OF MULTI-DIMENSIONAL PARTIAL  
DIFFERENTIAL EQUATIONS USING AN ADAPTIVE WAVELET METHOD

A Dissertation

Submitted to the Graduate School  
of the University of Notre Dame  
in Partial Fulfillment of the Requirements  
for the Degree of

Doctor of Philosophy

by

Damrongsak Wirasaet, B.Eng., M.S.

---

Samuel Paolucci, Director

Graduate Program in Aerospace and Mechanical Engineering

Notre Dame, Indiana

July 2007

# NUMERICAL SOLUTIONS OF MULTI-DIMENSIONAL PARTIAL DIFFERENTIAL EQUATIONS USING AN ADAPTIVE WAVELET METHOD

Abstract

by

Damrongsak Wirasaet

In this work, we describe an adaptive wavelet method for the solution of time-independent and time-dependent partial differential equations in  $d$ -dimensions. The method is based on  $d$ -dimensional interpolating wavelets constructed from tensor products of 1-D interpolating wavelets. The connection between interpolating wavelets and dyadic grid points, and the fact that wavelet amplitudes indicate the local regularity of solutions are used in the construction of a computational grid of irregular points. Operations, such as the wavelet transform, its inverse, and interpolation are performed efficiently. In the spatial discretization, the derivative approximation on the irregular grid is obtained by means of consistent finite differences. An extension of the adaptive method to problems defined on more complicated domains is achieved by a domain transformation technique. For time-independent problems, the method is tested on 2- and 3-D Poisson and Helmholtz problems with exact manufactured solutions in order to numerically study the connection between the order of the wavelet, the order of finite difference, the threshold values, and the accuracy of numerical solutions. The combustion of a 2-D flame ball-vortex interaction is used as a test problem for the time-independent algorithm. Application of adaptive method to incompressible Navier-Stokes equations is accomplished through the use of the Chorin

projection method for time discretization. We apply the algorithm to simulate the flow in the 2-D lid-driven cavity at moderate Reynolds numbers and in the 2-D differentially-heated cavity at high Rayleigh numbers, and in the 3-D differentially heated cavity for various values of Rayleigh numbers. It is found that numerical results, while requiring a relatively small number of degrees of freedom, are in good agreement with the most accurate results available in the literature.

# CONTENTS

FIGURES . . . . .	iv
TABLES . . . . .	ix
ACKNOWLEDGMENTS . . . . .	x
CHAPTER 1: INTRODUCTION . . . . .	1
1.1 Motivation . . . . .	1
1.2 Introduction to wavelets . . . . .	3
1.3 Overview of current literature . . . . .	7
1.3.1 Wavelet-Galerkin method . . . . .	7
1.3.2 Multilevel preconditioning . . . . .	15
1.3.3 Wavelet-collocation and finite difference methods . . . . .	18
1.3.4 Multiresolution methods . . . . .	24
1.3.5 Remarks on adaptive wavelet methods . . . . .	25
1.4 Outline . . . . .	27
CHAPTER 2: ONE-DIMENSIONAL INTERPOLATING WAVELETS . .	30
2.1 Interpolation subdivision scheme . . . . .	30
2.2 One-dimensional interpolating wavelet bases . . . . .	34
2.2.1 Fast interpolating wavelet transform . . . . .	37
2.2.2 Remarks . . . . .	40
2.2.3 Approximation property . . . . .	41
2.2.4 Wavelet transform on irregular grid . . . . .	43
CHAPTER 3: CONSTRUCTION OF IRREGULAR GRIDS . . . . .	47
3.1 Higher-dimensional interpolating scaling function . . . . .	48
3.2 MRA Approach . . . . .	49
3.2.1 $d$ -D interpolating wavelet transform . . . . .	52
3.2.2 Approximation property . . . . .	55
3.2.3 Wavelet transform on a $d$ -dimensional grid of irregular points	57



3.3	Sparse Wavelet Representation(SWR) and irregular sparse grid . .	63
3.3.1	Determining the SWR of a function . . . . .	72
3.4	Interpolation on irregular grid . . . . .	74
3.5	Algebra of the SWR . . . . .	76
3.5.1	Addition and subtraction . . . . .	76
3.5.2	Multiplication . . . . .	77
3.5.3	Nonlinear function evaluation . . . . .	79
3.6	Derivative Approximation . . . . .	80
3.7	MRA- $d$ Approach . . . . .	88
3.7.1	Derivative approximation for MRA- $d$ . . . . .	95
CHAPTER 4: ADAPTIVE WAVELET METHODS FOR PARTIAL DIFFERENTIAL EQUATIONS . . . . .		100
4.1	Spatial discretization on irregular grid . . . . .	101
4.2	Adaptive wavelet technique for elliptic PDE problems . . . . .	106
4.3	Remark on preconditioning . . . . .	110
4.4	Numerical Experiments . . . . .	112
4.4.1	Adaptive method using the MRA approach . . . . .	112
4.4.2	Adaptive method with MRA- $d$ approach . . . . .	124
4.5	Adaptive wavelet technique for time-dependent problems . . . . .	128
4.5.1	Numerical experiment . . . . .	132
4.5.2	Application to a flame ball problem . . . . .	137
4.6	Application of the method on a bounded curvilinear domain . . .	146
4.6.1	Numerical results . . . . .	152
4.7	Some comments on lifted wavelets . . . . .	158
CHAPTER 5: APPLICATION OF WAVELET METHODS TO INCOMPRESSIBLE NAVIER-STOKES EQUATIONS . . . . .		161
5.1	Governing equations . . . . .	162
5.2	Fractional step method . . . . .	162
5.3	Dynamically adaptive algorithm for incompressible flow . . . . .	164
5.4	Poisson-Neumann problem . . . . .	166
5.5	2-D lid-driven cavity . . . . .	170
5.6	2-D differentially-heated cavity . . . . .	177
5.7	3-D differentially heated cavity . . . . .	195
CHAPTER 6: CONCLUDING REMARKS . . . . .		209
6.1	Summary . . . . .	209
6.2	Recommendations for future work . . . . .	214
BIBLIOGRAPHY . . . . .		217

## FIGURES

2.1	The interpolating scaling functions, $\phi_{3,k}, k = 0, \dots, 4$ for $p = 4$ . . . . .	33
2.2	Points required in the calculation of wavelet coefficients $d_{j,k}$ for $p = 6$ (a) $k = 0, 1$ , (b) $k = 2, \dots, 2^{l-1} - 2$ . Note that the larger circles denote the points in $V_l$ . The filled circles represent points participating in the calculation of $d_{j,k}$ . . . . .	38
3.1	Filled circles denotes grid points needed for the calculation of the 2-D wavelet coefficients $d_{j,\mathbf{k}}^e$ away from boundaries for $p = 4$ . (a) $d_{j,\mathbf{k}}^{(0,1)}$ (b) $d_{j,\mathbf{k}}^{(1,0)}$ , and (c) $d_{j,\mathbf{k}}^{(1,1)}$ . Note that the larger circles denote grid points in $\mathbf{V}_j$ and the smaller circles denote grid points in $\mathbf{W}_j$ . . . . .	58
3.2	The irregular grid ( $p = 4, l_0 = 4$ ), satisfying the minimum index set condition, obtained from the augmentation of the grid of the single point $x_{4,(0,0)}^{(1,1)}$ : $\blacksquare$ represents $x_{4,(0,0)}^{(1,1)}$ and $\circ$ denote augmented points. . . . .	63
3.3	(a) Test function $f(x) = 0.2/( 0.4 - x^2  + 0.2)$ and (b) grid points corresponding to $ d_{j,\mathbf{k}}^e  \geq 5 \times 10^{-3}$ and the distribution of such grid points at each level for $p = 6, l_0 = 3$ . . . . .	64
3.4	(a) Test function $f(x, y) = 0.2/( 0.4 - x^2 - y^2  + 0.2)$ , (b) grid points corresponding to $ d_{j,\mathbf{k}}^e  \geq 5 \times 10^{-3}$ , and (c) grid points corresponding to $ d_{j,\mathbf{k}}^e  \geq 5 \times 10^{-3}$ at each level for $p = 6, l_0 = 3$ . . . . .	65
3.5	(a) Sections of test function $f(x, y, z) = 1/( 0.5 - x^2 - y^2 - z^2  + 0.1)$ , (b) grid points corresponding to $ d_{j,\mathbf{k}}^e  \geq 5 \times 10^{-3}$ for $p = 6, l_0 = 3$ . . . . .	66
3.6	Relationship between the approximation error $\ f(\mathbf{x}) - f^\varepsilon(\mathbf{x})\ _\infty$ and $N = \dim \mathbf{V}^\varepsilon$ as a results of varying $\varepsilon$ (left) and $\ f(\mathbf{x}) - f^\varepsilon(\mathbf{x})\ _\infty$ as a function of $\varepsilon$ (right) for sparse wavelet approximation with different values of $p$ . (a) test function (3.34); (b) test function (3.35). . . . .	71
3.7	Relationship of the error $\ f(\mathbf{x}) - f^\varepsilon(\mathbf{x})\ _\infty$ of the test function (3.36) and $N = \dim \mathbf{V}^\varepsilon$ as a result of varying $\varepsilon$ (left) and the error as a function of $\varepsilon$ (right) for sparse wavelet approximation with different values of $p$ . . . . .	72

3.8	Convergence in $L_\infty$ -norm of an addition of the SWR of the test function (3.34) and the SWR of the test function (3.35) for various $p$ and $\varepsilon$ . Left: error vs numbers of points in extended grids. Right: error vs. threshold values $\varepsilon$ . . . . .	78
3.9	Convergence in $L_\infty$ -norm of a multiplication of the SWR of the test function (3.34) and the SWR of the test function (3.35) for various $p$ and $\varepsilon$ . Left: error vs. numbers of points in extended grids. Right: error vs. threshold values. . . . .	79
3.10	Error in $L_\infty$ -norm of $g(f^\varepsilon)$ of the SWR of the test function (3.34) as a function of threshold value $\varepsilon$ . . . . .	81
3.11	$L_{\mathbf{V},\infty}$ -error of the derivative approximation on irregular grids, $\ \partial^i f/\partial x - D_{x_1}^{(i)} f^\varepsilon\ _{\mathbf{V}^\varepsilon, \infty}$ , of the test function (3.34) as a function of $N = \dim \mathbf{V}^\varepsilon$ for different $p$ , $n$ , and $\varepsilon$ . (a) error of the first derivative approximation with respect to $x_1$ -direction. (b) error of the second derivative approximation with respect to $x_1$ -direction. . . . .	87
3.12	MRA- $d$ irregular grid associated with the SWR of the function $f(x, y) = 0.2/( 0.4 - x^2 - y^2  + 0.2)$ for $\varepsilon = 5 \times 10^{-3}$ and $p = 6$ . . . . .	94
3.13	Regular sparse grid $V_J^{(1)}$ with $J = 7$ and $l_0 = 2$ . . . . .	97
3.14	$L_{\mathbf{V},\infty}$ -error of the derivative approximation on MRA- $d$ irregular grids, $\ \partial^i f/\partial x - D_{x_1}^{(i)} f^\varepsilon\ _{\mathbf{V}^\varepsilon, \infty}$ , of the test function (3.34) as a function of $N = \dim \mathbf{V}^\varepsilon$ for different $p$ , $n$ , and $\varepsilon$ . (a) error of the first derivative approximation with respect to $x_1$ -direction. (b) error of the second derivative approximation with respect to $x_1$ -direction. . . . .	99
4.1	Two refinement strategies with $P = 1$ : (a) strategy (4.9) (b) strategies (4.10); $\bullet$ grid points flagged for refinement. $\blacksquare$ neighboring points. . . . .	109
4.2	Solution of Test 1 (left) and the adaptively refined irregular grid (right) for $p = 6$ , $n = 4$ and $\varepsilon = 1 \times 10^{-4}$ . . . . .	114
4.3	Sequences of adaptively refined irregular grids $\mathbf{V}^m$ for $m = 0, \dots, 5$ in Test 1 for $p = 6$ , $n = 4$ and $\varepsilon = 5 \times 10^{-4}$ . . . . .	114
4.4	Test 1: (a)-(d) $L_{\mathbf{V},\infty}$ -error as a function of the number of grid points with different $p$ , $n$ , and $\varepsilon$ and using 2 <sup>nd</sup> to 8 <sup>th</sup> finite difference methods, (e) the number of grid points as a result of varying the threshold value $\varepsilon$ ; Bold solid lines represent the expected convergence, $\ \cdot\ _{\mathbf{V},\infty} \sim N^{\text{Conv}/d}$ . . . . .	115
4.5	Solution of Test 2 (left) and the adaptively refined irregular grid (right) generated by the adaptive method with $p = 4$ , $n = 4$ , and $\varepsilon = 10^{-4}$ . . . . .	117
4.6	Test 2: $L_{\mathbf{V},\infty}$ -errors as functions of the number of grid points as a result of varying $\varepsilon$ ; (a) $p = 4$ and $n = 2$ , (b) $p = 4$ and $n = 4$ , (c) $p = 6$ and $n = 4$ , and (d) $p = 8$ and $n = 7$ . . . . .	118

4.7	Test 2: the number of grid points produced by the adaptive algorithm as a result of varying $\varepsilon$ ; (a) $p = 4$ and $n = 2$ , (b) $p = 4$ and $n = 4$ , (c) $p = 6$ and $n = 4$ , and (d) $p = 8$ and $n = 7$ . . . . .	119
4.8	Solution of Test 3 (left) and the adaptively refined irregular grid (right) generated with $p = 4$ , $n = 4$ , and $\varepsilon = 10^{-4}$ . . . . .	121
4.9	Test 3: $L_{\mathbf{V},\infty}$ -errors as functions of the number of grid points as a result of varying $\varepsilon$ ; (a) $p = 4$ , $n = 2$ and 4; (b) $p = 6$ , $n = 4$ and 6; (c) $p = 8$ , $n = 6$ and $n = 8$ . . . . .	122
4.10	Solution of Test 4 (left) and the adaptively refined irregular grid (right) generated by the adaptive method with $p = 6$ , $n = 5$ , and $\varepsilon = 5 \times 10^{-6}$ . . . . .	124
4.11	Test 4: (a) $L_{\mathbf{V},\infty}$ -error as a function of number of grid points, $N$ , as a result of varying $\varepsilon$ and (b) $\varepsilon$ versus $N$ . . . . .	125
4.12	Solution of Test 5 (left) and the adaptively refined irregular grid (right) generated by the adaptive method based on MRA- $d$ with $p = 4$ , $n = 4$ , and $\varepsilon = 10^{-4}$ . . . . .	126
4.13	Test 5: $L_{\mathbf{V},\infty}$ -errors as a function of the number of grid points $N$ as a result of varying $\varepsilon$ (left) and the dependence of $N$ on $\varepsilon$ (right) using MRA- $d$ ; (a) $p = 4$ and $n = 4$ , (b) $p = 6$ and $n = 6$ . . . . .	127
4.14	Approximate solution of Test 6 (left) and the irregular grid (right) generated by the adaptive method with $p = 4$ , $n = 4$ , and $\varepsilon = 5 \times 10^{-4}$ . . . . .	134
4.15	Number of points in the computational grid as a function of time by the adaptive algorithm with $p = 4$ , $n = 4$ and different threshold values. . . . .	135
4.16	$L_{\mathbf{V},\infty}$ -error in the approximate solution as a function of time for $p = 4$ , $n = 4$ , and different threshold values. . . . .	135
4.17	$L_{\mathbf{V},\infty}$ -error as a functions of the number of grid points, $N$ , as a result of varying $\varepsilon$ (left) and $\varepsilon$ versus $N$ (right), at $t = 1/2$ and $t = 1$ . . . . .	136
4.18	Solution of the flame ball problem with $\Gamma = 100$ at $t = 0, 0.33, 0.79$ , and $1.0$ (top-bottom). Left: isolines of $T$ from $0.1$ to $0.9$ with increment of $0.1$ . Middle: corresponding isolines of $\omega$ . Right: irregular sparse grid for $p = 6$ , $n = 4$ , and $\varepsilon = (10^{-3}, 10^{-3})$ . . . . .	141
4.19	Solution of the flame ball problem with $\Gamma = 100$ at specific planes obtained with $p = 6$ , $n = 4$ , and $\varepsilon = (10^{-3}, 10^{-3})$ . (a) $T$ and $\omega$ on the plane $y = 0$ at $t = 0$ (left) and $t = 0.79$ (right), (b) $T$ and $\omega$ on the plane $x = 0$ at $t = 0$ (left) and $t = 0.79$ (right). . . . .	142
4.20	Solution of the flame ball problem with $\Gamma = 100$ at $t = 0, 0.33, 0.79, 1$ (top-bottom). Left: isolines of $T$ from $0.1$ to $0.9$ with increment of $0.1$ . Middle: corresponding isolines of $\omega$ . Right: irregular sparse grid from MRA- $d$ approach with $p = 6$ , $n = 4$ , and $\varepsilon = (10^{-3}, 10^{-3})$ . . . . .	143

4.21	Number degrees of freedom as a function of time required by the adaptive algorithm based on (a) the MRA approach and (b) the MRA- $d$ approach . . . . .	144
4.22	Integral of reaction rate, $R$ , as a function of time for different values of $\Gamma$ . . . . .	146
4.23	Solution of the flame ball problem with $\Gamma = 1000$ , $\gamma = 0$ at $t = 0$ , 0.05, 0.39, and 0.60 (top-bottom). Left: temperature $T$ . Middle: reaction rate $\omega$ . Right: irregular sparse grid generated dynamically by the adaptive wavelet scheme with $p = 6$ , $n = 4$ , $\varepsilon = (10^{-3}, 10^{-3})$ . . . . .	147
4.24	Coordinate transformation in two spatial dimension . . . . .	149
4.25	Physical domain $\tilde{\Omega}$ of Test 7. . . . .	152
4.26	Test 7: Numerical solution (left), and the adaptively refined irregular grid (right), generated by the adaptive method with $p = 6$ , $n = 4$ , and $\varepsilon = 10^{-3}$ . . . . .	154
4.27	Test 7: (a) $L_{\mathbf{V},\infty}$ -error as a function of number of grid points, $N$ , as a result of varying $\varepsilon$ and (b) $\varepsilon$ versus $N$ . . . . .	155
4.28	Physical domain $\tilde{\Omega}$ of Test 8. . . . .	156
4.29	Test 8: Numerical solution (left), and the adaptively refined irregular grid (right), generated by the adaptive method with $p = 6$ , $n = 4$ , and $\varepsilon = 10^{-4}$ . . . . .	157
4.30	Test 8: (a) $L_{\mathbf{V},\infty}$ -error as a function of number of grid points, $N$ , as a result of varying $\varepsilon$ , and (b) $\varepsilon$ versus $N$ . . . . .	157
5.1	Evolutions of streamlines and dynamically adaptive grids for $Re = 1000$ at $t = 2.5, 5.0, 7.5, 12.5$ and steady state. The number of grid points at each time are respectively $N = 3378, 3910, 4075, 4180$ and 4372. . . . .	173
5.2	Streamlines and dynamically adaptive grids at steady state for $Re = 400$ and 3200. . . . .	174
5.3	Comparison of steady state velocity profiles $u(1/2, y)$ and $v(x, 1/2)$ with those of [64] for (a) $Re = 400$ , (b) $Re = 1000$ , and (c) $Re = 3200$ . . . . .	176
5.4	Unsteady flow for $Ra = 10^8$ obtained with $\varepsilon = \{10^{-3}, 10^{-3}, 10^{-3}\}$ at three different times: stream function (left), isotherms (middle), and dynamically adaptive grid (right). . . . .	185
5.5	Evolution of number of grid points required for $Ra = 10^8$ . . . . .	186
5.6	Steady state solutions for $Ra = 10^6$ (top), $Ra = 10^7$ (middle), and $Ra = 10^8$ (bottom), obtained with $\varepsilon = \{10^{-3}, 10^{-3}, 10^{-3}\}$ : stream function (left), isotherms (middle), and dynamically adaptive grid (right). . . . .	187

5.7	Unsteady flow for $Ra = 5 \times 10^8$ at different times obtained with $\epsilon = \{10^{-3}, 4 \times 10^{-3}, 4 \times 10^{-3}\}$ : stream function (left), isotherms (middle), and dynamically adaptive grid (right). . . . .	192
5.8	Unsteady flow for $Ra = 5 \times 10^8$ at different times obtained with $\epsilon = \{10^{-3}, 4 \times 10^{-3}, 4 \times 10^{-3}\}$ : stream function (left), isotherms (middle), and dynamically adaptive grid (right) . . . . .	193
5.9	Evolution of the number of grid points required for $Ra = 5 \times 10^8$ . . . . .	194
5.10	Time trace of the $u$ -velocity component at the point $\mathbf{x} = (0.0478, 0.9522)$ for $Ra = 5 \times 10^8$ . . . . .	194
5.11	Schematic diagram of the differentially-heated cubic cavity. . . . .	196
5.12	Dynamically adaptive grid $\mathbf{V}$ for $Ra = 10^5$ obtained with $\epsilon = 3.75 \times 10^{-3}$ at three different times; (a) $t = 1.2$ , $N = 30764$ , (b) $t = 3.6$ , $N = 83328$ , (c) $t = 5.4$ , $N = 60800$ . . . . .	197
5.13	Isotherms (contour level: 0.375 (red), 0.25 (yellow), 0 (green), -0.25 (purple), -0.375 (blue)) for $Ra = 10^5$ obtained with $\epsilon = 3.75 \times 10^{-3}$ at three different times: (a) $t = 1.2$ (b) $t = 3.6$ (c) $t = 5.4$ . . . . .	198
5.14	Velocity field $u-v$ (top row) and distribution of grid points (bottom row) in the plane $z = 1/2$ for $Ra = 10^5$ obtained with $\epsilon = 3.75 \times 10^{-3}$ at three different times. . . . .	199
5.15	Velocity field $u-w$ (top row) and distribution of grid points (bottom row) in the plane $y = 1/2$ for $Ra = 10^5$ obtained with $\epsilon = 3.75 \times 10^{-3}$ at three different times. . . . .	200
5.16	Velocity field $v-w$ (top row) and distribution of grid points (bottom row) in the plane $x = 1/2$ for $Ra = 10^5$ obtained with $\epsilon = 3.75 \times 10^{-3}$ at three different times. . . . .	201
5.17	Evolution of the number of grid points required by the dynamically adaptive algorithm for $Ra = 10^5$ using different threshold values: (solid line) $\epsilon = 3.75 \times 10^{-3}$ , (dash line) $\epsilon = 5.0 \times 10^{-3}$ . . . . .	202
5.18	Steady flow for $Ra = 10^5$ obtained with $\epsilon = 3.75 \times 10^{-3}$ . (a) temperature contour levels: 0.375 (red), 0.25 (yellow), 0 (green), -0.25 (purple), -0.375 (blue); (b) temperature in the plane $z = 1/2$ ; (c) velocity field $u-v$ in the plane $z = 1/2$ , (d) velocity field $u-w$ in the plane $y = 1/2$ , and (e) velocity field $v-w$ in the plane $x = 1/2$ . . . . .	203
5.19	Dynamically adaptive grid at steady state for $Ra = 10^5$ obtained with $\epsilon = 3.75 \times 10^{-3}$ : (a) irregular grid in the cavity, (b) irregular grid in the plane $z = 1/2$ , (c) irregular grid in the plane $y = 1/2$ , and (d) irregular grid in the plane $x = 1/2$ . . . . .	204

## TABLES

2.1	THE VECTOR OF NONZERO FILTER COEFFICIENTS $h_{2k+1}^{l,r}$ FOR $p = 6$ . . . . .	41
3.1	SLOPES OF THE LOG-LOG PLOTS IN FIGURE 3.11 . . . . .	86
4.1	VALUES OF THE PEAK IN THE INTEGRAL $R(t)$ AND TIME AT WHICH IT OCCURS FOR $\Gamma = 100$ and $\Gamma = 1000$ . . . . .	148
5.1	COMPARISONS OF THE INTENSITIES OF THE PRIMARY VORTEX.178	
5.2	COMPARISONS OF THE INTENSITIES OF THE LOWER LEFT CORNER SECONDARY VORTEX. . . . .	179
5.3	COMPARISONS OF THE INTENSITIES OF THE LOWER RIGHT CORNER SECONDARY VORTEX . . . . .	180
5.4	RESULTS FOR $Ra = 10^6$ COMPARED WITH OTHER ACCURATE SOLUTIONS. . . . .	188
5.5	RESULTS FOR $Ra = 10^7$ COMPARED WITH OTHER ACCURATE SOLUTION. . . . .	189
5.6	RESULTS FOR $Ra = 10^8$ COMPARED WITH OTHER ACCURATE SOLUTION. . . . .	190
5.7	SPECIFIC RESULTS FOR $Ra = 10^3$ . . . . .	206
5.8	SPECIFIC RESULTS FOR $Ra = 10^4$ . . . . .	207
5.9	SPECIFIC RESULTS FOR $Ra = 10^5$ . . . . .	208

## ACKNOWLEDGMENTS

Many thanks go to my advisor, Dr. Samuel Paolucci, who has continuously provided me guidance, technical insights, invaluable discussions, and support all along the course of this study. I also would like to thank Dr. Joseph Powers, Dr. Mihir Sen, and Dr. Joannes Westerink for serving on my dissertation committee and reviewing this dissertation. I acknowledge Dr. Yevgenii Rastigejev for many fruitful discussions.

I am grateful for the help and friendship of my fellow graduate students, especially, Dr. Muhammad Iqbal Owais, Dr. Sandeep Singh, Dr. Greg Brooks, Weiming Li, Andrew Henrick, Benito Torres, Dr. Brett McMickell, Jason Mayes, John Kamel, Ashraf Al-Kateeb, Gianluca Puliti, Joel Jimenez, Williams Calderon, and Dr. Kamthorn Chailuek. I would like to express my gratitude to Jerome Ziliak of South Bend, Indiana, for his countless help and advice on things outside academic world since the day I arrived at Notre Dame. Finally, I thank my late father, my mother, my sisters, and my brothers for their love and support throughout my life.



## CHAPTER 1

### INTRODUCTION

#### 1.1 Motivation

In dealing with physical problems formulated as partial differential equations, one often encounters problems whose solution contains localized features, or sharp variations, in which their locations may vary with time. Since a high resolution discretization is necessary to resolve sharp variations, accurate numerical simulations of such problems require a large number of uniform grids, and hence enormous computational resources, in term of computing time and memory storage, are required. To reduce the number degrees of freedom, and at the same time obtain a solution of similar accuracy, a sophisticated adaptive discretization method, which reflects the local demands of the physical solution, becomes necessary. Different types of adaptive discretization methods have been discussed in recent years. Adaptive mesh refinements (*e.g.*, [8]), adaptive finite element methods (*e.g.*, [102]), and adaptive boundary element methods (*e.g.*, [87]) have been developed. These methods typically use mesh refinement criteria which are based on a local error indicator obtained from a previous calculated solution. In this work, we consider an adaptive discretization method based on wavelets as an attractive alternative.

Wavelets are multiscale basis functions with localization in both physical and Fourier spaces [54]. In addition, polynomials up to a certain order are in the span

of wavelets. Owing to these properties, wavelet amplitudes indicate the local regularity of a function. More precisely, wavelet amplitudes are large in regions where a function changes sharply, and small where it is smooth. Neglecting wavelet functions of small amplitude results in little loss of accuracy. As a consequence, a function with near singularities at a relatively small number of locations can be approximated within a specified error criteria with a relative small number of wavelet functions. This suggests that wavelets may be a promising candidate for an adaptive method where a strategy of resolution adaptation can be designed simply by examining the wavelets amplitudes. Another important property of wavelets is that the Sobolev norms of a function and a sequence norm of its wavelet coefficients are equivalent [45, 54]. This norm equivalence has an important implication on the efficient preconditioning of a large linear system resulting from discretization. It guarantees that a simple preconditioning based on a diagonal scaling is optimal [45]. That is, the condition number of the preconditioned linear system is bounded regardless of the size of the system. It is the combination of the compression property and the efficient preconditioning that explains, to some degree, the interest that wavelet methods have generated in recent years. This work is indeed motivated by such properties.

The application of wavelets to the solution of *Partial Differential Equations* (PDEs) has developed in several directions. Despite the growing number of solution algorithms in the literature, the field is relative young, and many technical issues must be overcome. In the following, brief introductions to wavelets and notations are given first. Then, an overview of applications of wavelets to the solution of PDEs as well as some remarks on relevant technical issues are given.

## 1.2 Introduction to wavelets

The objectives of this section are to provide a brief introduction to wavelets on the real line and to establish the notation used in the subsequent sections.

Consider a sequence of spaces  $\{V_l\}_{l \in \mathbf{Z}} \in L_2(\mathbf{R})$  which is a span of  $\{\phi_{l,k}, k \in \mathbf{Z}\}$  (where  $\mathbf{R}$  denotes the set of real numbers and  $\mathbf{Z}$  represents the set of integers). The functions  $\phi_{l,k}$ , called scaling functions on scale of resolution  $l$  and location  $k$ , are the dilation and translation of a single function  $\phi$ , more precisely,

$$\phi_{j,k}(x) = \phi(2^j x - k). \quad (1.1)$$

The function  $\phi$  is the solution of a scaling relation

$$\phi(x) = \sum_k h_k \phi(2x - k) \quad (1.2)$$

where  $\{h_k\}$  are known filter coefficients that determine the scaling function of the particular wavelet systems [54, 58, 104]. In addition, the function  $\phi(x)$  should have the following properties:

1. The function  $\phi$  has compact support or decays sufficiently fast such that it can be truncated. As a consequence,  $|\text{supp}(\phi_{l,k})| \sim O(2^{-l})$ . Note that a function  $f$  is said to have compact support if it is zero outside a bounded set.
2. There is a  $p \in \mathbf{N}$  (where  $\mathbf{N}$  is the set of natural numbers), which depends on the particular  $\phi$ , such that polynomials less than degree  $p$  can be written as linear combinations of  $\{\phi_{0,k}, k \in \mathbf{Z}\}$  (and hence of  $\{\phi_{l,k}, k \in \mathbf{Z}\}$ ).

A simple example of  $\phi(x)$  is a box function  $\phi = \mathcal{X}_{[0,1]}$ . In this case, one has  $\{h_0, h_1\} = \{1, 1\}$ ,  $\text{supp}(\phi) = [0, 1]$ , and  $p = 1$  (any constant is in the span of  $\{\phi_{j,k}, k \in \mathbf{Z}\}$ ). Another simple example of  $\phi$  is the hat function  $\phi = \max(0, 1 - |x|)$ . In general, a closed form of  $\phi$  is unavailable. However, it is not necessary since only the coefficients  $\{h_k\}$  are needed in a practical implementation.

The scaling relation (1.2) implies that the spaces  $V_l$  are nested, *i.e.*  $V_l \subset V_{l+1}$ . Now define the space  $W_l$  to be the complement of  $V_l$  in  $V_{l+1}$ , *i.e.*

$$V_{l+1} = V_l \oplus W_l. \quad (1.3)$$

The basis functions  $\{\psi_{l,k}, k \in \mathbf{Z}\}$  of  $W_l$  are called wavelets. They are the dilation and translation,

$$\psi_{l,k}(x) = \psi(2^l x - k), \quad (1.4)$$

of the single function  $\psi(x)$  defined by

$$\psi(x) = \sum_k g_k \phi(2x - k), \quad (1.5)$$

where the coefficients  $\{g_k\}$ , which in fact are related to  $\{h_k\}$ , determine the particular wavelet system (see details on the construction of coefficients  $\{g_k\}$  in, *e.g.* [36, 54, 84, 104]). In addition, the function  $\psi$  has compact support or decay sufficiently fast so that  $|\text{supp}(\psi_{l,k})| \sim O(2^{-l})$ . The decomposition of  $V_l$  until  $V_{l_0}$  is reached, where  $V_{l_0}$  denotes the space associated with the coarsest scale, yields the multiresolution analysis (MRA) of  $V_l$ ,

$$V_l = V_{l_0} \oplus \left( \bigoplus_{j=l_0}^{l-1} W_j \right). \quad (1.6)$$

Equation (1.6) implies that a function  $f^l \in V_l$  can be written either in the single scale representation

$$f^l(x) = \sum_k c_{l,k} \phi_{l,k}(x), \quad (1.7)$$

or, equivalently, in the multiscale scale representation

$$f^l(x) = \sum_k c_{l_0,k} \phi_{l_0,k}(x) + \sum_{j=l_0}^{l-1} \sum_k d_{j,k} \psi_{j,k}(x), \quad (1.8)$$

where  $\{c_{l,k}\}$  and  $\{d_{l,k}\}$  are denoted as scaling function and wavelet coefficients, respectively. Note that since the collection of  $\{\phi_{l_0,k}\}$  and  $\{\psi_{j,k}\}_{j=l_0}^{l-1}$  forms a basis of  $V_l$ , the coefficients  $\{c_{l_0,k}\}$  and  $\{d_{j,k}\}_{j=l_0}^{l-1}$  are determined uniquely.

Let  $f^l \in V_l$  be an approximation of a function  $f$ , and consider the difference of the approximation of  $f$  in two successive spaces  $V_l$  and  $V_{l+1}$ :

$$w^l = f^{l+1} - f^l = \sum_k d_{l,k} \psi_{l,k}. \quad (1.9)$$

Properties 1 and 2 imply that the wavelet coefficient  $d_{l,k}$  is large when the function  $\psi_{l,k}$  lies in the vicinity of a singularity or, near singularity, of  $f$ . Therefore, the number of wavelets required to represent the function  $f$  having near singular behavior, within a prescribed accuracy, can be substantially reduced if one uses the multiscale representation instead of the single scale representation.

Another important property of  $\{\phi_{l,k}, k \in \mathbf{Z}\}$  and  $\{\psi_{l,k}, k \in \mathbf{Z}\}$  is that the former and the latter constitute a Riesz basis (a stable basis) of  $V_l$  and  $W_l$ , respectively. In addition, the collection of  $\{\phi_{l_0,k}, k \in \mathbf{Z}\}$  and  $\{\psi_{l,k}, k \in \mathbf{Z}\}_{l \geq l_0}$  forms a Riesz basis of  $L_2(\mathbf{R})$ . The  $\{\theta_k, k \in \mathbf{Z}\}$  are said to be a Riesz basis of  $V \subset L^2(\mathbf{R})$  if and only if they span  $V$  and for all  $\mathbf{c} := \{c_k, k \in \mathbf{Z}\} \in l^2(\mathbf{Z})$ , there exist  $0 < C_1 \leq C_2$  such that

$$C_1 \|\mathbf{c}\|_{l^2(\mathbf{Z})} \leq \left\| \sum_k c_k \theta_k \right\|_{L^2(\mathbf{R})} \leq C_2 \|\mathbf{c}\|_{l^2(\mathbf{Z})}. \quad (1.10)$$

The above bound implies that, first, the series  $\sum_k c_k \theta_k$  converges unconditionally in  $L^2$  for a finite  $\sum_k |c_k|^2$ . Second, any  $f \in V$  can be decomposed in a unique way according to  $f = \sum_k c_{l,k} \phi_{l,k}$  with  $\mathbf{c} \in l^2(\mathbf{Z})$  and the norm equivalence (1.10) also holds in this case. Wavelets also provide unconditional bases and characterizations for many other functional space. For a Sobolev space  $H^s$ , with  $s$  in a certain range, the following norm equivalence holds [45, 54]

$$C_1 \sum_{l,k \in \mathbf{Z}} 2^{2sl} |d_{l,k}|^2 \leq \left\| \sum_{l,k \in \mathbf{Z}} d_{l,k} \psi_{l,k} \right\|_{H^s} \leq C_2 \sum_{l,k \in \mathbf{Z}} 2^{2sl} |d_{l,k}|^2 \quad (1.11)$$

where  $0 < C_1 \leq C_2$ . This relationship between the Sobolev norm and the sequence norm of wavelet coefficients plays an important role in the construction of an efficient preconditioning technique for the linear system resulting from discretization [45, 78], or for error control in an adaptive scheme [46].

Since in general neither the scaling or wavelet bases are orthogonal (in which case one has  $C_1 = C_2$  in (1.10) and they equal to unity if orthonormal), dual scaling and wavelet functions,  $\{\tilde{\phi}_{l,k}, k \in \mathbf{Z}\}$  and  $\{\tilde{\psi}_{l,k}, k \in \mathbf{Z}\}$ , can be considered. They satisfy the following bi-orthogonal relations:

$$\langle \phi_{l,m}, \tilde{\phi}_{l,n} \rangle = \delta_{m,n}, \quad \langle \psi_{l,m}, \tilde{\psi}_{p,n} \rangle = \delta_{l,p} \delta_{m,n} \quad \langle \phi_{l,m}, \tilde{\psi}_{p,n} \rangle = 0, \quad p \geq l,$$

where  $\langle \cdot, \cdot \rangle$  is the usual  $L^2$  inner product, or a dual pair in the case where the dual functions are linear combinations of Dirac delta functions. Dual functions are necessary for understanding (1.11) [45] and for the establishment of a fast wavelet transform. In addition, dual functions can be used as weight functions in

a Petrov-Galerkin discretization formulation.

This brief overview of wavelet theory is intended to serve as a background for the subsequent sections. An in depth treatment of wavelet theory can be found in [54, 84, 104].

### 1.3 Overview of current literature

Wavelet-based methods for the numerical solution of PDEs have evolved in several directions. They include, but are not limited to, wavelet-Galerkin, wavelet-collocation, and multilevel preconditioning methods. In the following subsections, a brief overview of the literature is provided. This overview of the literature is undoubtedly incomplete. The unsettled and scattered nature of the field makes it difficult to capture all work in this area. Note that, for up-to-date reasons and to reflect how the field evolves, this overview also includes works that appears after the author began this work. In the reader wishes, he/she may skip this part and proceed directly to read the rest of the thesis.

#### 1.3.1 Wavelet-Galerkin method

The wavelet-Galerkin method is probably the earliest approach considered. It has received a great deal of attention over the past years. Undoubtedly, this is attributed to the generality that the Galerkin formulation provides.

In early works, a number of researchers (*e.g.* [2, 65, 79, 93, 109]) make use of compactly supported orthonormal Daubechies scaling functions [54]. In them, the unknown solution is expanded in the single scale representation (1.7) at a selected finest scale  $J$  and the approximate solution is solved using the Galerkin approach (*i.e.* the test functions are chosen to be the trial functions). These methods are

applicable to problems defined on parallelepiped domains with periodic boundary conditions. As shown in [79], for periodic boundary conditions one obtains a method that exhibits *super convergence* at grid points, the order of approximation being  $2p$ , where  $p$  is the highest degree of polynomial that is in the span of  $\{\phi_{l,k}, k \in \mathbf{Z}\}$  (see Property 2 in Section 1.2). A similar result is also found in the case of spline-based scaling function bases [80]. While the aforementioned works are important in their own right, the use of scaling functions prevent exploiting the compression property of wavelets. The methods are therefore non-adaptive.

Using orthonormal cubic spline wavelets and a Galerkin formulation, Maday et al. [103] solves an evolution problem using an adaptive algorithm that is based on the thresholding of wavelet coefficients. The results obtained by applying the method to solve a 1-D periodic Burgers equation indicate a substantial reduction in the required number of degrees of freedom and exponential-like convergence rate in  $L^2$ . Nevertheless, the authors note the difficulties in treating the nonlinear term.

Shult and Wyld [118] describe a Galerkin method based on the orthonormal Daubechies wavelets. They provide details of the computations of wavelet integrals needed in the Galerkin procedure. These integrals are of the form

$$A_{l,m}^{p,q} = \int \frac{d^n \psi_{l,p}}{dx^n} \psi_{m,q} dx, \quad B_m^{p,q} = \int \frac{d^n \phi_{l_0,p}}{dx^n} \psi_{m,q} dx, \quad C_m^{p,q} = \int \frac{d^n \psi_{m,q}}{dx^n} \phi_{l_0,p} dx. \quad (1.12)$$

The procedure consists of applying (1.5), 1.4 and (1.1) recursively until reaching the finest level  $J$  and subsequently making use of known integrals of scaling functions [18, 45, 93]. They also apply the method in the solution of the 1-D periodic Burgers equation. Although the required number of degrees of freedom is greatly reduced, they report that the gain in compression has little benefit in



terms of computing time. The reason for this is the time-consuming algorithm for computing the stiffness matrix (which requires  $A_{l,m}^{p,q}$ ,  $B_m^{p,q}$  and  $C_m^{p,q}$  as ingredients), which is of  $O(N^3)$  where  $N$  is the number degrees of freedom retained, and the integrals arising from the nonlinear term.

An early attempt to explore the use of wavelet-Galerkin methods for elliptic PDEs was made by Jaffard [78]. He shows that, for a second order elliptic boundary value problem, as a direct consequence of the norm equivalence (1.11), there exists a diagonal preconditioning which makes the condition number of the wavelet-Galerkin matrix bounded uniformly. Also, the estimate of the decay of entries of the stiffness matrix (see also [50]) shows that, for a given accuracy, a large number of entries can be neglected. Such estimate can be used to increase the sparseness of the wavelet-Galerkin matrix.

It is worth noting that due to the different sizes of support wavelets have on different scales, the stiffness matrix of the wavelet-Galerkin discretization is not a sparse matrix in the strict sense. In the case of a linear differential operator, it can be shown that, for uniform refinements, the number of entries in the wavelet-Galerkin matrix is of the order  $O(J2^{dJ})$ , where  $d$  is the dimension of the problem and  $J$  is the highest (or finest) scale in the approximation.

Liandrat and collaborators [98, 99, 101] introduce a Petrov-Galerkin method and a dynamically adaptive algorithm for an evolution problem. Orthonormal wavelets are used as trial functions; however, the choice of test functions is adapted to the operator being considered such that the stiffness matrix becomes a diagonal matrix. Such test functions are so-called vaguelettes [104, 125] or pseudo-wavelets. Because the stiffness matrix is diagonal, the solution procedure is reduced to the calculation of integrals of products between the right-hand-side of the equation

and the test functions. The method is applied to the solution of the 1-D periodic Burgers equation. A similar technique similar to that used in pseudo-spectral schemes is used to treat the nonlinear term.

The adaptive wavelet-vaguelette method has been pursued by a number of other researchers [20, 61, 62, 117]. The fast algorithm for computing integrals of products between a function and a vaguelette, a key ingredient of this approach, is outlined in [62] for the 1-D case and in [20] for the 2-D case. Such algorithm is based on filters that are composed of inner products of vaguelettes and associated interpolation bases. In the case of periodic boundary conditions, such filters can be evaluated efficiently in the Fourier domain. An application of the method to a combustion problems with periodic boundary conditions can be found in [20, 61, 62]. In [117], the wavelet-vaguelette method with cubic orthonormal spline wavelets is applied to solve the problem of a 2-D mixing layer in the stream function-vorticity formulation with periodic boundary conditions. The results demonstrate that the method can track the displacement and deformation of active flow regions and they are quantitatively in good agreement with those obtained by a pseudo-spectral method. The total complexity of their algorithm, which is the number of operations required in each integration step, is of order  $O(N)$ , where  $N$  denotes the number of wavelets retained. Due to the non-optimized implementation of the algorithm, the authors report that the computing time is of the same order as required by the pseudo-spectral method.

An attempt to construct a wavelet Petrov-Galerkin method that adapts to the differential operator has also been made by Sweldens [123]. In this work, the trial and test functions are derived from biorthogonal wavelets. The construction is done in a way that they are adapted to the square root of the operator at hand

so that the stiffness matrix is diagonal. Application to one-dimensional boundary value problems with constant and variable coefficients is demonstrated. However, the ability to apply the method to complicated operators or to higher dimensional problems is doubtful. The idea of adapting wavelets to a differential operator is also suggested by Dahlke and Weinreich [42]. In their work, the trial and test functions are constructed so that they are semi-orthogonal with respect to the operator-induced norm. As a result, the stiffness matrix is decoupled at different scales.

A different approach is taken by Charton and Perrier [29]. They take advantage of the wavelet compression of the solution and the inverse operator. Unlike a finite-difference or a finite-element method for which the stiffness matrix is sparse but its inverse is full, a wavelet-Galerkin matrix of the heat operator and its inverse have the same structure of non-zero entries (see also [50]). In addition, for a given accuracy, a large number of entries in the inverse wavelet-Galerkin matrix can be discarded (see also [17]). By computing the inverse of the Galerkin matrix and storing its thresholded version, the solution procedure amounts to the multiplication of the compressed matrix with the compressed vector of the wavelet coefficients of right-hand-side of the equation. To determine the wavelet coefficients of the nonlinear term, they use a pseudo-spectral approach. It consists of performing nonlinear operations in the physical domain and making use of quadrature formula and a wavelet transform to go back and forth to the wavelet coefficients space. Using Daubechies wavelets, the method is applied to solve a 2-D vortex interaction problem written in the stream function-vorticity formulation with periodic boundary conditions. They estimate that the total complexity of the method in solving the problem is of order of  $O(2^{2J})$  where  $J$  is the finest scale

used in the approximation. As one might expect, they report a large overhead in computational time.

It should be noted that most of wavelet-based methods mentioned are applied to problems with periodic boundary conditions. Additionally, all such problems in 2-D are defined on rectangular domains. Due to the fact that wavelets are bases that naturally reside on the real line, in order to solve problems defined on an interval, they must be modified. The simplest means of accomplish this is by periodization [104]. As a consequence, this limits the boundary conditions of a problem to one of periodic type. For some of the methods, such as those of [61, 62, 117], periodic boundary conditions are essential in the construction of an efficient algorithm. In addition, typically higher-dimensional wavelet bases are constructed by tensor products of one-dimensional bases. Thus, the application of wavelet-based methods to higher dimensions has been limited to those defined on rectangular domains.

The problem of applying boundary conditions on a finite interval has been investigated by several workers. Amaratunga et al. [2] use a capacitance approach in a Galerkin method based on the Daubechies scaling functions to solve a one-dimensional boundary value problem with Dirichlet boundary conditions. Albeit with additional cost for the boundary treatment, the convergence rate of the method does not deteriorate. Xu and Shann [141] construct bases which consist of anti-derivatives of the Daubechies wavelets that belong to the space  $H_0^1([a, b])$ , thus automatically incorporating homogeneous Dirichlet conditions. They also present bases for the space  $H^1([a, b])$ , which is the space required for solving problems with Neumann boundary conditions within the Galerkin formulation. Bertoluzza [15] uses scaling functions of wavelets on an interval (constructed in

[37]) in a Galerkin scheme to solve a one-dimensional boundary value problem. This method is applicable to problems with both Neumann and Dirichlet boundary conditions since such scaling functions belong to  $H^2([0, 1])$  and, by excluding the specific scaling function at each boundary, they belong to  $H_0^2([0, 1])$ .

The inclusion of boundary conditions in solving PDEs is considered in a number of works. Compactly supported wavelets on an interval that satisfy homogeneous Dirichlet or Neumann boundary conditions are considered in [32, 91, 106]. The biorthogonal wavelets on the interval  $[0, 1]$ , so-called boundary adapted, is introduced in [27, 97]. In such biorthogonal wavelets, at each scale  $l$ , only one basis function (*i.e.* a wavelet, or scaling function if  $l = l_0$ , and its dual basis) has a non-zero value at each boundary. Only slight modification is needed in order to make the boundary-adapted biorthogonal wavelets satisfy homogeneous Dirichlet boundary conditions.

Extension of wavelet-based methods to problems defined on general domains has been attempted by a number of authors. In [57, 66, 67], the wavelet-Galerkin method is used in conjunction with the fictitious domain approach to solve a 2-D elliptic problem with Dirichlet/Neumann boundary conditions on a non-rectangular domain. In the fictitious domain approach, the original domain is embedded into an artificial rectangular domain. The equivalent problem defined for the whole domain is then reformulated in a weak formulation, where the original boundary conditions are taken into account by the introduction of an auxiliary condition and a Lagrange multiplier. This permits one to use the wavelet-Galerkin method designed for 2-D periodic boundary conditions to solve the reformulated problem. In those works, only scaling functions are used in the single scale representation. Details on the use of the fictitious domain approach in an adaptive

wavelet setting can be found in [47, 51, 92].

Canuto et al. [27, 28] present a construction of a multiresolution analysis using biorthogonal wavelets on fairly general domains. The construction applies to domains  $\Omega$  which can be represented as a union of disjoint smooth parametric images of the unit  $d$ -dimensional cube  $(0, 1)^d$ . The approach may be described as follows. First, construct bi-orthogonal wavelet bases on the unit interval  $(0, 1)$  that are boundary adapted. Then, the tensor product of these 1-D bases are used to form bases on the unit  $d$ -dimensional cube  $(0, 1)^d$ , and subsequently on each subdomain by means of parametric mapping. Finally, appropriately patch the local bases at the subdomain boundaries to obtain the global bases on  $\Omega$ . Patching wavelets across subdomain boundaries subject to biorthogonal constraints reduces to a study of certain systems of homogeneous linear equations. The global bases verify the biorthogonal relations with respect to a modified inner product. Application of such bases to a second-order elliptic boundary value problem on an L-shaped domain is demonstrated in [26]. A software realization for the solution of second order elliptic PDEs is described in Berrone and Emmel [9]. They use the method to solve a Poisson equation defined on a square box with an elliptic hole. A different strategy of constructing wavelets on fairly general domains can be found in [35, 49].

Considerable progress on theoretical as well as practical aspects of wavelet-Galerkin methods applied to the solution of elliptic PDEs has been made in a series of works [5, 35, 39, 43, 44, 46, 47] (see also references therein). Such works discuss localization, cancellation properties, norm equivalences, and nonlinear approximations. In addition to implying an optimal diagonal preconditioning, norm equivalences lead to the design of a powerful adaptive algorithm [39] and data

structure [5]. Such an adaptive scheme is proven to be asymptotically optimal [39], *i.e.* converge at the same rate as a best  $N$ -term approximation for a wide class of elliptic operators. Reviews of such works can be found in [45, 46]. Note that the target error, in which the estimate relating the computational work and the number of degree of freedom generated is conducted, is defined in an energy norm (*i.e.* it targets a solution for the whole domain). Dahmen et. al. [53] consider adaptive wavelet methods which are goal-oriented, *i.e.* the target errors are based on some quantity of interest, such as point values or line integrals.

It should be emphasized that a gain in compression of the solution or operator does not necessarily translate to a reduction in computing time. The computational time is very much affected by specific implementations used to calculate entries of the wavelet-Galerkin stiffness matrix. An efficient computation of the stiffness matrix and right-hand-side entries has been developed recently [10, 16]. In addition, it should be noted that the treatment of nonlinear terms in wavelet-Galerkin methods is not straightforward and their computation expensive. Algorithms for treating nonlinear terms are in fact active areas of research for the method developed in [39]. Such algorithms can be founded in [38, 52, 100] and more recently in [6, 41].

### 1.3.2 Multilevel preconditioning

Wavelets can be taken advantage of in multilevel preconditioning [142] for the solution of elliptic PDEs. So-called multilevel preconditioning (or change-of-basis preconditioning) is used to accelerate the numerical linear algebra of the Galerkin discretization. The idea consists of replacing a single scale basis by an equivalent multiscale one that forms a stable basis of the Sobolev space, and subsequently using a fast transformation between the single scale representation and the multi-

scale one (and vice versa). Application of a stable multiscale basis preconditioner to the stiffness matrix results in a matrix having a uniformly bounded condition number. This ensures that the number of iterations required to solve an algebraic system using a conjugate gradient solver (or variants) [7] is bounded regardless of the number of degrees of freedom used. The iterative solution of the discrete problem subsequently depends only on the efficiency of a matrix/vector multiplication. The transformation between the two representations allows one to avoid having to use an explicit form of the stiffness matrix in the multiscale representation, which is usually a nearly-sparse matrix. In fact, it allows one to carry out the matrix/vector multiplication in  $O(N_J)$  operations where  $N_J$  is the number of degrees of freedom at the finest scale, provided that the stiffness matrix in the single scale representation is a sparse matrix and the transform between the two representations can be carried out in  $O(N_J)$  operations.

Yserentant [142] discusses multilevel preconditioning in linear finite element spaces. His approach consists of replacing the usual finite element nodal basis by an equivalent hierarchical basis. The effect of the change-of-basis results in a preconditioned stiffness matrix that has a uniformly bounded condition number in one dimension and a condition number that grows proportionally as  $O(\log(h^{-2}))$  in two dimensions (in fact, for the 1-D case, the hierarchical basis belongs to the family of interpolating wavelets [58]). Tong et al. [127] also make use of the nodal change of basis as a preconditioner. They demonstrate that their method yields a condition number that grows as  $O(\log(h^{-1}))$  in 2-D and  $O(1)$  in solving their model Poisson equation.

Xu and Shann [141] discuss bases constructed from anti-derivatives of scaling functions and the equivalent multiscale bases constructed from anti-derivatives of



wavelets. The change-of-basis method is used to reduce operations during the iterative solution procedure.

The use of wavelets as a particular multilevel method is investigated by Dahmen and Kunoth [48]. Due to the sparseness of the stiffness matrix in the single scale representation and the fast wavelet transform, wavelet preconditioners are asymptotically optimal in terms of operations count; it is precisely given by  $O(2^{dJ})$ , where  $d$  denotes the dimension of the problem and  $J$  the finest level of approximation. An attempt to construct a wavelet-like basis with small support size in the finite element context can be found in [121]. Their motivation stems from the fact that the cost of transformation of sophisticated wavelets with better accuracy and stability, although being of the right order, increases with the size of their support.

In Canuto et al. [26], the wavelet Galerkin method, based on the multilevel preconditioning, is applied to solve the Poisson equation defined on an L-shaped domain. Their results indicate that, to reduce the residual by some fixed factor, the number of iterations required by the conjugate gradient solver is basically independent of the number of levels and hence the number of degrees of freedom.

As pointed out by Dahmen [46], the main drawback in the use of wavelets in the context of multilevel preconditioning is that one must work with uniform refinements. If an adaptive solution process is used, the number of wavelets required,  $N$ , may be significantly smaller than that of the corresponding uniform resolution  $N_J$ . In this case, the cost of using the multilevel preconditioning method is much higher than  $N$ , and thus the previously gained efficiency of this approach may result in no benefit.

### 1.3.3 Wavelet-collocation and finite difference methods

A number of researchers use wavelets within a collocation framework, and some among them utilize wavelets that are constructed to be especially effective within this framework.

Cai and Wang [23, 24] and Cai and Zhang [25] construct cubic spline wavelets on an interval that are bases of the Sobolev space  $H_0^2([0, L])$  with the inner product  $\int f'' g'' dx$ . Such wavelets are constructed so that the bases at a scale, say  $j$ , vanish at all collocation points associated with bases at the larger scales,  $l < j$ . With this property, they construct a discrete wavelet transform that maps samples of a function at nonuniform collocation points (with a specific restriction) to its wavelet coefficients. The first and second derivative matrices are determined, as in the construction of the cubic spline, from non-uniform samples. In this method, it is necessary to solve particular tri-diagonal systems in order to obtain the discrete wavelet transform and derivative matrices. In [23, 24], the method is applied in the solution of the 1-D linear wave equation and the 1-D inviscid Burgers equation. In [25], the adaptive method is used in conjunction with an alternating direction implicit method to solve a 2-D reaction-diffusion equation.

Bertoluzza et al. [14, 15] use auto-correlation functions of the Daubechies scaling functions as test functions in a collocation scheme to solve 1-D and 2-D linear and nonlinear second order boundary value problems. Such functions, which are the scaling functions of interpolating wavelets, have the so-called interpolation property at dyadic points, *i.e.* each scaling function being unity only at a particular dyadic grid point and vanish at the others. Details on interpolating wavelets are found in [56, 58, 116]. Several ways of treating Dirichlet boundary conditions are discussed. One of the approaches, based on the use of the interpolating wavelet

on the interval [58], yields uniform error distribution throughout the domain. For the (linear) Laplace equation with periodic boundary conditions, it is shown that the resulting system of algebraic equations arising from discretization is identical to that of the wavelet-Galerkin method using the Daubechies scaling functions. Bertoluzza and Naldi [14] investigate the use of interpolating wavelets in the multilevel preconditioning of the collocation matrix. Results indicate that although a diagonal preconditioning does not yield a bounded condition number, it yields a condition number that grow relatively slow.

Bertoluzza [11] proposes a dynamically adaptive wavelet collocation method to an evolution problem. The method uses interpolating wavelets on an interval as trial functions. A new set of collocation points (and hence corresponding wavelet functions) is selected upon examination of the wavelet amplitudes of the solution at a previously calculated time step. The author applies the adaptive method to solve the 1-D Burgers equation with Dirichlet boundary condition, while in [12, 13] she applies the method in the solution of boundary value problems. In [13], 1-D model problems, with solutions having particular features, are used in assessing the accuracy and adaptive reliability of the method. The results clearly indicate that, for problems whose solutions contain sharp variations or discontinuities in higher derivatives, the adaptive method performs better than the uniform refinement strategy in terms of convergence rate with respect to the number of degrees of freedom. Qualitative results obtained by such an adaptive method applied to various 2-D model problems are reported in [12, 13]. A comparison of the complexities between the wavelet-collocation and the wavelet-Galerkin methods is made in [13]. In the collocation method, each entry in the system of equations resulting from discretization of a linear term in PDEs can be calculated

in a unit cost, while in the wavelet-Galerkin method it may take up to  $O(2^{dJ})$  cost with a straight-forward implementation. The number of operation required in calculating nonlinear terms at each collocation point is of  $O(J)$  while in the wavelet-Galerkin approach it may take up to  $O(2^{dJ})$  to calculate each integral involved in the nonlinear term.

Vasilyev et al. [133] present a multilevel wavelet-collocation method for a finite domain. They expand an unknown solution in a multilevel redundant representation. The set of such expansion functions, which are dilations and translations of the wavelet function, form a frame of a Hilbert space. Simply, a frame  $\{\theta_k\}$  is a complete set and in general is a linearly dependent set; If the frame is linearly independent, it is in fact a Riesz basis. Details on the topic can be found in [54]. By imposing a certain relationship between the set of collocation points at each levels, they use the multilevel redundant representation to construct basis functions which satisfy the interpolation property. The dimension of such basis functions equals that of the set corresponding to the union of all sets of collocation points at the different levels. Such basis functions are used as trial functions in the collocation scheme. Two different approaches are used to treat general boundary conditions. They apply the method using the Mexican hat wavelet and the auto-correlation function of the Daubechies scaling function to solve the 1-D Burgers equation. They observe that the method exhibits spectral-like accuracy. Vasilyev and Paolucci [134] extend the method of [133] to include discretization on adaptive collocation points. Their adaptive algorithm is based upon examination of the amplitude of expansion functions of the numerical solution at the previous time step to obtain a new set of collocation points for the subsequent integration. The authors apply the method using the auto-correlation function of

the Daubechies scaling function to solve a 1-D modified Burgers equation and a 1-D nonlinear thermo-acoustic wave problem. The results indicate a substantial reduction in the number of degrees of freedom required in the simulations. However, in this approach, the computational cost of calculating spatial derivatives is  $O(N^2)$ , where  $N$  is the numbers of collocation points used. Subsequently, Vasilyev and Paolucci [134] present an algorithm which reduces the cost of calculating spatial derivatives to  $O(N)$ . In addition, the authors describe the extension of the method to solve a 2-D problem defined on a square domain. They use the 1-D and 2-D Burgers equations and the 2-D nonlinear thermo-acoustic wave problem to asses their numerical method.

Jameson [79] points out that, for the non-truncated representation, the derivative matrix derived from the Galerkin projection is effectively equivalent to a finite difference method [79] in which the order of finite difference scheme depends on the order of wavelets. From such an observation, Jameson [81, 82] introduces the wavelet optimized finite difference method. The method utilizes the Daubechies wavelet transform in establishing grids of non-uniform points and the finite difference method on the resulting non-uniform grids to solve PDEs. The primary reason the author uses finite difference methods is to avoid difficulties in treating boundary conditions and nonlinear terms arising in wavelet approaches.

In adaptive wavelet-collocation methods using a straightforward collocation discretization, the resulting system of algebraic equations is, as in adaptive Galerkin approaches, a near sparse matrix as a consequence of different support sizes of wavelets on different levels. In fact, it can be shown that the number of nonzero entries of such system is approximately  $O((J - j_0)N)$ , where  $J$  and  $j_0$  are respectively the finest level and the coarsest level of the approximation and  $N$  is

the number of collocation points. In the method of Vasilyev and Paolucci [134], the calculation of derivative approximations can be accomplished in  $O(N)$  operations, however with a large constant multiplier because the evaluation of the interpolation basis at each collocation point involves several expansion functions on different levels. To avoid this somewhat costly calculation, several authors have proposed the use of a consistent finite difference approximation instead.

Holmström [76] presents a fast (inverse) interpolating wavelet transform from a set of  $N$  functional values on sparse dyadic grids to a set of  $N$  associated wavelet coefficients (and vice versa). The transform, which takes  $O(N)$  operations with a small constant multiplier, takes advantage of the connection between interpolating wavelets and the interpolating subdivision scheme [56]. The author introduces the *Sparse Point Representation* (SPR), which is a collection of functional values and their grid points whose wavelet amplitudes are greater than a threshold value. Instead of determining the derivative of a function by direct differentiation of the sparse wavelet representation, the author suggests instead the use of a finite difference approximation on the SPR. The centered difference scheme on a locally reconstructed uniform stencil is employed in such work. The author introduces an adaptive method based on the use of the interpolating wavelet transform to obtain the SPR and the finite-difference method to solve PDEs on the SPR. Numerical experiments on 1-D and 2-D wave equations, and the 2-D Burgers equation, are presented.

Rastigejev and Paolucci [111], and Rastigejev [110], develop a wavelet-based adaptive multiresolution representation algorithm for problems defined in 3-D parallelepiped domains. The algorithm utilizes the connection between the properties of interpolating wavelets, semi-structured dyadic grid points, and the interpolat-

ing subdivision scheme, in order to obtain a fast wavelet transform on irregular grids. This, and the use of an efficient data structure [110], results in a relatively inexpensive grid adaptation procedure. In addition, they develop a fast algorithm for derivative calculations using finite-differences on irregular grids, and present an adaptive method for evolution problems that utilize the above ingredients. In [111], the method is applied to solve a 1-D reacting flow in a shock tube governed by the compressible Navier-Stokes equations and using realistic chemical reactions. In [110], the algorithm is extended to solve the 2-D lid-driven cavity benchmark problem at high Reynolds number. Early results for the 3-D lid-driven cavity problem are reported. In [139], this technique (with further improvement) are applied to numerically simulate the 2-D differentially heated cavity with large Rayleigh numbers and subsequently the 3-D heated cavity [140]. Early results of the simulation of flow in the presence of obstacles using the combination of this method with the Brinkman penalization technique can be found in [138]. Note that in the Brinkman penalization approach [3], the Navier-Stokes equations defined on a complex domain are reformulated to Navier-Stokes/Brinkman equations defined on a the rectangular domain, where the obstacles are represented by volumetric Darcy drag terms with near zero porosity.

Griebel and Koster [71] present an adaptive scheme that exploits the properties of interpolating wavelets and the so-called multiresolution analysis- $d$  (MRA- $d$ ). The interpolating wavelets permit the construction of a fast 1-D interpolating wavelet on irregular dyadic grids. The use of MRA- $d$  (which is not a multiresolution in the strict sense) reduces a  $d$ -D wavelet transform and its inverse to the application of sequential 1-D transforms. In this work, the fast wavelet transform and its inverse play key roles in the evaluation of nonlinear terms and the calcu-

lation of the discrete version of a differential operator. The authors suggest two different approaches for evaluating discrete differential operators. One is based on the Petrov-Galerkin scheme where interpolating wavelets are used as trial functions and their dual wavelets are used as test functions. The other approach makes use of finite-differences (see also [70] for a similar technique based on piecewise linear interpolating wavelets). They apply the method to solve two different 2-D incompressible flow problems: the interaction of three vortices and a mixing layer problem. Subsequently, this adaptive method is used to simulate a 3-D shear flow [72].

Vasilyev [132] presents a wavelet adaptive collocation scheme based on lifted interpolating wavelets. Such lifted wavelets allow one to work with grids that are not necessarily dyadic. Furthermore, coefficients associated with lifted wavelets provide better information on the regularity of the function than conventional interpolating wavelets. The author suggests the use of a hierarchical finite-difference scheme for derivative approximations, and assesses the performance of the method by applying it to the solution of a 1-D advection problem, a 2-D Burgers equation, and a 2-D laminar flame-vortex interaction problem. This adaptive technique is combined with the Brinkman penalization technique to calculate the 2-D flow around a cylinder at high Reynolds numbers [86].

#### 1.3.4 Multiresolution methods

Substantially research efforts have been devoted to the so-called multilevel (or multiresolution) methods; see [1, 19, 31, 74, 120] and reference therein, for example. The approaches aims at reducing the computing time of modern high-resolution shock capturing schemes for hyperbolic system of conservation laws. Note that shock capturing schemes are expensive due to costly flux evaluations



(which typically involve at least one eigenvalues-eigenvector decomposition of the Jacobian matrix of the system). In essence, multiresolution methods use a multiscale decomposition, such as cell average multiresolution transform [19, 74] or interpolating wavelets [31], to categorize cells or points according to the smoothness of the solution. Subsequently, numerical fluxes are evaluated only for cells or points where the so-called detail coefficients are large, while in the smooth regions they are approximated by (much more) inexpensive means. In these methods, numerical values on the finest grid must be available since the computation of correct numerical fluxes, when required, use values on the finest grid. Thus, such schemes are not fully adaptive. While these methods reduce the computing time, they require the same number of degree of freedom (and hence the same amount of memory) as in the uniform case. A fully adaptive approach that shares the spirit of works described earlier is developed in [40] for hyperbolic conservation laws. This method has been extended to solve 2- and 3-D convection-diffusion, reaction-diffusion equations [113], and a 2-D flame ball problem (diffusion-convection-reaction equation) [112]. The key to these methods are strategies for evaluating fluxes without the need of full knowledge of cell-average values on the finest grid.

### 1.3.5 Remarks on adaptive wavelet methods

In previous sections, one may gather that wavelet-Galerkin and wavelet-collocation methods each have advantages and disadvantages. Wavelet-Galerkin methods are more amenable to theoretical analysis. In fact, considerable progress on theoretical aspect of the methods for elliptic PDEs have been made. In such methods, one usually has a simple and efficient preconditioning for matrices. There exist convergence proofs of some particular adaptive wavelet-Galerkin algorithms. Indeed,

for some wavelet Galerkin algorithms, there exists a proof and numerical evidence that the method converge at the same rate as a best  $N$ -term approximation for a wide class of elliptic operators. In addition, in Galerkin schemes it is easier to incorporate properties of the differential operator into the construction of the basis. However, these advantages may not translate directly to a decrease in computing time in an adaptive scheme if the implementation is not optimized [34]. In the wavelet-vaguelette method, boundary conditions are required to be of periodic type in order to construct an optimal solution algorithm. Moreover, the treatment of nonlinear terms in wavelet-Galerkin methods poses a challenge in obtaining an algorithm that is both efficient and accurate. In several works, nonlinear terms are treated by a procedure that is analogous to that used in pseudo-spectral methods. In such case, a nonlinear term is calculated pointwise in the physical domain and a quadrature formula and a wavelet transform are employed in going back and forth to/from the wavelet domain. Such a procedure is quite computationally expensive for some specific wavelets and when many nonlinear terms are present in the problem.

On the other hand, point-oriented adaptive methods, *i.e.* adaptive wavelet collocation methods and wavelet/finite difference methods, in general, can be devised so that they are free of the above difficulties. More precisely, the calculation of derivative approximations and the handling of nonlinear terms are inexpensive and algorithmically straightforward. In addition, general boundary conditions can be applied easily within these schemes. Indeed, these types of technique have been used for numerical solution of more realistic problems (see for example [86, 91, 111, 139]). However, compared with the wavelet-Galerkin methods, the point-oriented technique is less amenable to theoretical analysis. There is no

guarantee that the numerical solution would be asymptotically optimal. One may not have the benefit of having the discrete system which is well-conditioned.

An additional property that is common among all wavelet-based methods is that higher-dimensional wavelets are conventionally constructed as tensor products of one-dimensional wavelets. As a consequence, these methods are restricted to rectangular computational domains in higher dimensions, thereby restricting the applicability of these methods. Although, several approaches, for instance a second generation wavelet [124], can be used to construct wavelets on a domain of interest, they usually result in a loss of simplicity and efficiency, which make them less appealing for numerical solution for PDEs.

## 1.4 Outline

In the remainder of this thesis, a brief detail of the one-dimensional multiscale basis of the so-called interpolating wavelets on the interval  $[0, 1]$  are discussed in Chapter 2. Here, the interpolating scaling functions on interval that yields the multiresolution analysis are constructed through the use of the interpolation subdivision scheme (as a result, one can see immediately several properties of the basis function, which are simply consequences of the subdivision scheme). A property that is unique to the interpolating wavelet is the so-called interpolation function, which leads to the construction of a fast (forward and inverse) wavelet transform, and allows one to transform function values on a grid of irregular points to their associated wavelet coefficients. Subsequently, we introduce the multiscale basis on  $[0, 1]^d$ , where  $d$  is the spatial dimension, constructed by considering the tensor product of one dimensional bases (see Chapter 3).

In Chapter 3, we provide elements used for the adaptive solution of par-

tial differential equation in  $d$  spatial dimensions. Means for determining the  $d$ -dimensional adaptive grid are among them. In this work, such task is accomplished by taking advantage of the compression property of an approximation of the  $d$ -dimensional interpolating wavelets together with the connection between basis functions and dyadic points. Crucial algorithms related to adaptive grid such as the fast wavelet transform and its inverse, interpolation, and derivative approximation (with finite differences) are provided in this chapter. Note that in this work, although we focus on material relating to adaptive grids constructed from conventional MRA, brief detail of those based on MRA- $d$  is also included.

Subsequently, in Chapter 4, algorithms for the adaptive solution of time-independent and time-dependent PDEs are described. In principle, such algorithms are designed based on the analysis of wavelet amplitudes. The algorithm adapts resolution automatically according to the local demand of solution without having knowledge of the solution before-hand. The adaptive technique for time-independent problems is tested on 2-D and 3-D Poisson and Helmholtz problems. The test problems considered are those using manufactured solutions so that we can use them to assess the accuracy of the adaptive algorithm. For the time-dependent problem, a 2-D flame ball (which a diffusion-convection-reaction) problem is used as a test problem. We then describe an approach that combines the adaptive method and a domain transformation technique for a numerical solution of problems defined on a bounded curvilinear domain.

In Chapter 5, the dynamically adaptive algorithm for incompressible flow is presented. The algorithm uses a Chorin-type projection method for time discretization. The projection method decouples the pressure and velocity variables by splitting the problem into (predictor-corrector) sub-problems. We present re-

sults of numerical experiments of the adaptive scheme in simulating a 2-D flow in a lid-driven cavity, a 2-D differentially heated cavity with large Rayleigh numbers, and a 3-D differentially heated cavity. Note that whenever possible the numerical results are compared with those obtained by other computational approaches.

In the last Chapter, conclusions from the current study are drawn. We include brief comments on the software implementation (the actual software itself is of course still a work in progress) and recommendations for the work that may be undertaken in the future.

## CHAPTER 2

### ONE-DIMENSIONAL INTERPOLATING WAVELETS

The adaptive solution of partial differential equations used in this work is based on the use of a multiscale basis constructed from the one-dimensional interpolating wavelets by mean of a tensor product. We devote this chapter to discussing the one-dimensional interpolating wavelets and set up notations that will be used in the subsequent chapters.

#### 2.1 Interpolation subdivision scheme

Below, we describe an interpolation subdivision scheme introduced by [56] for data defined for  $x$  on an interval. The so-called interpolating scaling function that yields the multiresolution analysis on the interval arises naturally from such a scheme. Let  $p$  be an even positive integer,  $l$  be an integer such that  $l \geq \log_2(p)$ , and

$$V_l = \{x_{l,k} = k2^{-l} : k \in k_l^0\}, \quad k_l^0 = \{0, 1, \dots, 2^l - 1, 2^l\} \quad (2.1)$$

be a grid of dyadic points at scale level  $l$ . In addition, let  $\{f_{l,k} \equiv f(x_{l,k}) : x_{l,k} \in V_l, k = 0, \dots, 2^l\}$  be a set of function values at the dyadic points of a given function  $f(x)$  defined on  $x \in [0, 1]$ . The interpolation subdivision scheme is a process to interpolate from a given data  $\{f_{l,k}\}_{k \in k_l^0}$  to obtain a function  $\tilde{f}(x)$  defined on  $x \in [0, 1]$ , such that  $\tilde{f}(x_{l,k}) = f_{l,k}$ . The scheme begins with determining

the interpolated values on the grid  $V_{l+1}$ . This is accomplished by computing the interpolated values at odd grid points  $x_{l+1,2k+1}$  by means of polynomial interpolation. For the even grid points, since  $x_{l+1,2k} = x_{l,k}$ , the function values remain unchanged, *i.e.*  $\tilde{f}_{l+1,2k} = f_{l,k}$ . More specifically, the determination of  $\tilde{f}_{l+1,2k+1}$  is computed by fitting a polynomial  $\pi_{l,k}$  of degree  $p-1$  to the function values at points in  $X_{l,k}$ :

$$X_{l,k} = \{x_{l,m} : m \in \mathcal{X}_{l,k}\} \quad (2.2)$$

where the index set  $\mathcal{X}_{l,k}$  is defined by

$$\mathcal{X}_{l,k} = \begin{cases} \{0, \dots, p-1\} & \text{for } 0 \leq k \leq p/2 - 2, \\ \{k - p/2 + 1, \dots, k + p/2\} & \text{for } p/2 - 1 \leq k \leq 2^l - p/2, \\ \{2^l - p + 1, \dots, 2^l\} & \text{for } 2^l - p/2 + 1 \leq k \leq 2^l - 1, \end{cases} \quad (2.3)$$

and the polynomial  $\pi_{l,k}$  is given by

$$\pi_{l,k}(x) = \sum_{m=\min \mathcal{X}_{l,k}}^{\max \mathcal{X}_{l,k}} f_{l,m} L_{l,m}^k(x), \quad (2.4)$$

where  $L_{l,m}^k$  is the Lagrange polynomial defined by

$$L_{l,m}^k \equiv \prod_{r=\min \mathcal{X}_{l,k}, r \neq m}^{\max \mathcal{X}_{l,k}} \frac{(x - x_{l,r})}{(x_{l,m} - x_{l,r})} \quad (2.5)$$

so that

$$\tilde{f}(x_{l+1,2k+1}) \equiv \pi_{l,k}(x_{l+1,2k+1}). \quad (2.6)$$

Since the function values  $\{\tilde{f}_{j+1,k}\}_{k \in k_{l+1}^0}$  are now known, the interpolated values on  $V_{l+2}$  can be calculated using the same scheme with  $\{\tilde{f}_{j+1,k}\}_{k \in k_{l+1}^0}$  for the starting

data. By applying this scheme iteratively, one obtains the interpolation function  $\tilde{f}(x)$  for  $x \in [0, 1]$  in the limit  $l \rightarrow \infty$ . For  $p = 2$ , the subdivision scheme yields linear interpolation. For higher  $p$ , the interpolation subdivision scheme defines a function whose continuity increases with  $p$  [58], *i.e.* with the order of the polynomial used in the interpolation process.

Let  $\phi_{l,k}(x)$  for  $k \in \{0, \dots, 2^l\}$  denote a function resulting from applying the subdivision scheme of order  $p$  to the Kronecker data  $\{\delta_{r,k} : r \in k_l^0\}$  (a set of function values whose entries are zeros except at the point  $x_{l,k}$ ). Figure 2.1 depicts as an example  $\phi_{3,k}(x), k = 0, \dots, 4$ , for  $p = 4$ . As discussed above, the function  $\phi_{l,k}$  is continuous and its continuity increases as a function of  $p$ .

It can be noticed that the subdivision scheme is linear. Consequently, the interpolation function  $\tilde{f}(x)$  resulting from applying the scheme to  $\{f_{l,k}\}_{k \in k_l^0}$  is equivalent to a superposition of those interpolation functions of the data sets  $\{f_{l,r}\delta_{r,k} : r = k_l^0\}, k = 0, \dots, 2^l$ . More precisely, the interpolation function  $\tilde{f}$  is a linear combination of  $\phi_{l,k}(x)$ ,

$$\tilde{f}(x) = \sum_{k=0}^{2^l} f_{l,k} \phi_{l,k}(x) \quad (2.7)$$

The function  $\phi_{l,k}(x)$  is known as the interpolating scaling function [15, 58] (they are linearly independent and hence form a basis). These functions can be used to form an interpolating multiresolution analysis on the interval  $[0, 1]$  and subsequently a multiscale basis. Before providing more details on the properties of  $\phi_{l,k}$ , we note that these functions can be constructed using a different approach [15, 88]. In the later approach, the construction is based on functions which are dilation and translation of an auto-correlation of the Daubechies scaling function [54].



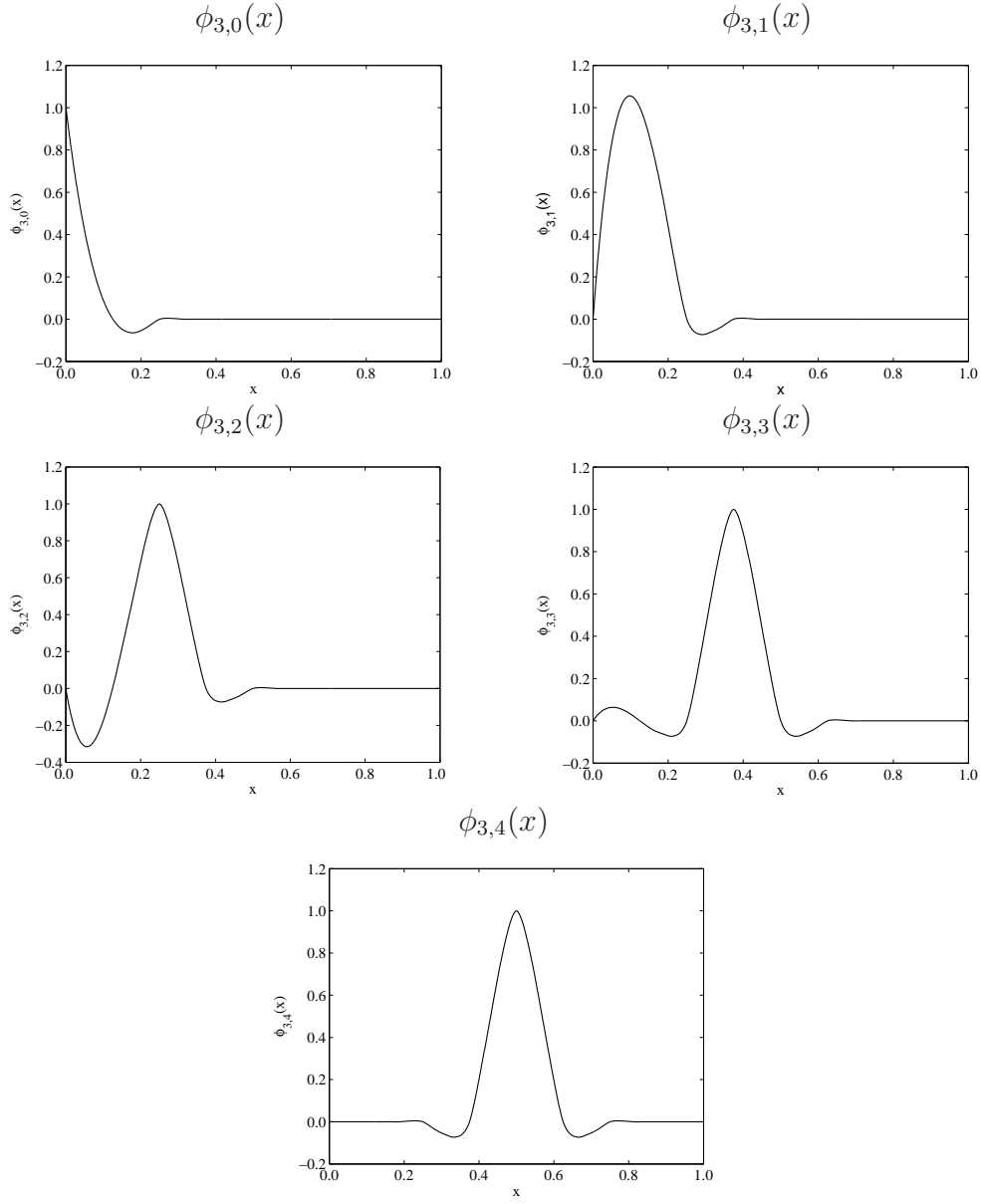


Figure 2.1: The interpolating scaling functions,  $\phi_{3,k}$ ,  $k = 0, \dots, 4$  for  $p = 4$

## 2.2 One-dimensional interpolating wavelet bases

The interpolating scaling function  $\phi_{j,k}(x)$  has the following properties:

i)  $\phi_{l,k}$  satisfied the so-called interpolation property on the dyadic grid  $V_l$ , *i.e.*

$$\phi_{l,k}(x_{l,r}) = \delta_{k,r}.$$

ii) The support of  $\phi_{j,k}$  is compact and  $|\text{supp } \phi_{l,k}|$  is of  $O(2^{-l})$ , more precisely,

$$\text{supp } \phi_{l,k} = [\max(0, 2^{-l}(k - p + 1)), \min(2^{-l}(k + p - 1), 1)] \quad (2.8)$$

iii) The function  $\phi_{l,k}$  is the solution of the two-scale relation

$$\phi_{l,k}(x) = \sum_{r=0}^{r=2^{l+1}} h_r^{l,k} \phi_{l+1,r}(x), \quad (2.9)$$

where  $h_r^{l,k}$  are the so-called filter coefficients. For  $r \in \{\min(0, 2k - p + 1), \dots, \max(2k + p - 1, 2^{l+1})\}$ , the filter coefficients are defined by

$$h_r^{l,k} = \phi_{l,k}(x_{l+1,r}) = \begin{cases} \delta_{r,2k} & \text{for } r \text{ even,} \\ L_{l,k}^{(r-1)/2}(x_{l+1,r}) & \text{for } r \text{ odd,} \end{cases} \quad (2.10)$$

and all other  $h_r^{l,k}$  are zero. Note that  $L_{l,k}^m$  is the Lagrange polynomial defined by (2.5).

iv) Polynomials on  $[0, 1]$  of degree less than  $p$  can be written as linear combinations of  $\{\phi_{l,k} : k = 0, \dots, 2^l\}$ . More precisely,

$$x^i = \sum_{k=0}^{k=2^l} (2^{-l}k)^i \phi_{l,k}(x), \quad i = 0, 1, \dots, p-1. \quad (2.11)$$

It is the property i) that distinguishes the interpolating scaling function from other non-interpolating scaling functions. In addition, this property implies that they are linearly independent and thus form a basis. Properties i), iii), and iv) are obvious from the subdivision scheme. Property ii) can be verified using the fact that, for a nonzero data set with finite support on  $V_l$ , one iteration of the subdivision scheme adds nonzero layers of size  $2^{-j-1}(p-1)$  on both side. Therefore, with infinite iterations, starting with the Kronecker data on  $V_l$  with nonzero element at the point  $x_{l,k}$ , the layer of total width  $\sum_{i=1}^{\infty} 2^{-l-i}(p-1) = (p-1)/2^l$  is added to both side of the point at  $2^{-l}k$ . Since the subdivision scheme is restricted to the interval  $[0, 1]$  (*i.e.* for a interpolated point near boundary, a set of points participating in the construction of  $\pi_{j,k}$  is adjusted accordingly.), one observed that (2.8) holds.

Let  $V_l$  denotes the span of  $\{\phi_{l,k}(x) : k \in k_l^0\}$ . Here the symbol  $V_l$  is overloaded to a space of functions. The specific meaning should be clear from the context. Owing to the interpolation property of the scaling functions, we define the interpolation operator  $I_l$  that maps a given function  $f(x)$  defined for  $x \in [0, 1]$  to the space  $V_l$ :

$$(I_l f)(x) \equiv \sum_{k=0}^{k=2^l} f_{l,k} \phi_{l,k}(x) \quad (2.12)$$

where  $f_{l,k} = f(x_{l,k})$ . Property iii) implies that the spaces  $V_l$  are nested, *i.e.*

$$V_{l-1} \subset V_l. \quad (2.13)$$

One therefore obtain a multiresolution analysis of  $V_l$ . Now, consider the complement space

$$W_l = V_{l+1} \ominus V_l. \quad (2.14)$$

In this particular setting, interpolating wavelets which form a basis of  $W_l$  can be simply chosen as noted by [58]:

$$\psi_{l,k}(x) = \phi_{l+1,2k+1}(x), \quad k \in k_l^1 = \{0, 1, \dots, 2^l - 2, 2^l - 1\}. \quad (2.15)$$

With wavelets defined as above, there exists a fast transform (to be described later) that maps function values to wavelet coefficients and *vice versa*.

Further decomposition of the space  $V_l$  until reaching  $V_{l_0}$  yields results similar to (1.6). Therefore, the interpolation function  $I_l f$  can be written in the multi-scale representation

$$(I_l f)(x) = \sum_{k=0}^{2^{l_0}} f_{l_0,k} \phi_{l_0,k}(x) + \sum_{j=l_0}^{l-1} \sum_{k=0}^{2^j-1} d_{j,k} \psi_{j,k}(x), \quad (2.16)$$

where  $d_{j,k}$  are wavelet coefficients. For a large classes of functions, [58] has shown that  $f = \lim_{l \rightarrow \infty} I_l f$ , *i.e.*

$$f(x) = \sum_{k=0}^{2^{l_0}} f_{l_0,k} \phi_{l_0,k}(x) + \sum_{j \geq l_0} \sum_{k=0}^{2^j-1} d_{j,k} \psi_{j,k}(x), \quad (2.17)$$

which represents the decomposition of  $f$  into a superposition of contributions from levels  $l \geq l_0$ . Information of these contributions is contained in the coefficients  $\{f_{l_0,k}\}_{k \in k_{l_0}^0}$  (the function values at the coarsest level) and wavelet coefficients  $\{\{d_{l,k}\}_{k \in k_l^1}\}_{l > l_0}$  on the finer scales.

Note that in the context of interpolating wavelets, each basis function is associated with one grid point. To be more specific, the scaling function  $\phi_{j,k}(x)$  is associated with the grid point  $x_{j,k}$  of  $V_j$  and the wavelet function  $\psi_{j,k}$  is associated with the grid point  $x_{j,k}^1$  of

$$W_j = \{x_{j,k}^1 = x_{j+1,2k+1} : k \in k_j^1\}, \quad (2.18)$$

the complement grid of  $V_j$  in  $V_{j+1}$ . It is obvious that the grid  $V_l$  is equivalent to  $V_{l_0} \cup W_{l_0} \cup \dots \cup W_{l-1}$ . Due to this decomposition of the grid and the notation used in (2.18), it can be seen that for any given dyadic point  $x_{q,k} \notin V_{l_0}$ , there is a unique index  $j, k$ , where  $j < q$ , such that  $x_{j,k}^1 = x_{q,k}$ . Thus, we have means to switch between the index of basis functions and their corresponding grid points. In this way, an action on the grid, *e.g.* discarding and including grid points, also implies an appropriate action on the index of the basis functions and *vice versa*.

### 2.2.1 Fast interpolating wavelet transform

For notational simplicity, let us denote (2.16) by  $f^l$ . Consider the difference of the approximation of  $f$  between the two consecutive space  $V_l$  and  $V_{l+1}$ :

$$w^l = f^{l+1} - f^l = \sum_{k=0}^{2^{l+1}} f_{l+1,k} \phi_{l+1,k} - \sum_{k=0}^{2^l} f_{l,k} \phi_{l,k} = \sum_{k=0}^{2^l-1} d_{l,k} \psi_{l,k}(x). \quad (2.19)$$

From (2.15), (2.10), and the interpolation property (property i)), the wavelet coefficients obtained by evaluating  $w_j(x)$  at  $x_{j,k}^1$  are given by

$$d_{l,k} = f_{l+1,2k+1} - \sum_{r=0}^{2^l} \underbrace{\phi_{l,r}(x_{l,k}^1)}_{= h_{2k+1}^{l,r}} f_{l,r}. \quad (2.20)$$

It can be verified that, for each  $k$ , there are only  $p$  filter coefficients  $h_{2k+1}^{l,r}$  having non-zero values and they are those with  $r \in \mathcal{X}_{l,k}$  (see (2.3) for the definition of the index set  $\mathcal{X}_{l,k}$ ). From (2.10), (2.5) and (2.4), it can be shown that

$$\begin{aligned}
\sum_{r=0}^{2^l} h_{2k+1}^{l,r} f_{l,r} &= \sum_{r=\min \mathcal{X}_{l,k}}^{\max \mathcal{X}_{l,k}} L_{l,r}^k(x_{l,k}^1) f_{l,r} = \pi_{l,k}(x_{l,k}^1) \\
&= \pi_{l,k}(x_{l+1,2k+1})
\end{aligned} \tag{2.21}$$

Notice that to compute  $d_{l,k}$ , one requires only  $f_{l+1,2k+1}$  and those  $f_{l,r}$  for  $r \in \mathcal{X}_{l,k}$  (see figure 2.2 for illustration). In addition, it is evident that the wavelet coefficient indicates how large  $f_{l+1,2k+1}$  deviates from the value predicted by the local interpolation polynomial.

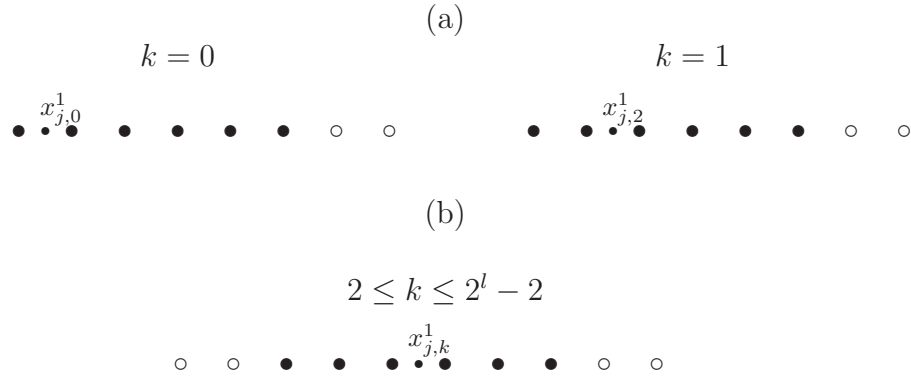


Figure 2.2: Points required in the calculation of wavelet coefficients  $d_{j,k}$  for  $p = 6$  (a)  $k = 0, 1$ , (b)  $k = 2, \dots, 2^{l-1} - 2$ . Note that the larger circles denote the points in  $V_l$ . The filled circles represent points participating in the calculation of  $d_{j,k}$ .

By applying (2.20) until reaching the coarsest level, one obtains an algorithm for determining the wavelet coefficients of  $I_l f$  of any given  $f$ . The interpolating wavelet transform mapping the function values to the wavelet coefficients is summarized in the Algorithm 1. Notice that the scaling function coefficients  $f_{j,k}$  are not altered by the wavelet transform (they correspond to the function values at

their associated points.). Thus, it is not necessary to perform the the wavelet transform in a top-to-bottom manner (*i.e.* from level  $j = l - 1$  down to  $l_0$ ). This feature is unique to this particular wavelet basis and useful in particular for an on-the-fly compression of function (see Chapter 2). The transform being done in the top-to-bottom manner has the advantage in that one can overwrite in place  $f_{j+1,2k+1}$  by  $d_{j,k}$  and hence require only one vector for storage.

---

**Algorithm 1** *1DFWT*

---

Given function values  $\{f_{l,k}\}_{k \in k_l^0}$   
**for**  $j = l - 1$  to  $l_0$  **do**  
  **for**  $k = 0$  to  $2^{j-1} - 1$  **do**  
     $m = 2^{l-j-1}(2k + 1)$   
     $f_{l,m} = f_{l,m} - \sum_{r=0}^{2^j} h_{2k+1}^{j,r} f_{l,r} 2^{l-j}$   
  **end for**  
**end for**

---

For a given set of wavelet coefficients, one can obtain the associated function values using the inverse wavelet transform. The procedure starts from the coarsest level  $j = l_0$  to obtain  $f_{j+1,2k+1}$  by adding  $d_{j,k}$  to the value predicted by the local polynomial, *i.e.*

$$f_{j+1,2k+1} = d_{j,k} + \sum_{r=0}^{2^l} h_{2k+1}^{l,r} f_{l,r}. \quad (2.22)$$

By applying (2.22) recursively until reaching level  $j = l - 1$ , one obtains the function values on the grid level  $l$ . It is important to note that the inverse wavelet transform must be performed strictly in the bottom-to-top order since the data  $\{f_{j,k}\}_{k \in k_l^0}$  must be available before  $f_{j+1,2k+1}$  can be computed. Algorithm (2)

summarizes the inverse wavelet transform procedure. Note that this algorithm assumes that the wavelet coefficient  $d_{j,k}$  is stored in the entry  $f_{l,2^{l-j-1}(2k+1)}$  of the input data.

---

**Algorithm 2** *1DIWT*

---

Given wavelet coefficients  $\{f_{l,k}\}_{k \in k_l^0}$   
**for**  $j = l_0$  to  $l - 1$  **do**  
  **for**  $k = 0$  to  $2^j - 1$  **do**  
     $m = 2^{l-j-1}(2k + 1)$   
     $f_{l,m} = f_{l,m} + \sum_{r=0}^{2^j} h_{2k+1}^{j,r} f_{l,2^{l-j}r}$   
  **end for**  
**end for**

---

It can be seen that the calculation of one wavelet coefficient requires  $p$  operations. The number of operations required for the transform or its inverse at the level  $j$  is of order  $p2^j$  and thus the total number of operations is of order  $p \sum_{j=l_0}^{l-1} 2^j = p(2^l - 1) \sim O(2^l)$ . This indicates that the cost of the wavelet transform or its inverse is proportional to the total number of data values. The proportionality constant varies linearly with the order of basis.

### 2.2.2 Remarks

The calculation of wavelets coefficients  $d_{l,k}$  requires the associated nonzero filter coefficients  $h_{2k+1}^{l,r}, r = \min \mathcal{X}_{l,k}, \dots, \max \mathcal{X}_{l,k}$ . Since the grid is composed of equally distant points, the values of filter coefficients are independent of the level  $l$ . For  $p/2 - 1 \leq k, m \leq 2^l - p/2$ , the filter coefficients  $\{h_{2m+1}^{l,r}\}_{r \in k_l^0}$  are just the shifted versions of  $\{h_{2k+1}^{l,r}\}_{r \in k_l^0}$ . Furthermore, the filter coefficients associated



with  $2^l - p/2 \leq k \leq 2^l - p/2 - 2$  correspond to those of  $2^l - k$  with reverse-order entries. Thus, it is sufficient to calculate the filters  $\{h_{2k+1}^{l,r}\}$  associated with  $k = 0, 1, \dots, p/2 - 1$ . Table 2.1 lists, for example, the non-zero filter coefficients  $h_{2k+1}^{l,r}$ ,  $r \in X_{l,r}$ , needed for the calculation of wavelet coefficients for  $p = 6$ .

TABLE 2.1

THE VECTOR OF NONZERO FILTER COEFFICIENTS  $h_{2k+1}^{l,r}$  FOR  $p = 6$ .

$r$	0	1	2	3	4	5
$k = 0$	63/256	315/256	-105/128	63/128	-45/256	7/256
$k = 1$	7/256	105/256	105/128	-35/128	21/256	-3/256
$k = 2$	3/256	-25/256	75/128	75/128	-25/256	3/256

### 2.2.3 Approximation property

There are several approaches used to estimate the accuracy of the interpolation function  $I_j f$ . Here, we discuss an estimate based on the connection between the Lagrange interpolation and interpolating wavelets. Since the second term on the right hand side of (2.20) is the interpolated value of the Lagrange interpolation, for a smooth function, by using the remainder term of Lagrange interpolation, it can be verified that the wavelet coefficient obeys the following equation:

$$d_{j,k} = \frac{f^{(p)}(\xi)}{(p+1)!} \prod_{i=\min(\mathcal{X}_{j,k})}^{\max(\mathcal{X}_{j,k})} (x_{j+1,2k+1} - x_{j,i}) = C_k 2^{-pj} f^{(p)}(\xi), \quad (2.23)$$

where  $C_k$  is an appropriate constant (it is independent of  $f$  and  $j$ ) and  $\xi \in [\min(X_{j,k}), \max(X_{j,k})]$  (see (2.2) for the definition of  $X_{j,k}$ ). This suggests that, for a given  $p$ , the wavelet coefficients decay exponentially with respect to level  $j$ . Furthermore, the higher the order  $p$  is, the faster the decay becomes.

Now consider the approximation error of  $I_j f$ ,

$$f - I_j f = \sum_{l=j}^{\infty} \sum_{k=0}^{2^l-1} d_{l,k} \psi_{l,k}(x). \quad (2.24)$$

Assuming that  $f$  is  $p$  time differentiable everywhere and using (2.23) and the fact that  $\psi_{l,k}$  has compact support, it can be found that

$$\begin{aligned} \|f - I_j f\|_{\infty} &= \max_{x \in [0,1]} \left| \sum_{l=j}^{\infty} \sum_{k=0}^{2^l-1} d_{l,k} \psi_{l,k}(x) \right| \\ &\leq C_1 C_2 \max_{x \in [0,1]} |f^{(p)}(\xi)| \sum_{l=j}^{\infty} 2^{-pj} \\ &= C_1 C_2 \max_{x \in [0,1]} |f^{(p)}(\xi)| 2^{-pj} \underbrace{\sum_{l=0}^{\infty} (1/2^p)^l}_{= 1/(1-2^{-p})} \\ &= C(f, p) 2^{-pj} \end{aligned} \quad (2.25)$$

where  $C_1$  denotes the maximum number of wavelets at any level whose support intersect any single point in  $[0, 1]$ ,  $C_2$  the maximum norm of the wavelet basis, and  $C(f, p)$  the appropriate constant that depends on  $f$  and  $p$ . This estimate implies that for a sufficiently smooth function,  $I_j f$  converge to  $f$  uniformly in the  $L_{\infty}$  norm and that the error of the interpolation  $I_j f$  is  $O(2^{-pj})$ . In other word,  $I_j f$  can be computed to any prescribed accuracy by using a sufficient large  $j$ .

Estimates for cases where  $f$  has weaker smoothness have been investigated by [58] and [88]. In [88], it has been proven that for  $f \in \mathcal{C}^n([0, 1])$ ,  $n \geq 1$ , the space of  $f$  that is  $(n - 1)$ -times continuously differentiable, with finite associated semi-norm

$$|u|_n := \sup \left\{ \left| \frac{u^{n-1}(x+h) - u^{n-1}(x)}{h} \right|_x, (x+h) \in [0, 1] \right\}, \quad (2.26)$$

where  $\sup$  denotes the least upper bound, the following estimate holds:

$$\|f - I_j f\|_\infty \leq C_1 2^{-\min(p,n)j} |f|_{\min(p,n)}, \quad (2.27)$$

where  $C_1$  is a constant. It is noted that the space  $\mathcal{C}^n$  is larger than the space  $C^n$  of  $n$ -times continuously differentiable functions.

The estimate for the Sobolev norm can be found in [13]. In addition, it has been proven that interpolating wavelet coefficients can characterize various classes of functional spaces (see [15, 45, 58]). In particular, the norm equivalence between the Sobolev norm of a function and the sequence norm of wavelet coefficients, *i.e.* a relation similar to (1.11) holds. However, such norm equivalence is valid for a relatively small range of Sobolev spaces.

#### 2.2.4 Wavelet transform on irregular grid

The wavelet transform given above can be extended straightforwardly to compute exactly, from given function values on a grid of irregular points, the corresponding wavelet coefficients associated such points, provided that the irregular grid satisfies a so-called minimum index set. This condition ensures that the points participating in the calculation of the wavelet coefficients belong to the grid.

Let  $\mathcal{V}_l = \{x_{l,k} : k \in \kappa_l^0 \subset k_l^0\}$  denote a grid of irregular points. As in the regular grid case,  $\mathcal{V}_l$  can be decomposed into complement grids, *i.e.*

$$\mathcal{V}_l = \mathcal{V}_{l_0} \cup \mathcal{W}_{l_0} \cup \cdots \cup \mathcal{W}_{l-1}, \quad (2.28)$$

where  $\mathcal{W}_j = \{x_{j,k}^1\}_{k \in \kappa_j^1}$ ,  $\kappa_j^1 = \{k : k \in k_j^1 \text{ and } x_{j,k}^1 \in \mathcal{V}_l\}$  and, for  $j < l$ ,  $\mathcal{V}_j = \{x_{j,k}\}_{k \in \kappa_j^0}$ ,  $\kappa_j^0 = \{k : k \in k_l^0 \text{ and } x_{j,k} \in \mathcal{V}_l\}$ . As a reminder, for any given dyadic point  $x_{q,k} \notin V_{l_0}$ , one can find uniquely an index  $(j, m)$  such that  $x_{j,m}^1 = x_{q,k}$ . Subsequently, (2.20) applies if the point  $x_{j,k}^1 \in \mathcal{W}_j$  and

$$\{x_{j,r} : r \in \mathcal{X}_{j,k}\} \quad (2.29)$$

belong to the irregular grid  $\mathcal{V}_l$ , so that the coefficient  $d_{j,k}$  can be computed exactly. Consequently, if the condition above holds for every points in  $\mathcal{W}_j$  for all  $j = l_0, \dots, l-1$ , one can compute the exact coefficients corresponding to the following wavelets

$$\{\{\phi_{l_0,k}(x)\}_{k \in \mathcal{V}_{l_0}}, \{\{\psi_{j,k}(x)\}_{k \in \kappa_j^1}\}_{j=l_0}^{j=l-1}\}. \quad (2.30)$$

Conversely, if the irregular grid associated with wavelets (2.30) satisfies the above condition and the coefficients of these wavelets are known exactly, one can recover the original function values on such irregular grid.

We refer to the grid verifying the conditions discussed above as a *minimal index set* for the interpolating wavelet transform. Subsequently, the wavelet transform and its inverse can be generalized immediately to grid satisfying the minimal index set. Moreover, since computing one wavelet coefficient requires  $p$  operations, the total number of operations required for the wavelet transform or its inverse is  $O(N)$ , where  $N = \dim \mathcal{V}_l$ . This means that the cost of the wavelet transform or

its inverse remains proportional to the number of data.

In practice, one normally obtains the index set  $\kappa_j^1$  (and hence  $\mathcal{W}_j$ ) from a thresholding procedure, *i.e.* retaining wavelets whose coefficients are larger than a prescribed threshold value. In general, the grid obtained does not satisfy the minimal set condition. This issue can be remedied by simply including the additional points such that the augmented grid verifies such the condition. The augmented grid can then be used in the wavelet transform. The enlargement of the grid/index set can be carried naturally from the finest level down to the coarsest level. Assuming that  $\mathcal{V}_l$  is given (thus one also has the associated index sets  $\kappa_{l_0}^0, \kappa_{l_0}^1, \dots, \kappa_{l-1}^0$ ), one first determines the grid points needed in the calculation of  $d_{l-1,k}$ ,  $k \in \kappa_{l-1}^1$  (they are  $x_{j-1,k}^1$  and those points in (2.29)). If one (or more) of these points does not belong to the irregular grid, it is included in the grid. Note that the missing point reside in either one of the complement grid of level lower than  $l - 1$  or in the coarsest grid  $V_{l_0}$ . Then, repeat this process by adding missing points needed for the calculation of wavelet coefficients associated with points  $x_{l-2,k}^1$  with  $k$  in the index set  $\kappa_{l-2}^1$  augmented previously. After repeating this process until reaching level  $l_0$ , one obtains the index set and grid verifying the minimal index set condition. Such a procedure is outlined in Algorithm 3. This algorithm uses the `fill` function outlined in Algorithm 4. In essence, `fill`

---

**Algorithm 3** *AUGMENT*

---

Establish index set  $\kappa_{l_0}^0, \kappa_{l_0}^1, \dots, \kappa_{L-1}^1$  for given  $\mathcal{V}_L$   
**for**  $j = l - 1$  down to  $l_0 + 1$  **do**  
    **for**  $k \in \kappa_j^1$  **do**  
        call `fill`( $j, k$ )  
    **end for**  
**end for**

---

constructs a grid of irregular points with minimum index set from a grid of single point. In this function,  $\mathcal{K}(j, 0, r)$  denote the determination of an index  $(i, 1, q)$  such that  $x_{l,q}^1 = x_{j,r}$  for  $x_{j,r} \notin V_{l_0}$ . For  $x_{j,r} \in V_{l_0}$ ,  $\mathcal{K}(j, 0, r)$  returns  $(l_0, 0, m)$ , where  $m$  is an appropriate index such that  $x_{l_0,m} = x_{j,r}$ .

---

**Algorithm 4** fill( $l, k$ )

---

```

 $j = l + 1$ 
for  $n = \min \mathcal{X}_{l,k}$  to  $\max \mathcal{X}_{l,k}$  do
   $r = 2n, (i, e, q) = 2\mathcal{K}(j, 0, r)$ 
  #  $L$  is the finest level
  if  $x_{j,r} \notin \mathcal{V}_L$  then
    Enlarge grid  $\mathcal{V}_l = \mathcal{V}_l \cup \{x_{j,r}\}$ 
    if  $x_{l,r} \notin V_{l_0}$  then
      Enlarge index set  $\kappa_l^1 = \kappa_l^1 \cup \{q\}$ 
    else
      Enlarge index set  $\kappa_{l_0}^0 = \kappa_{l_0}^0 \cup \{q\}$ 
    end if
  end if
  if  $x_{l,r} \notin V_{l_0}$  then
    call fill( $j, m$ )
  end if
end for

```

---

## CHAPTER 3

### CONSTRUCTION OF IRREGULAR GRIDS

In this chapter, we describe elements required in an algorithm for the adaptive solution of Partial Differential Equations (PDEs) in  $d$  spatial dimension. These elements include mainly means for determining a  $d$ -dimensional adaptive grid, for accomplishing tasks such as addition, multiplication and interpolation, and for computing the derivative approximation on such a grid. Here, the  $d$ -D adaptive grid (we often refer to it as the  $d$ -D irregular grid) is induced from an approximation of higher-dimensional interpolating wavelets. Algorithms related to such a grid are dictated or facilitated by the connection between interpolating wavelets and function values on dyadic points. The  $d$ -D wavelet basis on  $[0, 1]^d$  is usually constructed by mean of tensor products. There are two approaches to accomplish such a construction: an MRA approach [54] and an MRA- $d$  approach [70]. The former is more widely used in the wavelet literature. A method for adaptive solutions of the so-called Wavelet Adaptive Multiresolution Representation (WAMR), originally developed by [111] and further developed in this work, is based on the use of wavelets constructed from this approach. Although wavelets constructed from the MRA- $d$  approach have received less attention, the adaptive methods of [71, 73, 91] based on this type of wavelets are algorithmically elegant and powerful. With slight modification of the data structure designed for the

MRA approach [110], we can implement such methods. From this reason, we include brief detail of the MRA- $d$  approach.

Below, we first introduce the higher dimensional wavelets on  $[0, 1]^d$  constructed by the conventional MRA approach and then describe associated elements required in an algorithm for adaptive solutions of PDEs. In addition, brief detail of the MRA- $d$  approach is included at the end of this chapter. Note that while we use the material in this chapter for the adaptive solution of PDEs, the material can be applied in other applications as well.

### 3.1 Higher-dimensional interpolating scaling function

Higher-dimensional bases are conventionally constructed by considering the tensor product of one-dimensional multiresolution analyses. In particular, to construct an interpolating basis for a function defined on  $[0, 1]^d$ ,  $d \in \mathbf{N}$ ,  $d \geq 1$ , one considers

$$\mathbf{V}_j = \underbrace{V_j \otimes V_j \otimes \cdots \otimes V_j}_{d\text{-times}} = \text{span}\{f_1(x_1)f_2(x_2) \cdots f_d(x_d), f_n \in V_j\}, \quad (3.1)$$

where  $V_j$  is the space of 1-D functions defined in Chapter 2. Consequently,  $\mathbf{V}_j$  is therefore nested,

$$\mathbf{V}_j \subset \mathbf{V}_{j+1}. \quad (3.2)$$

Since the  $\{\phi_{j,k}\}_{k \in k_j^0}$ , forms a basis for  $V_j$ , the product functions

$$\Phi_{j,\mathbf{k}}(\mathbf{x}) = \prod_{i=1}^d \phi_{j,k_i}(x_i), \quad \mathbf{k} \in \mathbf{k}_j^0 = \underbrace{k_j^0 \times k_j^0 \times \cdots \times k_j^0}_{d\text{-times}} \quad (3.3)$$

constitute a basis for  $\mathbf{V}_j$ . Here,  $\mathbf{k} = (k_1, \dots, k_d) \in \mathbf{N}^d$ ,  $\mathbf{x} \in [0, 1]^d$  and  $k_j^0 =$



$\{0, \dots, 2^j\}$ . These product functions are referred to as the higher-dimensional scaling functions (or simply as scaling functions). As a consequence of the one-dimensional scaling functions, the higher-dimensional scaling functions have properties that are generalization of Properties i)-iv) (see Chapter 2, section 2.2).

As in the one-dimensional case, higher order scaling functions verify the interpolation property; more precisely

$$\Phi_{j,\mathbf{k}}(\mathbf{x}_{j,\mathbf{m}}) = \delta_{k_1-m_1} \cdots \delta_{k_d-m_d}, \quad \mathbf{x}_{j,\mathbf{k}} \equiv (x_{j,k_1}, \dots, x_{j,k_d}), \quad (3.4)$$

where  $\mathbf{x}_{j,\mathbf{k}}$  are the dyadic points. Therefore, the projection of a given function  $f(\mathbf{x})$  on the space  $\mathbf{V}_j$  can be obtained through the interpolation operator  $I_j$  defined by:

$$(I_j f)(\mathbf{x}) = \sum_{\mathbf{k} \in \mathbf{k}_j^0} f_{j,\mathbf{k}} \Phi_{j,\mathbf{k}}(\mathbf{x}), \quad (3.5)$$

where the scaling function coefficients are simply  $f(\mathbf{x}_{j,\mathbf{k}})$ , the function values at points  $\mathbf{x}_{j,\mathbf{k}}$ .

### 3.2 MRA Approach

This approach has been widely used in the construction of the higher-dimensional wavelets [54]. The main idea of this approach is to form an ordered sequence of approximation spaces and subsequently seek the wavelet functions that form a basis of the complement spaces of these approximation spaces. As in the one-dimensional case, let the space  $\mathbf{W}_j$  be the complement of  $\mathbf{V}_j$  in  $\mathbf{V}_{j+1}$ . For notational simplicity, let  $\mathbf{e} = (e_1, \dots, e_d) \in \{0, 1\}^d \setminus \{\mathbf{0}\}$ ,  $W_l^0 \equiv V_l$  and  $W_l^1 \equiv W_l$ . Then from (3.1) and (2.14), we have that

$$\begin{aligned}
\mathbf{V}_{l+1} &= \underbrace{(W_l^0 \oplus W_l^1) \otimes (W_l^0 \oplus W_l^1) \otimes \cdots \otimes (W_l^0 \oplus W_l^1)}_{d\text{-times}} \\
&= \underbrace{(W_l^0 \otimes W_l^0 \otimes \cdots \otimes W_l^0)}_{d\text{-times}} \oplus \left[ \bigoplus_{\mathbf{e} \in \{0,1\}^d \setminus \{\mathbf{0}\}} \mathbf{W}_l^{\mathbf{e}} \right] \\
&= \mathbf{V}_l \oplus \mathbf{W}_l
\end{aligned} \tag{3.6}$$

where the product spaces  $\mathbf{W}^{\mathbf{e}}$  are given by

$$\mathbf{W}_l^{\mathbf{e}} = \bigotimes_{i=1}^d W_l^{e_i}. \tag{3.7}$$

It follows that for  $\lfloor \log_2(p) \rfloor \leq l_0 \leq l$

$$\mathbf{V}_{l+1} = \mathbf{V}_{l_0} \oplus \left( \bigoplus_{j=l_0}^l \mathbf{W}_j \right). \tag{3.8}$$

where  $l_0$  denotes the coarsest level.

For each level  $l$ , the complement space  $\mathbf{W}_l$  is composed of  $2^d - 1$  product spaces and as a result the basis functions for  $\mathbf{W}_l$  consist of the basis functions for those  $2^d - 1$  different product spaces:

$$\Psi_{l,\mathbf{k}}^{\mathbf{e}}(\mathbf{x}) = \prod_{i=1}^d \psi_{l,k_i}^{e_i}(x_i), \quad \mathbf{k} \in \mathbf{k}_i^{\mathbf{e}} = k_l^{e_1} \times k_l^{e_2} \times \cdots \times k_l^{e_d}, \tag{3.9}$$

for  $\mathbf{e} \in \{0,1\}^d \setminus \{\mathbf{0}\}$ , where  $\psi_{j,k}^0 \equiv \phi_{j,k}$  and  $\psi_{j,k}^1 \equiv \psi_{j,k}$ . For example, the 2-D wavelet basis functions consist of the following three different product functions:

$$\begin{aligned}
\Psi_{j,\mathbf{k}}^{(1,0)}(x_1, x_2) &= \psi_{j,k_1}(x_1) \phi_{j,k_2}(x_2), \quad \mathbf{k} \in k_l^1 \times k_l^0, \\
\Psi_{j,\mathbf{k}}^{(0,1)}(x_1, x_2) &= \phi_{j,k_1}(x_1) \psi_{j,k_2}(x_2), \quad \mathbf{k} \in k_l^0 \times k_l^1, \\
\Psi_{j,\mathbf{k}}^{(1,1)}(x_1, x_2) &= \psi_{j,k_1}(x_1) \psi_{j,k_2}(x_2), \quad \mathbf{k} \in k_l^1 \times k_l^1.
\end{aligned} \tag{3.10}$$

Note that the number of basis function for  $\mathbf{W}_l^e$  is  $\prod_{i=1}^d (2^l - e_i + 1)$  and the total number of basis functions for  $\mathbf{W}_l$  is  $(2^{l+1} + 1)^d - (2^l + 1)^d$ . It is evident from (3.8) that the interpolation  $I_l f$  can be rewritten in the multiscale representation

$$f_l(\mathbf{x}) = (I_l f)(\mathbf{x}) = \sum_{\mathbf{k} \in \mathbf{k}_{l_0}^0} f_{l_0, \mathbf{k}} \Phi_{l_0, \mathbf{k}}(\mathbf{x}) + \sum_{j=l_0}^{l-1} \sum_{\mathbf{e} \in \{0,1\}^d \setminus \{\mathbf{0}\}} \sum_{\mathbf{k} \in \mathbf{k}_j^e} d_{j, \mathbf{k}}^e \Psi_{j, \mathbf{k}}^e(\mathbf{x}), \quad (3.11)$$

where  $f_{l_0, \mathbf{k}}$  denote the scaling function coefficients at the coarsest level and  $d_{j, \mathbf{k}}^e$  denote the wavelet coefficients.

As in 1-D case, within the context of the interpolating wavelet, each basis function is associated with one grid point. To describe this aspect, let  $\mathbf{V}_l$  be a  $d$ -D grid  $\mathbf{V}_l = \{\mathbf{x}_{j, \mathbf{k}}\}_{\mathbf{k} \in \mathbf{k}_j^0}$ . It is the so-called regular or full grid. Furthermore, let  $\mathbf{W}_l$  be the complement grid of  $\mathbf{V}_l$  in  $\mathbf{V}_{l+1}$ . Analogous to the complement space, the complement grid  $\mathbf{W}_l$  consists of  $2^d - 1$  different grids, *i.e.*

$$\mathbf{W}_l = \bigcup_{\mathbf{e} \in \{0,1\}^d \setminus \{\mathbf{0}\}} \mathbf{W}_l^e, \quad \mathbf{W}_l^e = \{\mathbf{x}_{j, \mathbf{k}}^e = \mathbf{x}_{j, 2\mathbf{k} + \mathbf{e}}, \mathbf{k} \in \mathbf{k}_j^e\}. \quad (3.12)$$

Obviously, the grid point  $\mathbf{x}_{l, \mathbf{k}}$  of  $\mathbf{V}_l$  associates with the scaling function  $\Phi_{j, \mathbf{k}}(\mathbf{x})$  and the grid point  $\mathbf{x}_{l, \mathbf{k}}^e$  of  $\mathbf{W}_l^e$  with the wavelet function  $\Psi_{j, \mathbf{k}}^e(\mathbf{x})$ . It can be seen that the grid  $\mathbf{V}_l$  is equivalent to  $\mathbf{V}_{l_0} \cup \mathbf{W}_l \cup \dots \cup \mathbf{W}_{l-1}$ . Because of this decomposition of the grid and the notation used in (3.12), there are means to switch reciprocally between indices of basis functions and their associated grid points. More precisely, for any given dyadic point  $\mathbf{x}_{l, \mathbf{k}} \notin \mathbf{V}_{l_0}$ , there is a unique index  $(j, \mathbf{e}, \mathbf{m})$ , where  $j < l$ , such that  $\mathbf{x}_{j, \mathbf{m}}^e = \mathbf{x}_{l, \mathbf{k}}$ . For a given index  $(l, \mathbf{e}, \mathbf{k})$ , the following relation holds  $\mathbf{x}_{l, \mathbf{k}}^e = \mathbf{x}_{l+i+1, 2^i(2\mathbf{k} + \mathbf{e})}$ ,  $i \geq 0$ .

### 3.2.1 $d$ -D interpolating wavelet transform

The wavelet coefficients  $d_{l,\mathbf{k}}^{\mathbf{e}}$  can be computed directly using the following formula:

$$d_{l,\mathbf{k}}^{\mathbf{e}} = \mathbf{I}_{l,\mathbf{k}}^{\mathbf{e}} f \equiv \left( \prod_{i=1}^d I_{l,k_i}^{e_i} \right) f, \quad (3.13)$$

where

$$I_{l,k}^1 = \left[ -h_{2k+1}^{l,\min(\mathcal{X}_{l,k})}, -h_{2k+1}^{l,\min(\mathcal{X}_{l,k})+1}, \dots, -h_{2k+1}^{l,k}, 1, -h_{2k+1}^{l,k+1}, \dots, -h_{2k+1}^{l,\max(\mathcal{X}_{l,k})-1}, -h_{2k+1}^{l,\max(\mathcal{X}_{l,k})} \right], \quad (3.14)$$

(see (2.10) for the definition of  $h_k^{l,r}$ ) denotes a 1-D stencil associated with the points

$$X_{l,k}^1 = \left[ x_{l,\min(\mathcal{X}_{l,k})}, x_{l,\min(\mathcal{X}_{l,k})+1}, \dots, x_{l,k}, x_{l+1,2k+1}, x_{l,k+1}, \dots, x_{l,\max(\mathcal{X}_{l,k})-1}, x_{l,\max(\mathcal{X}_{l,k})} \right], \quad (3.15)$$

and  $I_{l,k}^0 = [1]$  is a stencil associated with the point  $X_{l,k}^0 = [x_{l,k}]$ . The multiplication  $\mathbf{I}_{l,\mathbf{k}}^{\mathbf{e}}$  must be understood as a  $|\mathbf{e}|_1$ -dimensional stencil ( $|\mathbf{e}|_1 = \sum e_i$ ). Here,  $f$  on the right hand side of (3.13) are the function values at the associated stencil points. It can be observed from (3.13) that  $d_{j,\mathbf{k}}^{\mathbf{e}}$  is in fact the difference between  $f(\mathbf{x}_{j,\mathbf{k}}^{\mathbf{e}})$  and a value predicted by polynomial approximation, and thus provides information on the local regularity of the function. Conversely, the function value at point  $x_{l,\mathbf{k}}^{\mathbf{e}}$  can be recovered by adding the predicted value to  $d_{l,\mathbf{k}}^{\mathbf{e}}$ , *i.e.*

$$f_{l+1,2\mathbf{k}+\mathbf{e}} = d_{l,\mathbf{k}}^{\mathbf{e}} - \mathbf{I}_{l,\mathbf{k}}^{\mathbf{e}} \tilde{f} = d_{l,\mathbf{k}}^{\mathbf{e}} - \left( \prod_{i=1}^d I_{l,k_i}^{e_i} \right) \tilde{f}, \quad (3.16)$$

where  $\tilde{f} = 0$  for the point  $x_{l,\mathbf{k}}^{\mathbf{e}}$  and corresponds to the function values at other points in the stencil. If wavelet coefficients on level  $l$  are zero, (3.16) reduces to the interpolation scheme. While (3.14) and (3.15) allow us to gain insight on wavelet

coefficients, they are computationally inefficient. The calculation of  $d_{l,\mathbf{k}}^{\mathbf{e}}$  requires  $p^{|\mathbf{e}|_1}$  number of operations. The number of operation required can be reduced by taking advantage of the tensor product nature of the basis.

Due to the nature of the tensor product, the calculation of wavelet coefficients on the level  $l$  from given data  $\{f_{l,\mathbf{k}}\}_{\mathbf{k} \in \mathbf{k}_l^0}$  reduces simply to the application of  $d$ -consecutive single-step 1-D wavelets transforms: first in the  $x_1$ -direction, then in the  $x_2$ -direction, ..., and finally in the  $x_d$ -direction (the single-step 1-D wavelet transform maps the 1-D data  $\{f_{l,k}\}_{k \in k_l^0}$  to the associated wavelet coefficients  $\{\{f_{l-1,k}\}_{k \in k_{l-1}^0}, \{d_{l-1,k}\}_{k \in k_{l-1}^1}\}$  by means of (2.20)). To provide more detail of this procedure, we introduce the following notations. Let  $\widehat{\mathbf{k}}^i = (k_1, \dots, k_{i-1}, k_{i+1}, \dots, k_d)$  denote a subvector of  $\mathbf{k} \in N^d$  and  $H_l$  denote the 1-D single-step wavelet transform. Note that  $H_l$  can be written as a matrix of dimension  $(2^l + 1) \times (2^l + 1)$  with an entry  $(i, j)$  given by

$$(H_l)_{i,j} = \begin{cases} \delta_{i-j} & \text{for } i \text{ even,} \\ \delta_{i-j} & \text{for } i \text{ odd and } j \text{ odd,} \\ -h_i^{l,j/2} & \text{for } i \text{ odd and } j \text{ even,} \end{cases} \quad (3.17)$$

for  $i, j \in k_l^0$ . With  $H_l$  given above, entries with odd index of the resulting vector are scaling function coefficients and entries with even index are wavelet coefficients. Note that the entries of the resulting vector can be arranged into a different order by simply using a permutation matrix. The procedure for computing the wavelet coefficients on level  $l$  thus reads

$$\{d_{l,\mathbf{k}}\}_{\mathbf{k} \in \mathbf{k}_l^0} = \{H_l\{H_l \cdots \{H_l\{H_l\{f_{j,\mathbf{k}}\}_{\widehat{\mathbf{k}}^1 = \text{const}}\}_{\widehat{\mathbf{k}}^2 = \text{const}}\} \cdots \}_{\widehat{\mathbf{k}}^d = \text{const}}\}. \quad (3.18)$$

where  $\hat{\mathbf{k}}^i$  is the subvector of index  $\mathbf{k} \in \mathbf{k}_l^0$  and  $\{f_{j,\mathbf{k}}\}_{\hat{\mathbf{k}}^i = \text{const}}$  represents the vector of dimension  $2^l + 1$  obtained by holding  $\hat{\mathbf{k}}^i$  unchanged and allowing the index in the  $x_i$ -direction to vary. Note that the  $\{f_{l,\mathbf{k}}\}_{\hat{\mathbf{k}}^i = \text{const}}$  is a subvector if  $\{f_{l,\mathbf{k}}\}_{\mathbf{k} \in \mathbf{k}_l^0}$  is viewed as a vector and it is a  $\hat{\mathbf{k}}^i$ -th row vector if  $\{f_{l,\mathbf{k}}\}_{\mathbf{k} \in \mathbf{k}_l^0}$  is defined as a  $d$ -dimensional array. Here  $\{H_l\{g_{l,\mathbf{k}}\}_{\hat{\mathbf{k}}^i = \text{const}}\}$  is understood as the data  $\{\tilde{g}_{l,\mathbf{k}}\}_{\mathbf{k} \in \mathbf{k}_j^0}$  with  $\{\tilde{g}_{j,\mathbf{k}}\}_{\hat{\mathbf{k}}^i = \text{const}} = H_l\{g_{j,\mathbf{k}}\}_{\hat{\mathbf{k}}^i = \text{const}}$  for the given subvector  $\hat{\mathbf{k}}^i$ . In (3.18), the wavelet coefficient  $d_{l-1,\mathbf{k}}^{\mathbf{e}}$  is kept in the entry  $d_{l,2\mathbf{k}+\mathbf{e}}$  and the scaling function coefficient  $f_{l-1,\mathbf{k}}$  is kept in the entry  $d_{l,2\mathbf{k}}$ . The inversion procedure can be accomplished simply by replacing  $H_l$  in (3.18) by  $H_l^{-1}$ , a single step 1-D inverse transform. In this case,  $\{f_{l,\mathbf{k}}\}_{\mathbf{k} \in \mathbf{k}_l^0}$  denotes wavelet coefficients (*i.e.*, the entry  $f_{l,2\mathbf{k}+\mathbf{e}}$  corresponds to  $d_{l-1,\mathbf{k}}^{\mathbf{e}}$  and  $f_{l,2\mathbf{k}}$  to  $f_{l-1,\mathbf{k}}$ ). In other words, the inversion procedure consists of the application of  $d$ -consecutive single-step 1-D inverse wavelet transforms.

An overall algorithm for the  $d$ -dimensional wavelet transform is a generalization of the 1-D algorithm. It consists of performing the single-step transform (3.18) until reaching the coarsest level  $l_0$ . The pseudo-code for the  $d$ -dimensional wavelet transform is given in Algorithm 5. In this algorithm, the wavelet coefficients overwrite in place the given function values. To undo Algorithm 5, simply replace the addition in line 8 of Algorithm 5 with the subtraction and replace line 2 with  $l = l_0$  to  $L - 1$ .

It can be seen that the number of operations required for computing  $d_{l,\mathbf{k}}^{\mathbf{e}}$  is  $p|\mathbf{e}|_1$  (it is less expensive than that of (3.13) which is  $p^{|\mathbf{e}|_1}$ ). Consequently, the number of operations required for the wavelet transform or its inverse on level  $l$  is  $pd2^{dl-1}$  and thus the total number of operations needed is  $O(dp2^L) \sim O(2^L)$ , where  $L$  is the highest level. This indicates that the cost of the  $d$ -dimensional wavelet transform or its inverse is proportional to the total number of data points.

---

**Algorithm 5**  $nDFWT$ 

---

```
Given function values  $\{f_{L,\mathbf{k}}\}_{\mathbf{k} \in \mathbf{k}_L^0}$ 
# for inversion, replace line below with  $l = l_0$  to  $L - 1$ 
for  $l = L - 1$  down to  $l_0$  do
  for  $n = 1$  to  $d$  do
    for all  $(\mathbf{e}, \mathbf{k})$ ,  $\mathbf{k} \in \mathbf{k}_l^e$  and  $\mathbf{e} \in \{0, 1\}^d \setminus \{\mathbf{0}\}$  with  $e_n = 1$  do
       $\mathbf{m} = 2^{L-l-1}(2\mathbf{k} + \mathbf{e})$ ,  $\mathbf{q} = \mathbf{m}$ 
      for  $r = \min \mathcal{X}_{l,k_n}$  to  $\max \mathcal{X}_{l,k_n}$  do
         $q_n = 2^{L-l}r$ 
         $f_{L,\mathbf{m}} = f_{L,\mathbf{m}} - h_{2k_n+1}^{l,r} f_{L,\mathbf{q}}$ 
      end for
    end for
  end for
end for
```

---

Futhermore, the proportionality constant varies linearly with the order of basis  $p$  and the dimension  $d$ .

Note that the values of scaling function coefficients on each level are not affected by the wavelet transform, it is not necessary to perform the wavelet transform in a top-to-bottom order. This feature is unique to this particular wavelet basis. The wavelet transform being done in a top-to-bottom order has an advantage in that one can overwrite in place the wavelet coefficients on the scaling function coefficients (as in Algorithm 5), therefore avoiding the need of extra storage in the transform process. On the other hand, the inverse transform must be carried out strictly in a bottom-to-top order because the scaling functions coefficients on the immediate lower level must be available before scaling function coefficients on the current level can be computed.

### 3.2.2 Approximation property

A Taylor series analysis of (3.13) indicates that for a sufficiently smooth function  $f$  the wavelet coefficient  $d_{l,\mathbf{k}}^e$  satisfies

$$|d_{l,\mathbf{k}}^{\mathbf{e}}| \leq C \max_{\xi} |D^{p\mathbf{e}} f(\xi)| 2^{-p|\mathbf{e}|_1} \quad (3.19)$$

where  $D^{p\mathbf{e}} f \equiv \partial^{p|\mathbf{e}|_1} f / \partial^{pe_1} x_1 \cdots \partial^{pe_d} x_d$ ,  $|\mathbf{e}|_1 = \sum e_i$ ,  $\xi$  is in an appropriate domain determined by the stencil size, and  $C$  is a constant independent of  $f$  and  $l$ . The estimate implies that wavelet coefficients decay exponentially with respect to level  $l$ . In addition, the larger the degree  $p$  is, the faster the decay becomes. Assuming that  $f(\mathbf{x})$  is  $D^{\mathbf{p}}$  differentiable everywhere and making use of the geometric series, it can be shown that the error of the interpolation  $I_j f$  is bounded by

$$\|f(\mathbf{x}) - (I_l f)(\mathbf{x})\|_{\infty} \leq C 2^{-lp} \quad (3.20)$$

where  $C$  is a constant depending on the function and  $p$ . This estimate suggests that  $I_l f$  converge to  $f$  uniformly in the maximum norm and the error of the interpolation  $I_l f$  is  $O(2^{-lp})$ . Notice that while the approximation error is  $O(2^{-lp})$ , the number of grid points increase like  $O(2^{dl})$  as  $d$  increases. This reflects the main disadvantage of using the full grid approximation in that the number of grid points increase substantially, more precisely, at an exponential rate, with respect to the dimension of a problem.

For a function with weaker smoothness, (3.19) is still valid in regions sufficiently away from singularities. However, the wavelet coefficients in the vicinity of singularities decay at a slower rate than given by (3.19) (see [58] for one-dimensional case). In this case,  $I_j f$  still converge uniformly to  $f$  but at slower rate than that of (3.20). It will become evident later that  $I_j f$  can be made accurate to a prescribed accuracy by simply inspecting the magnitude of wavelet coefficients. The fact that the interpolating wavelet transform can be performed



in a bottom-to-top order allows one to perform this task efficiently. One can start with examining the wavelet coefficients on the coarse grid. If there is a wavelet coefficient whose magnitude is larger than a prescribed value, one simply adds points from the higher level(s) of resolution to obtain a finer grid and repeat this process until there is no wavelet coefficient whose magnitude is larger than the prescribed value. In this way, the need of a very fine grid to ensure the accuracy can be avoided.

In the next subsection, we describe the  $d$ -dimensional wavelet transform on a grid with irregular grid points. It is an important ingredient for determining the compressed wavelet representation and interpolation process.

### 3.2.3 Wavelet transform on a $d$ -dimensional grid of irregular points

The wavelet transform described previously can be extended straightforwardly to compute, from given function values on a  $d$ -D grid of irregular points, wavelet coefficients associated with those irregular points. As in the 1-D case, this is the case where the irregular grid satisfies the so-called minimum index set condition. This condition ensures that points participating in the calculation of the wavelet coefficient  $d_{j,\mathbf{k}}^{\mathbf{e}}$  belong to the irregular grid. Figure (3.1) depicts, as an example, points needed in the calculation of the 2-D wavelet coefficients  $d_{l,\mathbf{k}}^{(1,0)}$ ,  $d_{l,\mathbf{k}}^{(0,1)}$ , and  $d_{l,\mathbf{k}}^{(0,1)}$  away from boundaries for  $p = 4$ .

Now, we denote  $\mathcal{V}_l = \{\mathbf{x}_{l,\mathbf{k}} : \mathbf{k} \in \kappa_l^0 \subset \mathbf{k}_l^0\}$  as a grid of irregular points. The grid of irregular points  $\mathcal{V}_l$  can be written as the union of the associated complement grids, *i.e.*

$$\mathcal{V}_l = \mathcal{V}_{l_0}^0 \cup \left( \bigcup_{j=l_0}^{l-1} \left( \bigcup_{\mathbf{e} \in \{0,1\}^d \setminus \{\mathbf{0}\}} \mathcal{W}_j^{\mathbf{e}} \right) \right), \quad (3.21)$$

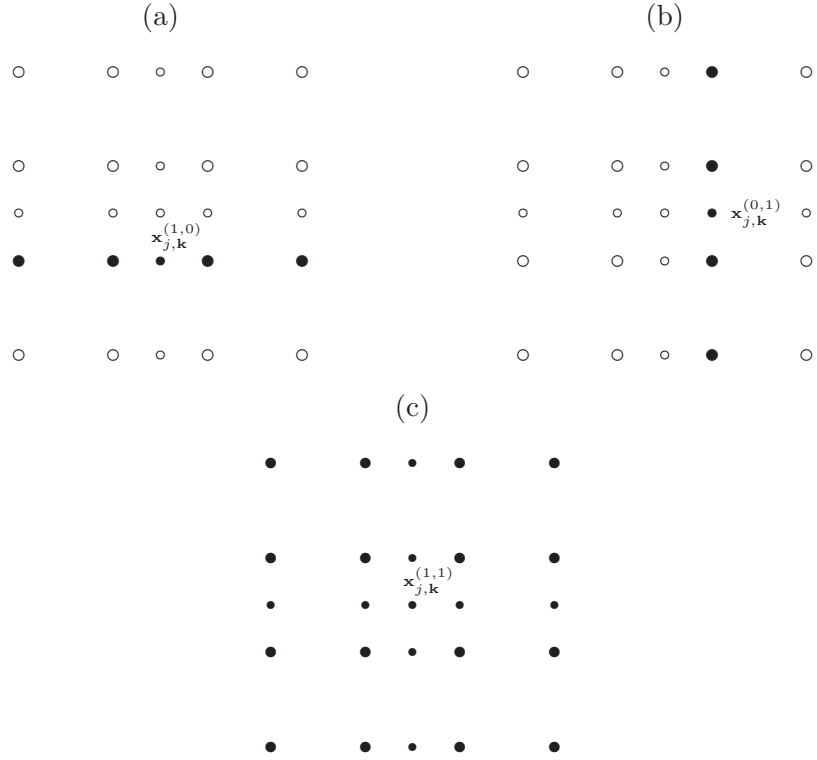


Figure 3.1: Filled circles denotes grid points needed for the calculation of the 2-D wavelet coefficients  $d_{j,\mathbf{k}}^e$  away from boundaries for  $p = 4$ . (a)  $d_{j,\mathbf{k}}^{(0,1)}$  (b)  $d_{j,\mathbf{k}}^{(1,0)}$ , and (c)  $d_{j,\mathbf{k}}^{(1,1)}$ . Note that the larger circles denote grid points in  $\mathbf{V}_j$  and the smaller circles denote grid points in  $\mathbf{W}_j$ .

where  $\mathcal{W}_j^{\mathbf{e}} = \{\mathbf{x}_{j,\mathbf{k}}^{\mathbf{e}}\}_{\mathbf{k} \in \kappa_j^{\mathbf{e}}}$ ,  $\kappa_j^{\mathbf{e}} = \{\mathbf{k} : \mathbf{k} \in \mathbf{k}_j^{\mathbf{e}} \text{ and } \mathbf{x}_{j,\mathbf{k}}^{\mathbf{e}} \in \mathcal{V}_l\}$  and  $\mathcal{V}_j = \{\mathbf{x}_{j,\mathbf{k}}\}_{\mathbf{k} \in \kappa_j^0}$ ,  $\kappa_j^0 = \{\mathbf{k} : \mathbf{k} \in \mathbf{k}_j^0 \text{ and } \mathbf{x}_{j,\mathbf{k}} \in \mathcal{V}_l\}$  for  $j < l$ . As a reminder, for any dyadic point  $\mathbf{x}_{l,\mathbf{k}} \notin \mathbf{V}_{l_0}$ , one can find uniquely an index  $(j, \mathbf{e}, \mathbf{m})$  such that  $\mathbf{x}_{j,\mathbf{m}}^{\mathbf{e}} = \mathbf{x}_{q,\mathbf{k}}$ . That is for a given  $\mathcal{V}_l$ , one can find the associated index set

$$\begin{aligned} \Lambda(l, l_0) = \{&(j, \mathbf{e}, \mathbf{k}) : j \in \{l_0, \dots, l-1\}, \\ &\mathbf{e} \in \{0, 1\}^d \setminus \{\mathbf{0}\}, \mathbf{k} \in \mathbf{k}_j^{\mathbf{e}} \text{ and } \mathbf{x}_{l,\mathbf{k}}^{\mathbf{e}} \in \mathcal{W}_j^{\mathbf{e}}\}. \end{aligned} \quad (3.22)$$

The irregular grid with minimum index set is any grid  $\mathcal{V}_l$  such that, for any grid point in the associated complement grids with index  $(l, \mathbf{e}, \mathbf{k}) \in \Lambda(l, l_0)$ , the following grid points

$$\mathbf{x} = (x_1, x_2, \dots, x_d) \in \mathbf{X}_{j,\mathbf{k}}^{\mathbf{e}} = X_{j,k_1}^{e_1} \times X_{j,k_2}^{e_2} \times \dots \times X_{j,k_d}^{e_d} \setminus \{\mathbf{x}_{j,\mathbf{k}}^{\mathbf{e}}\}, \quad (3.23)$$

where

$$X_{j,k}^0 \equiv \{x_{j+1,2k}\} \text{ and } X_{j,k}^1 \equiv X_{j,k} \cup \{x_{j+1,2k+1}\}, \quad (3.24)$$

belong to the irregular grid  $\mathcal{V}_l$  (or equivalently to either the appropriate complement grid  $\mathcal{W}_r^{\mathbf{e}}$  or the coarsest grid  $\mathbf{V}_{l_0}$ ). It can be observed that number of points in  $\mathbf{X}_{j,\mathbf{k}}^{\mathbf{e}}$  depends on  $\mathbf{e}$ , more precisely  $\dim \mathbf{X}_{j,\mathbf{k}}^{\mathbf{e}} = (p+1)^{|\mathbf{e}|_1} - 1$ . In addition, the coordinate of these points varies only in the  $i$ -direction where  $e_i$  is unity. It is important to note that for any  $\mathcal{V}_l$  with minimum index set,  $\mathcal{V}_{l_0} = \mathbf{V}_{l_0}$ . In other words, the presence of every points in the full coarsest grid is crucial for the wavelet transform on the irregular grid.

Since, for any point in this irregular grid, the points needed for computing the associated wavelet coefficient or for recovering the function value are always avail-

---

**Algorithm 6** *AFWT*


---

Given function values  $\{f_{L,\mathbf{k}}\}_{\mathbf{k} \in \kappa_L^0}$   
**Require:**  $\{\mathbf{x}_{L,\mathbf{k}}\}_{\mathbf{k} \in \kappa_L^0}$  be a grid with minimum index set  
**for**  $l = L - 1$  **down to**  $l_0$  **do**  
     **for**  $n = 1$  **to**  $d$  **do**  
         **for all**  $(l, \mathbf{e}, \mathbf{k}) \in \Lambda(L, l_0)$  such that  $e_n = 1$  **do**  
              $\mathbf{m} = 2^{L-l-1}(2\mathbf{k} + \mathbf{e}), \mathbf{q} = \mathbf{m}$   
             **for**  $r = \min \mathcal{X}_{j,k_n}$  **to**  $\max \mathcal{X}_{j,k_n}$  **do**  
                  $q_n = 2^{L-l}r$   
                  $f_{l,\mathbf{m}} = f_{l,\mathbf{m}} - h_{2^{k_n}+1}^{l,r} f_{L,\mathbf{q}}$   
             **end for**  
         **end for**  
     **end for**  
**end for**

---

able, the wavelet transform or its inverse can be applied straightforwardly. The overall procedure for the wavelet transform on an irregular grid is summarized in Algorithm 6 and the inverse wavelet transform in Algorithm 7. Note that in Algorithm 6, wavelet coefficients overwrite in place on scaling function coefficients. In Algorithm 6, it is assumed that wavelet coefficients  $d_{j,\mathbf{k}}^{\mathbf{e}}$  are stored in entries  $f_{L,2^{L-j-1}(2\mathbf{k}+\mathbf{e})}$  of the input data. The number of operations required in the calculation of the wavelet coefficient  $d_{j,\mathbf{k}}^{\mathbf{e}}$  is  $p|\mathbf{e}|_1$ . It thus follows that the total number of operations involved in the wavelet transform or its inverse on the irregular grid is  $cpN$  where  $N$  is the number of points in the irregular grid, *i.e.*  $N = \dim \mathbf{V}_L$  and  $c$  is a constant (it can be verified that its value is smaller than  $d$ ).

As mentioned earlier, in practical situations, an irregular grid is usually obtained from the thresholding process, *i.e.* discarding wavelets, and hence grid points, whose coefficients are smaller than a prescribed threshold values. In general, the grid obtained from thresholding does not satisfy the minimum index set condition. In order to use the algorithm described above, one simply enlarges the grid so that the augmented grid meets the minimum index set condition (and assumes zero value for the wavelet coefficients of the additional points).

---

**Algorithm 7** *AIWT*


---

Given wavelet coefficient  $\{f_{L,\mathbf{k}}\}_{\mathbf{k} \in \kappa_L^0}$   
**Require:**  $\{\mathbf{x}_{L,\mathbf{k}}\}_{\mathbf{k} \in \kappa_L^e}$  be a grid with minimum index set  
**for**  $l = l_0$  down to  $L$  **do**  
    **for**  $n = 1$  to  $d$  **do**  
        **for all**  $(l, \mathbf{e}, \mathbf{k}) \in \Lambda(L, l_0)$  such that  $e_n = 1$  **do**  
             $\mathbf{m} = 2^{L-l-1}(2\mathbf{k} + \mathbf{e}), \mathbf{q} = \mathbf{m}$   
            **for**  $r = \min \mathcal{X}_{j,k_n}$  to  $\max \mathcal{X}_{j,k_n}$  **do**  
                 $q_n = 2^{L-l}r$   
                 $f_{l,\mathbf{m}} = f_{l,\mathbf{m}} + h_{2^{k_n}+1}^{l,r} f_{L,\mathbf{q}}$   
            **end for**  
        **end for**  
    **end for**  
**end for**

---

Algorithm 8 is used to check whether an irregular grid  $\mathbf{V}_L$  satisfies the minimum index set condition. If it is not the case, the algorithm adds recursively additional grid points to  $\mathbf{V}_L$  such that the resulting grid satisfies it. This algorithm uses a function `dfill` given in Algorithm 9. The  $\mathcal{K}(j, \mathbf{0}, \mathbf{r})$  represents the procedure of the determination of an index  $(i, \mathbf{q}, \mathbf{m})$  such that  $\mathbf{x}_{i,\mathbf{m}}^{\mathbf{q}} = \mathbf{x}_{j,\mathbf{r}}$  for  $\mathbf{x}_{j,\mathbf{r}} \notin \mathbf{V}_{l_0}$ . For  $\mathbf{x}_{j,\mathbf{r}} \in \mathbf{V}_{l_0}$ ,  $\mathcal{K}(j, \mathbf{0}, \mathbf{r})$  yields  $(l_0, \mathbf{0}, \mathbf{m})$  where  $\mathbf{m}$  is an appropriate index such that  $\mathbf{x}_{l_0,\mathbf{m}} = \mathbf{x}_{j,\mathbf{r}}$ . Note that line 16 – 18 of `dfill` can be omitted without affecting the outcome. In principle, the function `dfill` constructs an irregular grid with minimum index set from a grid of one point. Figure 3.2 illustrates the grid resulting from the application of *dAUGMENT* to the grid of one point (in this case it is equivalent to run `dfill`). The function `dfill` has an advantage when it is used to add an additional point to the irregular grid already satisfying the minimum index set. It assures that the resulting grid is the minimum index grid without the need of using Algorithm 8.

---

**Algorithm 8** *dAUGMENT*

---

Establish index sets  $\kappa_{l_0}^0$ ,  $\Lambda(L, l_0)$  from a given  $\mathcal{V}_L$   
**for**  $l = L - 1$  **to**  $l_0$  **do**  
  **for**  $id = d$  **to**  $1$  **do**  
    **for all**  $(l, \mathbf{e}, \mathbf{k}) \in \Lambda(L, l_0)$  such that  $|\mathbf{e}|_1 = id$  **do**  
      call  $\text{dfill}(l, \mathbf{e}, \mathbf{k})$   
    **end for**  
  **end for**  
**end for**

---

---

**Algorithm 9**  $\text{dfill}(l, \mathbf{e}, \mathbf{k})$ ,  $l \geq l_0$ 

---

Given index  $(l, \mathbf{e}, \mathbf{k})$ ,  $l \geq l_0$   
 $j = l + 1$   
**for**  $id = 1$  **to**  $d$  **do**  
  **if**  $e_{id} = 1$  **then**  
     $\mathbf{r} = 2\mathbf{k} + \mathbf{e}$   
    **for**  $n = \min \mathcal{X}_{l, k_{id}}$  **to**  $\max \mathcal{X}_{l, k_{id}}$  **do**  
       $r_{id} = 2n$ ,  $(i, \mathbf{q}, \mathbf{m}) = \mathcal{K}(j, \mathbf{0}, \mathbf{r})$   
      **if**  $\mathbf{x}_{j, \mathbf{r}} \notin \mathcal{V}_L$  **then**  
        Enlarge grid  $\mathcal{V}_L = \mathcal{V}_L \cup \{\mathbf{x}_{j, \mathbf{r}}\}$   
        **if**  $\mathbf{x}_{j, \mathbf{r}} \notin \mathbf{V}_{l_0}$  **then**  
          enlarge index  $\Lambda(L, l_0) = \Lambda(L, l_0) \cup \{(i, \mathbf{q}, \mathbf{m})\}$   
        **else**  
          enlarge index  $\kappa_{l_0}^0 = \kappa_{l_0}^0 \cup \{(i, \mathbf{m})\}$   
        **end if**  
      **end if**  
      # if block below can be commented out when used with dAUGMENT  
      **if**  $\mathbf{x}_{j, \mathbf{r}} \notin \mathbf{V}_{l_0}$  **then**  
        call  $\text{dfill}(i, \mathbf{q}, \mathbf{m})$   
      **end if**  
    **end for**  
  **end if**  
**end for**

---

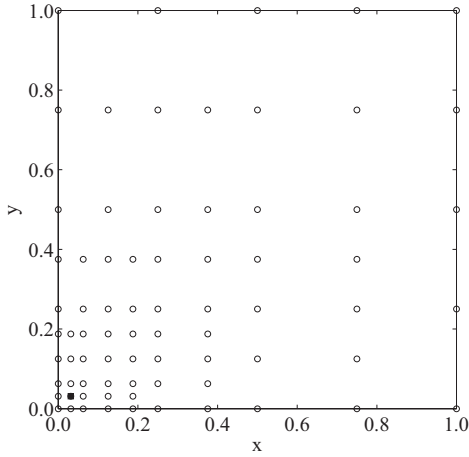


Figure 3.2: The irregular grid ( $p = 4, l_0 = 4$ ), satisfying the minimum index set condition, obtained from the augmentation of the grid of the single point  $x_{4,(0,0)}^{(1,1)}$ : ■ represents  $x_{4,(0,0)}^{(1,1)}$  and o denote augmented points.

### 3.3 Sparse Wavelet Representation(SWR) and irregular sparse grid

It is evident from the way that wavelet coefficients are defined that they indicate the deviation of values predicted by local polynomial approximations from actual function values at the associated points. The magnitude of the coefficients decay fast where the function is locally smooth and decay at a slower rate where the function is locally less smooth. For a function with isolated (near) singularities, only a small number of wavelet functions have large amplitude and such wavelet functions are located in the vicinity of the (near) singularities. This aspect is illustrated in Figures 3.3, 3.4 and 3.5. Figure 3.3 depicts the wavelet approximations of a 1-D function having a discontinuity in their first derivative and Figure 3.4 and 3.5 depict 2-D and 3-D functions having singularities in their first derivative on circular and spherical surface, respectively. Notice that points associated with wavelets coefficients with absolute values larger than the threshold

value concentrate mainly in the vicinity of the singularities. Furthermore, they form a cone-like structure pointing toward the location of the singularities.

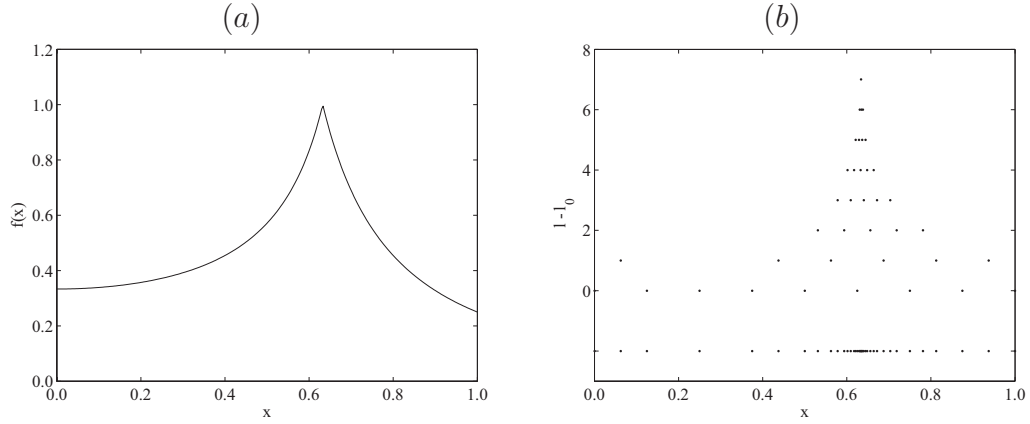


Figure 3.3: (a) Test function  $f(x) = 0.2/(|0.4 - x^2| + 0.2)$  and (b) grid points corresponding to  $|d_{j,\mathbf{k}}^e| \geq 5 \times 10^{-3}$  and the distribution of such grid points at each level for  $p = 6$ ,  $l_0 = 3$ .

The multiscale representation (3.11) can be decomposed as sums of wavelets whose amplitudes are above and below a user-selected threshold value  $\varepsilon$ :

$$f_J(\mathbf{x}) = f_J^\varepsilon(\mathbf{x}) + R_J^\varepsilon(\mathbf{x}), \quad (3.25)$$

where

$$f_J^\varepsilon(\mathbf{x}) = \sum_{\mathbf{k}} f_{l_0,\mathbf{k}} \Phi_{l_0,\mathbf{k}}(\mathbf{x}) + \sum_{(l,\lambda) \in \Lambda^\varepsilon(J,l_0)} d_{l,\lambda} \Psi_{l,\lambda}(\mathbf{x}) \quad (3.26)$$

and

$$R_J^\varepsilon(\mathbf{x}) = \sum_{j=l_0}^{J-1} \sum_{\{\lambda \mid |d_{j,\lambda}| \leq \varepsilon\}} d_{j,\lambda} \Psi_{j,\lambda}(\mathbf{x}). \quad (3.27)$$



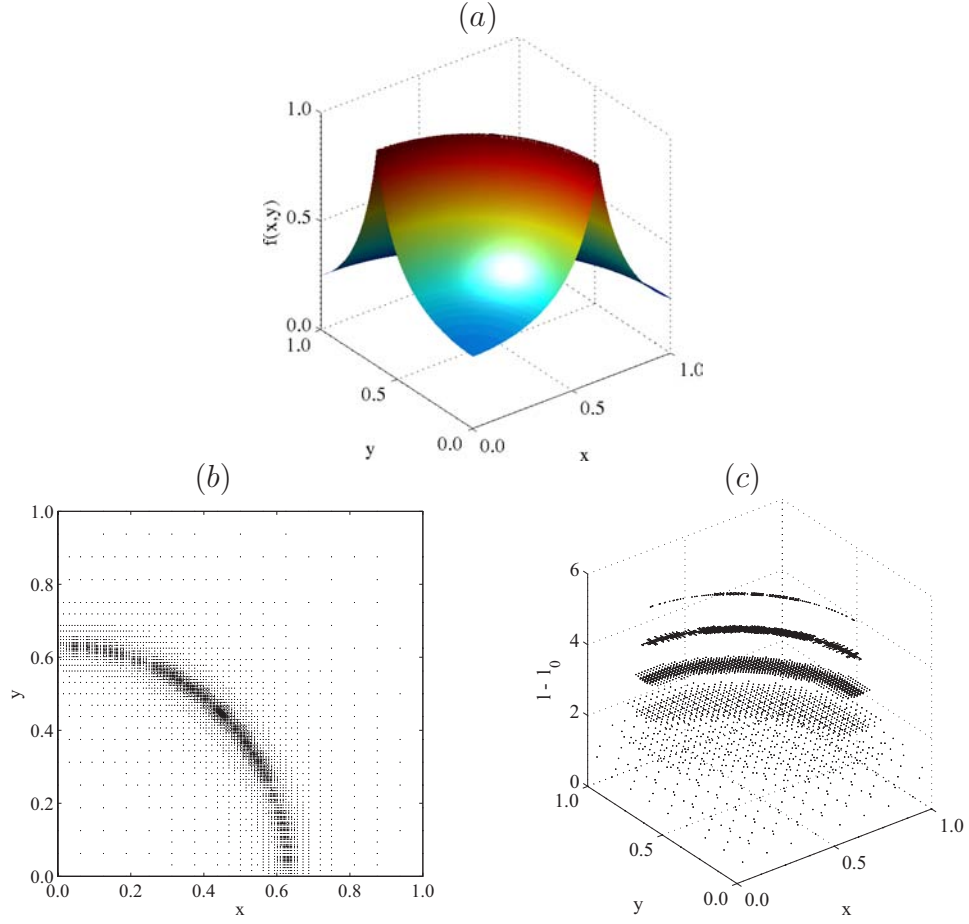


Figure 3.4: (a) Test function  $f(x, y) = 0.2/(|0.4 - x^2 - y^2| + 0.2)$ , (b) grid points corresponding to  $|d_{j,\mathbf{k}}^e| \geq 5 \times 10^{-3}$ , and (c) grid points corresponding to  $|d_{j,\mathbf{k}}^e| \geq 5 \times 10^{-3}$  at each level for  $p = 6$ ,  $l_0 = 3$

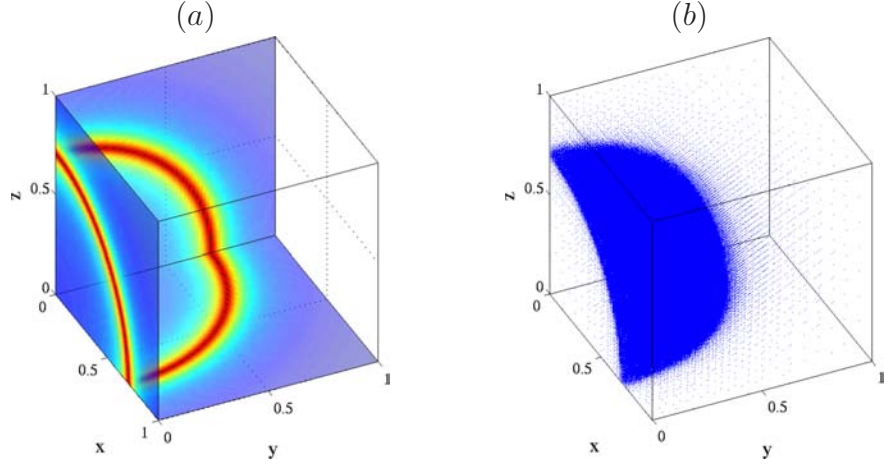


Figure 3.5: (a) Sections of test function  $f(x, y, z) = 1/(|0.5 - x^2 - y^2 - z^2| + 0.1)$ , (b) grid points corresponding to  $|d_{j,\mathbf{k}}^{\mathbf{e}}| \geq 5 \times 10^{-3}$  for  $p = 6$ ,  $l_0 = 3$ .

Here,

$$\begin{aligned} \Lambda^\varepsilon(J, J_0) &= \{(l, \lambda) = (l, \mathbf{e}, \mathbf{k}) : l \in \{J_0, \dots, J-1\}, \\ &\quad \mathbf{e} \in \{0, 1\}^d \setminus \{\mathbf{0}\}, \mathbf{k} \in \mathbf{k}_l^{\mathbf{e}} \text{ and } |d_{j,\lambda}| \geq \varepsilon\}. \end{aligned} \quad (3.28)$$

and, to simplify the notation, we use the multi-index  $\lambda = (\mathbf{e}, \mathbf{k})$ ,  $\Psi_{j,\lambda} = \Psi_{j,\mathbf{k}}^{\mathbf{e}}$  and  $\mathbf{x}_{j,\lambda} = \mathbf{x}_j^{\mathbf{e}}$ . In addition, the value of the threshold parameter  $\varepsilon$  is assumed to be small, provided that the function  $f$  is appropriately normalized. Note that, subsequently, we sometime use  $\Lambda^\varepsilon$  to refer generically to an index set  $\Lambda^\varepsilon(\cdot, J_0)$ . It is worth noting that the second term on the right hand side of (3.26) is equivalent to

$$\sum_{j=l_0}^{J-1} \sum_{\{\lambda \mid |d_{j,\lambda}| \geq \varepsilon\}} d_{j,\lambda} \Psi_{j,\lambda}(\mathbf{x}) \quad (3.29)$$

The approximation  $f_J^\varepsilon$  is the so-called *sparse wavelet representation* (SWR)[76] or the *compressed representation* of  $f$ . In fact, neglecting the term  $R_J^\varepsilon$  results in an error which in general is different from the threshold value  $\varepsilon$  but it is closely related to it, provided that  $J$  is sufficiently large so that the interpolation function  $f_J$  is accurate within a prescribed accuracy value of less than  $\varepsilon$ . This point is made more precise in the estimate below.

It can be observed from (3.19) that indeed there is a  $J$  such that for  $j \geq J$ ,  $|d_{j,\mathbf{k}}^\varepsilon| \leq \varepsilon$ , more precisely  $J = \lfloor ((1/p) \log_2(C\|D^{\mathbf{q}}f\|_{p,\infty}/\varepsilon)) \rfloor$ , where  $\|D^{\mathbf{q}}f\|_{p,\infty} = \max_{|\mathbf{q}|_1 \leq p} \|D^{\mathbf{q}}f\|_\infty$  (here, we assume that the function is sufficiently smooth). Let  $d_{j,\lambda}^\varepsilon = 0$  for index  $(j, \lambda) \in \Lambda^\varepsilon(J, l_0)$  (*i.e.*, index such that  $|d_{j,\lambda}| \geq \varepsilon$ ) and  $d_{j,\lambda}^\varepsilon = d_{j,\lambda}$  otherwise. It can be seen that

$$\begin{aligned}
\|f - f_J^\varepsilon\|_\infty &\leq \left\| \sum_{j=l_0}^{J-1} \sum_{\lambda} d_{j,\lambda}^\varepsilon \Psi_{j,\lambda} \right\|_\infty + \left\| \sum_{j \geq J} \sum_{\lambda} d_{j,\lambda} \Psi_{j,\lambda} \right\|_\infty \\
&\leq (J - l_0) C_1 C_2 \varepsilon + C_1 C_2 \sum_{j \geq J} \underbrace{\max_{\lambda} \|d_{j,\lambda}\|}_{\leq C\|D^{\mathbf{q}}f\|_{p,\infty} 2^{-jp}} \\
&\leq C_1 C_2 (J - l_0 + C_3 C_4) \varepsilon
\end{aligned} \tag{3.30}$$

where  $C_1 \sim O(p^d)$  is the maximum number of wavelets at any level intersecting any single point in  $[0, 1]^d$ ,  $C_2 \sim O(1)$  is the maximum norm of wavelets,  $C_3$  is the constant depending on the function, and  $C_4$  is the constant arising from the geometric series (the larger  $p$  is, the closer  $C_4$  is to unity). In other word, the error of  $f_J^\varepsilon$  is proportional to the threshold value  $\varepsilon$ ,

$$\|f - f_J^\varepsilon\|_\infty \leq C_5 \varepsilon \tag{3.31}$$

where  $C_5$  is a constant depending on the function  $f$ ,  $d$ ,  $p$ , and mildly on the threshold value  $\varepsilon$ . Furthermore, the number of basis functions in the SWR, denoted by  $N$ , satisfies (see [58, 76])

$$N^{1/d} \leq C_6 \varepsilon^{-1/p}, \quad (3.32)$$

where  $d$  is a spatial dimension of the function and the constant  $C_6$  depends on the function  $f$ . Equations (3.31) and (3.32) imply the following bound:

$$\|f - f_J^\varepsilon\|_\infty \leq CN^{-p/d}. \quad (3.33)$$

Subsequently, these bounds imply that  $f_J^\varepsilon$  approximates  $f$  within  $\varepsilon$  using approximately  $\varepsilon^{-d/p}$  wavelets (a more accurate approximation of  $f_J^\varepsilon$  can be obtained by decreasing the value of  $\varepsilon$  with a commensurate increase in the number of wavelets required).

Notice that instead of requiring  $2^{dJ}$  degrees of freedom (DOFs) to approximate  $f$  within  $\varepsilon$ , only  $O(\varepsilon^{-d/p})$  DOFs are needed in the SWR case. This indicates that wavelets furnish one with a systematic way to reduce the amount of data necessary in the representation of a function. To exploit the compression property of wavelets, one discards wavelets whose amplitudes are smaller than  $\varepsilon$ . Subsequently, one is required to work instead with the subset of wavelets retained. As mentioned in the previous section, tasks such as wavelet transform, its inverse, and interpolation can be performed in an efficient manner on the subset of wavelets and the associated grid of irregular points. Note that an irregular grid obtained from the thresholding process in general does not satisfy the minimum index set condition. As a consequence, algorithms given in the previous sections are not immediately applicable. To alleviate this issue, one can simply keep additional

points necessary for the augmented grid to meet the minimum index set condition. Wavelet coefficients associated with the additional points can take either their original values or zero values. In the latter case, the function values obtained from the inverse wavelet transform at few points are not identical to the values. The discrepancy between these values is naturally only of order  $\varepsilon$ .

To numerically verify the estimates given above, we consider the SWRs of the following 2-D test functions:

$$f(\mathbf{x}) = \frac{0.05(x_1 + 0.05)}{(x_1 + 0.05)^2 + (x_2 - 0.4)^2} \quad (3.34)$$

and

$$f(\mathbf{x}) = \exp \left[ -200 \left( (x_1 - \frac{1}{2})^2 + (x_2 - \frac{1}{2})^2 \right) \right] + \frac{1}{5} \sin(2\pi x_1) \sin(2\pi x_2) \quad (3.35)$$

The former test function changes sharply near the middle-left of the boundary and the latter test function is superposition a Gaussian bump and a sine function. Note that both functions are  $C^\infty$ . Figure 3.6 shows the error in  $L_\infty$ -norm of SWR  $f^\varepsilon$  with different  $p$  as a results of varying the value of the threshold parameter  $\varepsilon$ . The error of SWR associated with the function (3.34) is plotted in Figure 3.6(a) and with the function (3.35) in Figure 3.6(b). The plots in the right column of this figure which are log-log plots of the error  $\|f(\mathbf{x}) - f^\varepsilon(\mathbf{x})\|_\infty$  versus the threshold parameter  $\varepsilon$  used, clearly indicate that error of  $f^\varepsilon$  is of the same of with  $\varepsilon$ . The slope of the log-log plot of error versus the number of basis functions  $N$  required for different choices of  $p$  (see left column of Figure 3.6) is approximately  $-p/2$  (without providing such precise numbers, we note that this can be seen easily by

comparing the plots with the plots of  $O(N^{-p/2})$  indicated by lines without any maker). The results are in good agreement with the estimate (3.33).

To examine an SWR of the function with less smoothness, we consider the test function with discontinuities in derivative:

$$f(\mathbf{x}) = g(x_1)g(x_2), \quad \text{where } g(s) = \begin{cases} \sin(s) & \text{for } s < 1/2 \\ as^2 + b & \text{for } s \geq 1/2 \end{cases} \quad (3.36)$$

where  $a$  and  $b$  are chosen such that  $g$  is  $C^1$ . Figure 3.7 shows the error of  $f^\varepsilon$  as function of the threshold value  $\varepsilon$  and of the number of basis functions  $N$  required. The plots on the left column of the figure indicates that  $f^\varepsilon$  is accurate to the threshold value used. Since now the test function is not sufficiently smooth, the rate of convergence of the error with respect to  $N$  is therefore expected to be lower than  $O(N^{-p/d})$  for approximation with large  $p$  is used. The plots on the right column of Figure 3.7 support such an expectation. Note that in this case, the rate of convergence of the error with respect to  $N^{-1/d}$  is approximately of order 2 for  $p = 2$  and is approximately of order 4 for  $p \geq 4$ .

It is noted that, in the numerical results shown above, we use an Algorithm described in the next section to determine wavelet functions whose coefficients are larger than the threshold values. Instead of thresholding of the wavelet coefficients on the full grid that is very fine, such an algorithm performs thresholding starting from the coarsest level and advancing progressively up to the finest level necessary. The maximum error is computed on a very fine grid where the function values of  $f^\varepsilon$  at points that do not belong to the associated irregular grid are obtained by mean of the wavelet interpolation (see 3.4 for detail).

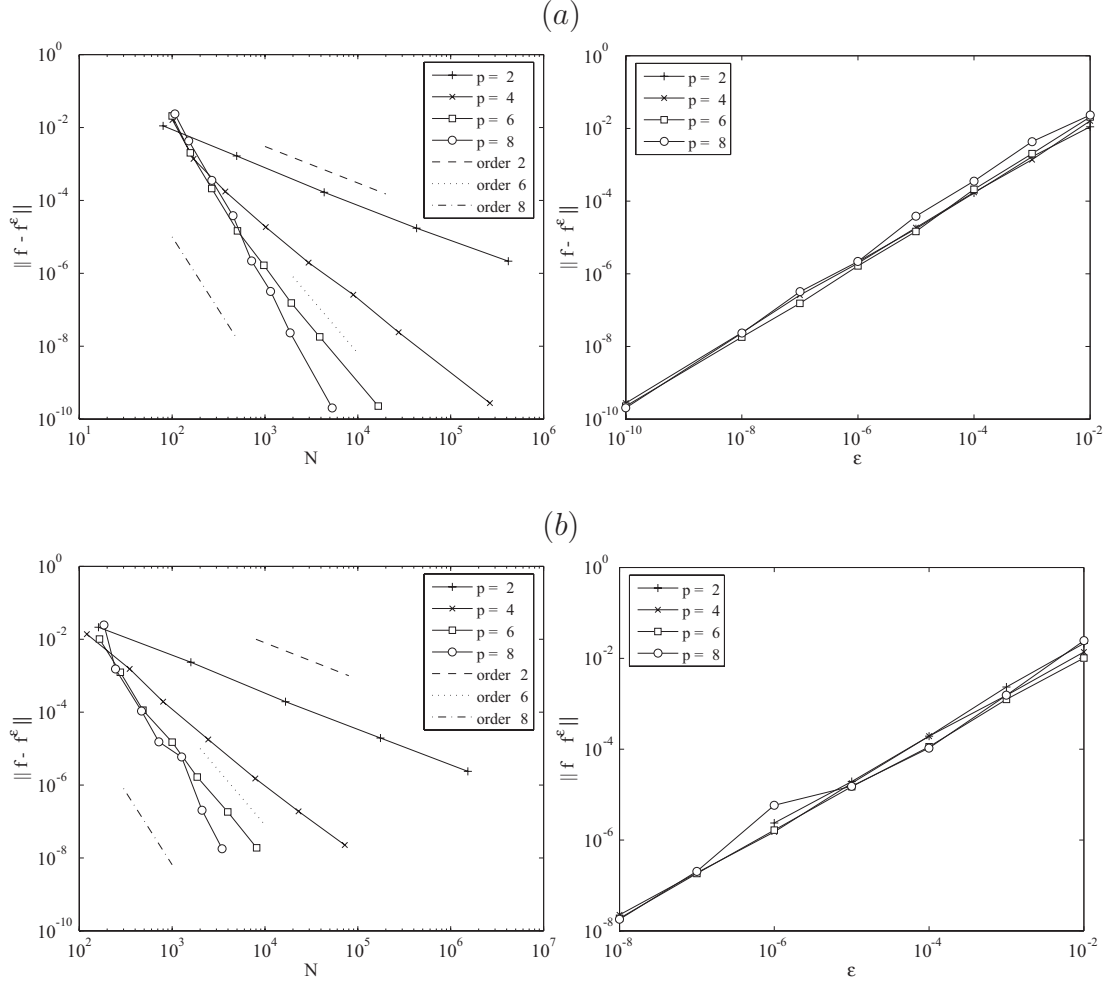


Figure 3.6: Relationship between the approximation error  $\|f(\mathbf{x}) - f^\varepsilon(\mathbf{x})\|_\infty$  and  $N = \dim \mathcal{V}^\varepsilon$  as a results of varying  $\varepsilon$  (left) and  $\|f(\mathbf{x}) - f^\varepsilon(\mathbf{x})\|_\infty$  as a function of  $\varepsilon$  (right) for sparse wavelet approximation with different values of  $p$ . (a) test function (3.34); (b) test function (3.35).

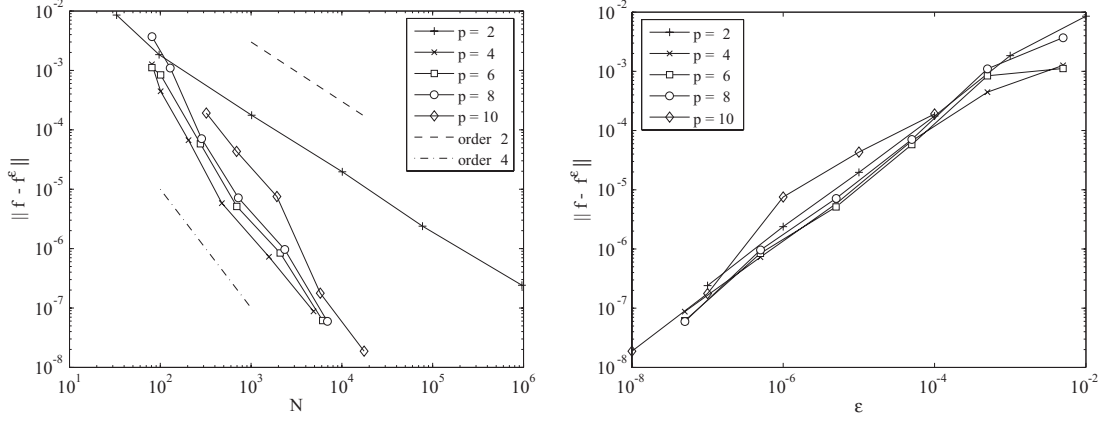


Figure 3.7: Relationship of the error  $\|f(\mathbf{x}) - f^\epsilon(\mathbf{x})\|_\infty$  of the test function (3.36) and  $N = \dim \mathcal{V}^\epsilon$  as a result of varying  $\epsilon$  (left) and the error as a function of  $\epsilon$  (right) for sparse wavelet approximation with different values of  $p$ .

### 3.3.1 Determining the SWR of a function

From a practical standpoint, it may not always be desirable to first compute the wavelet coefficients on the full grid that is very fine and then omit entries by means of thresholding. Instead, it is preferable to proceed in a bottom-to-top order with on-the-fly thresholding starting from the coarsest level and advancing progressively up to the finest level necessary. This procedure is demonstrated in Algorithm 10. It constructs gradually the compressed wavelet representation (and the associated irregular grid). It can be described roughly as follow: (i) compute the wavelet coefficients on the full grid of level  $l_0 + 1$  and perform thresholding to obtain the irregular grid (ii) for each point in the resulting grid, collect a few surrounding points on the same and next level(s) and include them to the grid. (iii) perform wavelet transform on the resulting grid and check if there is any coefficient in the newly added level having absolute value larger than  $\epsilon$ . If this is the case, perform thresholding and go back step (ii) otherwise perform thresholding



---

**Algorithm 10** *COMPRESS*

---

Given continuous function  $f$   
 $AFWT\{f(\mathbf{x}_{j,\mathbf{k}})\}_{\mathbf{k} \in \mathbf{k}_{l_0+1}^0} \rightarrow \{\{f_{l_0,\mathbf{k}}\}, \{d_{l_0+1,\lambda}\}\}$   
 $i = 1$   
 $\Lambda_i = \{(l_0 + 1, \lambda) : |d_{l_0+1,\lambda}| \geq \varepsilon\}, \Lambda^0 = \Lambda_1$   
**while**  $\Lambda_i \neq \emptyset$  **do**  
    Determine the neighboring region  $\Lambda^n = \cup_{(l,\lambda) \in \Lambda^0} \mathcal{N}_{l,\lambda}$   
    Set  $\Lambda^1 = \Lambda^0 \cup \Lambda^n$   
     $AFWT\{\{f(\mathbf{x}_{l_0,\mathbf{k}})\}_{\mathbf{k} \in \mathbf{k}_{l_0}^0}, \{f(\mathbf{x}_{l,\lambda})\}_{(l,\lambda) \in \Lambda^1}\} \rightarrow \{\{f_{l_0,\mathbf{k}}\}_{\mathbf{k} \in \mathbf{k}_{l_0}^0}, \{d_{l,\lambda}\}_{(l,\lambda) \in \Lambda^1}\}$   
     $i = i + 1,$   
     $\Lambda_i = \{(l_0 + i, \lambda) : (l_0 + i, \lambda) \in \Lambda^1 \text{ and } |d_{l_0+i,\lambda}| \geq \varepsilon\}$   
    Threshold  $\Lambda^0 = \{(j, \lambda) : (j, \lambda) \in \Lambda^1 \text{ and } |d_{j,\lambda}| \geq \varepsilon\}$   
**end while**  
**Results:**  
     $\Lambda^\varepsilon = \Lambda^0$   
     $\mathcal{D}^\varepsilon = \{\{f_{l_0,\mathbf{k}}\}_{\mathbf{k} \in \mathbf{k}_{l_0}^0}, \{d_{j,\lambda}\}_{(l,\lambda) \in \Lambda^\varepsilon}\}$

*Note:*  $\mathcal{N}_{l,\lambda}$  is the index set of few neighboring points residing on the same and next level(s) of resolution.

---

and terminate the process. Note that in step (ii) the few points included to the irregular grid are those in which their associated wavelet coefficients are expected to be possibly larger than  $\varepsilon$ . Owing to the fact that wavelet coefficients decay as cone-like structure with respect to level, one can expect the value of coefficients associated with the points away from such cone to be than less the threshold value. Thus it is adequate to include only few points on the same and on the higher level(s) around each point considered. In addition, not being stated in the algorithm is that irregular grids in this process are enlarged such that they satisfy the minimum index set. More precisely, after thresholding, we add additional points necessary to make the irregular grid becomes that of the minimum index set.

For many cases, Algorithm 10 yields an irregular grid that is identical with that of the traditional thresholding procedure (by first computing coefficients on the

finest grid and then thresholding). It is important to note that it is possible that this procedure terminates prematurely and fail to resolve all function features. An example that leads to this circumstance is when the function  $f(\mathbf{x})$  has a small spike on the very fine level. Note that the thresholding used in (3.26) is based on the magnitude of wavelet coefficients. However, in some cases, this choice can be too sensitive and may result in a non-terminating algorithm. This situation occurs especially in the case where functions contain discontinuities. Alternatively, one can threshold based on the following criteria [73],

$$\|d_{j,\lambda}\Psi_{j,\lambda}(\mathbf{x})\| = |d_{j,\lambda}|\|\Psi_{j,\lambda}(\mathbf{x})\| \geq \varepsilon \quad (3.37)$$

with a given norm  $\|\cdot\|$ . Note that in fact the thresholding based the absolute value of the coefficients is effectively similar to the use of max-norm. For  $L_1$  and  $L_2$  norms, we have  $\|\Psi_{j,\lambda}(\mathbf{x})\| = c_1 2^{-j|\mathbf{e}|_1}$  and  $c_2 2^{-j|\mathbf{e}|_1/2}$ , where  $c_1$  and  $c_2$  are appropriate constants depending on  $d$  and the location of wavelets. It follows that additional damping is added when these norms are used, hence ensuring that the algorithm is a terminating one.

### 3.4 Interpolation on irregular grid

In some circumstance, one may wish to find the interpolated values of the SWR on an irregular grid  $\{\{\mathbf{x}_{l_0,\mathbf{k}}\}_{\mathbf{k} \in \mathbf{k}_{l_0}^0}, \{\mathbf{x}_{l,\lambda}\}_{(l,\lambda) \in \Lambda}\}$  differing from the irregular grid associated with that of the SWR (*i.e.*,  $\Lambda \neq \Lambda^\varepsilon$ ). The simple way to accomplish this task is to include points to be interpolated and set respective coefficients of points in  $\Lambda \setminus \Lambda^\varepsilon$  to zero and then use the inverse wavelet transform (*AIWT*) to obtain the interpolated values on  $\Lambda \cup \Lambda^\varepsilon$ . More precisely, interpolated values can be obtained

from the inverse wavelet transform of the coefficients  $\{\{f_{l_0, \mathbf{k}}\}_{\mathbf{k} \in \mathbf{k}_{l_0}^0}, \{d_{l, \lambda}\}_{(l, \lambda) \in \Lambda \cup \Lambda^\varepsilon}\}$  where  $d_{l, \lambda} = 0$  for  $(l, \lambda) \in \Lambda \setminus \Lambda^\varepsilon$ . The number of operations required is of order  $N = \dim\{\{f_{l_0, \mathbf{k}}\}_{\mathbf{k} \in \mathbf{k}_{l_0}^0}, \{d_{l, \lambda}\}_{(l, \lambda) \in \Lambda \cup \Lambda^\varepsilon}\}$ . Alternatively, interpolated values at points  $\mathbf{x}_{l, \lambda}$  with  $(l, \lambda) \in \Lambda \setminus \Lambda^\varepsilon$  can be computed directly from the function values  $\{\{f_{l_0, \mathbf{k}}\}_{\mathbf{k} \in \mathbf{k}_{l_0}^0}, \{f_{l, \lambda}\}_{(l, \lambda) \in \Lambda^\varepsilon}\}$  (without having to perform the complete inverse wavelet transform). It consists of the application of the inversion formula (3.16) directly at these interpolated points, *i.e.*

$$f_{l, (\mathbf{e}, \mathbf{k})} = - \left( \prod_{i=1}^d I_{l, k_i}^{e_i} \right) \tilde{f} \equiv \mathbf{I}_{l, \mathbf{k}}^{\mathbf{e}} \tilde{f}, \quad (l, \lambda) \in \Lambda \setminus \Lambda^\varepsilon, \quad (3.38)$$

where  $\tilde{f} = 0$  at the point  $\mathbf{x}_{l, \lambda}$  and  $\tilde{f}$  correspond to the function value at the other stencil points. Notice that, in order to compute the interpolated value of a given point, the values of points at the lower level are required. It is possible that one (or few) of those points may be in  $\Lambda \setminus \Lambda^\varepsilon$ . In this case, the interpolated value(s) of such point(s) must be available before (3.38) can be applied. Therefore, to be certain that all data necessary is available, it is important to apply the direct formula from the coarsest level to the finest level. Furthermore, to compute interpolated values of a point  $\mathbf{x}_{l, \mathbf{k}}^{\mathbf{e}}$  with a certain value  $|\mathbf{e}|_1 = c$ , the values of points on the same level with  $|\mathbf{e}|_1 < c$  are needed. Thus, at each level, the calculation of interpolated values must be performed sequentially for points with  $|\mathbf{e}|_1 = 1, 2, \dots, d$ . Algorithm 11 outlines this alternative means of computing interpolated values at the points  $\Lambda \setminus \Lambda^\varepsilon$  from given function values  $\{\{f_{l_0, \mathbf{k}}\}_{\mathbf{k} \in \mathbf{k}_{l_0}^0}, \{f_{l, \lambda}\}_{(l, \lambda) \in \Lambda^\varepsilon}\}$ .

Note that the computation of one interpolated value  $f_{l, (\mathbf{e}, \mathbf{k})}$  using the direct inversion formula requires  $p^{|\mathbf{e}|_1}$  operations while the inverse wavelet transform needs  $p|\mathbf{e}|_1$  operations. When considering the cost of computing a single interpolated value, the direct inversion formula is more expensive than the inverse wavelet

---

**Algorithm 11** *LAIWT*

---

Given function values  $\{\{f_{l_0, \mathbf{k}}\}_{\mathbf{k} \in \mathbf{k}_{l_0}^0}, \{f_{l, \lambda}\}_{(l, \lambda) \in \Lambda^\varepsilon}\}$  and  $\Lambda$   
**for**  $l = l_0$  to  $J - 1$  **do**  
  **for**  $id = 1$  to  $d$  **do**  
    Compute  $f_{l, \lambda}$ , for  $(l, \mathbf{e}, \mathbf{k}) \in \Lambda \setminus \Lambda^\varepsilon$  and  $|\mathbf{e}|_1 = id$  by means of (3.38)  
  **end for**  
**end for**

---

transform. However, the direct inversion formula allow one to compute only the interpolated values at points of interest (*i.e.*, points in  $\Lambda \setminus \Lambda^\varepsilon$ ), while the inverse wavelet transform interpolates every point in the grid  $\Lambda \cup \Lambda^\varepsilon$ . The direct inversion formula has an advantage when  $\dim(\Lambda \setminus \Lambda^\varepsilon)$  is small compared to  $\Lambda \cup \Lambda^\varepsilon$  and one is given the function values  $\{\{f_{l_0, \mathbf{k}}\}_{\mathbf{k} \in \mathbf{k}_{l_0}^0}, \{f_{l, \lambda}\}_{(l, \lambda) \in \Lambda^\varepsilon}\}$  instead of the wavelet coefficients  $\{\{f_{l_0, \mathbf{k}}\}_{\mathbf{k} \in \mathbf{k}_{l_0}^0}, \{d_{l, \lambda}\}_{(l, \lambda) \in \Lambda^\varepsilon}\}$ .

### 3.5 Algebra of the SWR

#### 3.5.1 Addition and subtraction

The addition of functions in the multiscale representation can be obtained easily by simply adding or subtracting their coefficients. This can be extended immediately to the addition of sparse representations. In this case, the summation involves an index set that is the union of index sets induced by the functions (and possibly different threshold values). More precisely, supposed we are given the SWR with threshold value  $\varepsilon$  of  $f$  and  $g$ . Addition of these functions is given by

$$f^\varepsilon \pm g^\varepsilon = \sum_{\mathbf{k} \in \mathbf{k}_{l_0}^0} (f_{l_0, \mathbf{k}} \pm g_{l_0, \mathbf{k}}) \Phi_{l_0, \mathbf{k}}(\mathbf{x}) + \sum_{(l, \lambda) \in \Lambda^{\varepsilon, f} \cup \Lambda^{\varepsilon, g}} (d_{l, \lambda}^f \pm d_{l, \lambda}^g) \Psi_{l, \lambda}(\mathbf{x}) \quad (3.39)$$

where  $d_{j, \lambda}^f$  are the wavelet coefficients of the function  $f$ ,  $\Lambda^{\varepsilon, f} = \{(j, \lambda) \mid j \geq l_0, |d_{j, \lambda}^f| \geq \varepsilon\}$ , and the coefficients  $d_{j, \lambda}^f$  with  $(j, \lambda) \notin \Lambda^{\varepsilon, f}$  are assumed to be zero.

It can be estimated that

$$\begin{aligned}
\|(f(\mathbf{x}) \pm g(\mathbf{x})) - (f^\varepsilon(\mathbf{x}) \pm g^\varepsilon(\mathbf{x}))\|_\infty &\leq \|f(\mathbf{x}) - f^\varepsilon(\mathbf{x})\|_\infty + \|g(\mathbf{x}) - g^\varepsilon(\mathbf{x})\|_\infty \\
&\leq c_1\varepsilon + c_2\varepsilon \\
&\leq 2c_3\varepsilon.
\end{aligned}
\tag{3.40}$$

This means that the addition error of  $f^\varepsilon(\mathbf{x}) \pm g^\varepsilon(\mathbf{x})$  is still of order  $\varepsilon$ . Figure 3.8 shows for example the accuracy of the addition of SWRs of the test function (3.34) and of the test function (3.35) with different  $p$ . The plots on right column of this figure demonstrates clearly that error in  $L_\infty$  norm of the addition of two SWRs is of the same order with the threshold values  $\varepsilon$ . The rate of convergence for the addition with respect to  $N^{-1/d}$ , where  $N$  is the dimension of the union of two index sets (or equivalently of two irregular sparse grids) associated with these particular SWRs, is approximately of order  $p$ .

### 3.5.2 Multiplication

The direct multiplication of two functions in the multiscale representation results in the summation of pairs of basis functions (instead of the summation of basis function like (3.11)). Evaluating the summation of this type involves substantially more operations. Alternatively, one can compute the pointwise multiplication in physical space and use the wavelet transform to compute the corresponding result in the wavelet representation. More precisely, for a given  $f^\varepsilon$  and  $g^\varepsilon$ , the SWR of  $f$  and  $g$ , the multiplication procedure consists of the following steps: (i) compute by means of the inverse wavelet transform their associated function values on the extended grid (the grid of points  $\{\mathbf{x}_{l_0, \mathbf{k}} : \mathbf{k} \in \mathbf{k}_{l_0}^0\} \cup \{\mathbf{x}_{l, \lambda} : \Lambda^{\varepsilon, f} \cup \Lambda^{\varepsilon, g}\}$ ), (ii) evaluate the pointwise product on the

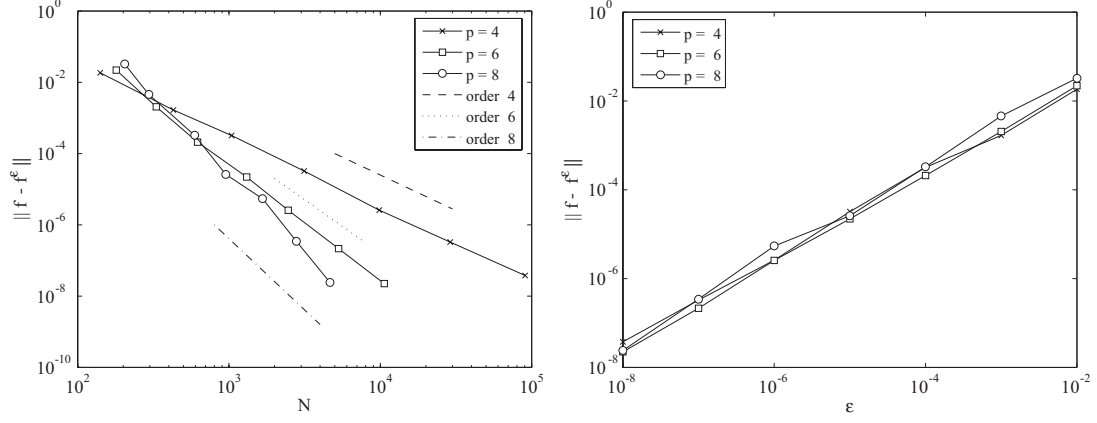


Figure 3.8: Convergence in  $L_\infty$ -norm of an addition of the SWR of the test function (3.34) and the SWR of the test function (3.35) for various  $p$  and  $\varepsilon$ . Left: error vs numbers of points in extended grids. Right: error vs. threshold values  $\varepsilon$ .

extended grid (*i.e.*,  $\{f(\mathbf{x})g(\mathbf{x}) : \mathbf{x} \in \{x_{l_0, \mathbf{k}} : \mathbf{k} \in \mathbf{k}_{l_0}^0\} \cup \{x_{l, k} : \Lambda^{\varepsilon, f} \cup \Lambda^{\varepsilon, g}\}\}$ ), and (iii) perform the wavelet transform on the obtained values to get the respective wavelet coefficients. Since the cost of the wavelet transform and its inverse is proportional to the number of grid points, the number of operations required for this procedure is proportional to the dimension of  $\Lambda^{\varepsilon, f} \cup \Lambda^{\varepsilon, g}$ . The error resulting from this procedure is bounded by

$$\begin{aligned}
\|f(\mathbf{x})g(\mathbf{x}) - f^\varepsilon(\mathbf{x})g^\varepsilon(\mathbf{x})\|_\infty &= \|f(\mathbf{x})g(\mathbf{x}) - f^\varepsilon(\mathbf{x})g^\varepsilon(\mathbf{x}) + f(\mathbf{x})g^\varepsilon(\mathbf{x}) - f(\mathbf{x})g^\varepsilon(\mathbf{x})\|_\infty, \\
&\leq \|f(\mathbf{x})(g(\mathbf{x}) - g^\varepsilon(\mathbf{x}))\|_\infty + \|g^\varepsilon(\mathbf{x})(f(\mathbf{x}) - f^\varepsilon(\mathbf{x}))\|_\infty, \\
&\leq c_1\varepsilon\|g^\varepsilon(\mathbf{x})\|_\infty + c_2\varepsilon\|f(\mathbf{x})\|_\infty.
\end{aligned} \tag{3.41}$$

Analogously, it can be shown that

$$\|f(\mathbf{x})g(\mathbf{x}) - f^\varepsilon(\mathbf{x})g^\varepsilon(\mathbf{x})\|_\infty \leq c_1\varepsilon\|g(\mathbf{x})\|_\infty + c_2\varepsilon\|f^\varepsilon(\mathbf{x})\|_\infty. \tag{3.42}$$

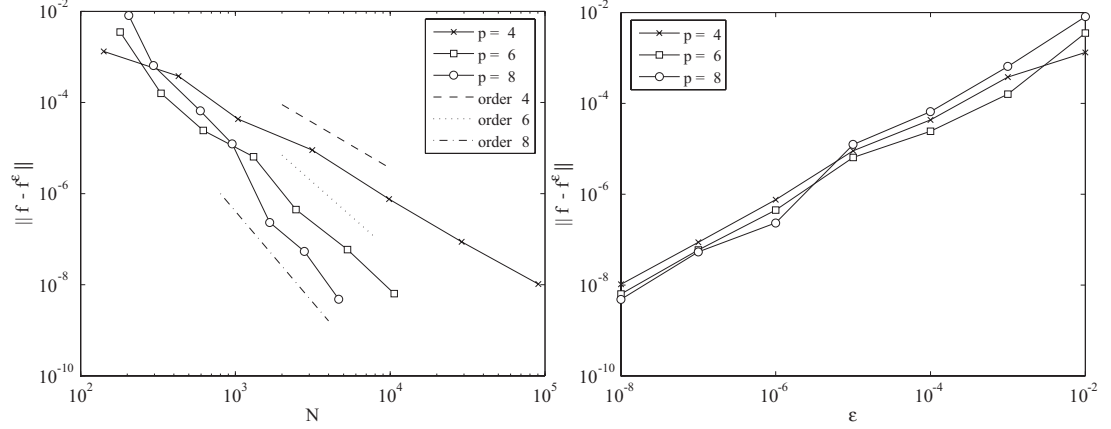


Figure 3.9: Convergence in  $L_\infty$ -norm of a multiplication of the SWR of the test function (3.34) and the SWR of the test function (3.35) for various  $p$  and  $\epsilon$ . Left: error vs. numbers of points in extended grids. Right: error vs. threshold values.

Thus, for multiplication, the error depends on the functions. If they are properly normalized, the error in this procedure remains proportional to the threshold value  $\epsilon$ . Note that an analogous result can be shown for the division of two functions (assuming that the denominator is not zero anywhere). The above estimate is numerically demonstrated for the multiplication of the SWR of the test function (3.34) and the SWR of the test function (3.35) (see plots on right column of Figure 3.9). In this particular case, the error arising from the multiplication is proportional approximately to  $O(N^{-p/2})$ .

### 3.5.3 Nonlinear function evaluation

The evaluation of a nonlinear function on the SWR,  $g \circ f^\epsilon$ , can be accomplished in a way similar as in the multiplication described above. More precisely, for a given  $\{\{f_{l_0, \mathbf{k}}\}_{\mathbf{k} \in \mathbf{k}_{l_0}^0}, \{d_{l, \lambda}^f\}_{(l, \lambda) \in \Lambda^\epsilon}\}$ , the procedure for evaluating  $g \circ f^\epsilon$  reads as follows: (i) calculate  $\{\{f_{l_0, \mathbf{k}}\}_{\mathbf{k} \in \mathbf{k}_{l_0}^0}, \{f_{l, \lambda}\}_{(l, \lambda) \in \Lambda^\epsilon}\}$  by means of the in-

verse wavelet transform, and (ii) compute the pointwise values  $g(f^\varepsilon(\mathbf{x}))$ ,  $\mathbf{x} \in \{\{\mathbf{x}_{l_0, \mathbf{k}}\}_{\mathbf{k} \in \mathbf{k}_{l_0}^0}, \{\mathbf{x}_{l, \lambda}\}_{(l, \lambda) \in \Lambda^\varepsilon}\}$ . It can be verified that the operations required by this procedure is proportional to the number of grid points. Note that for any  $\mathbf{x}$ , there is an appropriate  $c$  such that  $f^\varepsilon(\mathbf{x}) + c\varepsilon$  and  $|c|$  is of course smaller than the constant defined in (3.31). If  $g$  is differentiable, the following estimate holds:

$$\begin{aligned} g(f(\mathbf{x})) &= g(f^\varepsilon(\mathbf{x}) + c\varepsilon) \\ &= g(f^\varepsilon(\mathbf{x})) + cO(\varepsilon) \end{aligned} \tag{3.43}$$

Therefore, one can expect that this procedure yields results with error of order  $\varepsilon$  (however, the constant associated with it could be large in certain cases). To check the validity of the above estimate, we consider the evaluation of the composite function,

$$g(f(\mathbf{x})) = \exp \left[ \frac{-5(1 - f(\mathbf{x}))}{1 - 1/2(1 - f(\mathbf{x}))} \right]. \tag{3.44}$$

where  $f(\mathbf{x})$  is the test function (3.34). Figure 3.10 show relationship between the error of  $g(f^\varepsilon)$  and the threshold parameter  $\varepsilon$ . In this particular test case, the error is of the same of with the threshold value.

### 3.6 Derivative Approximation

For a given SWR, the derivative approximation can be computed by direct differentiation of the multiscale basis (see [11, 12, 133, 134]). However, this approach is somewhat computationally costly because of the different support sizes of wavelets on different levels and the fact that a derivative of wavelet functions no longer satisfies the interpolation property. Evaluating the derivative at a specific grid point involves a number of derivatives of wavelets (approximately  $O(p(J - l_0))$  at each point, and hence  $O(p(J - l_0)N)$  overall, where



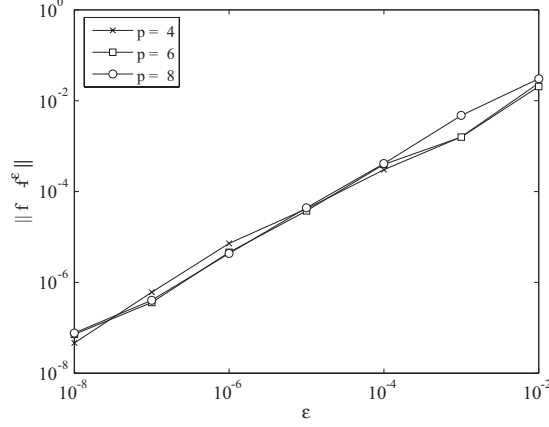


Figure 3.10: Error in  $L_\infty$ -norm of  $g(f^\epsilon)$  of the SWR of the test function (3.34) as a function of threshold value  $\epsilon$ .

$(J - l_0)$  is the number of levels used in the approximation). An alternative approach is to approximate a derivative using a finite difference approximation (see [70, 76, 81, 111, 132, 136, 138]). Let  $\mathcal{D}$  be the set of wavelet coefficients of the SWR, *i.e.*  $\mathcal{D} = \{\{f_{j_0, \mathbf{k}}\}_{\mathbf{k} \in \mathbf{k}_{l_0}^0}, \{d_{l, \lambda}\}_{(l, \lambda) \in \Lambda^\epsilon}\}$ . As a reminder, since  $\mathbf{x}_{j, \lambda}$  is nothing but  $\mathbf{x}_{j+1, 2\mathbf{k}+\mathbf{e}}$ , there exists a unique index at the finest level  $(J, \mathbf{m})$  such that  $\mathbf{x}_{J, \mathbf{m}} = \mathbf{x}_{j, \lambda}$ . We can therefore define  $\Xi^\epsilon = \{(J, \mathbf{k}) : \mathbf{x}_{J, \mathbf{k}} = \mathbf{x}_{l, \lambda}, (l, \lambda) \in \Lambda^\epsilon\} \cup \{(J, \mathbf{k}) : \mathbf{x}_{J, \mathbf{k}} = \mathbf{x}_{l_0, \mathbf{m}}, \mathbf{m} \in \mathbf{k}_{l_0}^0\}$ . With the above notations, the procedure of the derivative approximation using finite-differences can be described by follows:

1. For a given SWR, perform the fast inverse interpolating wavelet transform to yield the function values on an irregular grid of points  $\mathcal{V}$  with index  $(J, \mathbf{k}) \in \Xi$ , *i.e.*

$$AIWT(\mathcal{D}) \rightarrow \{f_{J, \mathbf{m}} = f(x_{J, \mathbf{m}}), (J, \mathbf{m}) \in \Xi\}, \quad (3.45)$$

where  $\mathbf{V}$  is the grid of points required in the finite difference approximation. In general, it is larger than  $\mathbf{V}_J^\varepsilon$  (the irregular grid associated with the SWR), *i.e.*  $\mathbf{V}_J^\varepsilon \subseteq \mathbf{V}$  (hence,  $\Xi^\varepsilon \subseteq \Xi$ ).

2. Use finite differences to approximate the derivative at each grid point. More precisely, at each grid point  $\mathbf{x} = \mathbf{x}_{J,\mathbf{m}}$ ,  $(J, \mathbf{m}) \in \Xi^\varepsilon$  (or equivalently at each  $\mathbf{x}$  in  $\mathbf{V}_J^\varepsilon$ ), the  $i^{\text{th}}$  derivative approximation with respect to  $x_r$  at such grid point is determined by

$$\left. \frac{\partial^i f_\varepsilon}{(\partial x_r)^i} \right|_{\mathbf{x}} \approx f_{J,\mathbf{m}}^{(x_r)^i} = \frac{1}{h^i} \sum_{l \in \mathcal{S}_{i,\mathbf{x}}^{x_r}} a_l^{r,\mathbf{x}} f(x_{J,(m_1,\dots,m_r+l,\dots,m_d)}), \quad (3.46)$$

where  $m_r$  is the  $r^{\text{th}}$ -component of  $\mathbf{m}$  and  $\mathcal{S}_{i,\mathbf{x}}^{x_r}$  denotes the index set of the finite-difference stencil  $\{\mathbf{x}_{J,(m_1,\dots,m_r+l,\dots,m_d)} : l \in \mathcal{S}_{i,\mathbf{x}}^{x_r}\}$  of the  $n^{\text{th}}$ -derivative with respect to  $x_r$  at point  $\mathbf{x}$ . Note that an index  $(J, (m_1, \dots, m_r+l, m_d))$ ,  $l \in \mathcal{S}_{i,\mathbf{x}}^{x_r}$  is not necessarily in the index set  $\Xi^\varepsilon$ . Here,  $h$  refers to the associated step size associated with the stencil  $\{\mathbf{x}_{J,(m_1,\dots,m_r+l,\dots,m_d)} : l \in \mathcal{S}_{i,\mathbf{x}}^{x_r}\}$ . It is defined as the minimum of the spacings between two consecutive stencil points and  $\{a_l^{r,\mathbf{x}}\}_{l \in \mathcal{S}_{i,\mathbf{x}}^{x_r}}$  denote the corresponding finite-difference coefficients for the  $i^{\text{th}}$  derivative. For a consistent approximation, the order of the finite difference scheme must be connected to the order of wavelets used. It is currently assumed that the finite difference approximation (3.46) is  $O(h^n)$ .

3. If required, apply the interpolating wavelet transform to the result to obtain the corresponding wavelet coefficients:

$$AFWT(\{f_{J,\mathbf{m}}^{(x_r)^i}, (J, \mathbf{m}) \in \Xi^\varepsilon\}) \rightarrow \mathcal{D}^{(x_r)^i} = \{\{f_{j_0,\mathbf{k}}^{(x_r)^i}\}, \{d_{l,\lambda}^{(x_r)^i}, \lambda \in \Lambda^\varepsilon\}\} \quad (3.47)$$

and, subsequently, the derivative approximation is given by

$$D_{x_r}^{(i)} f_\varepsilon^J(\mathbf{x}) = \sum_{\mathbf{k}} f_{l_0, \mathbf{k}}^{(x_r)^i} \Phi_{l_0, \mathbf{k}}(\mathbf{x}) + \sum_{(l, \lambda) \in \Lambda^\varepsilon} d_{l, \lambda}^{(x_r)^i} \Psi_{l, \lambda}(\mathbf{x}). \quad (3.48)$$

It can be verified that the above procedure requires only  $O(N)$  operations, where  $N = \dim \mathbf{V}_J^\varepsilon$  (assumed that  $\dim \mathbf{V}$  is of the same order with  $N$ ). Above, no specific detail on the strategy of choosing the finite difference stencil is provided. In what follows, two particular strategies are discussed.

In the first approach, we use the so-called  $\Lambda$ -cycle approach of [136] to choose the finite difference stencil. This approach results in a finite difference approximation on a locally reconstructed uniform stencil [76, 136]. At each level, we determine  $\Lambda^j$ , which is the set of points  $\mathbf{x}_{j, \mathbf{k}} \in \mathbf{V}_j^\varepsilon$  that have no immediate surrounding points on the next level of resolution, *i.e.*  $\mathbf{x}_{j+1, 2\mathbf{k} \pm \mathbf{1}} \notin \mathbf{V}_{j+1}^\varepsilon$ . The stencils associated with such points are those of the  $n^{\text{th}}$  order accurate uniform centered stencil (or those of the uniform one-sided stencil and non-symmetric stencil at and near boundaries, respectively) with step size of  $h = 2^{-j}$ . By doing this starting from level  $l_0$  and progressively advancing until reaching level  $J$ , one obtains the finite difference stencils for every grid point in  $\mathbf{V}_J^\varepsilon$ . It can be observed that this strategy requires points that do not belong to the grid  $\mathbf{V}_J^\varepsilon$ . Here, we use  $\mathbf{V}$  to denote a grid that is a union of these missing points and those of  $\mathbf{V}_J^\varepsilon$ , as defined in step 1 above. The approximate function values on  $\mathbf{V}$  can be obtained by setting wavelet coefficients associated with the points  $\mathbf{V} \setminus \mathbf{V}_J^\varepsilon$  to zero and subsequently computing the inverse wavelet transform (*AIWT*). Moreover, if the function values  $f(\mathbf{x})$ ,  $\mathbf{x} \in \mathbf{V}_J^\varepsilon$  are known (instead of the associated wavelet coefficients), one can also use *LAIWT* (see section 3.4) to find directly the approximate function values at the missing points.

In this setting, it is not straightforward to provide a rigorous accuracy estimate of the derivative approximation. The following discussion, which uses a similar argument given in [76], attempts to provide an approximate estimate. Unlike the SWR, the error of the derivative approximation, in general, does not vanish at the grid points in  $\mathbf{V}_j^\varepsilon$ . To obtain a norm of the error, we introduce the following maximum norm over the grid points:

$$\|g\|_{\mathbf{V},\infty} = \max_{\mathbf{x} \in \mathbf{V}} |g(\mathbf{x})|, \quad (3.49)$$

where  $\mathbf{V}$  is an irregular grid. Here, the error estimate relies on the fact that when  $|d_{j+1,\lambda}|$  is less than the threshold value  $\varepsilon$ , the actual function in the neighborhood of the point  $x_{j+1,\lambda}$  is well approximated by the local polynomial interpolation constructed from points with step size  $h = 2^{-j}$ , *i.e.*, it is locally well approximated by  $I_j f$ . In light of (3.20) and (3.31), we have that, in such neighborhood, the following relation holds

$$h^p \sim \varepsilon. \quad (3.50)$$

As noted earlier, stencils associated with points in  $\Lambda^j$  (as a reminder they are points that have no immediate surrounding point associated with wavelets on level  $j+1$ ) have step length  $h = 2^{-j}$  and the finite difference scheme is assumed to be  $O(h^n)$ . Thus at a grid point in which its associated stencil has no missing point(s), the truncation error of the scheme is approximately  $O(h^n) \sim O(\varepsilon^{n/p})$ . For a grid point in which its associated stencil contains points that do not belong to  $\mathbf{V}_j^\varepsilon$ , there is an additional error associated with the wavelet interpolation. In this case, the local error of the finite difference approximation is given by

$$\begin{aligned}
D_{x_r}^{(i)} f_\varepsilon^J \Big|_{\mathbf{x}} - \partial^i f / (\partial x_r)^i \Big|_{\mathbf{x}} &\sim O(\varepsilon/h^i) + O(h^n), \\
&\sim O(\varepsilon^{(p-i)/p}) + O(\varepsilon^{n/p}), \\
&\sim O(\varepsilon^{\min((p-i)/p, n/p)}).
\end{aligned}$$

From (3.32), it follows that

$$D_{x_r}^{(i)} f_\varepsilon^J \Big|_{\mathbf{x}} - \partial^i f / (\partial x_r)^i \Big|_{\mathbf{x}} \sim O(N^{-\min((p-i), n)/d}). \quad (3.51)$$

for a sufficiently smooth function. Hence, we conclude that the pointwise error of the derivative approximation has the following bound:

$$\|\partial^i f / (\partial x_r)^i - D_{x_r}^{(i)} f_\varepsilon^J\|_{\mathbf{V}_{J,\infty}^\varepsilon} \leq CN^{-\min((p-i), n)/d}. \quad (3.52)$$

The above result suggests that in order not to lose accuracy, a finite difference of order  $n \leq p - i$  should be employed. It is remarked while that (3.52) is by no means a rigorous error bound, results from numerical experiments agree reasonably well with it. Figure 3.11 shows the log-log plots of error of the first and second derivative approximation  $D^{(i)} f_\varepsilon$  of the test function (3.34) versus the number of grid points  $N$  for different combinations of  $p$  and  $n$  and for threshold values  $\varepsilon$  ranging from  $10^{-2}$  to  $10^{-8}$ . Table (3.1) summarize the slopes of the plots (more precisely, they are the slopes of the linear curve fitting in the least square sense.). It can be seen that, for the derivative approximation with  $p = 4$  and  $6$ , the relationship between error and  $N$  conform reasonably with the estimate (3.52). The rate of convergence for  $p = 8$  and  $n = 6$  seems to be much faster than the prediction; however, in this case, if the slope is computed using only data of the last few points of such plot (*i.e.* discarding data associated with large threshold

TABLE 3.1

SLOPES OF THE LOG-LOG PLOTS IN FIGURE 3.11

	$-d \times \text{slope}$	
	1 <sup>st</sup> derivative	2 <sup>nd</sup> derivative
$p = 4, n = 2$	2.1	2.1
$p = 4, n = 4$	3.8	2.1
$p = 6, n = 4$	4.7	4.2
$p = 6, n = 6$	5.6	3.5
$p = 8, n = 6$	7.8	8.6

values), the slopes of the plots are approximately 6, which agree reasonably well with the estimate.

We also use an approach developed by [111], which is more computationally efficient. This approach is based on the calculation of finite differences on non-uniform stencils. It is known that a stencil which is highly non-uniform, *i.e.* the ratio of the largest to the smallest spacing between points in the stencil is large, may yield a non-robust algorithm. For this reason, the stencil is chosen such that the ratio of the largest to the smallest spacing between points should be no greater than 2. If the stencil does not satisfy such restriction, a locally reconstructed uniform stencil is then used instead. In practice, it is found that the stencil satisfies the above restriction most of the times. Consequently, the need for interpolations is reduced substantially. In addition, this approach takes advantage of the fact that the irregular grid  $\mathcal{V}_J^\varepsilon$  is a dyadic semi-structured grid. Hence, coefficients of

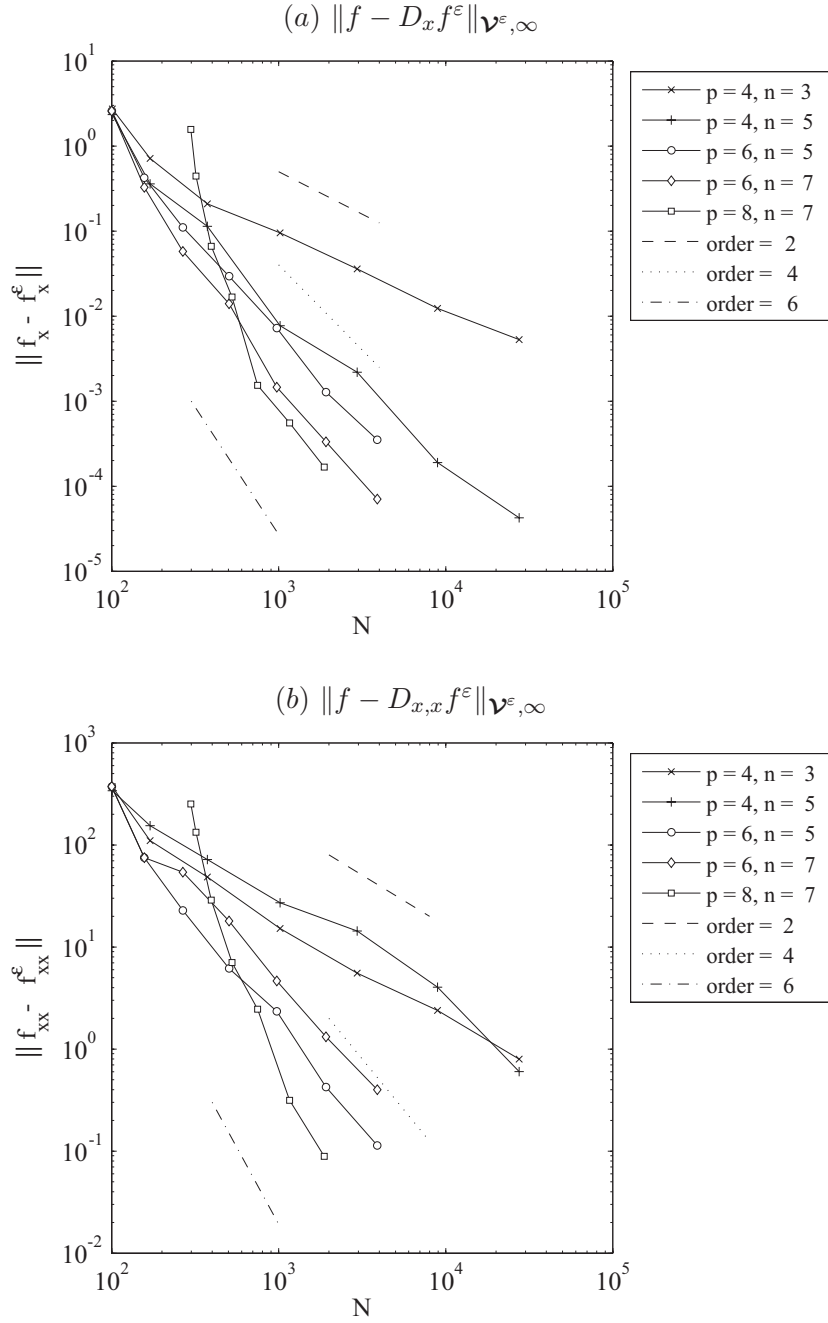


Figure 3.11:  $L_{\mathbf{V}, \infty}$ -error of the derivative approximation on irregular grids,  $\|\partial^i f / \partial x - D_{x_1}^{(i)} f^\varepsilon\|_{\mathbf{V}^\varepsilon, \infty}$ , of the test function (3.34) as a function of  $N = \dim \mathbf{V}^\varepsilon$  for different  $p$ ,  $n$ , and  $\varepsilon$ . (a) error of the first derivative approximation with respect to  $x_1$ -direction. (b) error of the second derivative approximation with respect to  $x_1$ -direction.

all possible finite-difference stencils can be calculated and stored *a priori*. The calculation of finite-difference coefficients for a particular grid then reduces to simply fetching the stored coefficients whose stencil matches the one being considered. It can be argued that the estimate (3.52) is also applicable for this approach.

When solving a (linear) second order partial differential equation using a finite difference approximation on an irregular grid, we expect that the accuracy of the numerical solution should conform (to some extent) with (3.52). This, in turn, implies that if the method is used with  $n \leq p - 2$ , the order of accuracy of the numerical solution is expected to be  $n$ . For  $n > p - 2$ , it is expected that the accuracy of the solution is of order  $p - 2$  and hence the  $n - p + 2$  order of accuracy (compared with the order of the finite difference scheme) is lost. However, numerical solutions to specific problems (see Section 4.4) show that such loss of accuracy may not be as large as that of the estimate.

### 3.7 MRA- $d$ Approach

The wavelet functions discussed previously are constructed based on the approach most commonly used in the wavelet literature. Alternatively, the multiscale basis can be constructed directly from the tensor products of 1-D wavelets. This so-called MRA- $d$  approach is investigated by [70, 71] and references therein. More precisely, the basis functions are obtained by considering the following decomposition

$$\begin{aligned} \mathbf{V}_j &= \underbrace{V_j \otimes \cdots \otimes V_j}_{d\text{-times}}, \\ &= \underbrace{\left[ V_{l_0} \oplus \left( \bigoplus_{l=l_0}^{j-1} W_l \right) \right] \otimes \cdots \otimes \left[ V_{l_0} \oplus \left( \bigoplus_{l=l_0}^{j-1} W_l \right) \right]}_{d\text{-times}}. \end{aligned} \quad (3.53)$$



Therefore, higher-dimensional basis functions are given by

$$\Psi_{(\mathbf{l}, \mathbf{k})}(\mathbf{x}) \equiv \psi_{l_1, k_1}(x_1) \cdots \psi_{l_d, k_d}(x_d), \quad \mathbf{k} \in \mathbf{k}_1, \quad \mathbf{l} \in \mathbf{T}(l_0, j), \quad (3.54)$$

where

$$\mathbf{k}_1 = k_{l_1}^1 \times \cdots \times k_{l_d}^1, \quad (3.55)$$

and

$$\mathbf{T}(l_0, j) = \underbrace{\{l_0 - 1, l_0, \dots, j - 1\} \times \cdots \times \{l_0 - 1, l_0, \dots, j - 1\}}_{d\text{-times}}. \quad (3.56)$$

Note that  $\psi_{l_0-1, k}(\cdot) \equiv \phi_{l_0, k}(\cdot)$  and  $k_{l_0-1}^1 \equiv k_l^0$ . As in the MRA approach, each tensor basis function  $\Psi_{(\mathbf{l}, \mathbf{k})}(\mathbf{x})$  is associated closely with the grid point  $\mathbf{x}_{(\mathbf{l}, \mathbf{k})}$ , where  $\mathbf{x}_{(\mathbf{l}, \mathbf{k})} \equiv (\tilde{k}_1/2^{\tilde{l}_1}, \dots, \tilde{k}_d/2^{\tilde{l}_d})$ ,  $\tilde{k}_i = 2k_i + 1$ ,  $\tilde{l}_i = l_i + 1$  for  $l_i \geq l_0$ , and  $\tilde{k}_i = k_i$ ,  $\tilde{l}_i = l_0$  for  $l_i = l_0 - 1$ . Note that for a given dyadic point  $\mathbf{x}$ , the index  $(\mathbf{l}, \mathbf{k})$  such that  $\mathbf{x}_{(\mathbf{l}, \mathbf{k})} = \mathbf{x}$  can be uniquely determined.

It follows directly from (3.53) and (3.54) that the interpolation function (3.5) can be written as,

$$(I_j f)(\mathbf{x}) = \sum_{(\mathbf{l}, \mathbf{k}) \in \mathbf{L}(l_0, j)} d_{(\mathbf{l}, \mathbf{k})} \Psi_{(\mathbf{l}, \mathbf{k})}(\mathbf{x}), \quad (3.57)$$

where  $d_{(\mathbf{l}, \mathbf{k})}$  is a coefficient associated with the basis function  $\Psi_{(\mathbf{l}, \mathbf{k})}(\mathbf{x})$  and,  $\mathbf{L}(l_0, j) = \{(\mathbf{l}, \mathbf{k}) : \mathbf{l} \in \mathbf{T}(l_0, j), \mathbf{k} \in \mathbf{k}_1\}$ . The coefficients  $d_{(\mathbf{l}, \mathbf{k})}$  can be computed directly from

$$d_{(\mathbf{l}, \mathbf{k})} = \begin{cases} \left( \prod_{i=1}^d I_{l_i, k_i} \right) f \equiv \mathbf{I}_{(\mathbf{l}, \mathbf{k})} f & \text{for } \mathbf{l} \neq \mathbf{l}_0 - \mathbf{1}, \\ f_{l_0, \mathbf{k}} & \text{for } \mathbf{l} = \mathbf{l}_0 - \mathbf{1}, \end{cases} \quad (3.58)$$

where  $\mathbf{l}_0 = \{l_0, \dots, l_0\}$ . Here, for  $l \geq l_0$ ,  $I_{l, k} = I_{l, k}^1$  is a 1-D stencil associated with the points  $X_{l, k}^1$  and for  $l = l_0 - 1$ ,  $I_{l, k} = [1]$  is a 1-D stencil associated with the

point  $X_{l,k}^0$  (see (3.14) and (3.15) for the detailed definition). Note that  $\mathbf{l}_{(\mathbf{l},\mathbf{k})}$  is understood as the  $(d-n)$ -dimensional stencil, where  $n$  is the number of entries in  $\mathbf{l}$  with value of  $l_0 - 1$ , and  $f$  denotes the function values at the stencil points. By using a Taylor series analysis, it can be shown that

$$|d_{\mathbf{l},\mathbf{k}}| \leq C \max_{\xi} \{D^{\mathbf{p}} f\} 2^{-p|\mathbf{l}|_1} \quad (3.59)$$

where  $|\mathbf{l}|_1 \equiv \sum_{i \in \{i : l_i \neq l_0 - 1\}} l_i$ . This indicates that the coefficients decay exponentially with respect to  $|\mathbf{l}|_1$ . Note that for a function with less smoothness, the coefficients decay at a rate slower than that given in the above equation. In fact, the coefficients  $d_{(\mathbf{l},\mathbf{k})}$  measure the difference between  $f(\mathbf{x}_{\mathbf{l},\mathbf{k}})$  and the values predicted from the local polynomial approximations. As a consequence, they can be used to detect local sharp variations in the function.

For a given threshold  $\varepsilon$ , the SWR of the MRA- $d$  is given by

$$f_J^\varepsilon = \sum_{(\mathbf{l},\mathbf{k}) \in \mathcal{T}(l_0,J)} d_{(\mathbf{l},\mathbf{k})} \Psi_{(\mathbf{l},\mathbf{k})}(\mathbf{x}), \quad (3.60)$$

where

$$\mathcal{T}(l_0, J) = \{(\mathbf{l}_0 - \mathbf{1}, \mathbf{k}) : \mathbf{k} \in \mathbf{k}_{l_0}^0\} \cup \{(\mathbf{l}, \mathbf{k}) \in \mathbf{L}(l_0, J) : |d_{(\mathbf{l},\mathbf{k})}| \geq \varepsilon\}. \quad (3.61)$$

Note that the thresholding given above is effectively based on the max-norm. To add an additional damping effect, one can use  $L_1$ - or  $L_2$ -norm in the thresholding process (see (3.37) for details). Analogous to the MRA approach, the error of the SWR for MRA- $d$  is also  $O(\varepsilon)$  [73]. Similarly, associated with the SWR  $f_J^\varepsilon$  is the irregular grid

$$\{\mathbf{x}_{\mathbf{l},\mathbf{k}} : (\mathbf{l},\mathbf{k}) \in \mathcal{T}(l_0, J)\}. \quad (3.62)$$

The computation of  $d_{(\mathbf{l},\mathbf{k})}$  using (3.58) is inefficient. In fact, the salient advantage of MRA- $d$  comes the fact that algorithms such as the wavelet transform, its inverse, interpolation and derivative approximation reduce to the application of 1-D algorithm (see [71, 73]). Algorithms of any  $d$ -D problem boils down to the application of 1-D algorithms in each direction sequentially. More specifically, let  $f_{(\mathbf{l},\mathbf{k})}$  represent the data at point  $\mathbf{x}_{(\mathbf{l},\mathbf{k})}$ , the wavelet transform mapping the given data  $\{f_{(\mathbf{l},\mathbf{k})}\}_{(\mathbf{l},\mathbf{k}) \in \mathcal{T}(l_0, J)}$  to the associated wavelet coefficients  $\{d_{(\mathbf{l},\mathbf{k})}\}_{(\mathbf{l},\mathbf{k}) \in \mathcal{T}(l_0, J)}$  can be accomplished by the following procedure

$$\mathcal{H}\{\mathcal{H} \cdots \{\mathcal{H}\{f_{\mathbf{l},\mathbf{k}}\}_{(\hat{\mathbf{l}}^1, \hat{\mathbf{k}}^1)=\text{const}}\}_{(\hat{\mathbf{l}}^2, \hat{\mathbf{k}}^2)=\text{const}} \cdots \}_{(\hat{\mathbf{l}}^d, \hat{\mathbf{k}}^d)=\text{const}}, \quad (3.63)$$

where  $\mathcal{H}$  is the 1-D wavelet transform on the irregular grid (see Chapter 2 for details),  $\{f_{(\mathbf{l},\mathbf{k})}\}_{(\hat{\mathbf{l}}^i, \hat{\mathbf{k}}^i)=\text{const}}$  is the data on the grid with index  $(\mathbf{l}, \mathbf{k})$  having a certain subindex  $(\hat{\mathbf{l}}^i, \hat{\mathbf{k}}^i)$ . One can think of it as the  $(\hat{\mathbf{l}}^i, \hat{\mathbf{k}}^i)$ -th column in  $i$ -direction. Here,  $\{\mathcal{H}\{g_{\mathbf{l},\mathbf{k}}\}_{(\hat{\mathbf{l}}^i, \hat{\mathbf{k}}^i)=\text{const}}\}$  represents the result obtained from applying the 1-D wavelet transform to every column in  $i$ -direction. The inverse transform can be accomplished by replacing  $\mathcal{H}$  by the 1-D inverse transform  $\mathcal{H}^{-1}$  and  $\{f_{(\mathbf{l},\mathbf{k})}\}$  now represents wavelet coefficients. The wavelet transform and its inverse with respect to the  $i$ -direction are outlined in Algorithms 12 and 13 respectively. Subsequently, the  $d$ -dimensional wavelet transform and inverse transform can be obtained by applying  $PAFWT2(i)/PAIWT2(i)$  with  $i = 1, 2, \dots, d$ , respectively, as given in Algorithms 14 and 15. Note that  $PAFWT2$  is not a direct implementation of the process  $\{\mathcal{H}\{f_{\mathbf{l},\mathbf{k}}\}_{(\hat{\mathbf{l}}^i, \hat{\mathbf{k}}^i)=\text{const}}\}$  described above; here the calculation with respect to the  $i$ -direction are done in a level-wise fashion (instead of column-wise manner). In

---

**Algorithm 12** *PAFWT2*( $i$ )

---

Given data  $\{f_{\mathbf{l},\mathbf{k}}\}_{(\mathbf{l},\mathbf{k}) \in \mathcal{T}(l_0, J)}$  and  $i$ -direction  
**for**  $j = J - 1$  to  $l_0$  **do**  
  **for all**  $(\mathbf{l}, \mathbf{k}) \in \mathcal{T}(l_0, J)$  such that  $l_i = j$  **do**  
     $\mathbf{x} = \mathbf{x}_{(\mathbf{l}, \mathbf{k})}$   
    **for**  $n = \min \mathcal{X}_{l_i, k_i}$  to  $\max \mathcal{X}_{l_i, k_i}$  **do**  
       $x_i = 2^{-l_i} n$  and find index  $(\mathbf{l}, \mathbf{n})$  such that  $\mathbf{x}_{\mathbf{l}, \mathbf{n}} = \mathbf{x}$   
       $f_{(\mathbf{l}, \mathbf{k})} = f_{(\mathbf{l}, \mathbf{k})} - h_{2^{k_i}+1}^{l_i, n} f_{(\mathbf{l}, \mathbf{n})}$   
    **end for**  
  **end for**  
**end for**

---

---

**Algorithm 13** *PAIWT2*( $i$ )

---

Given data  $\{f_{\mathbf{l},\mathbf{k}}\}_{(\mathbf{l},\mathbf{k}) \in \mathcal{T}(l_0, J)}$  and  $i$ -direction  
**for**  $j = l_0$  to  $J - 1$  **do**  
  **for all**  $(\mathbf{l}, \mathbf{k}) \in \mathcal{T}(l_0, J)$  such that  $l_i = j$  **do**  
     $\mathbf{x} = \mathbf{x}_{(\mathbf{l}, \mathbf{k})}$   
    **for**  $n = \min \mathcal{X}_{l_i, k_i}$  to  $\max \mathcal{X}_{l_i, k_i}$  **do**  
       $x_i = 2^{-l_i} n$  and find index  $(\mathbf{l}, \mathbf{n})$  of the point  $\mathbf{x}$   
       $f_{(\mathbf{l}, \mathbf{k})} = f_{(\mathbf{l}, \mathbf{k})} + h_{2^{k_i}+1}^{l_i, n} f_{(\mathbf{l}, \mathbf{n})}$   
    **end for**  
  **end for**  
**end for**

---

this way, we can use a (slightly modified) data structure [111], originally designed for the MRA approach, for the MRA- $d$  approach.

It can be verified that the  $i$ -directed transform requires  $pN$  operations, where  $N = \dim \mathcal{T}(l_0, J)$ . Thus, for MRA- $d$ , the total number of operations required in the wavelet transform or its inverse is  $dpN$ . The number of operations needed in this approach is slightly higher than that of the conventional MRA approach when  $N$  of two approaches are identical. However, generally, the number of points required by the two approaches are not identical. We remark that, as in the conventional MRA case, the index set  $\mathcal{T}$  must also satisfy the so-called minimum index set condition in order for *AFWT2*/*AIWT2* to work. In general, the set  $\mathcal{T}$  resulting from thresholding will not meet such a condition. This can be easily remedied by enlarging the index set. Owing to the tensor product nature, this can be done easily by applying the algorithm *AUGMENT* (see Chapter 2 for details) to enlarge grid in each direction successively.

---

**Algorithm 14** *AFWT2*

---

Given function values  $\{f_{\mathbf{l}, \mathbf{k}}\}_{(\mathbf{l}, \mathbf{k}) \in \mathcal{T}(l_0, J)}$   
**for**  $id = 1$  to  $d$  **do**  
    Compute the coefficients in place using *PAFWT2*( $id$ )  
**end for**

---



---

**Algorithm 15** *AIWT2*

---

Given wavelet coefficients  $\{f_{\mathbf{l}, \mathbf{k}}\}_{(\mathbf{l}, \mathbf{k}) \in \mathcal{T}(l_0, J)}$   
**for**  $id = 1$  to  $d$  **do**  
    Compute the function values in place using *PAIWT2*( $id$ )  
**end for**

---

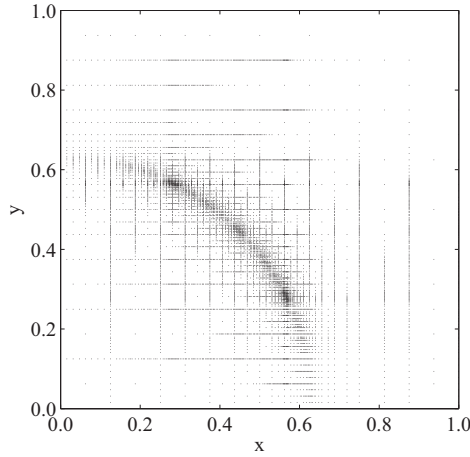


Figure 3.12: MRA- $d$  irregular grid associated with the SWR of the function  $f(x, y) = 0.2/(|0.4 - x^2 - y^2| + 0.2)$  for  $\varepsilon = 5 \times 10^{-3}$  and  $p = 6$ .

We note that the SWR of a continuous function can be obtained through a procedure that is similar to that of *COMPRESS*. Figure 3.12 shows for example the MRA- $d$  irregular grid associated with the SWR of the function having a singularity (in the first derivative) of circular shape (see Figure 3.4 for comparison with that of the conventional MRA approach).

The algebra of the SWR (*e.g.*, addition, subtraction, multiplication, and non-linear function evaluation) can be accomplished in similar ways as in the MRA approach (see section 3.5). Since the wavelet transform and its inverse transform requires only  $O(N)$  operations, the number of operations required in these procedures is subsequently  $O(N)$ . In other words, the cost of these procedures is proportional to the number of grid points. It can be easily shown that error in these procedures also obeys the estimates given in section 3.5.

### 3.7.1 Derivative approximation for MRA- $d$

As in the MRA approach, approximating a derivative of the SWR associated with the MRA- $d$  by mean of direct differentiation of the basis functions is computationally expensive. [70, 71, 88] introduce a procedure that makes use of finite differences. Such a procedure for approximating the  $i^{\text{th}}$  partial derivative with respect to  $x_r$  from given wavelet coefficients  $\{d_{\mathbf{l},\mathbf{k}}\}_{(\mathbf{l},\mathbf{k}) \in \mathcal{T}}$  can be described as follows:

1. For a given  $\mathcal{D} = \{d_{\mathbf{l},\mathbf{k}}\}_{(\mathbf{l},\mathbf{k}) \in \mathcal{T}}$ , perform the inverse wavelet transform with respect to the  $r$ -direction to yield the intermediate function values on grid points with index  $(\mathbf{l},\mathbf{k}) \in \tilde{\mathcal{T}}$ , *i.e.*,

$$PAIWT2(r)(\mathcal{D}) \rightarrow \{\tilde{f}_{\mathbf{l},\mathbf{k}}\}_{(\mathbf{l},\mathbf{k}) \in \tilde{\mathcal{T}}}. \quad (3.64)$$

Here,  $\tilde{\mathcal{T}}$  is an index set of points participating in the finite difference calculation. It is in general larger than  $\mathcal{T}$ , more precisely,  $\mathcal{T} \subseteq \tilde{\mathcal{T}}$ .

2. At each point  $\mathbf{x}_{\mathbf{l},\mathbf{k}}$ ,  $(\mathbf{l},\mathbf{k}) \in \mathcal{T}$ , apply the finite difference operator to the intermediate function values to obtain

$$\tilde{f}_{\mathbf{l},\mathbf{k}}^{(x_r)^i} = \frac{1}{h^i} \sum_{(\mathbf{l},\mathbf{m}) \in \mathcal{S}_{i,(\mathbf{l},\mathbf{k})}^{x_r}} a_{(\mathbf{l},\mathbf{m})}^{r,\mathbf{x}_{(\mathbf{l},\mathbf{k})}} \tilde{f}_{\mathbf{l},\mathbf{m}}, \quad (3.65)$$

where  $\mathcal{S}_{i,(\mathbf{l},\mathbf{k})}^{x_r}$  denotes the set of index  $(\mathbf{l},\mathbf{m})$  that determines the finite-difference stencil of  $i^{\text{th}}$  derivative with respect to  $x_r$  at point  $\mathbf{x}_{\mathbf{l},\mathbf{k}}$ . Note that it is not necessary for an index  $(\mathbf{l},\mathbf{k}) \in \mathcal{S}_{i,(\mathbf{l},\mathbf{k})}^{x_r}$  to belong to the index set  $\mathcal{T}$ . Here,  $h$  refers to the step size associated with the stencil  $\{\mathbf{x}_{\mathbf{l},\mathbf{m}} : (\mathbf{l},\mathbf{k}) \in \mathcal{S}_{i,(\mathbf{l},\mathbf{k})}^{x_r}\}$ . It is defined as the minimum of the spacing between two consecutive stencil points. The quantities  $\{a_{(\mathbf{l},\mathbf{m})}^{r,\mathbf{x}_{(\mathbf{l},\mathbf{k})}} : (\mathbf{l},\mathbf{m}) \in \mathcal{S}_{i,(\mathbf{l},\mathbf{k})}^{x_r}\}$

denote the corresponding finite difference coefficients. In addition, it is assumed that the accuracy of the finite difference scheme (3.65) is  $O(h^n)$ .

3. Perform the wavelet transform with respect to the  $r$ -direction on  $\{\tilde{f}_{\mathbf{l},\mathbf{k}}^{(x_r)^i}\}_{(\mathbf{l},\mathbf{k}) \in \mathcal{T}}$  to obtain the corresponding wavelet coefficients of the derivative approximation:

$$PAFWT2(r)(\{\tilde{f}_{\mathbf{l},\mathbf{k}}^{(i)}\}_{(\mathbf{l},\mathbf{k}) \in \mathcal{T}}) \rightarrow \{\hat{d}_{(\mathbf{l},\mathbf{k})}\}_{(\mathbf{l},\mathbf{k}) \in \mathcal{T}} = \mathcal{D}^{(x_r)^i}. \quad (3.66)$$

4. Values of the derivative approximation are then obtained by means of the inverse wavelet transform:

$$AIWT2(\mathcal{D}_r^{(x_r)^i}) \rightarrow \{f_{\mathbf{l},\mathbf{k}}^{(x_r)^i}\}_{(\mathbf{l},\mathbf{k}) \in \mathcal{T}}, \quad (3.67)$$

where  $f_{(\mathbf{l},\mathbf{k})}^{(x_r)^i}$  denotes the approximate value of the  $i^{\text{th}}$  derivative with respect to  $x_r$  of the function  $f$  at the point  $\mathbf{x}_{\mathbf{l},\mathbf{k}}$ .

In step 2, the finite difference stencil at point  $\mathbf{x}_{\mathbf{l},\mathbf{k}}$  is obtained by first locating the point in  $\{\mathbf{x}_{(\mathbf{j},\mathbf{m})}\}_{(\hat{\mathbf{j}}^r=\hat{\mathbf{l}}^r, \hat{\mathbf{m}}^r=\hat{\mathbf{l}}^r)}$  that is closest to  $\mathbf{x}_{\mathbf{l},\mathbf{k}}$ . The distance to that point,  $h$ , is chosen as the step length. The finite difference stencil associated with  $\mathbf{x}_{\mathbf{l},\mathbf{k}}$  is subsequently that of  $n^{\text{th}}$  order accurate uniformly centered stencil (with the appropriate modification at and near boundaries) with step length  $h$ .

Since the number of operations required in *PAIWT2*, *PAFWT2* or *AIWT2* is proportional to the number of data points, as a result, the procedure of computing the derivative approximation described above requires only  $O(N)$  operations. It is important to mention that, in this procedure, steps 1 and 3 are crucial in obtaining an accurate derivative approximation. Omitting these steps and applying a simple finite difference operator (as done by those noted in the previous paragraph) directly to the function values  $\{f_{\mathbf{l},\mathbf{k}}\}_{(\mathbf{l},\mathbf{k}) \in \mathcal{T}}$  in general results in large errors which



are not reduced as  $N$  is increased. Note that there is no rigorous error estimate of this procedure when the grid is irregular. However, for the special case of the so-called regular sparse grid denoted by  $\mathbf{V}_J^{(1)}$ , it has been shown by [71] that the procedure obeys

$$\|\partial^i f / \partial x_r^i - D_x^{(i)} f_J^{(1)}\|_{\mathbf{V}_J^{(1)}, \infty} \leq C J 2^{-\min(p,n)J}, \quad (3.68)$$

where  $f_J^{(1)}$  represents the SWR associated with the regular sparse grid. The regular sparse grid is defined by

$$\mathbf{V}_J^{(1)} = \{\mathbf{x}_{\mathbf{l}, \mathbf{k}} : (\mathbf{l}, \mathbf{k}) \in \mathbf{L}(l_0, J) \text{ and } |\mathbf{l}|_1 \leq J + (d-1)l_0 - d\} \quad (3.69)$$

where  $|\mathbf{l}|_1 = \sum_{i=1}^d l_i$ . Figure 3.13 for example of a regular sparse grid. Note that

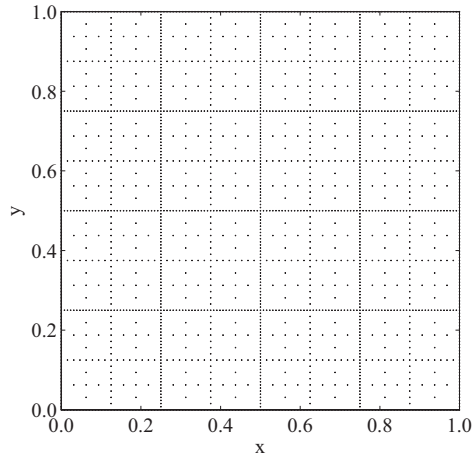


Figure 3.13: Regular sparse grid  $V_J^{(1)}$  with  $J = 7$  and  $l_0 = 2$ .

the  $\dim \mathbf{V}_J^{(1)}$  is  $O(J^{d-1}2^J)$ . This and (3.68) suggest that there is an advantage in using the discretization on  $\mathbf{V}_J^{(1)}$  in comparison to using finite differences on the full grid. While the finite difference approximation on a full grid of step size  $2^{-J}$  using  $2^{dJ}$  grid points yield an error of  $O(2^{-nJ})$ , the discretization on  $V_J^{(1)}$  with only  $O(J^{d-1}2^J)$  grid points yields almost the same accuracy, *i.e.*  $O(J2^{-nJ})$ . Although not covered by this estimate, one can expect this procedure to yield an accurate derivative approximation in the case of an irregular grid. Figure 3.14 shows as an example the accuracy of the derivative approximation  $D_{x_1}^{(i)} f^\varepsilon$  of the test function (3.34) for different combinations of  $p$  and  $n$  as the threshold value is varied from  $10^{-2}$  to  $10^{-8}$  (see 3.11 for similar plots with MRA). Let us note that, for this particular test function, the derivative approximation with MRA- $d$  yields a more accurate approximation (in term of both convergence rate and actual error for the same number of grid used) than that of MRA in the case where  $p = 4$ . For  $p > 4$ , the rate of convergence of the MRA- $d$  approach is either comparable (or even slightly higher for some combinations of  $p$  and  $n$ ) than that of MRA approach; however, actual error (in these specific range of threshold value) for the same number of grid used is higher than that of the MRA approach.

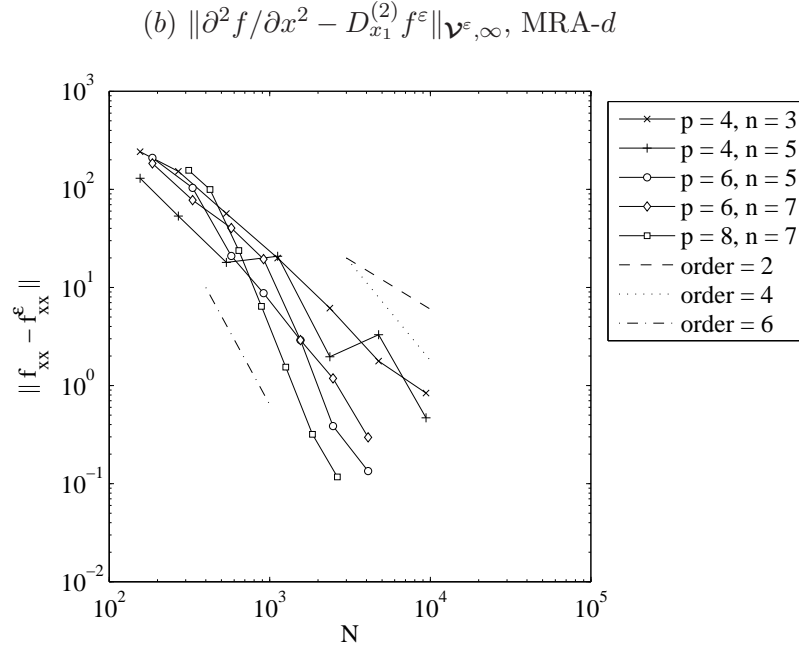
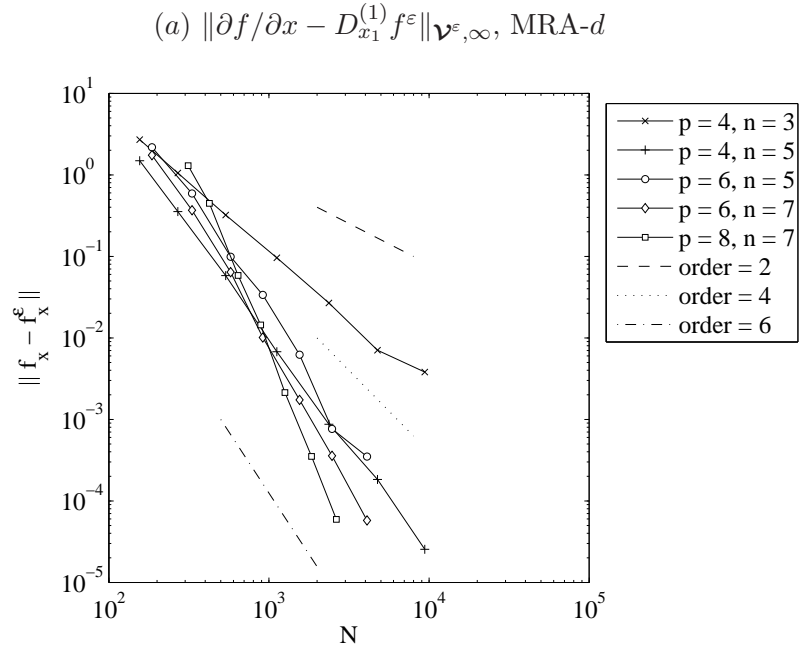


Figure 3.14:  $L_{\mathbf{V}^\varepsilon, \infty}$ -error of the derivative approximation on MRA- $d$  irregular grids,  $\|\partial^i f / \partial x - D_{x_1}^{(i)} f^\varepsilon\|_{\mathbf{V}^\varepsilon, \infty}$ , of the test function (3.34) as a function of  $N = \dim \mathbf{V}^\varepsilon$  for different  $p$ ,  $n$ , and  $\varepsilon$ . (a) error of the first derivative approximation with respect to  $x_1$ -direction. (b) error of the second derivative approximation with respect to  $x_1$ -direction.

## CHAPTER 4

### ADAPTIVE WAVELET METHODS FOR PARTIAL DIFFERENTIAL EQUATIONS

In this chapter, we describe algorithms for the adaptive solution of both time-independent and time dependent partial differential equations (PDEs). Such algorithms are in essence a sequence of performing two main tasks, namely solving the discretization of PDEs and adapting the computational grid. The latter is accomplished by analyzing the wavelet amplitudes of the solution of discretized version of the PDEs. In this work, the spatial discretization of PDEs is accomplished via a collocation procedure and the derivative approximation is computed by means of finite differences as described in the previous chapter. To make things more precise, we first provide detail on the spatial discretization on an irregular grid in the next section. Subsequently, we describe an adaptive algorithm, which use wavelet amplitude as indicators for refinement for adaptively solving time-independent problems without prior knowledge of which area of the domain need refinement. Some comments on preconditioning of the linear system is made in Section 4.3. Later in Section 4.4 the results of the numerical experiments on 2- and 3-D problems using the adaptive algorithm are presented. We use solutions to problems of which exact solutions are available so that the accuracy of the adaptive procedure can be assessed. In Section 4.5, we provide details of a dynamically adaptive algorithm for time-dependent PDEs. The algorithm is designed to

adapt the computational grid according to the solution features that may evolve with time. We then use the adaptive algorithm on a convection-diffusion-reaction problem corresponding to a 2-D flameball-vortex interaction.

#### 4.1 Spatial discretization on irregular grid

In this section, we describe the spatial discretization of PDE problems on a given irregular grid and also the application of different types of boundary conditions. To this end, consider the following 2-D model problem for  $u(\mathbf{x})$ :

$$\begin{cases} -\nabla^2 u + cu = f, & \text{for } \mathbf{x} \in \Omega = (0, 1)^2 \\ u = g, & \text{for } \mathbf{x} \in \Gamma_d \\ \partial u / \partial n = q, & \text{for } \mathbf{x} \in \Gamma_n \end{cases} \quad (4.1)$$

where  $c > 0$  is a positive constant and  $\Gamma_d \cup \Gamma_n = \partial\Omega$  and  $\Gamma_d \cap \Gamma_n = \emptyset$ . Here,  $f$  is a source term and  $g$  and  $q$  are specified Dirichlet and Neumann conditions, respectively. For a given irregular grid  $\mathcal{V} = \{\mathbf{x}_{J,\mathbf{k}} : (J, \mathbf{k}) \in \Xi\}$  and let  $u_{J,\mathbf{k}} = u(\mathbf{x}_{J,\mathbf{k}})$  denotes unknown nodal values of  $u$ . Then the approximate solution at nodal points is obtained by finding  $\mathbf{u} = \{u_{J,\mathbf{k}} : (J, \mathbf{k}) \in \Xi\}$  that satisfies the following problem

$$\begin{cases} -\left[\{\mathbf{D}_{x_1 x_1}^S \mathbf{u}\}_{J,\mathbf{m}} + \{\mathbf{D}_{x_2 x_2}^S \mathbf{u}\}_{J,\mathbf{m}}\right] + cu_{J,\mathbf{m}} = f_{J,\mathbf{m}}, & \text{for } \mathbf{x}_{J,\mathbf{m}} \in \mathcal{V} \cap (0, 1)^2, \\ u_{J,\mathbf{m}} = g_{J,\mathbf{m}}, & \text{for } \mathbf{x}_{J,\mathbf{m}} \in \mathcal{V} \cap \Gamma_d, \\ n_1|_{\mathbf{x}_{J,\mathbf{m}}} \{\mathbf{D}_{x_1}^S \mathbf{u}\}_{J,\mathbf{m}} + n_2|_{\mathbf{x}_{J,\mathbf{m}}} \{\mathbf{D}_{x_2}^S \mathbf{u}\}_{J,\mathbf{m}} = t_{J,\mathbf{m}}, & \text{for } \mathbf{x}_{J,\mathbf{m}} \in \mathcal{V} \cap \Gamma_n, \end{cases} \quad (4.2)$$

where  $f_{J,\mathbf{m}}$ ,  $g_{J,\mathbf{m}}$ , and  $t_{J,\mathbf{m}}$  denote respectively the function values of  $f$ ,  $g$ , and  $t$  at  $\mathbf{x}_{J,\mathbf{m}}$ ,  $n_i|_{\mathbf{x}_{J,\mathbf{m}}}$  denotes a  $i$ -component of the unit vector normal to  $\Gamma_n$  at  $\mathbf{x}_{J,\mathbf{m}}$ . The term  $\{\mathbf{D}_{x_1 x_1}^S \mathbf{u}\}_{J,\mathbf{m}}$  denotes an approximation of the second derivative with respect

to  $x_1$  at the point  $\mathbf{x}_{J,\mathbf{m}}$  by means of finite differences described in section 3.6. Note that  $\{\mathbf{D}_{x_2x_2}^S \mathbf{u}\}_{\mathbf{x}_{J,\mathbf{m}}}$ ,  $\{\mathbf{D}_{x_1}^S \mathbf{u}\}_{\mathbf{x}_{J,\mathbf{m}}}$ , and  $\{\mathbf{D}_{x_2}^S \mathbf{u}\}_{\mathbf{x}_{J,\mathbf{m}}}$  are defined in an analogous way. Since the derivative approximation procedure is linear, it can be represented by a matrix-vector multiplication with  $\mathbf{u}$  as an input vector and  $\mathbf{D}^S$  as a so-called derivative matrix [128] (note that we drop the subscript from the notation when it is referred in a generic fashion, *i.e.* when no specific order of derivative is specified.). Precisely, as an example, the derivative matrix  $\mathbf{D}_{x_ix_i}^S$  for the second derivative with respect to  $x_i$  is defined by

$$\mathbf{D}_{x_ix_i}^S = \mathbf{D}_{x_ix_i} \tilde{\mathbf{E}} \mathbf{H}, \quad (4.3)$$

or equivalently by

$$\mathbf{D}_{x_ix_i}^S = \mathbf{D}_{x_ix_i} \tilde{\mathbf{P}}, \quad (4.4)$$

where  $\mathbf{H}$ , a  $N \times N$  matrix where  $N = \dim \Xi$ , represents the operator associated with the adaptive wavelet transform  $AFWT$ ,  $\tilde{\mathbf{E}}$  is a  $\tilde{N} \times N$  matrix, where  $\tilde{N} = \dim \tilde{\Xi}$ , that denotes the wavelet interpolation operator that determines interpolated values on an auxiliary grid with index set  $\tilde{\Xi}$  by means of appending  $\tilde{N} - N$  entries of zero values to a vector and subsequently applying the fast inverse adaptive wavelet transform (step 1 in Section 3.6), and  $\mathbf{D}_{x_ix_i}$  is a  $N \times \tilde{N}$  matrix that represents finite difference operator for second derivative with respect to  $x_i$  (step 2 in section 3.6). As a reminder, the auxiliary grid with index set  $\tilde{\Xi}$ , which is as a reminder the grid of points needed for finite differences, is larger than the considered grid, *i.e.*  $\tilde{N} \geq N$ . In (4.4),  $\tilde{\mathbf{P}}$  is a  $\tilde{N} \times N$  matrix that represents the wavelet interpolation operator associated with  $LAIWT$  (Algorithm 11). It is remarked that when  $\mathbf{V}$  is a full grid,  $\mathbf{D}^S$  is a conventional  $n^{\text{th}}$ -order finite difference

differentiation matrix. For the irregular grid, this matrix is effectively nothing but a certain finite difference derivative matrix with stencil varying for each grid point.

The problem (4.2) is just a linear system of equations composed of  $N_i = \dim \mathbf{V} \cap \Omega$  equations associated with the discretization at interior grid points,  $N_d = \dim \mathbf{V} \cap \Gamma_d$  equations enforcing Dirichlet boundary condition, and  $N_n = \dim \mathbf{V} \cap \Gamma_n$  equations approximating the Neumann boundary condition. It can be represented in the standard form

$$\mathbf{A}\mathbf{u} = \mathbf{f}, \quad (4.5)$$

where the matrix  $\mathbf{A}$  contains  $N_i$  rows extracted correspondingly from  $-(\mathbf{D}_{x_1x_1}^S + \mathbf{D}_{x_1x_1}^S) + c\mathbf{I}$ , and  $N_d$  and  $N_n$  rows representing the imposition of the Dirichlet and Neumann boundary conditions. The vector  $\mathbf{f}$  consists of the values of the right hand side of the equation at interior grid points and values of specified Dirichlet and Neumann condition at the boundaries.

The numerical solution of (4.1) thus becomes a matter of solving the linear system (4.5). It is noted that  $\mathbf{A}$  is a sparse matrix, *i.e.* the number of nonzero entries is proportional to  $N$ . In general,  $\mathbf{A}$  is non-symmetric, even when the corresponding continuous operator is symmetric and self adjoint. A vector resulting from a matrix-vector multiplication with the derivative matrix  $\mathbf{D}^S$  can be obtained easily and efficiently through the use of fast algorithms *AFWT* and *AIWT* (or *LAIWT*), and subsequently simple finite differences. Note this procedure requires only  $O(N)$  operations. In this way, a vector resulting of a matrix-vector multiplication with  $\mathbf{A}$  can be determined without the need of assembling matrix  $\mathbf{A}$ . Based on these facts, we can employ the Bi-Conjugate Gradient STABilized (BiCGSTAB), the Gen-

eralized Minimum RESidual (GMRES), or the Transpose-Free Quasi-Minimum Residual (TFQMR) (which are non-stationary iterative solvers relying on Krylov subspace [7, 114]) in solving the linear system arising from the discretization. These iterative solvers are applicable to non-symmetric linear systems. In addition, solvers based on Krylov subspace require users to supply only a result of matrix vector multiplication not a matrix in the solution process. It is noted that entries of  $\mathbf{A}$  having nonzero values can be determined exactly before-hand in an efficient manner. The nonzero pattern of  $\mathbf{A}$  is determined by the non-zero pattern of derivative matrices  $\mathbf{D}_{x_i x_i}^S$ , which can be obtained by examining associated finite difference stencils (in step 2.). More precisely, for grid points with their associated stencils containing no missing points, non-zero entries of corresponding rows are simply the index of points in stencils. For grid points with their associated stencils containing missing points, nonzero entries are the index of those point that participate in the stencil and the index of those points needed in the wavelet interpolation of the missing points. Note, however, that it is somewhat cumbersome to determine nonzero entries of the discretization matrix involving mix derivatives without first determining stencils for the mixed derivative. In the case where the system matrix is explicitly assembled, the transpose of  $\mathbf{A}$  is available, thus permitting the use of the Bi-Conjugate Gradient (BiCG) or Conjugate Gradient on Normal Equation (CGNE) for solving the system of equations.

So far, we have discussed the discretization based on a grid of the conventional MRA. For MRA- $d$ , the discretization on a given irregular grid of the MRA- $d$  replaces the PDE problem (4.1) with a problem similar to (4.2) (with the use of suitable notations). In this case,  $\{\mathbf{D}^S \mathbf{u}\}_{\mathbf{l}, \mathbf{m}}$  denotes a derivative approximation at the point  $\mathbf{x}_{\mathbf{l}, \mathbf{m}}$  by means of finite differences described in Section 3.7.1. In matrix



form, as an example,  $\mathbf{D}_{x_i x_i}^S$  is given by

$$\mathbf{D}_{x_i x_i}^S = \mathbf{E} \mathbf{H}_i \mathbf{D}_{x_i x_i} \tilde{\mathbf{E}}_i \mathbf{H} \quad (4.6)$$

where  $\mathbf{H}$ ,  $\mathbf{E}$  are  $N \times N$  matrices that represent respectively the MRA- $d$  fast wavelet transform operator (*AFWT2*) and the fast inverse wavelet transform operator (*AIWT2*),  $\mathbf{H}_i$  is a  $N \times N$  matrix denoting the wavelet transform with respect to the  $x_i$ -direction (*PAFWT2(1)*),  $\tilde{\mathbf{E}}_1$  is a  $\tilde{N} \times N$  matrix representing the procedure consisting of appending a vector with  $\tilde{N} - N$  entries of zero value and subsequently applying the inverse wavelet transform with respect to the  $i$ -direction on the augmented vector to obtain the intermediate function values on the auxiliary grid (step 1 in Section 3.6), and  $\mathbf{D}_{x_i x_i}$  is a  $N \times \tilde{N}$  matrix that represents the finite difference operator for the second derivative with respect to  $x_i$  (step 2 in Section 3.6). As in the MRA case, the system matrix  $\mathbf{A}$  of the MRA- $d$  approach is sparse and non-symmetric. There are fast algorithms for calculating a matrix-vector multiplication with  $\mathbf{A}$  (without having to know the explicit form of  $\mathbf{A}$ ), but there is no algorithm for the transpose of  $\mathbf{A}$ . It is also important to note that determining non-zero entries of this matrix is computationally expensive and should be avoided in practice.

It can be observed that the discretization can be applied straightforwardly to PDE problems with non-constant coefficient, owing to the collocation-like nature of the discretization. To this end, we note that the discussion and numerical results given in Section (3.7.1) and (3.6) indicate that this numerical discretization is consistent. It is noted that in order for the numerical solution to converge, in addition to being consistent, the discretization must be stable. To the author's knowledge, there exists no proof of stability of a finite-difference discretization

on a general irregular grid. However, experience with the use of the derivative approximation making use of finite difference schemes applied to specific problems shows that the numerical solutions converge to the corresponding exact solutions.

#### 4.2 Adaptive wavelet technique for elliptic PDE problems

In the previous section, we describe the procedure for numerically solving an elliptic PDE on a given irregular grid. Since wavelet amplitudes provide information on the regularity of the function, one can take advantage of this information to refine the computational grid accordingly. With these ingredients on hand, one can establish the algorithm for adaptive solution of elliptic problems without knowledge of where to refine before-hand [12, 70]. In this section, we consider such an algorithm for the linear second order elliptic problem

$$\mathcal{L}u = -\nabla \cdot (\mathbf{a}\nabla u) + \mathbf{b} \cdot \nabla u + cu = f \quad \text{in } \Omega = (0, 1)^d, \quad (4.7)$$

$$\mathcal{B}u = g \quad \text{on } \partial\Omega, \quad (4.8)$$

where  $\mathcal{B}$  represents boundary conditions and  $\mathcal{L}$  denotes a second order elliptic operator with possibly variable coefficients  $\mathbf{a} \in R^d \times R^d$ ,  $\mathbf{b} \in R^d$ , and  $c \in R$ .

The algorithm for the adaptive solution of the elliptic problem consists of solving the problem on one grid and then use wavelet amplitude of the approximate solution obtained as an indicator for constructing a new (and better) refined grid to be used for the subsequent approximate solution. The procedure is performed iteratively starting from a specific initial grid until an error tolerance is satisfied or until the computational grid stop changing—whichever come first. Such a process for the presented work is elaborated in Algorithm 16. It can be described roughly

---

**Algorithm 16** Elliptic solver with adaptive wavelet refinement

---

Given  $\varepsilon$ ,  $\varepsilon_{tol}$ ,  $it_{max}$  and initial grid  $\mathcal{V}^0$   
 $m = 0$   
Solve PDE problem on  $\mathcal{V}^m$  for  $\mathbf{u}^m$   
**repeat**  
     $AFWT(\mathbf{u}^m) \rightarrow \{\{u_{j_0, \mathbf{k}}^m\}, \{d_{l, \lambda}^m\}_{(l, \lambda) \in \Lambda^m}\}$   
    **# flag essential grid points**  
    Gather  $\Lambda = \{(l, \lambda) \in \Lambda^m : |d_{l, \lambda}^m| > \varepsilon\}$  and  
    **# add neighboring points to grid of essential points**  
    Determine the neighbor region  $\Lambda_{\mathcal{N}} = \bigcup_{(l, \lambda) \in \Lambda} \mathcal{N}_{l, \lambda}$  and establish  $\Lambda^{m+1} = \Lambda \cup \Lambda_{\mathcal{N}}$   
    **# new irregular grid**  
    Construct  $\mathcal{V}^{m+1} = \{\{\mathbf{x}_{l_0, \mathbf{k}}\}, \{\mathbf{x}_{l, \lambda}\}_{(l, \lambda) \in \Lambda^{m+1}}\}$   
  
    Solve PDE problem on  $\mathcal{V}^{m+1}$  for  $\mathbf{u}^{m+1}$   
     $AFWT(\mathbf{u}^{m+1}) \rightarrow \{\{u_{j_0, \mathbf{k}}^{m+1}\}, \{d_{l, \lambda}^{m+1}\}_{(l, \lambda) \in \Lambda^{m+1}}\}$   
  
    **# compute quantity used in stopping criteria**  
    Determine  $\boldsymbol{\lambda} = \{\lambda : (j_0 + m + 1, \lambda) \in \Lambda^{m+1}, |d_{j_0+m+1, \lambda}| > \varepsilon\}$   
    Calculate  $E^m = \|u^{m+1} - u^m\|$   
  
     $m = m + 1$   
**until**  $E^m \leq \varepsilon_{tol}$  or  $\boldsymbol{\lambda} = \emptyset$  or  $m > it_{max}$ 

---

as follows:

- (i) Compute the approximate solution of the PDE problem on an irregular grid;
- (ii) Flag the points with  $|d_{l, \lambda}| > \varepsilon$  (they are referred as *essential points*) for refinement. For each point flagged, an associated set of *neighboring points*,  $\mathcal{N}_{l, \lambda}$ , composing of surrounding points at the same and next level are established. The next computational grid consist of the union of the essential points and neighboring points;
- (iii) Repeat (i) and (ii) until stopping criteria are satisfied.

The set of neighboring points  $N_{l,\lambda}$  associated with  $\mathbf{x}_{l,\lambda}$  (note that  $\mathbf{x}_{l,\lambda=(\mathbf{e},\mathbf{k})} = \mathbf{x}_{l+1,2\mathbf{k}+\mathbf{e}}$ ) is defined by

$$\mathcal{N}_{l,\lambda=(\mathbf{e},\mathbf{k})} = \{\mathbf{x}_{l+2,\mathbf{r}} : \mathbf{r} \in \mathbf{k}_{l+2}^0 \text{ and } \|\mathbf{r} - (4\mathbf{k} + 2\mathbf{e})\|_\infty < 2P\}, \quad (4.9)$$

where an integer  $P$  is a parameter controlling the size of region to be refined. This strategy includes  $(4P + 1)^d - 1$  points surrounding one essential point and in practice there is no benefit in using  $P > 1$ . Different strategies can be also employed in defining  $N_{l,\lambda}$ . Another possibility is to use the following strategy

$$\mathcal{N}_{l,\lambda} = \{\mathbf{x}_{l+2,\mathbf{r}} : \mathbf{r} \in \mathbf{k}_{l+2}^0 \text{ and } |r_i - (4k_i + 2e_i)| < 2P, i = 1, \dots, d\}, \quad (4.10)$$

which adds  $d(4P + 1) - 1$  points surrounding one essential points. Using  $\mathcal{N}_{l,\lambda}$  obtained from (4.10) results in an irregular sparse grid with larger sparseness. Figure 4.1 illustrates sets of neighboring points defined by the two strategies described above.

In this work, the computation is terminated when  $\|u^{n+1} - u^n\|_{\mathbf{V}^n,\infty}$  is less than a user prescribed tolerance  $\varepsilon_{tol}$  and when the computational grid converges<sup>1</sup>. Here,  $u^n$  and  $\mathbf{V}$  represent respectively the approximate solution and irregular grid at  $n^{\text{th}}$ -iteration. Owing to the fact that wavelet amplitudes decay in a cone-like structure in location scale space, convergence of computational grid is declared when there is no point on a newly added highest level with wavelet amplitude greater than  $\varepsilon$ . In addition to these simple stopping criteria, the calculation is also terminated when the number of iterations reaches a user-specified maximum ( $it_{max}$  in Algorithm 16).

---

<sup>1</sup>One must not confuse this terminology with a grid convergence used in the realm of the so-called verification and validation. Here we mean that the computational grid stops changing.

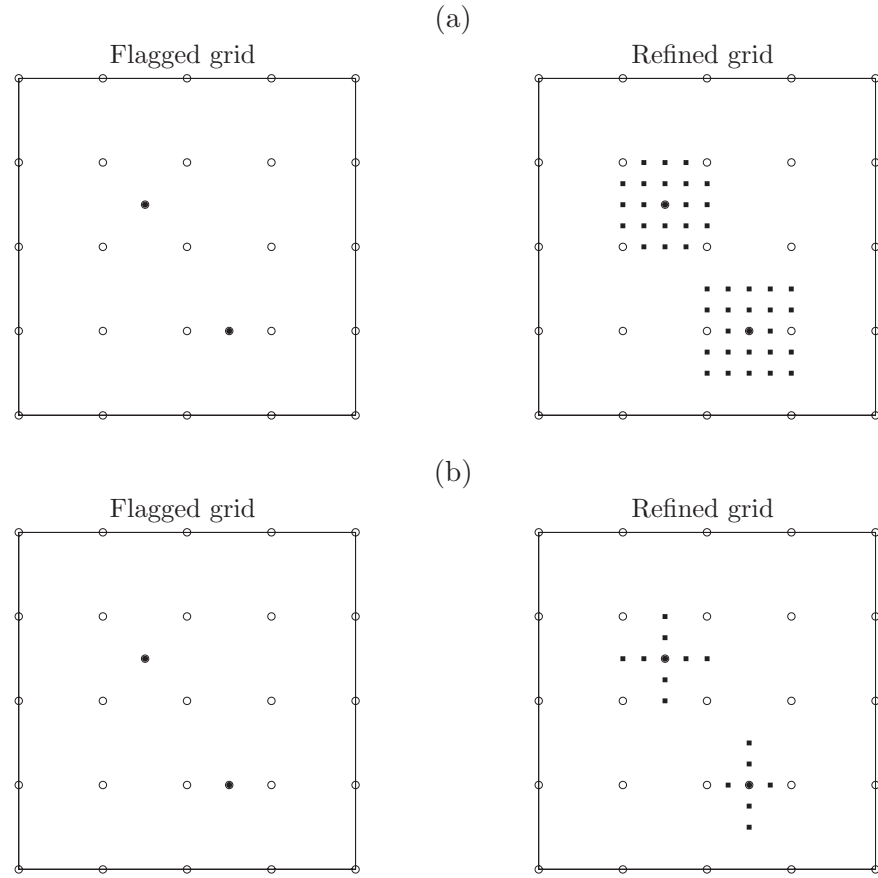


Figure 4.1: Two refinement strategies with  $P = 1$ : (a) strategy (4.9) (b) strategies (4.10);  $\bullet$  grid points flagged for refinement.  $\blacksquare$  neighboring points.

To this end, let us note that a similar adaptive algorithm based on MRA- $d$  can be obtained by simply substituting appropriate notations into Algorithm 16. In this case, convergence of the computational grid and the strategy for defining  $\mathcal{N}_{\mathbf{l},\mathbf{m}}$  are also modified to suit the nature of MRA- $d$ . More precisely, the step length used to define neighboring points in the  $i$ -direction depends on level  $l_i$  ( $i^{\text{th}}$ -component of level index  $\mathbf{l}$ ) as opposed to use the same step length in every direction. Convergence of computational grid in this case is determined by examine wavelet amplitudes on a newly added highest level in all  $i$ -directions.

We remark that the technique discussed is paradigmatically similar to other conventional adaptive refinement schemes [105]. The major difference lies on the fact that it is point-oriented not cell or element-oriented. Subsequently, the grid adaption procedure is inexpensive since it is simply a matter of adding or removing points (without any conformity requirement to be fulfilled as in cell- or element-oriented methods).

#### 4.3 Remark on preconditioning

In the next section, we apply the adaptive method to solve the Poisson equations on a square or a cubic domain. As mentioned in Section 4.1, we use a non-stationary iterative method (*e.g.* BiCGSTAB, GMRES, or TFQMR) to solve the resulting linear system of equations. Such system has a condition number  $\kappa$  that grows at least as  $O(4^J)$ , where  $J$  is the finest level of resolution. Solving such a system to a given accuracy with a modern non-stationary solver will require  $O(\sqrt{\kappa})$  [4, 7]. As a consequence, one can expect poor performance of iterative solvers when the value of  $J$  is large. To improve the performance, preconditioning must be used.

For the MRA- $d$  approach, Koster [90] finds that, with a Jacobi (diagonal) preconditioner  $\mathbf{M}$  with diagonal entries  $\sum_{i=1}^d 4^{l_i}$ , the number of iterations required to solve the preconditioned system  $\mathbf{HAEM}^{-1}\tilde{\mathbf{u}} = \tilde{\mathbf{f}}$ , where  $\tilde{\mathbf{u}} = \mathbf{MHu}$  and  $\tilde{\mathbf{f}} = \mathbf{Hf}$  (and, as a reminder,  $\mathbf{A}$  is the MRA- $d$  discretization matrix,  $\mathbf{H}$  and  $\mathbf{E}$  are respectively the MRA- $d$  forward and inverse wavelet transform) depends mildly on  $J$  for  $p \geq 4$  (despite the fact that his estimate, for the setting that assumes the full grid and periodic boundary condition, shows that  $\kappa \sim O((J2^J)^{1/2})$ , a relative rapid growth). Note that this technique may be viewed as a variant of a so-called multi-level preconditioning [48] used in a Galerkin technique. This type of preconditioning relies on the fact that the basis must be stable in Sobolev-spaces. In this work, we use two different types of preconditioners for the discretization matrix arising from the MRA approach. Since the discretization matrix is sparse and its non-zeros entries can be assembled explicitly, one preconditioner that we adopt is the Incomplete LU-factorization (ILU) with drop tolerance [114]. The ILU-preconditioner is quite efficient in reducing the number of iterations; however, the additional memory for storage is required (the amount usually can not be estimated a priori). The other preconditioner that we use borrows the idea from the Jacobi preconditioning of the MRA- $d$  discretization matrix as described above. In this particular case, the preconditioner  $\mathbf{M}$  is a diagonal matrix with entries  $d4^l$ . The system to be solved is then given by

$$\mathbf{HAEM}^{-1}\tilde{\mathbf{u}} = \tilde{\mathbf{f}}, \quad (4.11)$$

where  $\tilde{\mathbf{u}} = \mathbf{MHu}$  and  $\tilde{\mathbf{f}} = \mathbf{Hf}$  (and  $\mathbf{H}$  and  $\mathbf{E}$  represent the MRA forward and inverse wavelet transforms ( $AFWT$ ,  $AIWT$ ) respectively). Note that in this case, the additional memory storage is one vector of size  $N$  is needed and there is

not necessary to assemble the system matrix. Numerical experiments indicates that with the Jacobi preconditioner the number of iterations required in the solver depends mildly on  $J$ . Note that the numerical results presented in the next section are obtained by using this particular preconditioner.

#### 4.4 Numerical Experiments

In this section, we apply the adaptive algorithm to solve the 2- and 3-D model problems having mainly the Laplacian as operators in order to assess its performance in term of and accuracy. These test problems are linear PDEs and they are chosen such that their exact solution are available. Note that in this study we focus our attention main on the assessment of the method with MRA.

##### 4.4.1 Adaptive method using the MRA approach

**Test 1:** Consider the 2-D Poisson equation with Dirichlet boundary conditions:

$$-\nabla^2 u = f \quad \text{in } \Omega = (0, 1)^2, \quad (4.12)$$

$$u = g \quad \text{on } \partial\Omega, \quad (4.13)$$

where  $f(\mathbf{x})$  and  $g(\mathbf{x})$  are chosen such that the solution of the problem is

$$u(\mathbf{x}) = \tan^{-1} \left[ \frac{x_2}{x_1 + 0.01} \right]. \quad (4.14)$$

Note that the solution is harmonic, *i.e.*  $\nabla^2 u = 0$  and decays very sharply near  $\mathbf{x} = (-0.01, 0)$  (see Figure 4.2). If the domain were to be extended to  $x_1 = -0.01$ , there would be a jump in the boundary condition.



The numerical calculations are performed with different values of  $p$  (order of wavelet),  $n$  (order of finite differences), and threshold parameter  $\varepsilon$ . The coarsest level is set to  $j_0 = 3$  ( $9 \times 9$  grid points) in cases where  $p = 4$  and 6 and to  $j_0 = 4$  ( $17 \times 17$  grid points) in cases where  $p = 8$  and 10. The neighboring set is defined through (4.9) with  $P = 1$ . The maximum allowable resolution  $J - j_0$  is set to a large value and we check that the maximum level of resolution is never reached (*i.e.* no essential point exists at this level) by the adaptive algorithm. Subsequently, the accuracy of numerical solution is controlled primarily by the threshold parameter  $\varepsilon$ . We use  $\|u^{m+1} - u^m\|_{\mathbf{V}^{m+1}, \infty} \leq \varepsilon_{tol}$  with  $\varepsilon_{tol} = 15\varepsilon$  for the stopping criterion (see also the other two simple stopping criteria in Algorithm 16). As a reminder,  $u^m$  represents the numerical solution obtained with  $\mathbf{V}^m$ , the refined grid at the  $m^{\text{th}}$ -iteration.

Figure 4.2 shows for the numerical solution obtained by the adaptive method with  $p = 6$ ,  $n = 4$  and  $\varepsilon = 10^{-4}$ . It can be seen that grid points are concentrated in vicinity of the lower left corner, the area close to the singularity. The sequence of adaptively refined grids  $\mathbf{V}^m$  generated by the algorithm is depicted in Figure 4.3 for the first five iterations. It can be observed that the resolutions are added progressively and locally to the area where the solution changes sharply.

Figure 4.4 (a)-(d) shows the log-log plots of errors in the  $L_{\mathbf{V}, \infty}$ -norm versus the number degrees of freedom with different  $p$  and  $n$  as a result of varying the threshold parameter  $\varepsilon$ . Plots of error from discretization on a uniform grid (in this case, the method corresponds to a conventional higher order finite difference method) are included in the figure. It can be seen that, in this test problem, the discretization on uniform grids performs rather poorly. With the adaptive technique, the solution of similar accuracy is obtained with a substantially fewer

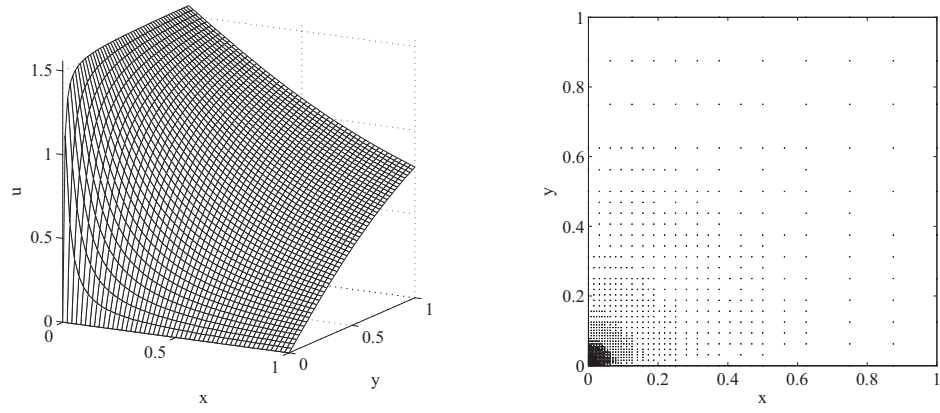


Figure 4.2: Solution of Test 1 (left) and the adaptively refined irregular grid (right) for  $p = 6$ ,  $n = 4$  and  $\varepsilon = 1 \times 10^{-4}$ .

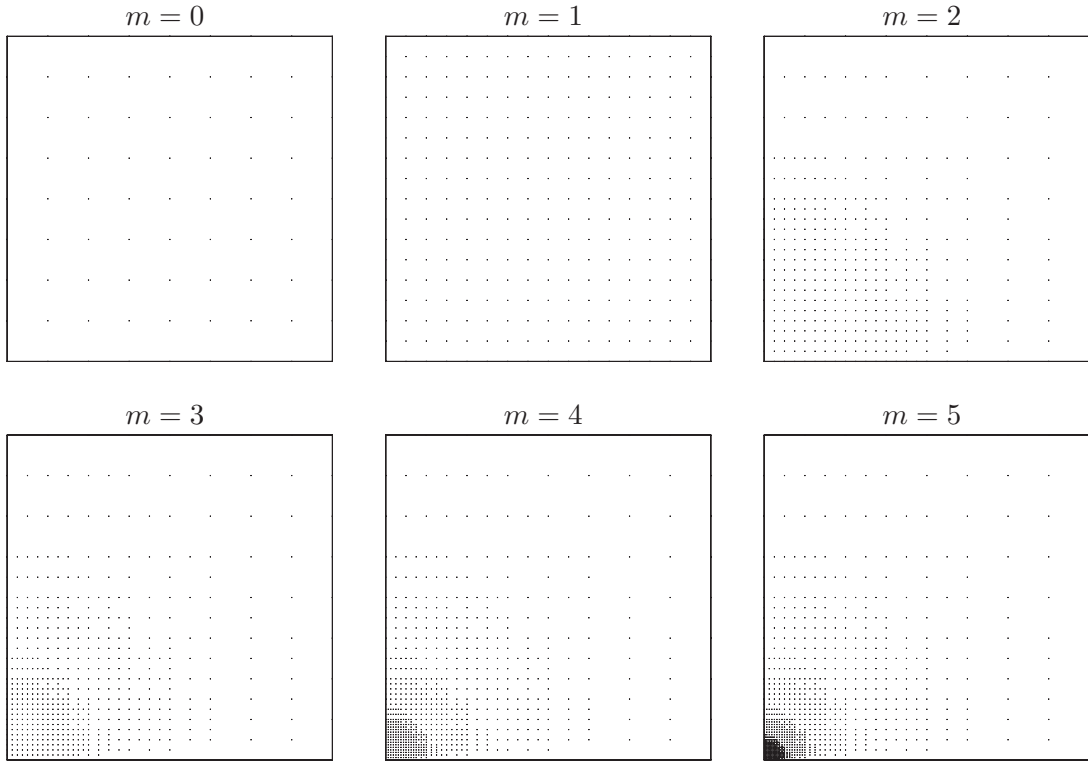


Figure 4.3: Sequences of adaptively refined irregular grids  $\mathbf{V}^m$  for  $m = 0, \dots, 5$  in Test 1 for  $p = 6$ ,  $n = 4$  and  $\varepsilon = 5 \times 10^{-4}$ .

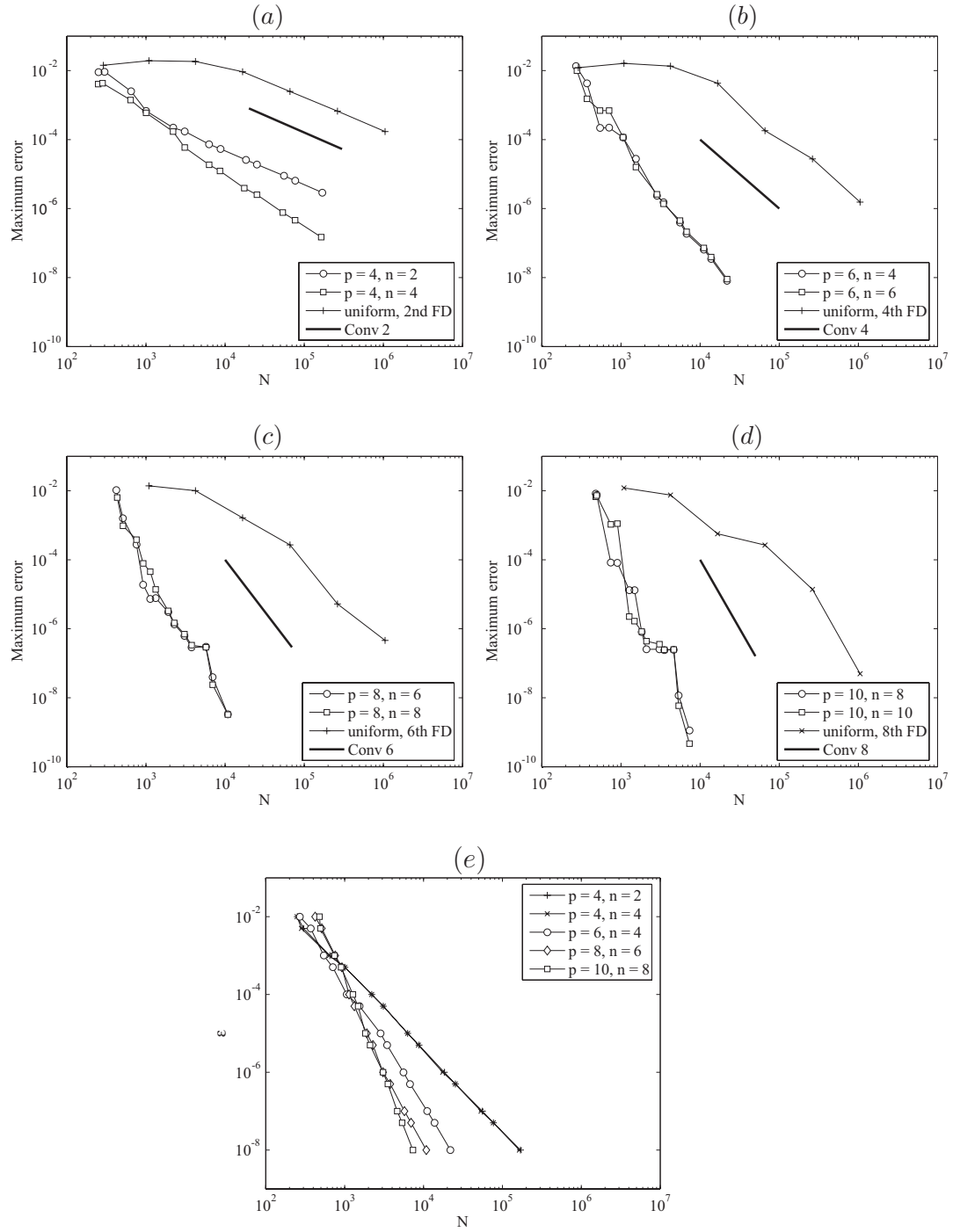


Figure 4.4: Test 1: (a)-(d)  $L_{\mathbf{v},\infty}$ -error as a function of the number of grid points with different  $p$ ,  $n$ , and  $\epsilon$  and using 2<sup>nd</sup> to 8<sup>th</sup> finite difference methods, (e) the number of grid points as a result of varying the threshold value  $\epsilon$ ; Bold solid lines represent the expected convergence,  $\|\cdot\|_{\mathbf{v},\infty} \sim N^{\text{Conv}/d}$

number of grid points. For instance, the 4<sup>th</sup> order finite difference uniform discretization requires approximately  $10^5$  points to produce the numerical solution with accuracy of approximately  $10^{-4}$ ; however, the similar accuracy is achieved with approximately only  $10^3$  points using the adaptive method with  $p = 6$  and  $n = 4$ . Except for the case with  $p = 4$ , the results indicate that there is no significant gain in using the finite difference scheme of order  $n \geq p$ . The order of convergence with respect to  $N^{-1/2}$  is approximately 2.01 for the adaptive method with  $p = 4$  and  $n = 2$  and 2.98 for  $p = 4$  and  $n = 4$ . For other combinations of  $p$  and  $n$ , the adaptive method exhibits rates of convergence of approximately  $p$ , two orders higher than  $O(N^{\min(p-2, n)/2})$  as given by (3.52). Figure (4.4) (e) shows the number of grid points  $N$  generated by the adaptive algorithm versus the threshold values  $\varepsilon$ . We note that  $N$  generated by the adaptive algorithm as  $\varepsilon$  is varied agrees well with (3.32), *i.e*  $N$  is of  $O(\varepsilon^{-d/p})$ . The slopes of plots are  $-2.10$  for  $p = 4$ ,  $-3.11$  for  $p = 6$ ,  $4.41$  for  $p = 8$  and  $4.99$  for  $p = 10$  respectively.

**Test 2:** Consider the Poisson problem (4.12)-(4.13) for which  $f$  and  $g$  are selected such that the exact solution is

$$u(\mathbf{x}) = \frac{x_c(x_1 + x_c)}{(x_1 + x_c)^2 + (x_2 - 0.4)^2}, \quad x_c > 0 \quad (4.15)$$

Note that  $u$  has a singularity at  $\mathbf{x} = (-x_c, 0.4)$ , which although it lies outside the domain, it is close to the boundary. As in Test 1, a uniform discretization is impractical for a small value of  $x_c$ .

We consider the problem with  $x_c = 0.25, 0.1, 0.01$ , and  $0.005$ . The adaptive method with different  $p$  and  $n$  is tested for different values of  $\varepsilon$  ranging from  $10^{-2}$  to  $10^{-9}$ . Note that the coarsest level is set to  $j_0 = 3$  for  $p = 4$  and  $6$  and  $j_0 = 4$  for  $p = 8$ . Figure 4.5 provide the numerical solution for  $x_c = 0.01$  and the adaptively

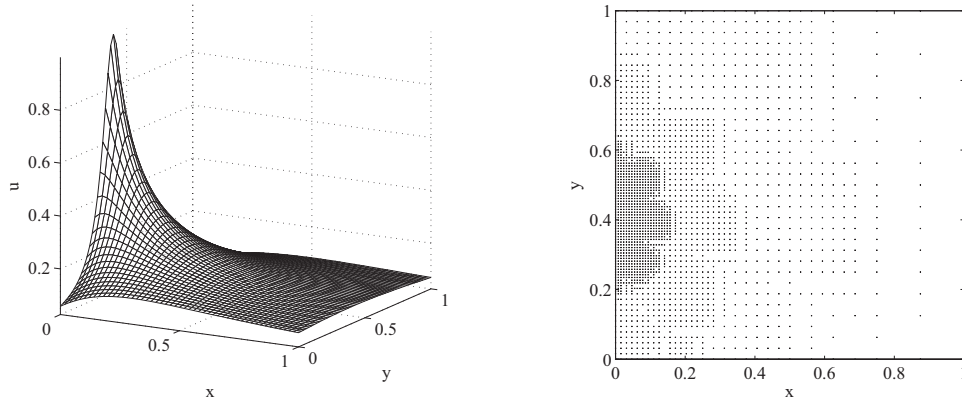


Figure 4.5: Solution of Test 2 (left) and the adaptively refined irregular grid (right) generated by the adaptive method with  $p = 4$ ,  $n = 4$ , and  $\varepsilon = 10^{-4}$ .

refined irregular grid produced with  $p = 4$ ,  $n = 4$ , and  $\varepsilon = 10^{-4}$ . Notice that the grid points are concentrated in the proximity of the singularity.

Figure 4.6 shows the log-log plots of the  $L_{\mathbf{v},\infty}$ -errors of the numerical solutions for four different values of  $x_c$  as functions of the number of grid points  $N$  required by the adaptive algorithm as a result of varying the threshold parameter. The dependence of  $N$  on the threshold values  $\varepsilon$  are shown in Figure 4.7. It can be observed that the relationship between  $N$  and  $\varepsilon$  is in good agreement with (3.32). The numerical solution obtained by the adaptive method with  $p = 6$  and  $n = 4$  and with  $p = 8$  exhibits approximately order  $p$  convergence with respect to  $N^{-1/2}$ . Such a rate is two orders higher than that predicted by (3.52). Indeed, in these cases, the error of the solutions (in the pointwise maximum norm) is of the same order as the threshold value  $\varepsilon$  (the linear curve fitting in the least square sense of the log-log plots of errors and threshold values indicate that the slopes of the plots are approximately 1 and the constants are also of order one). Without providing precise numbers, we note that this can also be seen easily by inspecting

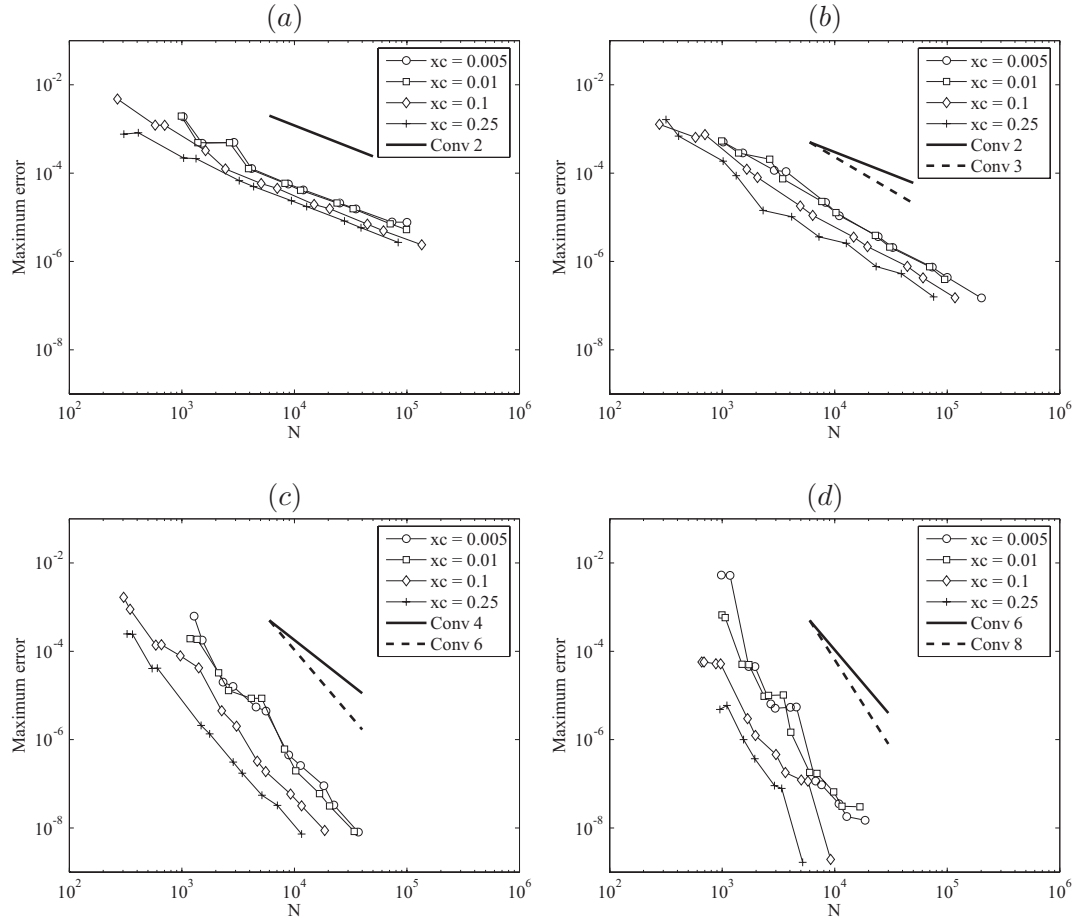


Figure 4.6: Test 2:  $L_{\mathbf{V},\infty}$ -errors as functions of the number of grid points as a result of varying  $\varepsilon$ ; (a)  $p = 4$  and  $n = 2$ , (b)  $p = 4$  and  $n = 4$ , (c)  $p = 6$  and  $n = 4$ , and (d)  $p = 8$  and  $n = 7$ .

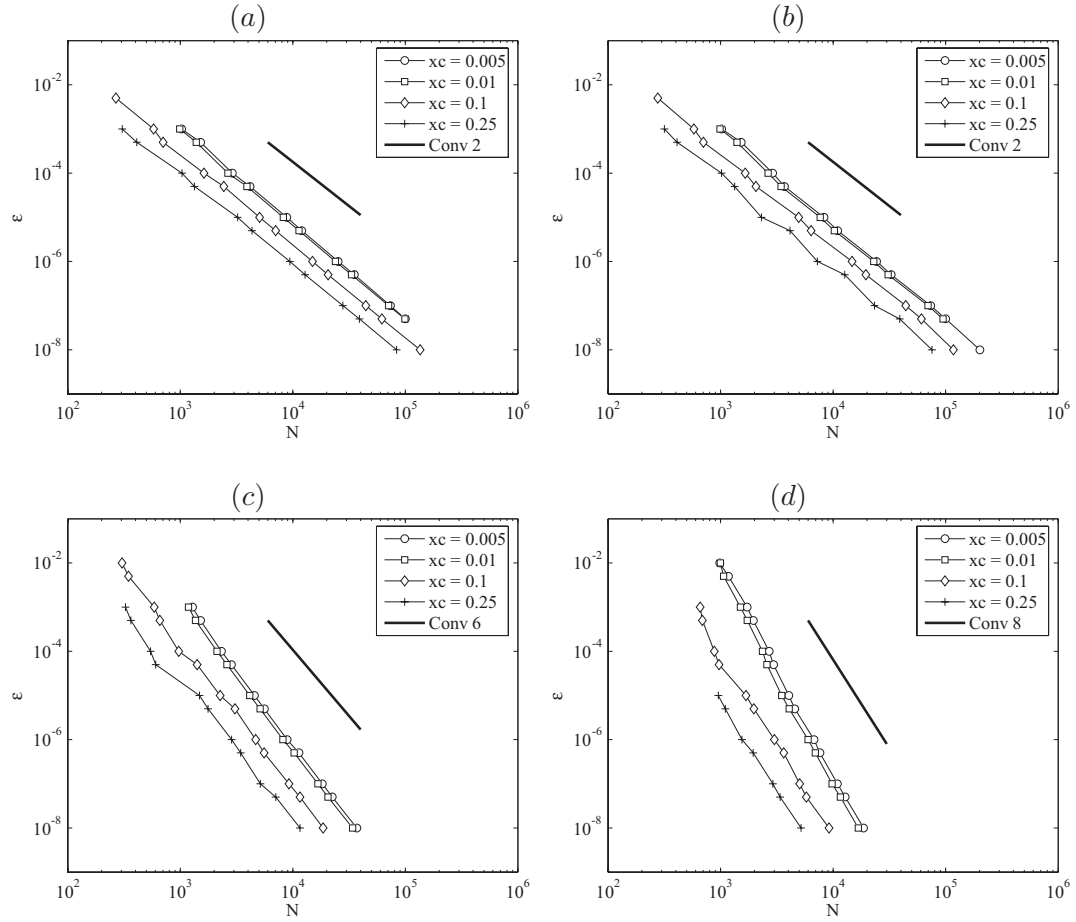


Figure 4.7: Test 2: the number of grid points produced by the adaptive algorithm as a result of varying  $\varepsilon$ ; (a)  $p = 4$  and  $n = 2$ , (b)  $p = 4$  and  $n = 4$ , (c)  $p = 6$  and  $n = 4$ , and (d)  $p = 8$  and  $n = 7$ .

Figure 4.6 (c) and (d) together with Figure 4.7 (c) and (d). The rate of convergence for the method with  $p = 4$  and  $n = 3$  is approximately 2 and it is consistent the estimate given in (3.52). For  $p = 4$  and  $n = 4$ , the rate of convergence is of order 3 which is one order higher than of the prediction. In theses cases, the method yields errors that are larger than the threshold values  $\varepsilon$  (the ratio of the error to the threshold value increases as the threshold values are decreased; more precisely, the error is proportional approximately to  $O(\varepsilon^{0.5})$  and to  $O(\varepsilon^{0.7})$  for  $p = 4, n = 3$  and  $p = 4, n = 4$  respectively).

Results from these test cases imply that the adaptive method yields solutions that conform reasonably with the estimate (3.52) and, in the best possible scenario, yields solution having an error of the order of threshold values.

**Test 3:** Consider the Helmholtz problem

$$-\nabla^2 u + cu = f \quad \text{in } \Omega = (0, 1)^2, \quad (4.16)$$

$$u = g \quad \text{on } \partial\Omega, \quad (4.17)$$

with  $c = 8$  and  $f(\mathbf{x})$  and  $g(\mathbf{x})$  chosen in such a way that the exact solution  $u(\mathbf{x})$  is given by

$$u(\mathbf{x}) = \frac{1}{5} \sin(3\pi x_1) \sin(2\pi x_2) + \exp \left[ -\alpha((x_1 - x_{01})^2 + (x_2 - x_{02})^2) \right] \quad (4.18)$$

where  $\alpha = 250$ ,  $x_{01} = 3/5$ , and  $x_{02} = 2/5$ .

The adaptive method with different  $p$  and  $n$  is applied in the solution of this problem. The coarsest level is set to  $j_0 = 3$  (9 points in each direction) for  $p = 4$  and 6 and  $j_0 = 4$  (17 points in each direction) for  $p = 8$ . Threshold values  $\varepsilon$  ranging from  $1 \times 10^{-2}$  to  $5 \times 10^{-9}$  are employed in the numerical calculations.



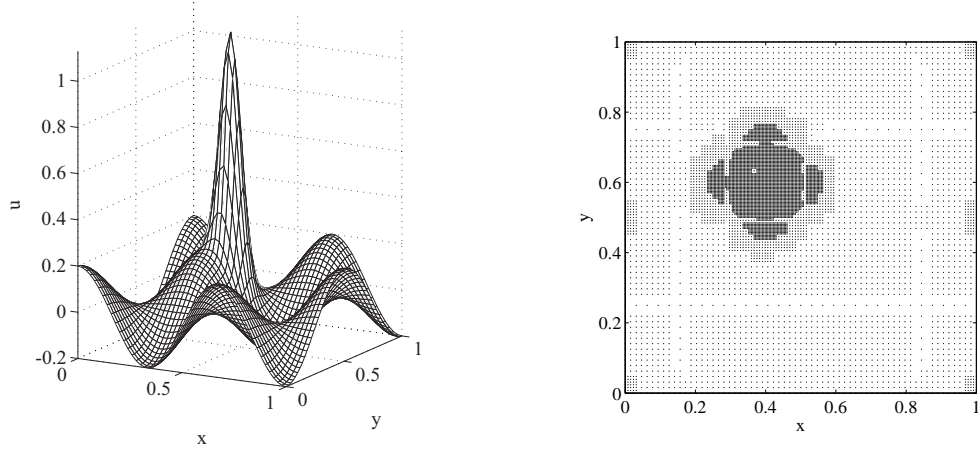


Figure 4.8: Solution of Test 3 (left) and the adaptively refined irregular grid (right) generated with  $p = 4$ ,  $n = 4$ , and  $\varepsilon = 10^{-4}$ .

Figure 4.8 shows, for example, the numerical solution of this problem and the adaptively refined irregular grid generated with  $p = 4$ ,  $n = 5$  and  $\varepsilon = 5 \times 10^{-5}$ .

Figure 4.9 shows the error of the numerical solution in the discrete maximum-norm as a function of the number of grid points on log-log scales for different  $p$  and  $n$  as the threshold value is varied. The figure includes also the dependence of  $N$  on the threshold values  $\varepsilon$ . By inspecting the slope of the plot, the trend of convergence is overall similar to that observed in the previous test problem. In this particular problem, the rate of convergence with respect to  $N^{-1/2}$  with  $p = 4$  and  $n = 2$  is approximately 2 and for  $p = 4$  and  $n = 4$  is approximately 3.4. For  $p = 6$  and  $n = 4$ , the rate of convergence is approximately 4.4 and for  $p = 6$  and  $n = 6$  it is about 5.4. The numerical solutions for  $p = 8$  and  $n = 6$ , and  $p = 8$  and  $n = 8$  are accurate with errors of  $\varepsilon$ . In these cases, the rates of convergence are approximately 9.2 and 10.6 with  $p = 8$  and  $n = 6$  and  $p = 8$  and  $n = 8$ , respectively. It is remarked that although the numerical results for  $p = 6$  and  $p = 8$  exhibit higher convergence rate when equal orders  $p$  and  $n$  are used, values

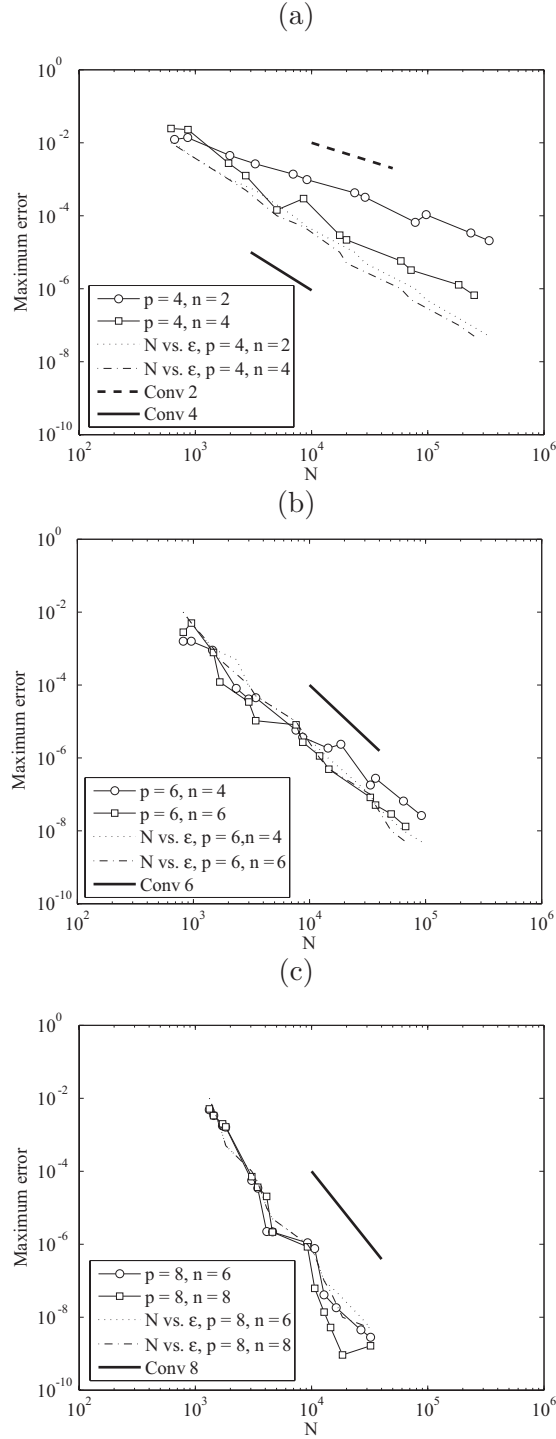


Figure 4.9: Test 3:  $L_{\mathbf{V},\infty}$ -errors as functions of the number of grid points as a result of varying  $\epsilon$ ; (a)  $p = 4$ ,  $n = 2$  and 4; (b)  $p = 6$ ,  $n = 4$  and 6; (c)  $p = 8$ ,  $n = 6$  and  $n = 8$ .

of errors (in the max-norm) are however approximately of the same order as when  $n = p - 2$  is used.

Results of this test problem indicate again to some degree that the adaptive method yield the solution that conforms reasonably well with the estimate given in (3.52). Note that (3.52) implies that when using equal order  $p$  and  $n$ , one can expect to lose two orders of accuracy comparing with the order of finite differences used. However, the numerical solutions above provides evidence that for some particular cases such loss may not be as large as that of the estimate.

**Test 4:** In this test, we consider a 3-D Poisson equation on a unit cube with Dirichlet boundary conditions,

$$\begin{aligned}\nabla^2 u &= f \quad \text{in } \Omega = (0, 1)^3 \\ u &= g \quad \text{on } \partial\Omega.\end{aligned}$$

where  $f(\mathbf{x})$  and  $g(\mathbf{x})$  are chosen in a such way that the exact solution is given by

$$u(\mathbf{x}) = [\delta^2 + (x_1 - x_{01})^2 + (x_2 - x_{02})^2 + (x_3 - x_{03})^2]^{1/2}. \quad (4.19)$$

In the following numerical experiments, we consider the problem with  $\delta = 0.075$  and  $x_{01} = x_{02} = x_{03} = 2/5$ . Note that  $\delta$  is a regularizing parameter (if  $\delta$  were to be zero, the derivative of  $u(\mathbf{x})$  would contain a singularity at  $\mathbf{x} = (x_{01}, x_{02}, x_{03})$ ). Here, the adaptive method with  $p = 4$  and  $6$  is used to solved the problem. The coarsest level is set to  $j_0 = 3$  (9 in each direction). In the calculations, we vary the values of  $\varepsilon$  from  $10^{-3}$  to  $5 \times 10^{-8}$ . The solution of the problem and the adaptively refined grid are shown in Figure 4.10 for the method with  $p = 6$ ,  $n = 4$  and  $\varepsilon = 5 \times 10^{-6}$ . Note that the adaptive algorithm provides high resolution in the

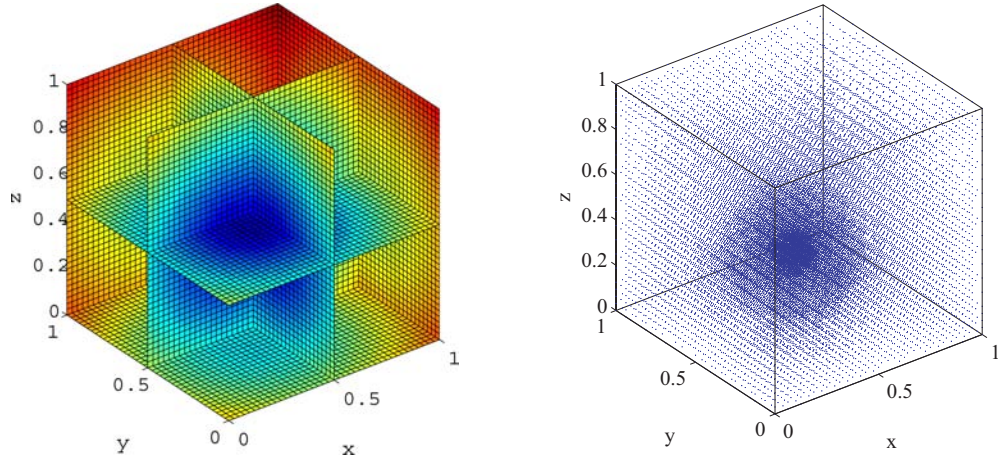


Figure 4.10: Solution of Test 4 (left) and the adaptively refined irregular grid (right) generated by the adaptive method with  $p = 6$ ,  $n = 5$ , and  $\varepsilon = 5 \times 10^{-6}$ .

vicinity of the quasi-singular point.

The error in the numerical solution in the discrete maximum norm versus the number of grid points,  $N$ , is plotted in Figure 4.11 for different  $p$  and  $n$ . The dependence of  $N$  on  $\varepsilon$  for the different cases are also included in this figure. The rate of convergence with respect to  $N^{-1/3}$  is approximately 3.1 for  $p = 4$  and  $n = 4$  and is approximately 6 for  $p = 6$  and  $n = 4$  and  $p = 6$  and  $n = 6$ . Such rates are higher than  $O(N^{\min(p-2,n)/2})$  predicted by (3.52). Note that in test cases with  $p = 6$ , the error in the numerical solution is indeed of the same order as the threshold value.

#### 4.4.2 Adaptive method with MRA- $d$ approach

**Test 5:** To test the implementation of the adaptive method using MRA- $d$  approach, we consider the 2-D Poisson problem (4.12) and (4.13) where  $f(\mathbf{x})$  and  $g(\mathbf{x})$  are chosen such that

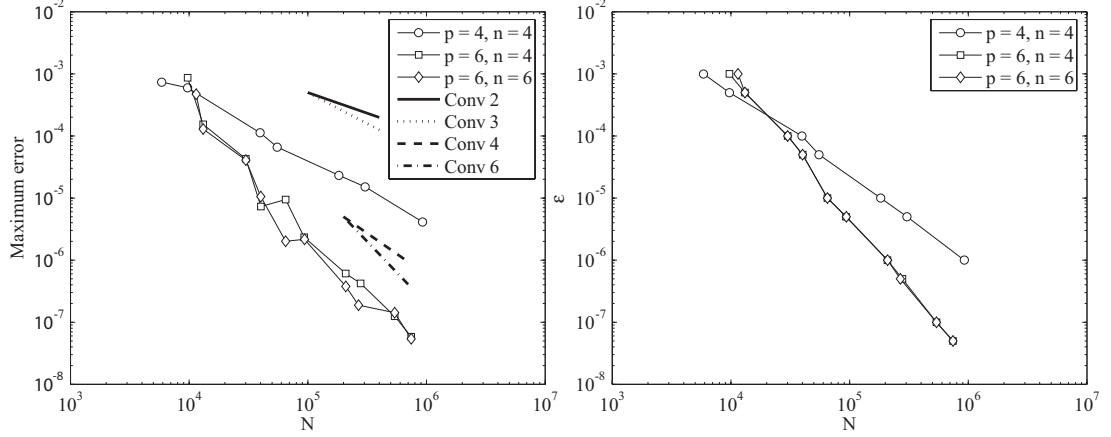


Figure 4.11: Test 4: (a)  $L_{\mathbf{V},\infty}$ -error as a function of number of grid points,  $N$ , as a result of varying  $\epsilon$  and (b)  $\epsilon$  versus  $N$ .

$$u(\mathbf{x}) = \left[ \delta^2 + \left( x_1 - \frac{3}{9} \right)^2 + \left( x_2 - \frac{4}{9} \right)^2 \right]^{1/4} \quad (4.20)$$

as a test problem. Such a problem is used in [89] to test the originally proposed MRA- $d$ . We use this testing to verify the implementation of MRA- $d$  method in the current computer code. It should be noted that the fact although we follow in principle the idea proposed in [70, 88, 89], the detail implementation (due to somewhat insufficient information *e.g.* how missing points participating the finite-differences are handled, choices of FD stencils, how the neighboring points are added, and *etc.*) in the current work may not be exactly as implemented in [89]. Consequently, we do not expect to reproduce exactly the specific results given in [89]. However, we expect numerical results to behave in the similarly.

In numerical calculations, we consider the problem with three different values of  $\delta$ , namely  $\delta = 10^{-1}$ ,  $10^{-2}$ , and  $10^{-3}$  and different values of  $p$ ,  $n$  and  $\epsilon$ . The set of neighboring points associated with an essential point is given by (4.9) (with

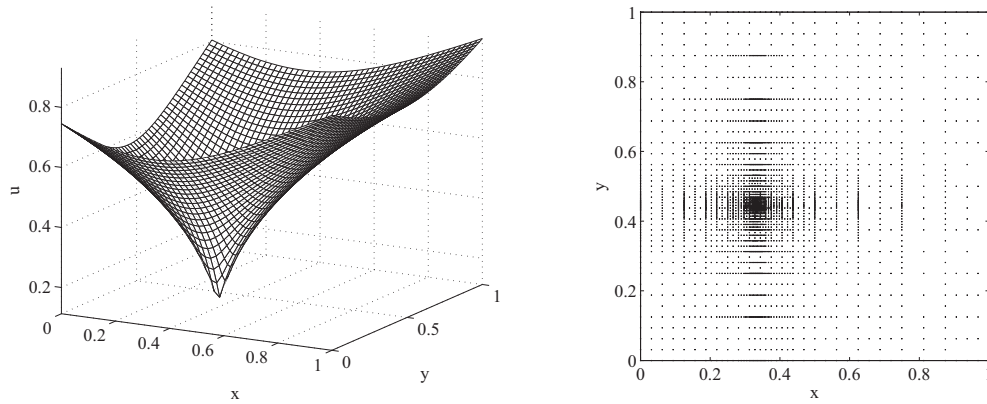


Figure 4.12: Solution of Test 5 (left) and the adaptively refined irregular grid (right) generated by the adaptive method based on MRA- $d$  with  $p = 4$ ,  $n = 4$ , and  $\varepsilon = 10^{-4}$ .

accordingly modified step length in each direction) with  $P = 1$ . The values of threshold parameter  $\varepsilon$  are varied from  $1 \times 10^{-2}$  to  $5 \times 10^{-8}$ .

Figure 4.12 shows for example the numerical result for the problem with the regularizing parameter  $\delta = 10^{-2}$  and the adaptively refined grid produced automatically by the adaptive method with  $p = 4$ ,  $n = 4$  and  $\varepsilon = 10^{-4}$ . Note that grid points are clustered in the proximity of the near-singularity. The log-log plots of errors in the discrete maximum-norm versus number of grid points  $N$  as a result of varying the threshold parameter  $\varepsilon$  are shown in Figure 4.13. The figure includes also the plots of number of grid points  $N$  versus  $\varepsilon$ . The plots indicate the convergence of numerical solution. In this particular problem, the overall rate of convergence for the adaptive method with  $p = 4$  and  $n = 4$  is approximately 4.4 and with  $p = 6$  and  $n = 6$  is approximately 6.4. However, it should be noted that the adaptive method displays somewhat irregular convergence behavior. Such behavior is indeed also reported in the numerical experiments of Koster [89] (see [89], pp. 107, for comparison).

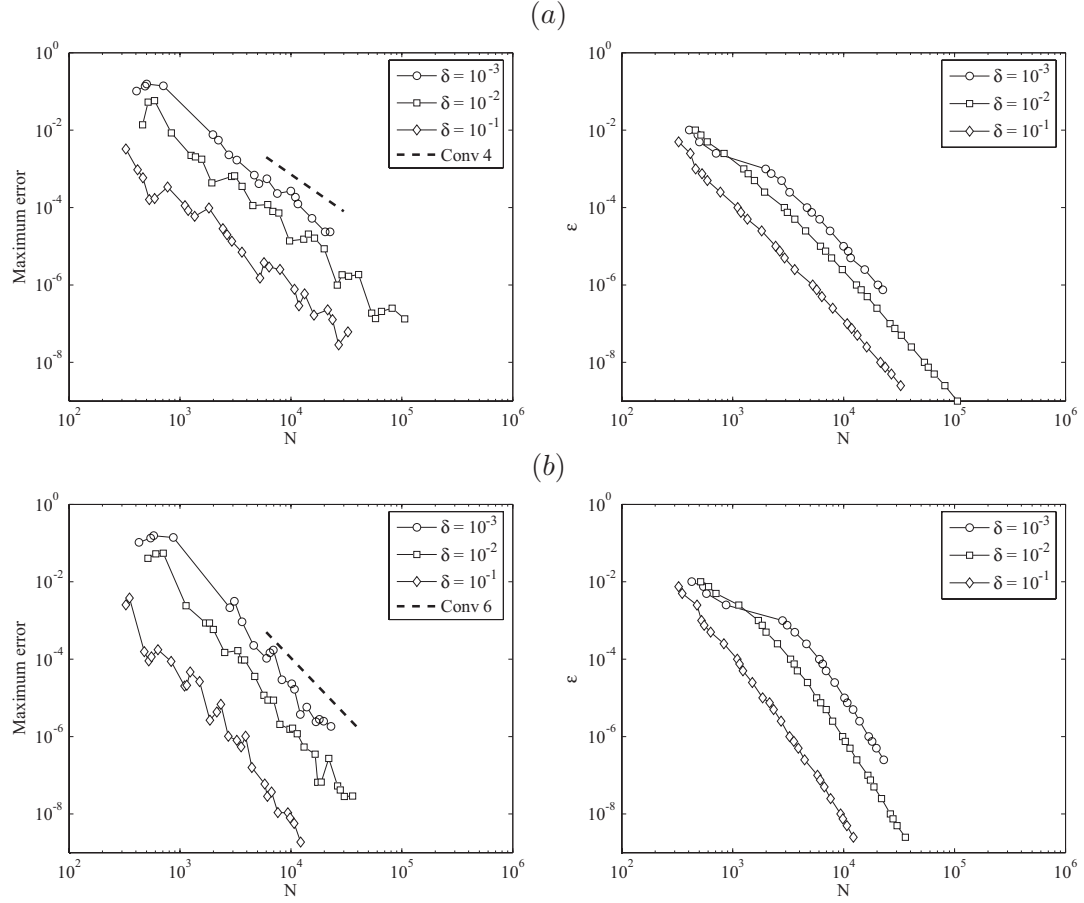


Figure 4.13: Test 5:  $L_{\mathbf{V},\infty}$ -errors as a function of the number of grid points  $N$  as a result of varying  $\varepsilon$  (left) and the dependence of  $N$  on  $\varepsilon$  (right) using MRA- $d$ ; (a)  $p = 4$  and  $n = 4$ , (b)  $p = 6$  and  $n = 6$ .

#### 4.5 Adaptive wavelet technique for time-dependent problems

In this section, we describe the adaptive technique for time-dependent problems

$$\frac{\partial u}{\partial t} = \mathcal{L}u + \mathcal{N}(u, \nabla u) + f(\mathbf{x}, t), \quad \text{for } (\mathbf{x}, t) \in (0, 1)^d \times [0, \infty), \quad (4.21)$$

where  $\mathcal{L}$  is a linear second order differential operator,  $\mathcal{N}$  denotes a nonlinear term of lower order operator, and  $f$  represent a source term. The equation is augmented with appropriate boundary and initial conditions. The solution of such problem may contain important localized structures, layers, or sharp variations whose locations may vary with time (note that such features could be induced by operators  $\mathcal{L}$  or  $\mathcal{N}$ , source term  $f$ , or boundary/initial conditions). In order to resolve structures appearing in the solution as they evolve, the computational grid needs to adapt dynamically to reflect the local change in the solution. We tackle such a task by employing a time discretization to reduce the time-dependent PDE to a time-marching procedure and use wavelet amplitudes of the approximate solution at the current time level to construct the irregular grid for the approximate solution of next time level. Below, we first discuss the time discretization, and subsequently provide details of the dynamically adaptive algorithm for time dependent problems.

Time discretization by a finite difference scheme reduces the numerical solution of the time-dependent problem to solving a sequence of equations of the form,

$$\mathcal{A}^{n+1}u = \mathcal{F}^{n+1}, \quad \text{for } \mathbf{x} \in (0, 1)^d \quad (4.22)$$



The form of  $\mathcal{A}$  depends on the time discretization scheme employed. Given the solution at the previous time steps, the approximate solution  $u^{n+1}$  can then be computed. In this study, we employ a scheme that is implicit for the linear term and either explicit or implicit with linearization for the nonlinear term, *e.g.*

$$\frac{u^{m+1} - u^m}{\Delta t} = \tilde{\mathcal{N}}(u^{m+s}, u^{m+s-1}, \dots) + \alpha(\mathcal{L}u^{m+1} + f^{m+1}) + (1-\alpha)(\mathcal{L}u^m + f^m) \quad (4.23)$$

where  $\Delta t$  is the size of time step,  $u^m$  is the solution at the time level  $t^m = m\Delta t$ , and  $\tilde{\mathcal{N}}$  is either an approximation of the nonlinear term using a specific explicit scheme, such as, the Adam-Bashforth schemes or with a linearized implicit scheme (note that  $s = 0$  for explicit treatment and  $s = 1$  for implicit treatment), and  $\alpha$  is a parameter defining a specific time discretization scheme. More precisely, the integration scheme is a backward Euler when  $\alpha = 1$ , trapezoidal when  $\alpha = 1/2$ , and forward Euler when  $\alpha = 0$ . Since  $\mathcal{L}u$  considered in this work is mainly that of the diffusive type, we use an implicit scheme to avoid having to use a severely small  $\Delta t$  dictated by the condition for numerical stability arising when using an explicit scheme (as an example, for  $\mathcal{L}u \equiv \nu \nabla^2 u$ , the condition of stability for the explicit Euler scheme is  $\Delta t < 4^{-J}/(2d\nu)$ ,  $J$  being the finest level).

The conceptual idea of the adaptive strategy used here is introduced in [99] (and later pursued by several others [11, 61, 71, 76, 111, 134]): at each time level, after obtaining the approximate solution, grid points associated with wavelets that represent the approximate solution within a prescribed accuracy are retained; to accommodate the possible advection and sharpening of solution features, grid points in the vicinity of essential points at the same and higher levels of resolution are included in the computational grid to be used in the numerical solution of the subsequent time level. This strategy is expected to perform well when the solution

changes smoothly in time, *i.e.* the solution is appropriately time-resolved. Detail of such a procedure is provided in Algorithm 17. It consists roughly of the following steps:

- (i) Calculate the approximate solution of (4.23) based on the known approximate solutions at previous time levels;
- (ii) Retain points  $\mathbf{x}_{l,\lambda}$  whose  $|d_{j,\lambda}^{m+1}| > \varepsilon$ . Then collect sets of neighboring point  $\mathcal{N}_{l,\lambda}$  associated with essential points. The computational grid for the subsequently time level  $\mathcal{V}^{m+1}$  is the union of the essential and neighboring points;
- (iii) Compute interpolated values of the approximate solution with respect to the new grid at time levels necessary for the numerical solution at the next time level;
- (iv) Repeat (i)-(iii) until reaching a desirable time.

The initial irregular grid  $\mathcal{V}^0$  is constructed by combining the essential points of the initial condition and adding neighboring points to them. We employ the strategy given by (4.9) or (4.10) in defining the set of neighboring  $\mathcal{N}_{l,\lambda}$  associated with  $\mathbf{x}_{l,\lambda}$ . Note that in the case where a system of PDEs is solved, the computational grid is defined as a union of irregular grids arising from the consideration of each dependent variable. One may use different grids for different variables. In this case, switching from one grid to another can be accomplished by using the wavelet interpolation. This is not done in the current study. Let us remark that by replacing the conventional MRA-related materials with those of the MRA- $d$ , one has a similar algorithm based on MRA- $d$ .

---

**Algorithm 17** Dynamically adaptive solver for time-dependent problem

---

Given  $\varepsilon$ ,  $t_{final}$ , and  $\Delta t$

Construct the initial irregular grid  $\mathcal{V}^0$  by thresholding  $u(\mathbf{x}, 0)$

**for**  $m = 0$  to  $t_{final}/\Delta t$  **do**

    Solve problem (4.23) for  $u^{m+1}$  based on  $u^m, u^{m-1}, \dots, u^{m-q}$  on grid  $\mathcal{V}^m$

    # find essential grid points

    Perform  $AFWT(\mathbf{u}^{m+1}) \rightarrow \{\{u_{j_0, \mathbf{k}}^{m+1}\}, \{d_{l, \lambda}^{m+1}\}_{(l, \lambda) \in \Lambda^m}\}$

    Gather  $\Lambda_s = \{(l, \lambda) \in \Lambda^m : |d_{l, \lambda}^m| \geq \varepsilon\}$

    # add neighboring points to grid of essential points

    Determine the neighbor region  $\Lambda_{\mathcal{N}} = \bigcup_{(l, \lambda) \in \Lambda_s} \mathcal{N}_{l, \lambda}$  and establish  $\Lambda^{m+1} = \Lambda_s \cup \Lambda_{\mathcal{N}}$

    # new irregular grid

    Construct  $\mathcal{V}^{m+1} = \{\{\mathbf{x}_{l_0, \mathbf{k}}\}, \{\mathbf{x}_{l, \lambda}\}_{(l, \lambda) \in \Lambda^{m+1}}\}$

    # compute  $u(n+1), \dots, u(n-q-1)$  with respect to new grid

**for**  $r = 0$  to  $q - 1$  **do**

        Establish  $\tilde{\mathcal{D}} = \{\{u_{j_0, \mathbf{k}}^{m+1-r}\}, \{\tilde{d}_{l, \lambda}\}_{(l, \lambda) \in \Lambda^{m+1}}\}$  with

$$\tilde{d}_{l, \lambda} = \begin{cases} d_{l, \lambda}^{m+1-r}, & (l, \lambda) \in \Lambda^{m+1} \cap \Lambda^m \\ 0, & (l, \lambda) \in \Lambda^{m+1} \setminus \Lambda^m \end{cases}$$

        Perform  $AIWT(\tilde{\mathcal{D}}) \rightarrow u^{n+1-r}$

**end for**

**end for**

---

#### 4.5.1 Numerical experiment

**Test 6:** In order to test the adaptive algorithm for time dependent problems, we consider the following 2-D model problem:

$$\begin{aligned} \frac{\partial u}{\partial t} + (\mathbf{V} \cdot \nabla)u &= \nu \nabla^2 u + f(\mathbf{x}, t), \quad (\mathbf{x}, t) \in \Omega \times [0, +\infty), \Omega = (0, 1)^2, \\ u(\mathbf{x}, t) &= g(\mathbf{x}, t), \quad \mathbf{x} \in \partial\Omega \quad \text{and} \quad u(\mathbf{x}, 0) = q(\mathbf{x}), \end{aligned} \quad (4.24)$$

where  $\mathbf{V} = (0, 1)^T$  and  $\nu = 1/100$ . The source term  $f$ , the specified Dirichlet condition  $g$ , and the initial condition  $q$  are chosen so that the exact solution is given by

$$u(\mathbf{x}, t) = \frac{0.05(x_1 + 0.05)}{(x_1 + 0.05)^2 + (x_2 - t)^2}. \quad (4.25)$$

In the numerical calculations, a trapezoidal scheme for temporal discretization is used. The calculation is performed until  $t = 1$  is reached. A small time step is employed in order to minimize the error in the time discretization, *i.e.* the error in the approximate solution is dominated by that of the spatial discretization. Different combinations of  $p$  and  $n$  are used in the simulations. The maximum allowable resolution  $J - j_0$  is set to a large value so that the accuracy of the approximate solution is controlled by the threshold parameter  $\varepsilon$ . Note that the coarsest scale  $j_0$  is set to 3 ( $9 \times 9$  for the coarsest grid). Threshold values are varied from  $5 \times 10^{-3}$  to  $1 \times 10^{-7}$ . Figure 4.14 shows snapshots of the approximate solution and the irregular grid produced dynamically by the adaptive algorithm at different times. Results show in this figure are those computed with  $p = 4$ ,  $n = 4$ , and  $\varepsilon = 5 \times 10^{-4}$ . It can be noticed that the computational grid at any time level has fine resolution in the proximity of the near singularity and coarse

resolution in the smooth areas. This figure clearly indicates that the adaptive algorithm is able to follow the moving structure of the solution properly. The number of grid points required by the adaptive algorithm during the course of the simulation with different threshold values is plotted in Figure 4.15. Note that the integral of the exact solution as a function time is symmetric about  $t = 0.5$ . We thus expect the number of grid points generated by the adaptive algorithm to show this symmetry with respect to the specific time. Indeed, it can be observed that the plots appear symmetric about  $t = 1/2$  (of course this is only approximate in a strict sense). Note also that as the threshold parameter  $\varepsilon$  is decreased, the number of grid points demanded by the adaptive algorithm increase automatically. Figure 4.16 depicts the error, in the discrete maximum norm  $L_{\mathbf{V},\infty}$ , in the approximate solution obtained by the adaptive method with  $p = 4$  and  $n = 4$  during the course of the simulation. It can be observed that the error in the approximate solution stays almost constant (after the first few integration steps) as a function of time.

Figure 4.17 shows the log-log plots of the error in  $L_{\mathbf{V},\infty}$  in the approximate solution at  $t = 1/2$  and  $1$  versus  $N$ , the number of grid points required by the adaptive algorithm, for different combination of  $p$  and  $n$  as a result of varying the threshold parameter  $\varepsilon$ . The log-log plots of  $N$  versus  $\varepsilon$  are included in the right column of the figure. The slopes of the log-log plots indicate that at  $t = 1/2$  the number of grid points behaves like  $N = O(\varepsilon^{-2/c_1})$ , where  $c_1$  is approximately 4.4 for  $p = 4$  and  $n = 4$ , 7.1 for  $p = 6$  and  $n = 4$ , and 10.9 for  $p = 8$  and  $n = 6$ . At  $t = 1$ , the number of grid points  $N$  is proportional to  $O(\varepsilon^{-2/c_1})$ , where  $c_1$  is approximately 4.5 for  $p = 4$  and  $n = 4$ , 7.4 for  $p = 6$  and  $n = 4$ , and 9.4 for  $p = 8$  and  $n = 6$ . Note that the values of  $c_1$  are slightly larger than

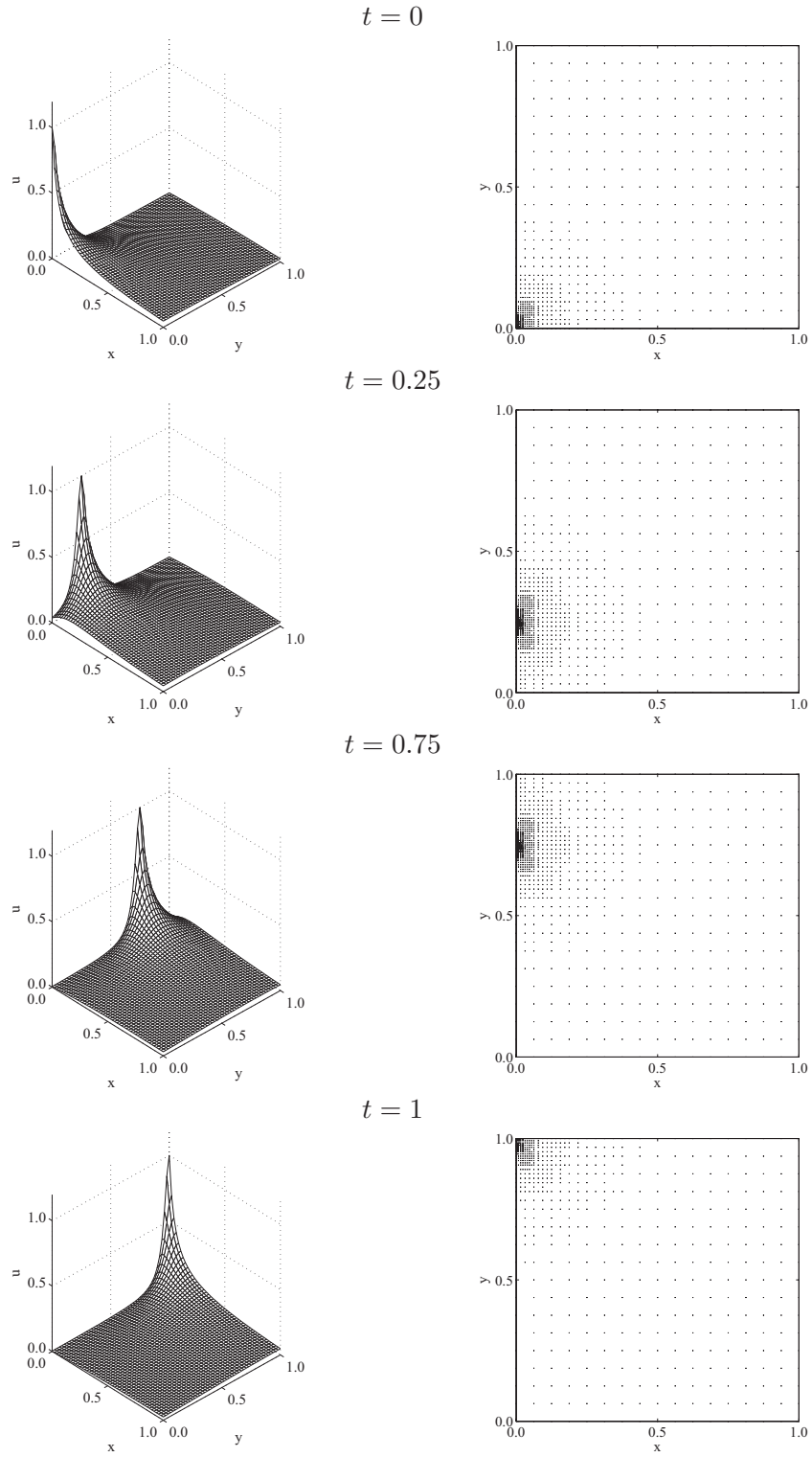


Figure 4.14: Approximate solution of Test 6 (left) and the irregular grid (right) generated by the adaptive method with  $p = 4$ ,  $n = 4$ , and  $\varepsilon = 5 \times 10^{-4}$ .

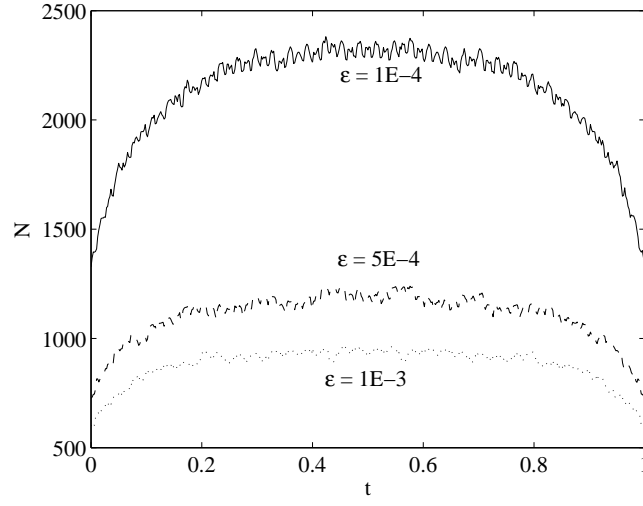


Figure 4.15: Number of points in the computational grid as a function of time by the adaptive algorithm with  $p = 4$ ,  $n = 4$  and different threshold values.

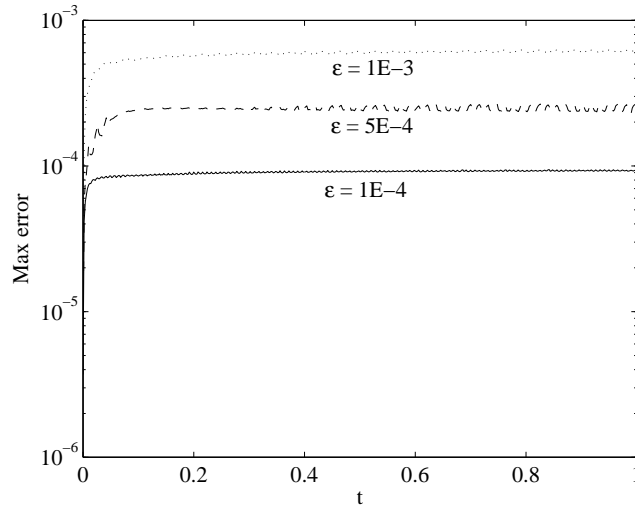


Figure 4.16:  $L_{\mathbf{V},\infty}$ -error in the approximate solution as a function of time for  $p = 4$ ,  $n = 4$ , and different threshold values.

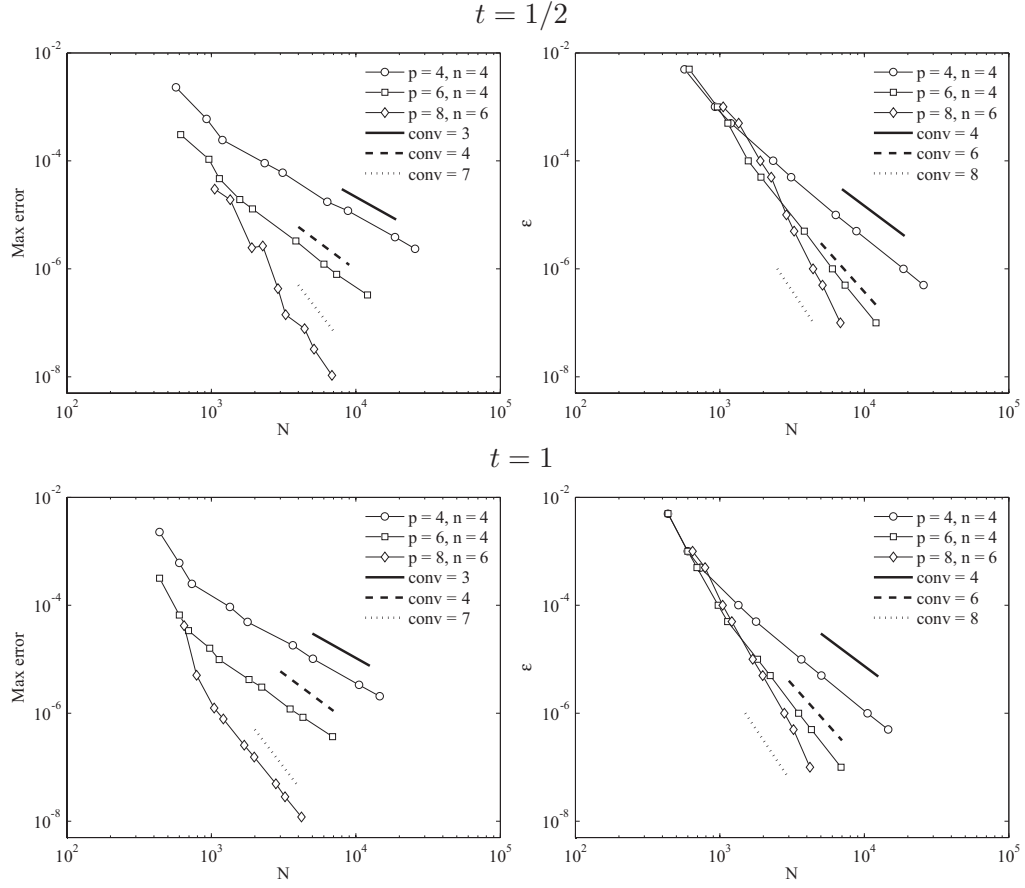


Figure 4.17:  $L_{\mathbf{V},\infty}$ -error as a functions of the number of grid points,  $N$ , as a result of varying  $\varepsilon$  (left) and  $\varepsilon$  versus  $N$  (right), at  $t = 1/2$  and  $t = 1$ .



that predicted by (3.32). The constant  $c_2$ , from a least square fit in the form  $\|u_{ext} - u_\varepsilon\|_{\mathbf{V},\infty} = O(N^{-c_2/2})$  at  $t = 1/2$ , is approximately 3 for  $p = 4$  and  $n = 4$ , 4.2 for  $p = 6$  and  $n = 4$ , and 9 for  $p = 8$  and  $n = 6$ . At  $t = 1$ , the constant  $c_2$  is approximately 3.2 for  $p = 4$  and  $n = 4$ , 4.1 for  $p = 6$  and  $n = 4$ , and 7.9 for  $p = 8$  and  $n = 6$ . Such rates of convergence for the adaptive algorithm with  $p = 4$  and  $p = 8$  are higher than  $O(N^{\min(p-2,n)/2})$  predicted by (3.52). For  $p = 6$ , the rate of convergence is in good agreement with the prediction. Note that the constant  $c_3$  from a least square fit in the form  $\|u_{ext} - u_\varepsilon\|_{\mathbf{V},\infty} = O(\varepsilon^{c_3})$  at  $t = 1/2$  is approximately 0.69 for  $p = 4$  and  $n = 4$ , 0.59 for  $p = 6$  and  $n = 4$ , and 0.83 for  $p = 8$  and  $n = 6$ . At  $t = 1$ , such a constant equals approximately 0.70 for  $p = 4$  and  $n = 4$ , 0.55 for  $p = 6$  and  $n = 4$ , and 0.81 for  $p = 8$  and  $n = 6$ . Note that for  $p = 4$  and  $n = 6$ , errors in the discrete maximum norm are larger than the threshold values as the threshold values are decreased (as indicated by the fact that the constants  $c_3$  for  $p = 4$  and  $p = 6$  are smaller than unity). For the range of threshold values used, the errors in the approximate solution for  $p = 8$  are of the same order as the threshold values (despite the fact that  $c_3$  for  $p = 8$  is smaller than unity).

#### 4.5.2 Application to a flame ball problem

In this section, the interaction of a flame ball with a single vortex in a combustion chamber filled with premixed fuel is used as a test problem. Roussel and Schneider [112] use this problem as a test problem for their cell-average multiresolution method with finite-volume discretization. The equations modeling the problem in the dimensionless form is as follows, for  $(\mathbf{x}, t) \in \Omega \times [0, \infty)$  [112, 137],

$$\frac{\partial T}{\partial t} + (\mathbf{u} \cdot \nabla)T = \nabla^2 T + \omega - s, \quad (4.26)$$

$$\frac{\partial Y}{\partial t} + (\mathbf{u} \cdot \nabla)Y = \frac{1}{Le} \nabla^2 Y - \omega,$$

$$\omega(T, Y) = \frac{Ze^2}{2Le} Y \exp \left[ -\frac{Ze(1-T)}{1-\alpha(1-T)} \right], \quad (4.27)$$

$$s(T) = \gamma \left[ (T + \alpha^{-1} - 1)^4 - (\alpha^{-1} - 1)^4 \right], \quad (4.28)$$

where  $T$  and  $Y$  represent respectively temperature and the partial mass of the fresh premixed gas, and  $\mathbf{u}$  the velocity of gaseous mixture, and  $\omega$  and  $\alpha$  are respectively the dimensionless reaction rate and heat loss due to radiation, and  $\alpha$  and  $\gamma$  denote respectively the burnt-unburnt temperature ratio and the radiation coefficient. The dimensionless parameters appearing in the equations are Zeldovich number,  $Ze$ , and the Lewis number,  $Le$ . The above model is based upon the hypothesis of the thermodiffusive approximation, which is valid for a slowly propagating flame [112, 137]. In this case, the fluid flow is not effected by the chemical reaction and the velocity field of the mixture  $\mathbf{u}$  is that of incompressible flow (*i.e.*,  $\nabla \cdot \mathbf{u} = 0$ ). Radiative heat losses are assumed to obey the Stefan-Boltzmann law.

To be able to check the results of the present algorithm, we use the velocity field, initial profiles of  $T$  and  $Y$  and boundary conditions given in [112]. That is, we consider  $\mathbf{u}$  corresponding to the Hamel-Oseen vortex flow which, in polar coordinates, is given by

$$\mathbf{u} = \frac{\Gamma}{2\pi r} \left( 1 - \exp \left[ -\frac{r^2}{4Pr(t + \tau)} \right] \right), \quad (4.29)$$

where  $\Gamma$  is the dimensionless circulation,  $Pr$  denotes the Prandtl number, and  $\tau > 0$  represents a reference time. The domain  $\Omega$  is square  $[-L, L]^2$ , where  $L$  is sufficiently large. The initial conditions for  $T$  and  $Y$  are given by

$$T(r, 0) = \begin{cases} 1 & \text{if } r \leq r_0, \\ \exp(1 - r/r_0) & \text{if } r > r_0, \end{cases} \quad (4.30)$$

$$Y(r, 0) = \begin{cases} 0 & \text{if } r \leq r_0, \\ 1 - \exp(Le(1 - r/r_0)) & \text{if } r > r_0, \end{cases} \quad (4.31)$$

where  $r_0 = \sqrt{(x - X_0)^2 + y^2}$ ,  $r_0$  is the radius of the flame ball at  $t = 0$ , and  $X_0$  is the initial  $x$ -coordinate of the center of the flame ball. The boundary conditions on the square domain are approximated by

$$\left. \frac{\partial T}{\partial n} \right|_{\partial\Omega} = \left. \frac{\partial Y}{\partial n} \right|_{\partial\Omega} = 0. \quad (4.32)$$

In the all test cases, the gaseous mixture ( $\text{H}_2$ -air) considered has the temperature ratio of  $\alpha = 0.64$ , and the Lewis number and Zeldovich number of  $Le = 0.3$ , and  $Ze = 10$ , respectively. The initial radius of the flame ball is taken to be  $r_0 = 0.5$  and initially the center of the flame ball is located at  $x_0 = 2.5$ . The velocity profile is that corresponding to the Prandtl number  $Pr = 0.01$ , and characteristic time  $\tau = 0.1$ . We consider the problem with circulation  $\Gamma$  ranging from 0 to 1000. Here, we consider only cases where heat loss due to radiation is neglected, *i.e.*  $\gamma = 0$ .

In the following numerical calculations, we use the trapezoidal scheme for the convection and diffusion terms and the 2<sup>nd</sup>-order Adam-Bashforth scheme for the nonlinear source term. A time step of  $\Delta t = 5 \times 10^{-5}$  is used for the test case where  $\Gamma = 0, 10, 100$  and  $\Delta t = 2.5 \times 10^{-5}$  for  $\Gamma = 1000$ . The calculation is carried out until  $t = 1.6$  is reached. The computational domain is set to  $\Omega = [-10, 10]^2$ . The number of evolving resolution levels is set to  $J - j_0 = 7$  where  $j_0 = 4$ . This results in a grid with  $17 \times 17$  points at the coarsest level and the finest grid possible is of size  $\Delta x = \Delta y = 2L/2048 = 5/512$ . The set of neighboring points is defined according to (4.9) with  $P = 1$ . Threshold values of  $\varepsilon = (5 \times 10^{-3}, 5 \times 10^{-3})$ ,  $(10^{-3}, 10^{-3})$ , and  $(5 \times 10^{-4}, 5 \times 10^{-4})$  are used in the simulations. Unless otherwise noted, we use  $p = 6$ , and derivative approximations are obtained with  $n = 4$ . The simulations based on the MRA- $d$  approach with  $p = 6$  and  $n = 4$  are also obtained for the test case with  $\Gamma = 100$ .

Figure 4.18 shows snap shots of temperature, reaction rate, and irregular grid, generated automatically by the adaptive algorithm, for  $\Gamma = 100$  at  $t = 0, 0.33, 0.79$ , and  $1.0$ . The profiles of  $T$  and  $\omega$  on the planes  $x = x_0$  and  $y = 0$  at specific time levels are shown in Figure 4.19. Results shown in these figures are obtained with  $\varepsilon = (10^{-3}, 10^{-3})$ . Additionally, the numerical solution obtained from MRA- $d$  with  $\varepsilon = (10^{-3}, 10^{-3})$  are displayed in Figure 4.20. Results from the two techniques are in good agreement and agree well with those provided in [112]. It can be observed the flame ball is advected counter-clockwise around the center of the vortex. As observed in [112], the structure of the flame ball, which is initially circular, becomes increasingly distorted as time evolves and eventually form a snail-like (or ammonite-like structure). Note the splitting of the flame front into two parts after the flame rolls up around the center of the vortex. The distribution of points in

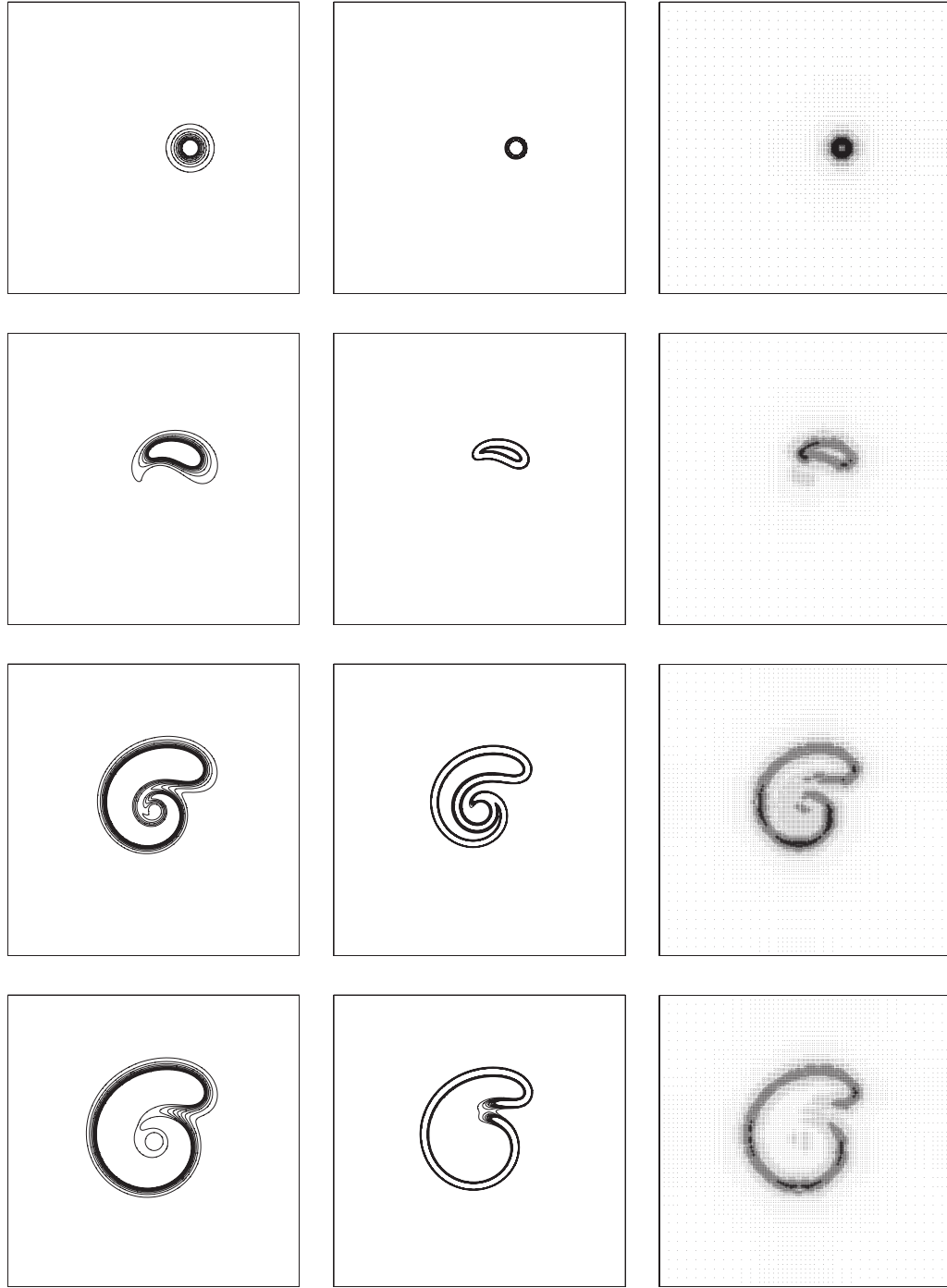


Figure 4.18: Solution of the flame ball problem with  $\Gamma = 100$  at  $t = 0, 0.33, 0.79$ , and  $1.0$  (top-bottom). Left: isolines of  $T$  from  $0.1$  to  $0.9$  with increment of  $0.1$ . Middle: corresponding isolines of  $\omega$ . Right: irregular sparse grid for  $p = 6$ ,  $n = 4$ , and  $\varepsilon = (10^{-3}, 10^{-3})$ .

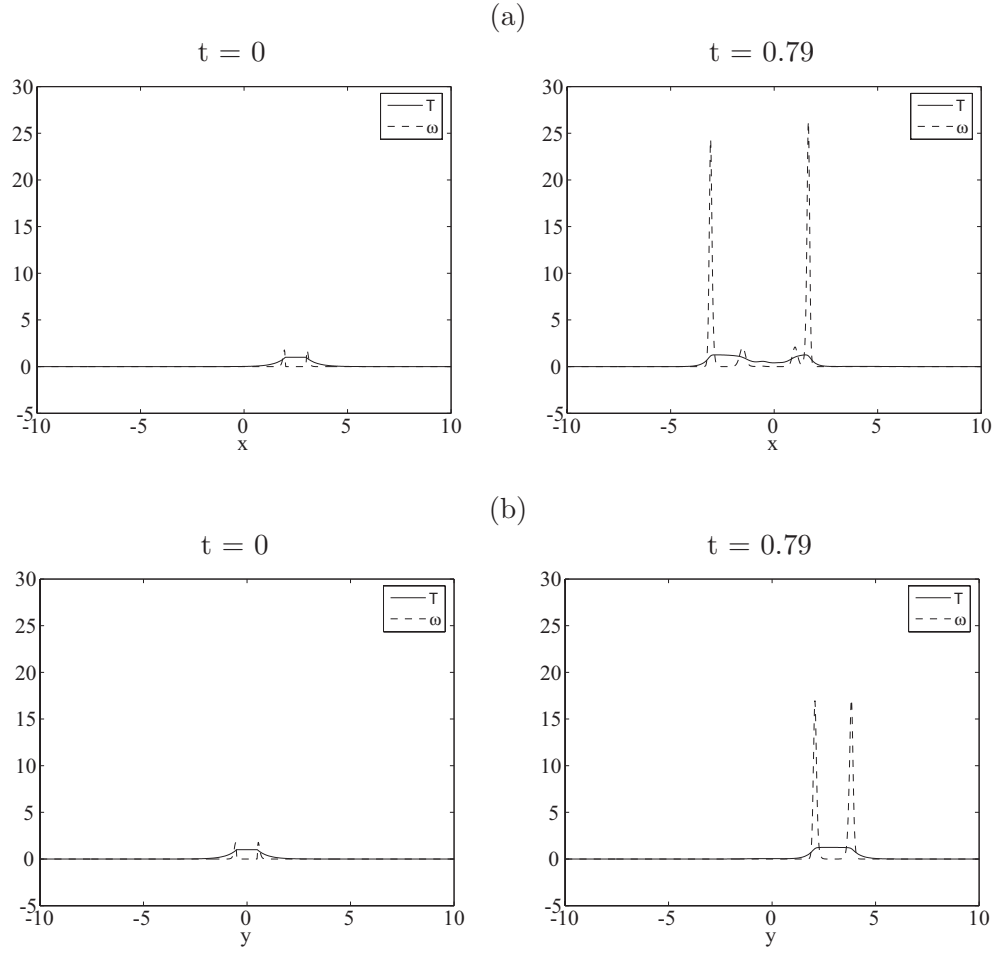


Figure 4.19: Solution of the flame ball problem with  $\Gamma = 100$  at specific planes obtained with  $p = 6$ ,  $n = 4$ , and  $\varepsilon = (10^{-3}, 10^{-3})$ . (a)  $T$  and  $\omega$  on the plane  $y = 0$  at  $t = 0$  (left) and  $t = 0.79$  (right), (b)  $T$  and  $\omega$  on the plane  $x = 0$  at  $t = 0$  (left) and  $t = 0.79$  (right).

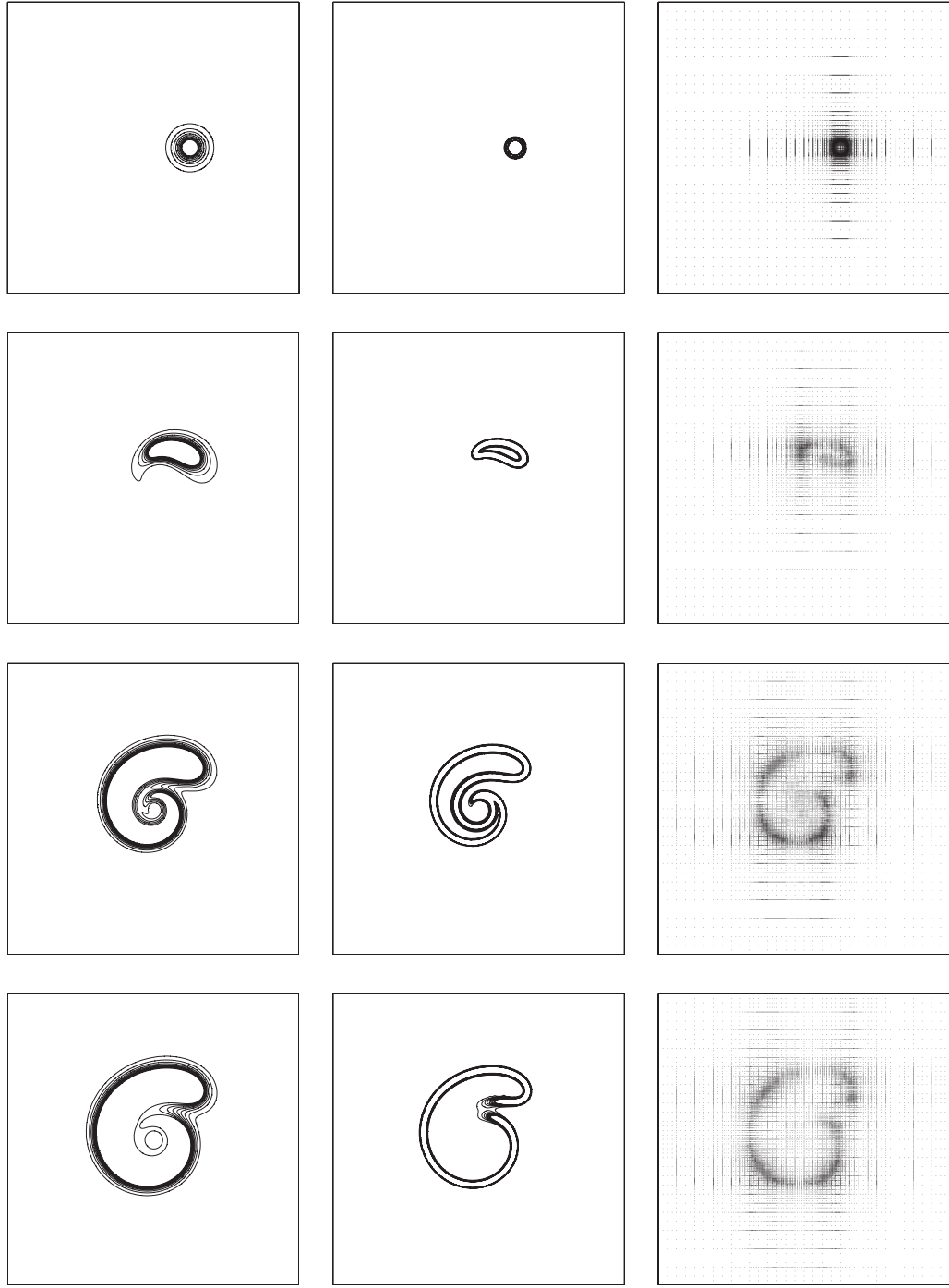


Figure 4.20: Solution of the flame ball problem with  $\Gamma = 100$  at  $t = 0, 0.33, 0.79, 1$  (top-bottom). Left: isolines of  $T$  from 0.1 to 0.9 with increment of 0.1. Middle: corresponding isolines of  $\omega$ . Right: irregular sparse grid from MRA- $d$  approach with  $p = 6$ ,  $n = 4$ , and  $\epsilon = (10^{-3}, 10^{-3})$ .

adaptive grids, generated automatically, clearly demonstrates that the adaptive algorithm based on both MRA approaches is able to track the evolving structures of the flame ball properly with the majority of points being clustered around the flame front. Figure 4.21 shows the number of degree of freedom required (*i.e.*  $N = \dim \mathbf{V}$ ) during the course of the simulations.

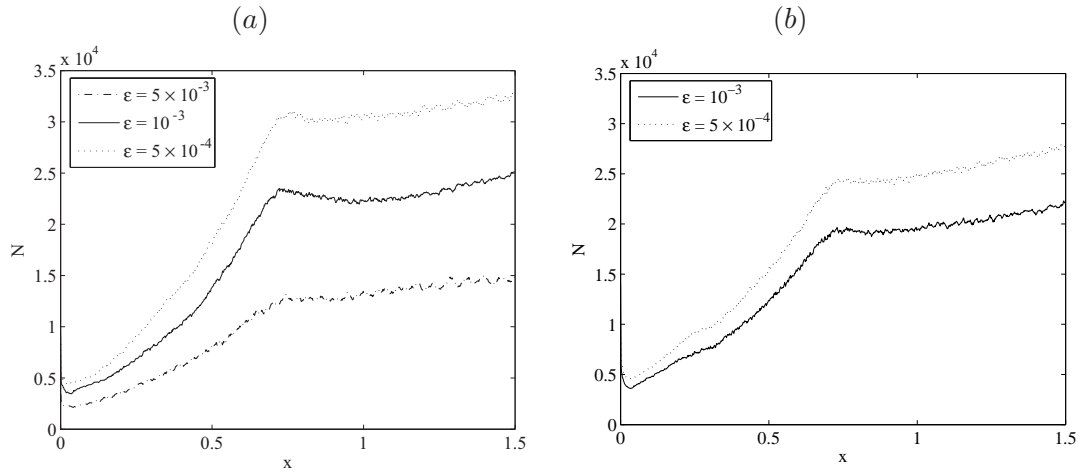


Figure 4.21: Number degrees of freedom as a function of time required by the adaptive algorithm based on (a) the MRA approach and (b) the MRA- $d$  approach

Note that as the threshold parameter  $\epsilon$  is decreased, the number of points required by the adaptive algorithm increases automatically. The average number of degrees of freedom required in the simulations up to  $t = 1.5$  using the MRA approach with  $\epsilon = (5 \times 10^{-3}, 5 \times 10^{-3})$ ,  $(10^{-3}, 10^{-3})$ , and  $(5 \times 10^{-4}, 5 \times 10^{-4})$  are approximately 10000, 17000, and 22000 respectively. For simulations using MRA- $d$  approach, the



average number of degree of freedom are approximately 15000 and 19000 when  $(1 \times 10^{-3}, 1 \times 10^{-3})$  and  $(5 \times 10^{-4}, 5 \times 10^{-4})$  respectively. In these simulations with  $\Gamma = 100$ , the highest level demanded by the solution never reaches the finest level  $J = 11$ . The maximum level required in these simulations is that of  $j = 10$  (note that in the case where  $\varepsilon = (5 \times 10^{-3}, 5 \times 10^{-3})$ , only few points ( $< 100$ ) belong to the maximum level. In this case, it is reasonable to state that the maximum level is  $j = 9$ ). Note the reduction of degrees of freedom from the case where a full grid of the maximum resolution were to be used.

To evaluate the results quantitatively, the integral of the reaction rate  $R$ , defined by

$$R(t) = \int_{\Omega} \omega(x, y, t) dx dy \quad (4.33)$$

is computed. Figure 4.22 shows the reaction rate  $R$  for  $\Gamma$  ranging from 0 to 1000. The integral  $R$  shown in the figure correspond to the integration of results obtained with the threshold value  $\varepsilon = (10^{-3}, 10^{-3})$ . The difference between the results given and those using other threshold parameters are indistinguishable by the naked eye and thus their plots are omitted. The plots of  $R$  are in good agreement with those given in [112]. Note that as the value of circulation  $\Gamma$  increases, the integral  $R$  deviates more and more from the pure reaction diffusion case ( $\Gamma = 0$ ). For large  $\Gamma$ , there is a peak in  $R$  and such a peak occurs earlier for the larger value as  $\Gamma$  (see Figure 4.23 for snap shots of solution at certain time levels for the problem with  $\Gamma = 1000$ )<sup>2</sup>. We summarize the magnitude of the peak and the time it occurs for problem with  $\Gamma = 100$ ,  $\Gamma = 1000$  in Table 4.1. Note that similar quantities are not provide in [112]. We include them here for the reason that they might be useful in the future for comparison between different numerical methods.

---

<sup>2</sup>For  $\Gamma > 1300$ , the chemical reaction extinguishes rapidly [112].

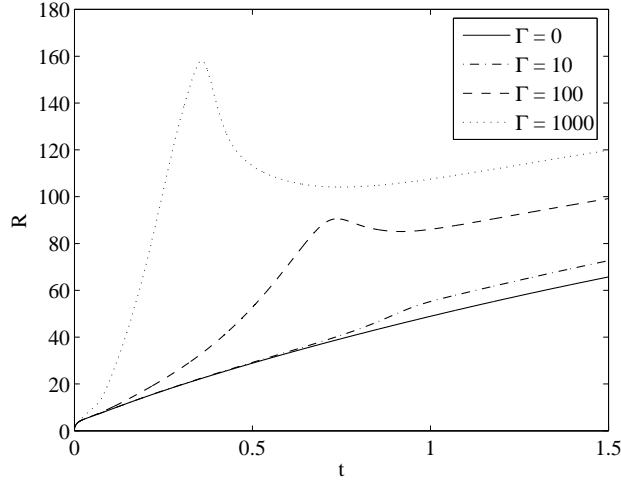


Figure 4.22: Integral of reaction rate,  $R$ , as a function of time for different values of  $\Gamma$ .

#### 4.6 Application of the method on a bounded curvilinear domain

Since higher-dimensional wavelets used in the adaptive methods are constructed from tensor product of one-dimensional wavelets, the method is inherently restricted to rectangular computational domains. In order to deal with more general geometries, we consider a domain transformation technique. Let  $\tilde{\Omega} \in R^d$ , a physical domain of interested, be a simply connected domain, and  $\Omega = [0, 1]^d$  be a computational domain. Note that if the domain is multiply-connected, then with use of branch cuts, the domain can be made simply connected. The domain transformation technique consists of two tasks: finding a  $C^2$  invertible map

$$\psi : \Omega \rightarrow \tilde{\Omega}, \quad \mathbf{x} = (x_1, \dots, x_d) \mapsto \boldsymbol{\xi} = (\xi_1, \dots, \xi_d) = \psi(\mathbf{x}), \quad (4.34)$$

which maps the computational domain one-to-one onto the physical domain (see Figure 4.24 for illustration), and subsequently solving the transformed equations

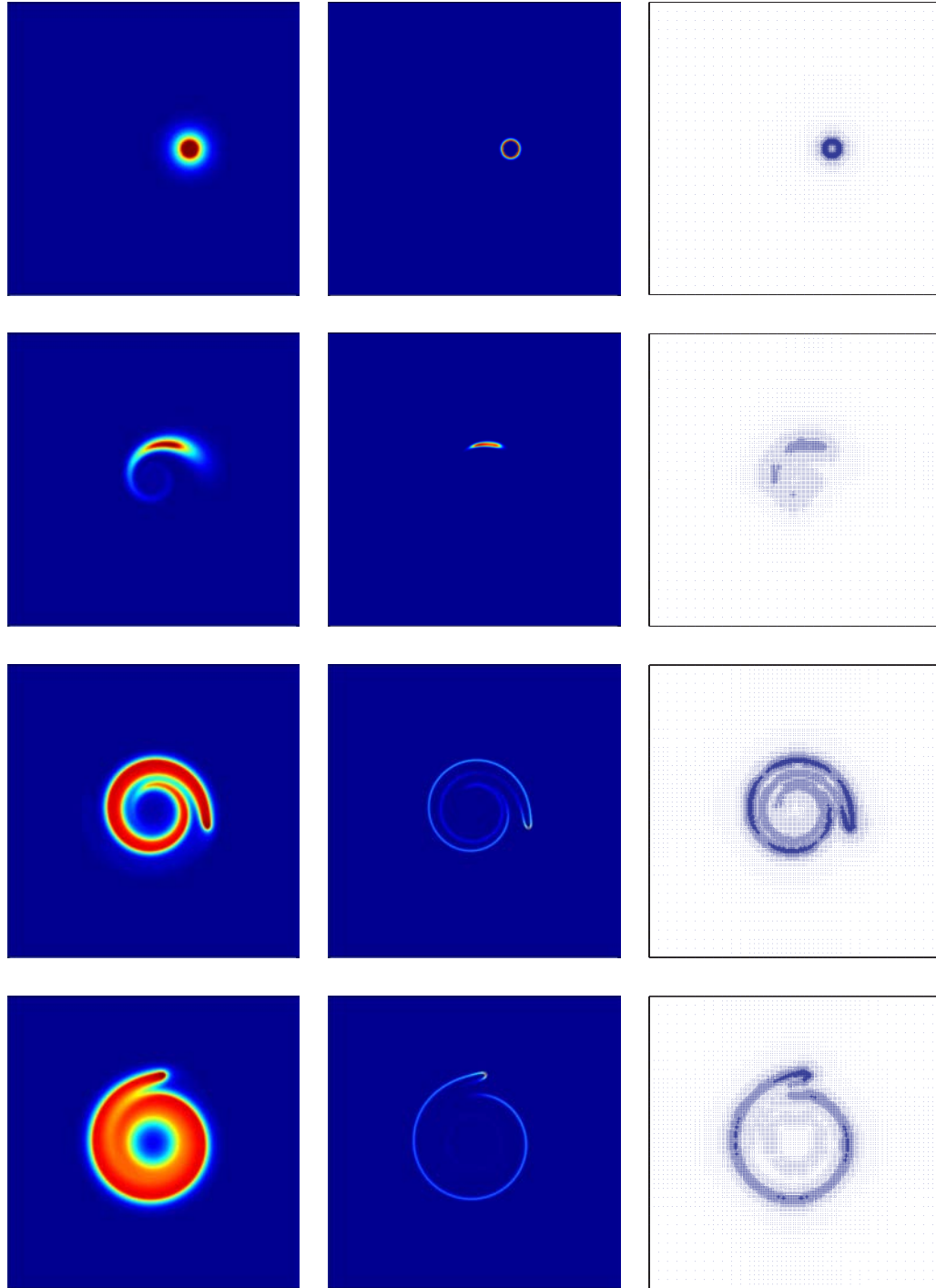


Figure 4.23: Solution of the flame ball problem with  $\Gamma = 1000$ ,  $\gamma = 0$  at  $t = 0$ ,  $0.05$ ,  $0.39$ , and  $0.60$  (top-bottom). Left: temperature  $T$ . Middle: reaction rate  $\omega$ . Right: irregular sparse grid generated dynamically by the adaptive wavelet scheme with  $p = 6$ ,  $n = 4$ ,  $\varepsilon = (10^{-3}, 10^{-3})$ .

TABLE 4.1

VALUES OF THE PEAK IN THE INTEGRAL  $R(t)$  AND TIME AT WHICH IT OCCURS FOR  $\Gamma = 100$  and  $\Gamma = 1000$

	$\epsilon$	$\Gamma = 100$		$\Gamma = 1000$	
		$R_{peak}$	$t_{peak}$	$R_{peak}$	$t_{peak}$
$p = 6, n = 4$	$(1e-3, 1e-3)$	90.564	0.737	158.160	0.357
	$(5e-4, 5e-4)$	90.589	0.738	158.141	0.357
$p = 8, n = 7$	$(1e-3, 1e-3)$	90.571	0.738		
	$(5e-4, 5e-4)$	90.584	0.738		
MRA- $d, p = 6, n = 4$	$(5e-4, 5e-4)$	90.629	0.740		

in the computational domain. Note that since  $\boldsymbol{\psi}$  is invertible, one also has the inverse mapping  $\boldsymbol{\psi}^{-1} : \tilde{\Omega} \rightarrow \Omega$ ,  $\boldsymbol{\xi} \mapsto \mathbf{x} = \boldsymbol{\psi}^{-1}(\boldsymbol{\xi})$ . Since transformed problems are now defined on  $[0, 1]^d$ , the adaptive method just described is therefore applicable.

It is noted that the corresponding PDEs in the generalized (Cartesian) coordinate (also known as boundary fitted coordinate) can be obtained systematically by making use of the following formulas for partial derivatives,

$$\frac{\partial}{\partial \xi_i} = \sum_{k=1}^d \frac{\partial x_k}{\partial \xi_i} \frac{\partial}{\partial x_k}, \quad (4.35)$$

$$\frac{\partial^2}{\partial \xi_i \partial \xi_j} = \sum_{k=1}^d \frac{\partial^2 x_k}{\partial \xi_i \partial \xi_j} \frac{\partial}{\partial x_k} + \sum_{k=1}^d \sum_{m=1}^d \frac{\partial x_k}{\partial \xi_i} \frac{\partial x_m}{\partial \xi_j} \frac{\partial^2}{\partial x_k \partial x_m}. \quad (4.36)$$

These formulas can be easily verified by using the chain rule and the fact that the inverse mapping exist, *i.e.* one has  $x_i = x_i(\xi_1, \dots, \xi_d)$ . For an example, the transformed equation corresponding to the  $d$ -D Poisson equation

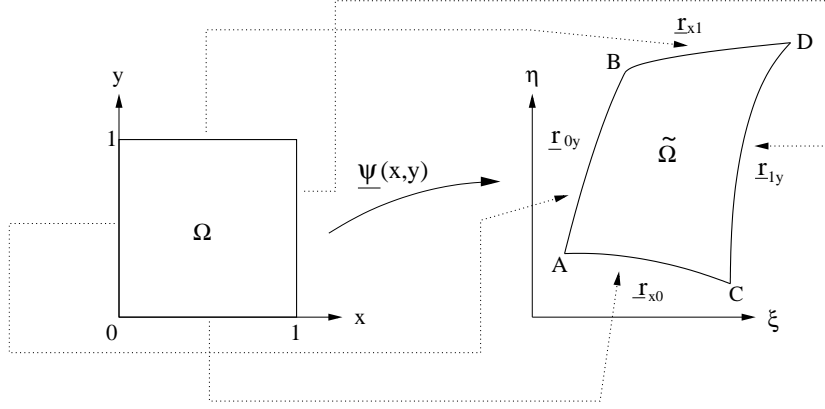


Figure 4.24: Coordinate transformation in two spatial dimension

$$\sum_{i=1}^d \frac{\partial^2 u(\boldsymbol{\xi})}{\partial \xi_i^2} = f(\boldsymbol{\xi}) \quad \text{in } \tilde{\Omega}, \quad (4.37)$$

is given by

$$\sum_{i=1}^d \left[ \sum_{k=1}^d \frac{\partial^2 x_k}{\partial \xi_i^2} \frac{\partial u(\mathbf{x})}{\partial x_k} + \sum_{k=1}^d \sum_{m=1}^d \frac{\partial x_k}{\partial \xi_i} \frac{\partial x_m}{\partial \xi_i} \frac{\partial^2 u(\mathbf{x})}{\partial x_k \partial x_m} \right] = f(\mathbf{x}) \quad \text{in } \Omega \quad (4.38)$$

where  $f(\mathbf{x}) = f(\boldsymbol{\psi}^{-1}(\boldsymbol{\xi}))$ .

The remaining task is to determine the (boundary fitted) mapping  $\boldsymbol{\psi} : \Omega \rightarrow \tilde{\Omega}$ , *i.e.*  $\xi_i = \xi_i(x_i, \dots, x_d)$  of points interior to the domain. Such a task can be accomplished by means of elliptic grid generation techniques or algebraic mapping techniques (see [59, 60, 126] for more detail). In this study, we consider a transfinite interpolation [68], which is a commonly used algebraic grid generation approach. In the 2-D case, the transfinite interpolation is defined by

$$\begin{aligned}
\boldsymbol{\xi} = \boldsymbol{\psi}(x_1, x_2) = & \{(1 - x_1)\mathbf{r}_{0x_2}(x_1) + x_1\mathbf{r}_{1x_2}(x_2)\} + \{(1 - x_2)\mathbf{r}_{x_10}(x_1) + \\
& x_2\mathbf{r}_{x_11}(x_1)\} - \{(1 - x_1)(1 - x_2)\mathbf{r}_{x_10}(0) + (1 - x_1)x_2\mathbf{r}_{x_11}(0) + \\
& (1 - x_2)x_1\mathbf{r}_{x_10}(1) + x_1x_2\mathbf{r}_{x_11}(1)\}, \tag{4.39}
\end{aligned}$$

where  $\mathbf{r}_{0x_2}(s)$ ,  $\mathbf{r}_{1x_2}(s)$ ,  $\mathbf{r}_{x_10}(s)$ , and  $\mathbf{r}_{x_11}(s)$  for  $s \in [0, 1]$  are the four physical boundary edges, in parametric form, associated with the computational boundary edges at  $x_1 = 0$ ,  $x_1 = 1$ ,  $x_2 = 0$ , and  $x_2 = 1$ , respectively (see Figure 4.24). Note that these four boundary lines must be consistent at the four vertices of the physical domain,

$$\begin{aligned}
\mathbf{r}_{x_10}(0) &= \mathbf{r}_{0x_2}(0), & \mathbf{r}_{x_10}(1) &= \mathbf{r}_{1x_2}(0), \\
\mathbf{r}_{1x_2}(1) &= \mathbf{r}_{x_11}(1), & \mathbf{r}_{0x_2}(1) &= \mathbf{r}_{x_11}(0).
\end{aligned}$$

It is easy to check that (4.39) maps exactly edges of the computational domain to edges of the physical domain. The corresponding formula for in the 3-D case consists of twenty-six terms of transfinite interpolation of six parametrized surfaces (see [59], pp. 96, for the detailed formula). It can be noticed from (4.39) that the value of  $\boldsymbol{\xi} = (\xi_1, \dots, \xi_d)$  can be evaluated at any point once the parametrized boundary edges are determined. In addition, partial derivatives can be computed at any point provided that the parametrized boundary edges have the necessary order of continuity.

Let us mention that the values of  $\xi_i$ ,  $\partial\xi_i/\partial x_j$ , and  $\partial^2\xi_i/\partial x_j\partial x_k$  can be computed easily. The latter two may be computed exactly or numerically. For adaptive grids, one may use the derivative approximation described in the previous chapter

to numerically evaluate these partial derivatives at the grid points. Note that, in solving transformed PDEs using the described adaptive techniques, we require  $\partial x_i / \partial \xi_j$  and  $\partial^2 x_i / \partial \xi_j \partial \xi_k$  (instead of  $\partial \xi_i / \partial x_j$  and  $\partial^2 \xi_i / \partial x_j \partial x_k$ ) at grid points. Although in general the closed form of the inverse mapping  $\psi^{-1}$  is not available (and of course we prefer not to seek such a closed form), such quantities can be easily computed by inverting the differential matrix associated with (4.35) and (4.36). The values of partial derivatives required correspond simply to entries of such inverse matrix. To see this, consider (4.35) and (4.36) in the 2-D case, in matrix form:

$$\begin{pmatrix} \frac{\partial u}{\partial x_1} \\ \frac{\partial u}{\partial x_2} \\ \frac{\partial^2 u}{\partial^2 x_1} \\ \frac{\partial^2 u}{\partial^2 x_2} \\ \frac{\partial x_1 \partial x_2}{\partial^2 u} \end{pmatrix} = \mathbf{Q} \cdot \begin{pmatrix} \frac{\partial u}{\partial \xi_1} \\ \frac{\partial u}{\partial \xi_2} \\ \frac{\partial^2 u}{\partial^2 \xi_1} \\ \frac{\partial^2 u}{\partial^2 \xi_2} \\ \frac{\partial \xi_1 \partial \xi_2}{\partial^2 u} \end{pmatrix} \quad (4.40)$$

where  $\mathbf{Q}$  is a  $5 \times 5$  matrix whose entries consist of  $\partial \xi_i / \partial x_j$  and  $\partial^2 \xi_i / \partial x_j \partial x_k$  (quantities that can be computed). We then obtain  $\partial \xi_i / \partial x_j$  and  $\partial^2 \xi_i / \partial x_j \partial x_k$  from appropriate entries of  $\mathbf{Q}^{-1}$ . In this case,  $\mathbf{Q}^{-1}$  is given by

$$\mathbf{Q}^{-1} = \begin{pmatrix} \frac{\partial x_1}{\partial \xi_1} & \frac{\partial x_2}{\partial \xi_1} & 0 & 0 & 0 \\ \frac{\partial x_1}{\partial \xi_2} & \frac{\partial x_2}{\partial \xi_2} & 0 & 0 & 0 \\ \frac{\partial^2 x_1}{\partial \xi_1^2} & \frac{\partial^2 x_2}{\partial \xi_1^2} & \left(\frac{\partial x_1}{\partial \xi_1}\right)^2 & 2 \frac{\partial x_1}{\partial \xi_1} \frac{\partial x_2}{\partial \xi_1} & \left(\frac{\partial x_2}{\partial \xi_1}\right)^2 \\ \frac{\partial^2 x_1}{\partial \xi_1 \partial \xi_2} & \frac{\partial^2 x_2}{\partial \xi_1 \partial \xi_2} & \frac{\partial x_1}{\partial \xi_2} \frac{\partial x_1}{\partial \xi_1} & \left(\frac{\partial x_1}{\partial \xi_1} \frac{\partial x_2}{\partial \xi_2} + \frac{\partial x_1}{\partial \xi_2} \frac{\partial x_2}{\partial \xi_1}\right) & \frac{\partial x_2}{\partial \xi_1} \frac{\partial x_2}{\partial \xi_2} \\ \frac{\partial^2 x_1}{\partial \xi_2^2} & \frac{\partial^2 x_2}{\partial \xi_2^2} & \left(\frac{\partial x_1}{\partial \xi_2}\right)^2 & 2 \frac{\partial x_1}{\partial \xi_2} \frac{\partial x_2}{\partial \xi_2} & \left(\frac{\partial x_2}{\partial \xi_2}\right)^2 \end{pmatrix} \quad (4.41)$$

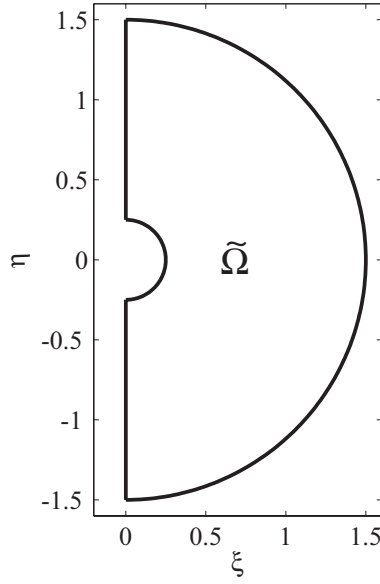


Figure 4.25: Physical domain  $\tilde{\Omega}$  of Test 7.

Note that in the 2-D case, computing  $\mathbf{Q}^{-1}$  requires  $O(5^3)$  operations (roughly  $(2/3) \times 5^3$  from the LU decomposition and  $5^2 \times 5$  from back substitution) for one grid point and  $O(9^3)$  operations in the 3-D case. Thus for a grid of  $N$  irregular grid points, the number of operations associated with the grid transformation is of  $O(5^3 N)$  for 2-D case and  $O(9^3 N)$  for the 3-D case. Note that this calculation is done once, and the only for the grid that change.

#### 4.6.1 Numerical results

To test the strategy described above, we consider a 2-D Poisson problem defined on a curvilinear domain with Dirichlet boundary conditions.

**Test 7:** Consider the Poisson equation defined on the domain  $\tilde{\Omega}$  shown in Figure 4.25. The right hand side of the equation and boundary conditions on  $\partial\tilde{\Omega}$  are chosen such that the exact solution of the problem is given by



$$u(\boldsymbol{\xi}) = \tan^{-1} \left[ \frac{\xi_2 - 0.85}{\xi_1 + 0.01} \right]. \quad (4.42)$$

In the transformation, the boundary edges, in parametric form, are given by

$$\begin{aligned} \mathbf{r}_{x_1 0}(s) &= \begin{bmatrix} 0, & (r_1 - r_2)s - r_1 \end{bmatrix}^T, \\ \mathbf{r}_{x_1 1}(s) &= \begin{bmatrix} 0, & (r_2 - r_1)s + r_1 \end{bmatrix}^T, \\ \mathbf{r}_{0x_2}(s) &= \begin{bmatrix} r_1 \cos(\pi(s - 1/2)), & r_1 \sin(\pi(s - 1/2)) \end{bmatrix}^T, \\ \mathbf{r}_{1x_2}(s) &= \begin{bmatrix} r_2 \cos(\pi(s - 1/2)), & r_2 \sin(\pi(s - 1/2)) \end{bmatrix}^T. \end{aligned} \quad (4.43)$$

where  $r_1 = 0.25$  and  $r_2 = 1.5$ . Of course one can use different choices for the parametrized equations (for instance, one may define a parametrization by mean of some interpolation). The adaptive method based on MRA approach with different combinations of  $p$  and  $n$  is applied to solve the (transformed) problem. The threshold value of  $\varepsilon$  is varied from  $5 \times 10^{-2}$  to  $1 \times 10^{-8}$  in the numerical calculations. Figure 4.26 shows the numerical solution and the adaptively refined irregular grid produced with  $p = 4$ ,  $n = 4$ , and  $\varepsilon = 10^{-3}$ . It can be seen that grid points are concentrated in the proximity of the singularity (see Test 1, Section 4.4 for a similar Poisson test problem defined on a square domain). The figure indicates qualitatively that the adaptive algorithm adds grid points to where the refinement is necessary. It demonstrates also that the use of the coordinate transformation permit the use the wavelet adaptive method to solve problems in more complicate domains. The error in  $L_{\mathbf{V},\infty}$ -norm in the numerical solution as a function of the number of grid points  $N$ , for different  $p$  and  $n$  as a result of varying the threshold parameter  $\varepsilon$ , is plotted in Figure 4.27. The dependence between  $N$  and  $\varepsilon$  is also included in the figure. The slopes of the log-log plots of  $\varepsilon$  versus  $N$  are

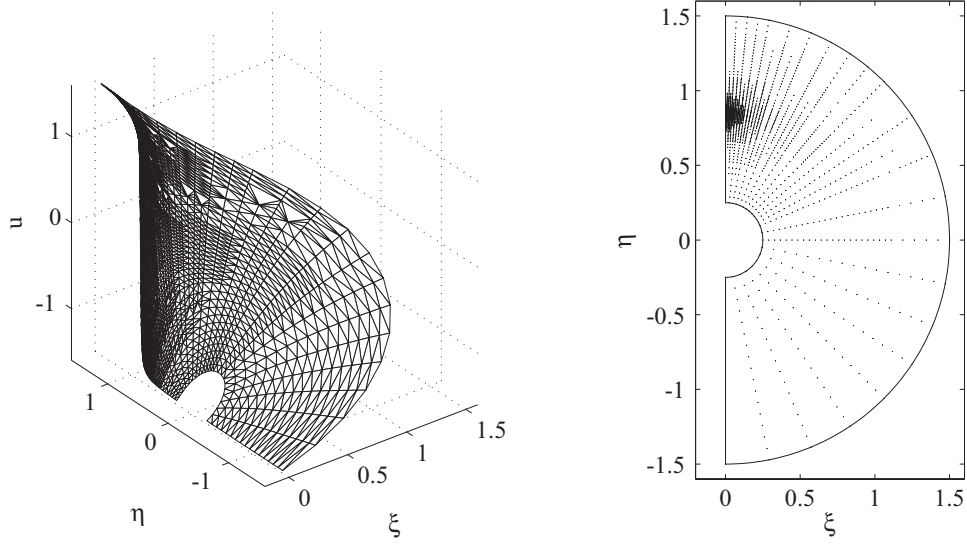


Figure 4.26: Test 7: Numerical solution (left), and the adaptively refined irregular grid (right), generated by the adaptive method with  $p = 6$ ,  $n = 4$ , and  $\varepsilon = 10^{-3}$ .

approximately  $-2.1$  for  $p = 4$  and  $n = 4$ ,  $-2.99$  for  $p = 6$  and  $n = 4$ , and  $-2.98$  for  $p = 6$  and  $n = 6$ . Thus, for these test problem,  $N$  is of  $O(\varepsilon^{-4.2/d})$  for  $p = 4$  and approximately  $O(\varepsilon^{6/d})$  for  $p = 6$ . It can be seen that they agree well with (3.32). Now let us examine the accuracy of the numerical solution. The slopes of the log-log plots of the errors in discrete maximum norm versus  $N$  indicate error in numerical solution behaves like  $O(N^{-3/d})$  for  $p = 4$  and  $n = 4$ ,  $O(N^{-5.6/d})$  for  $n = 6$  and  $n = 4$ , and  $O(N^{-5.3/d})$  for  $n = 6$  and  $n = 6$ . In this problem, the rate of convergence for each combination of  $p$  and  $n$  is higher than  $O(N^{\min(p-2,n)/d})$  predicted by (3.52). Indeed, for the range of threshold parameters employed, the errors in the numerical solutions obtained with  $p = 6$  are of the same order with the threshold values. For  $p = 4$  and  $n = 4$ , the error is approximately proportional to  $O(\varepsilon^{0.7})$ . It can be observed that results exhibit similar behavior to those observed in the numerical experiments of Section 4.4. The error of numerical solutions tend

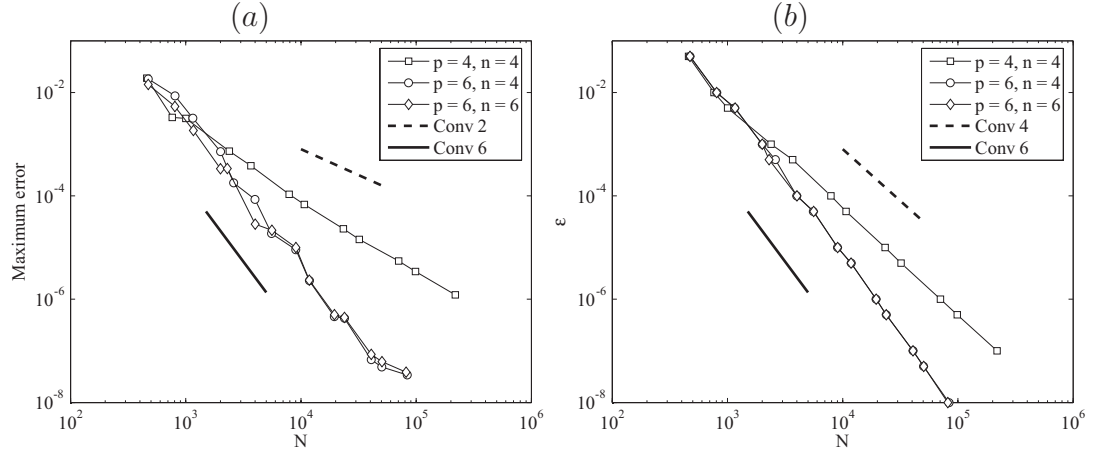


Figure 4.27: Test 7: (a)  $L_{\mathbf{V},\infty}$ -error as a function of number of grid points,  $N$ , as a result of varying  $\varepsilon$  and (b)  $\varepsilon$  versus  $N$ .

to agree to some degree with  $O(N^{\min(p-2,n)/d})$  and, in some cases, error is of the same order with the threshold value.

**Test 8:** Consider the Poisson equation defined on the domain shown in Figure 4.28 with Dirichlet boundary conditions. The right hand side of the equation and the Dirichlet boundary conditions are chosen so that the solution of the problem is given by

$$u(\mathbf{x}) = \left[ \delta^2 + \left( \xi_1 - \frac{3}{4} \right)^2 + \left( \xi_2 - \frac{1}{4} \right)^2 \right], \quad (4.44)$$

where the value of regularizing parameter,  $\delta$ , is set to  $1/100$ .

In the numerical calculation, the transfinite interpolation is computed with the following parametrized boundary edges:

$$\begin{aligned} \mathbf{r}_{x_1 0}(s) &= \left[ l(2s-1), \quad (1-h_2)(2s-1)^2 + h_2^2 \right]^T, \\ \mathbf{r}_{x_1 1}(s) &= \left[ l(2s-1), \quad -(1-h_2)(2s-1)^2 - h_2^2 \right]^T, \end{aligned}$$

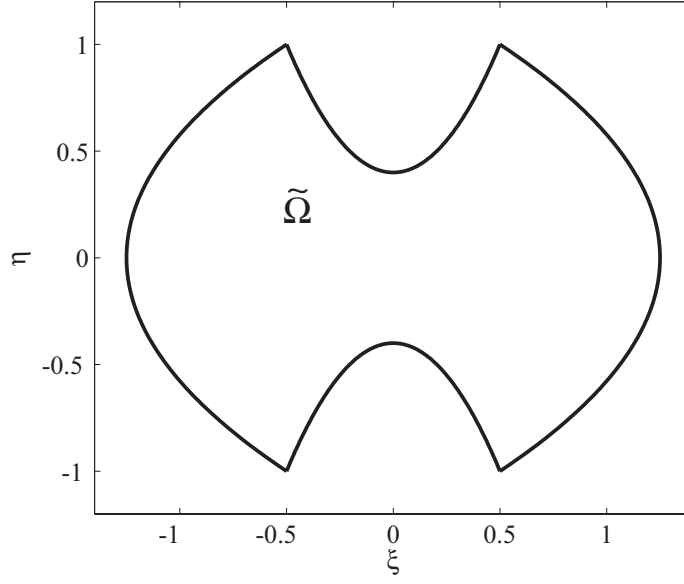


Figure 4.28: Physical domain  $\tilde{\Omega}$  of Test 8.

$$\begin{aligned} \mathbf{r}_{0x_2}(s) &= \begin{bmatrix} h_1(2s-1)^2 - h_1 - l, & 2s-1 \end{bmatrix}^T, \\ \mathbf{r}_{1x_2}(s) &= \begin{bmatrix} -h_1(2s-1)^2 + h_1 + l, & 2s-1 \end{bmatrix}^T. \end{aligned}$$

where  $h_1 = 3/4$ ,  $h_2 = 2/3$ , and  $l = 1/2$ . Figure 4.29 shows the approximate solution and the adaptively refined grid produced by the adaptive method with  $p = 6$ ,  $n = 4$ , and  $\varepsilon = 10^{-4}$ . It can be noticed that the adaptive grid has fine resolution in the vicinity of the near singularity and coarse resolution in the smooth regions. Figure 4.30(a) shows the log-log plots of the error in  $L_{\mathbf{V},\infty}$ -norm in the numerical solution versus the number of grid points,  $N$ , as a results of varying the threshold parameters  $\varepsilon$  (from  $10^{-2}$  to  $10^{-8}$ ) for different  $p$  and  $n$ . The dependence between  $N$  and  $\varepsilon$  is included in Figure 4.30(b). The slopes of the log-log plots of  $\varepsilon$  versus  $N$  are approximately  $-p/2$ , *i.e.*  $N = O(\varepsilon^{-2/c_1})$ , where

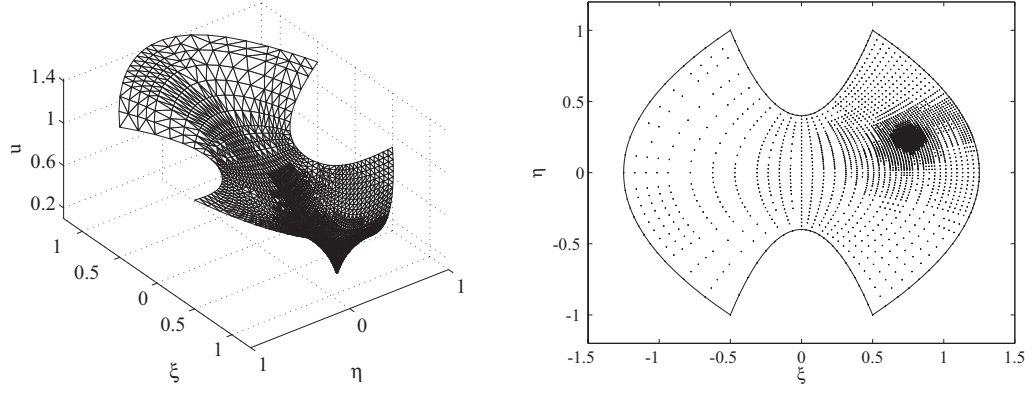


Figure 4.29: Test 8: Numerical solution (left), and the adaptively refined irregular grid (right), generated by the adaptive method with  $p = 6$ ,  $n = 4$ , and  $\varepsilon = 10^{-4}$ .

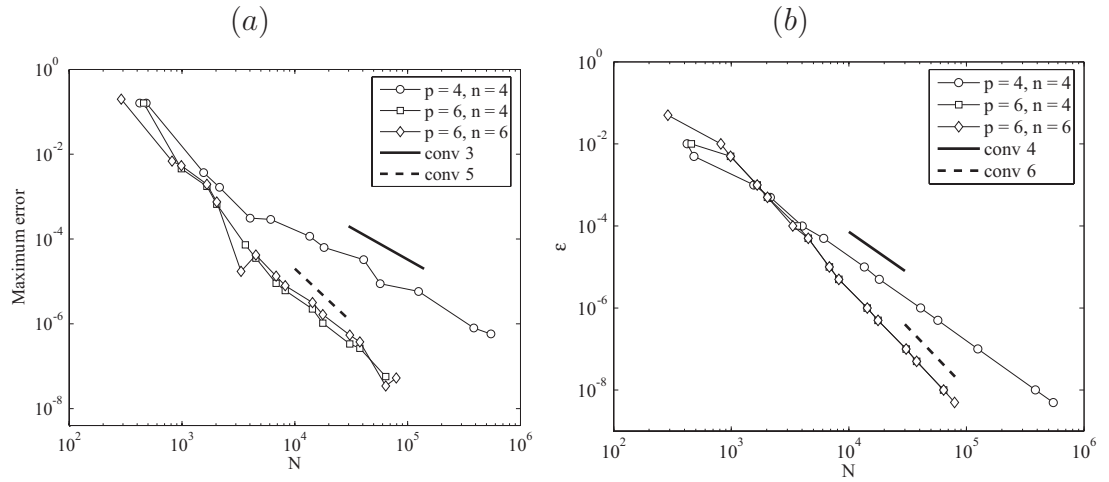


Figure 4.30: Test 8: (a)  $L_{\mathbf{V},\infty}$ -error as a function of number of grid points,  $N$ , as a result of varying  $\varepsilon$ , and (b)  $\varepsilon$  versus  $N$ .

$c_1$  is approximately  $p$ . Note that these results are in good agreement with (3.32). In this particular problem, the error in discrete maximum norm in the numerical solution behaves like  $O(N^{-c_2/2})$ , where  $c_2$  is approximately 2.8 for  $p = 4$  and  $n = 4$ , 5.6 for  $p = 6$  and  $n = 4$ , 4.9 for  $p = 6$  and  $n = 6$  (these values are the slopes of the linear curves fitted with a least square approximation). It can be observed that the rate of convergence for each combination of  $p$  and  $n$  is slightly higher than  $O(N^{\min(p-2,n)/2})$  predicted by (3.52). Note that, for this problem, the error in the discrete maximum norm is proportional to  $O(\varepsilon^{c_3})$ , where  $c_3$  is approximately 0.7 for  $p = 4$  and  $n = 4$ , 0.9 for  $p = 6$  and  $n = 4$ , 0.8 for  $p = 6$  and  $n = 6$ . For  $p = 6$ , although  $c_3$  indicates that the error converges slightly slower than  $O(\varepsilon)$ , for the range of threshold values used the errors in the numerical solutions for  $p = 6$  are in fact of same order as the threshold values. It is remarked that the adaptive method (in terms of grid points generated and errors in the numerical solution as a results of varying threshold values) shows a similar behavior as in solving the problems without using a transformation.

#### 4.7 Some comments on lifted wavelets

We have also investigated the use of lifted interpolating wavelets in the present framework; *i.e.* the use of  $d$ -D lifted wavelets for grid adaption and interpolation, while derivative approximations are still computed by finite differences. Simply explained, lifted interpolating wavelets can be thought of as better versions of interpolating wavelets. Note that interpolating wavelets are not wavelets in a strict sense since they do satisfy the so-called vanishing moment property (*i.e.*,  $\int x^n \psi_{j,k}(x) dx \neq 0$  for integer  $n \geq 0$ ), and, since their duals consists of a linear combination of Dirac delta functions, they do not form a bi-orthogonal system of

$L_2$  space (they are just dual pairs). A lifting scheme [124] can be used to construct the bi-orthogonal system of  $L_2$  by starting from interpolating wavelets<sup>3</sup>. The scheme results in lifted wavelets and their dual counterparts which now form a bi-orthogonal system of  $L_2$ . Thus lifted wavelets are applicable to a wider class of functions. The lifting scheme also facilitates a fast transform for calculating wavelet amplitudes (which are integral quantities) starting simply from function values (for the interpolating wavelet case)! In 1-D, the lifted wavelet transform on a full grid associated with interpolating wavelets can be described roughly as follows: (i) given the data  $\{f_{j,k}\}$ , one finds  $\{d_{j-1,k}\}$  by mean of (2.20); (ii) then the wavelet coefficients  $d_{j-1,k}$  are used to “update” the conventional interpolating scaling function values  $\{f_{j-1,k} \equiv f_{j,2k}\}$  to obtain the updated scaling function coefficients  $\{\tilde{f}_{j-1,k}\}$ ; (iii) assign  $\{\tilde{f}_{j-1,k}\} \rightarrow \{f_{j-1,k}\}$  and then repeat (i)-(iii) for level  $j - 1$ . Note that the “update” depends on the lifting scheme used. The inverse transform can be obtained by simply undoing the forward transform. We have considered  $d$ -D lifted wavelets resulting from lifting  $d$ -D interpolating wavelets associated with the MRA (this lifting scheme ensures the zero mean of lifted wavelets). The fast wavelet transform and its inverse are simply the generalization of the procedure described above. In this case, a fast transform and the inverse which maps function values on an irregular grid to (approximate) lifted wavelet amplitude and *vice versa*, can be devised provided that the minimum index condition for  $d$ -D interpolating wavelets is fulfilled (see Chapter 3). Note that these fast transforms are slightly more expensive than *AFWT* and *AIWT* due to the update step. We also note that in this case we do not have an algo-

---

<sup>3</sup>Strictly speaking, the lifting scheme furnish the so-called algebraic stage of the construction. The question on whether or not the resulting lifted wavelets and their duals are functions which form the bi-orthogonal system of  $L_2$  must be checked separately. However, it is common understanding that when mentioning lifted wavelets, one means lifted wavelet system that constitutes  $L_2$  space.

rithm similar to *LAIWT* for interpolation, the interpolation must be carried out by the inverse lifted wavelet transform. We have used these fast lifted transform and its inverse in place of *AFWT* and *AIWT* in the adaptive algorithms described in this Chapter for numerical solutions of PDEs. Numerical experiments on the Poisson problems show that there is not much gain (in fact mostly there is no gain) in terms of accuracy from this approach. With Jacobian preconditioning, it appears to help in reducing the number of iterations required in the iterative solvers. However, the reduction in the number of iterations is not substantial; this computational gain seems to be offset by the additional work required in the lifting procedure. Based on these observation, the use of lifting in the present framework appears not so appealing, thus it was decided not to pursue such direction for the moment. However, we believe that lifted wavelets are well suited to Galerkin-related methods since lifted wavelets form stable basis and can be fully exploited in a Galerkin scheme.



## CHAPTER 5

### APPLICATION OF WAVELET METHODS TO INCOMPRESSIBLE NAVIER-STOKES EQUATIONS

In this chapter, we describe the extension of the adaptive wavelet method to solve incompressible flows in 2- and 3-D. The algorithm for incompressible flow in the primitive variable formulation makes use of a so-called fractional step method for time discretization. The fractional step method in essence provides a means to decouple the pressure and velocity variables. In one time step, it splits the numerical solution of the problem into the numerical solution of convection-diffusion (prediction step) and Poisson (correction step) subproblems. Here, we consider exclusively the adaptive algorithm on the irregular grid of the MRA approach.

In the remainder of this chapter, the governing equations for incompressible flow are recalled first. Subsequently, we describe the dynamically adaptive algorithm for the solution of such flow problems. We then discuss an approach for solving the Poisson-Neumann problem (the subproblem of the correction step) which needs special treatment due to the non-uniqueness of the solution. To demonstrate the algorithm, we simulate the 2-D flow in the lid-driven cavity, the 2-D differentially-heated cavity with large Rayleigh numbers, and the 3-D differentially-heated cavity for various Rayleigh numbers. The numerical results, when possible, are compared with accurate previously published results.

## 5.1 Governing equations

In this study, we restrict our consideration to flows of incompressible (or Boussinesq) fluids. The governing equations, which consist of the continuity, momentum, and energy equations are given in the dimensionless form by

$$\nabla \cdot \mathbf{u} = 0, \quad (5.1)$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = -\nabla p + \frac{1}{Re} \nabla^2 \mathbf{u} - \frac{Gr}{Re^2} T \mathbf{n}, \quad (5.2)$$

$$\frac{\partial T}{\partial t} + (\mathbf{u} \cdot \nabla) T = \frac{1}{RePr} \nabla^2 T, \quad (5.3)$$

with appropriate boundary and initial conditions. The equations are made dimensionless with respect to the characteristic length  $L$ , velocity  $U$ , time  $L/U$ , and temperature scale  $\Delta T$ . Here,  $\mathbf{u} = \mathbf{u}^*/U$  represents the velocity vector (for 2D problems  $\mathbf{u} = (u, v)^T$  and for 3-D problems  $\mathbf{u} = (u, v, w)^T$ ),  $p = (p^* - P_r)/\rho U^2$  is the pressure,  $T = (T^* - T_r)/\Delta T$  denotes the temperature (note  $T_r$  and  $P_r$  are respectively the reference temperature and reference pressure; dependent variables with superscripts denote dimensional variables), and  $\mathbf{n}$  is the unit vector in the direction of gravity. The dimensionless parameters appearing in the equations are the Reynolds number,  $Re = UL/\nu^2$ , the Grashof number,  $Gr = \beta g \Delta T L^3/\nu^2$ , and the Prandtl number,  $Pr = \nu/\alpha$ . Here  $\nu$  denotes the kinematic viscosity,  $g$  the magnitude of the gravitational acceleration,  $\beta$  the volumetric thermal expansion coefficient, and  $\alpha$  the thermal diffusivity.

## 5.2 Fractional step method

To solve an incompressible flow problem described by (5.1)-(5.3), we discretize the equations in time using a variant of the Chorin-type projection method. At

each time step, we first compute the temperature by solving

$$\frac{T^{m+1} - T^m}{\Delta t} + \frac{1}{2}(\tilde{\mathbf{u}} \cdot \nabla)(T^{m+1} + T^m) = \frac{1}{2RePr} \nabla^2(T^{m+1} + T^m), \quad (5.4)$$

where  $\tilde{\mathbf{u}}$  is a second order extrapolation of the velocity field at  $t^{m+1}$  given by  $\tilde{\mathbf{u}} = (1 + r)\mathbf{u}^m - r\mathbf{u}^{m-1}$  with  $r = \Delta\tau/\Delta t$ ,  $\Delta t = t^{m+1} - t^m$  and  $\Delta\tau = t^m - t^{m-1}$ . Note that (5.4) is a 2<sup>nd</sup> order linearized trapezoidal approximation. An intermediate velocity field  $\hat{\mathbf{u}}$  is calculated by considering the linearized trapezoidal approximation of the momentum equation with a provisional pressure  $\tilde{p}$ :

$$\frac{\hat{\mathbf{u}} - \mathbf{u}^m}{\Delta t} + \frac{1}{2}(\tilde{\mathbf{u}} \cdot \nabla \hat{\mathbf{u}} + \mathbf{u}^m \cdot \nabla \mathbf{u}^m) = -\nabla \tilde{p} + \frac{1}{2Re} \nabla^2(\hat{\mathbf{u}} + \mathbf{u}^m) - \frac{Gr\mathbf{n}}{2Re^2}(T^{m+1} + T^m). \quad (5.5)$$

The provisional pressure  $\tilde{p}$  is a prediction of the pressure field  $p^{m+1}$ . It is usually set to the known pressure at time  $t^m$  or simply set to zero. As a results, the intermediate velocity field  $\hat{\mathbf{u}}$  is not divergence-free. In the next step, the true velocity field  $\mathbf{u}^{m+1}$  is obtained by solving the system

$$\frac{\mathbf{u}^{m+1} - \hat{\mathbf{u}}}{\Delta t} = -\nabla \phi, \quad (5.6)$$

$$\nabla \cdot \mathbf{u}^{m+1} = 0, \quad (5.7)$$

where  $\phi$  is an auxiliary potential function. Equations (5.6) and (5.7) lead to the following Poisson equation for  $\phi$ :

$$\nabla^2 \phi = \frac{1}{\Delta t} \nabla \cdot \hat{\mathbf{u}}. \quad (5.8)$$

When required, the pressure field at the new time step can be computed from the formula below

$$p^{m+1} = \tilde{p} + \phi - \frac{1}{2Re} \Delta t \nabla^2 \phi. \quad (5.9)$$

We remark that other time discretization schemes, such as second order backward difference can be used, in the calculation of temperature and intermediate velocity fields (in this case one would have a different formula for calculating  $p^{m+1}$ ).

With the fractional step method, finding the numerical solution of incompressible flow reduces to solving subproblems of convection-diffusion-type (or Helmholtz-type if the convection term is discretized via an explicit time discretization scheme) and Poisson-type. These subproblems constituting, of (5.4), (5.5) and (5.8), must be augmented by initially and boundary conditions. The proper choices of boundary conditions for (5.5) and (5.8) are unclear and in fact there is some disagreement in the literature. In this work, the boundary condition on  $\hat{\mathbf{u}}$  are taken to be the same as those for  $\mathbf{u}^{m+1}$ . Subsequently, boundary conditions for  $\phi$  of Neumann type are deduced from (5.6) in conjunction with the condition of global mass conservation.

### 5.3 Dynamically adaptive algorithm for incompressible flow

Above, we have reduced the time-dependent problem to a time-marching procedure. Now the algorithm of Section 4.5 can be used for an adaptive solution of the incompressible flow. In this study, the irregular grid  $\mathbf{\mathcal{V}}$  is constructed from thresholding the velocity and temperature fields only (we exclude  $p$  since in incompressible flow it plays the role of a Lagrange multiplier and does not have much meaning by itself). Certainly, one can also adapt the grid based on other derived flow variables, such as vorticity. For this purpose, we introduce the threshold parameter  $\varepsilon = \{\varepsilon_s, s = u, v, w, T\}$  associated with these variables. The dynamically

adaptive algorithm to integrate the equations can then be summarized as follows:

- (i) use the solution from the previous time step as initial condition, solve (5.4), (5.5) and (5.8) to obtain the approximate solutions  $T^{m+1}$ ,  $\mathbf{u}^{m+1}$  and  $\phi$  on the irregular grid  $\mathcal{V}^m$ . Note that we first solve (5.4), then (5.5), and thereafter (5.8). The new velocity field  $\mathbf{u}^{m+1}$  is subsequently obtained from (5.6);
- (ii) determine grids based on thresholding the magnitude of wavelet amplitudes, with  $s$  standing for either  $u$ ,  $v$ ,  $w$  or  $T$ :  $\hat{\mathcal{V}}_s = \{x_{j,\lambda} \mid (j, \lambda) \in \Lambda_{\mathcal{V}^m}, |d_{j,\lambda}^{s,m+1}| \geq \varepsilon_s\}$ , where  $\Lambda_{\mathcal{V}^m} = \{(j, \lambda) \mid x_{j,\lambda} \in \mathcal{V}^m\}$  and  $\{d_{j,\lambda}^{s,m+1}\}$  are the corresponding wavelet amplitudes of  $\mathbf{u}^{m+1}$  and  $T^{m+1}$ , respectively. With 5.5  $\hat{\mathcal{V}} = \bigcup_s \hat{\mathcal{V}}_s$ , the sparse grid for the next time step is then given by

$$\mathcal{V}^{m+1} = \{x_{J_0, \mathbf{k}}\} \cup \hat{\mathcal{V}} \bigcup_{(j,\lambda) \in \Lambda_{\hat{\mathcal{V}}}} \mathcal{N}_{j,\lambda}, \quad (5.10)$$

where  $\mathcal{N}_{j,\lambda}$  is the set of neighboring points of  $x_{j,\lambda}$ . Note that neighboring points are added in order to accommodate possible advection and sharpening of solution features;

- (iii) assign a zero value to wavelet amplitudes  $d_{j,\lambda}^{s,m+1}$  associated with new grid points, *i.e.* set  $d_{j,\lambda}^{s,m+1} = 0$  for  $x_{j,\lambda} \in \mathcal{V}^{m+1} \setminus \mathcal{V}^m$ , and compute the inverse adaptive wavelet transform to yield  $\mathbf{u}^{m+1}$  and  $T^{m+1}$  on the new sparse grid  $\mathcal{V}^{m+1}$ ;
- (iv) assign  $(\mathbf{u}^{m+1}, T^{m+1}) \rightarrow (\mathbf{u}^m, T^m)$  and  $\mathcal{V}^{m+1} \rightarrow \mathcal{V}^m$ , increment time, and go back to step (i).

The initial sparse grid  $\mathcal{V}^0$  is obtained from the thresholding of the initial conditions.

To this end let us mention that to solve systems of equations arising from the discretization of (5.4) and (5.5), we use nonstationary iterative solvers (such as BiCGSTAB, CGS, and TFQMR) without any preconditioning. The systems are well conditioned when  $\Delta t$  is small. For the Poisson-Neumann problem, the linear system of equations is solved using the nonstationary iterative solvers with ILU (Incomplete-LU decomposition) with drop tolerance as a preconditioner [114].

#### 5.4 Poisson-Neumann problem

As mentioned above and discussed in the next section, the boundary conditions for the Poisson equation (5.8) are of Neumann type. It is known that this type of problem has a solution only when the integral of the source term over the domain equals the boundary integral of the normal derivative of the solution (*i.e.* the boundary condition), and such a solution is only determined up to an arbitrary constant. Note that this condition is the so-called compatibility condition. When it is not fulfilled, the problem has no solution. The linear system resulting from discretization of the Poisson-Neumann problem inherits this type of singularity.

After discretization, the Poisson-Neumann problem reduces to the system of linear algebraic equations

$$\mathbf{A}\phi = \mathbf{f}, \quad (5.11)$$

where the matrix  $\mathbf{A}$  corresponds to the discrete Laplacian within the domain with incorporated Neumann boundary conditions, the vector  $\phi$  is the discrete representation of  $\phi$  at the grid points, and  $\mathbf{f}$  consist of the values of  $\nabla \cdot \hat{\mathbf{u}}$  within the domain and of the specified Neumann conditions at the boundaries. Since the boundary conditions are purely of Neumann type, the discretization matrix is singular with one zero eigenvalue (one-rank deficient). An eigenvector associated

with the null eigenvalue is a constant vector, *i.e.*  $C\mathbf{1}^T$ . In order for such a linear system to have a solution, the vector  $\mathbf{f}$  must belong to the column space of matrix  $\mathbf{A}$ . In other words,  $\mathbf{f}$  must be orthogonal (with respect to the usual inner product) to the left null space vector of  $\mathbf{A}$ , namely  $\mathbf{v}$  where,  $\mathbf{v}^T \mathbf{A} = \mathbf{0}^T$ , thereby yielding the discrete solvability condition

$$\mathbf{v}^T \mathbf{f} = \mathbf{0}. \quad (5.12)$$

Note that this condition is usually attained in numerical methods by using a staggered grid. In general, this condition is not fulfilled in the presented method and (5.11) does not always have a solution (and thus applying an iterative solver to (5.11), while not satisfying (5.12), will fail to yield a solution).

A straightforward way to remedy this difficulty is to impose a Dirichlet condition at one of the boundary points (*i.e.*, setting the value of  $\phi$  at one of the boundary points to an arbitrary constant), so that the following system is considered

$$\mathbf{A}^* \phi = \mathbf{f}^*. \quad (5.13)$$

The discretization matrix of the modified system is different from  $\mathbf{A}$  in only one row. This replaces the singular system (5.11) with a nonsingular linear system of equations. In some cases, this approach produces an accurate solution; however, from our numerical experiments, it often results in numerical artifacts (in the vicinity of where the Dirichlet condition is enforced) especially in cases where a large value of Reynolds number,  $Re$ , (or Rayleigh number,  $Ra$ ) is used. The reason for this is most likely due to the fact that the necessary Neumann condition is not forced explicitly and thus in general not satisfied at the chosen point.

In this study, to circumvent this difficulty, we use approaches that modify the source term in the Poisson equation [22, 75, 108]:

$$\mathbf{A}\boldsymbol{\phi} = \mathbf{f} - \alpha\mathbf{v}, \quad (5.14)$$

where  $\mathbf{v}$  is a suitable normalized vector (*i.e.*  $\mathbf{v}^T\mathbf{v} = 1$ ), and the constant  $\alpha$  is selected so that the resulting right hand side vector satisfies the solvability condition. When the left null space vector  $\mathbf{c}$  is known, one may impose the solvability condition on (5.14) to obtain  $\alpha = \mathbf{c}^T\mathbf{f}/\mathbf{c}^T\mathbf{v}$ . In practice, the left null space vector  $\mathbf{c}$  may be computed directly or in some cases obtained by inspection. Note that since the matrix  $\mathbf{A}$  in the presented method is non-symmetric, the left and right null space vectors are not necessarily identical (in most discretization methods, the right null space vector can be easily determined by inspection; unfortunately, this is not the case for the left null space vector).

Henshaw [75] suggests a method based on the above procedure. In his approach, (5.11) is replaced by the augmented linear nonsingular system

$$\begin{pmatrix} \mathbf{A} & \mathbf{r} \\ \mathbf{r}^T & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\phi} \\ \alpha \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \beta \end{pmatrix}, \quad (5.15)$$

where  $\beta$  is an arbitrary constant and  $\mathbf{r}$  is the nulls space vector of  $\mathbf{A}$ . Effectively, the solution of the augmented system (5.15) satisfies

$$\alpha = \mathbf{c}^T\mathbf{f}/\mathbf{c}^T\mathbf{r}, \quad \mathbf{A}\boldsymbol{\phi} = \mathbf{f} - \alpha\mathbf{r}, \quad \mathbf{r}^T\boldsymbol{\phi} = \beta. \quad (5.16)$$

While the above approach yields satisfactory results for many cases, we have noticed numerical artifacts, especially in the differentially-heated cavity problem



with large Rayleigh numbers.

The approach we use in the current work is based on the actual computation of the left null space vector  $\mathbf{c}$ . To determine the left null space vector, *i.e.* finding a vector  $\mathbf{c}$  such that

$$\mathbf{A}^T \mathbf{c} = \mathbf{0}, \quad (5.17)$$

where  $\mathbf{0} = [0, 0, \dots, 0, 0]^T$  is the zero vector, we consider the following non-homogeneous linear system of equations

$$\tilde{\mathbf{A}} \tilde{\mathbf{c}} = \mathbf{e}, \quad (5.18)$$

where the matrix  $\tilde{\mathbf{A}} = \mathbf{A}^T$ , except for the  $i^{\text{th}}$  row whose entries are given by  $[\tilde{\mathbf{A}}]_{i,j} = \delta_{i,j}$ , and  $\mathbf{e} = \mathbf{0}$ , except for the  $i^{\text{th}}$  entry which has a nonzero value (the choice of the  $i^{\text{th}}$  and value of entry  $e_i$  is arbitrary). This linear system of equations is nonsingular and has a unique solution. It can be observed that the solution  $\tilde{\mathbf{c}}$  of (5.18) satisfies every linear equation of (5.17). This is obvious for any  $m^{\text{th}}$  equation ( $m \neq i$ ). For the  $i^{\text{th}}$  equation, since the  $i^{\text{th}}$ -row of  $\mathbf{A}^T$  can be written as a linear combination of the other row vectors (owing to the fact that  $\mathbf{A}^T$  is rank-one deficient), the solution  $\tilde{\mathbf{c}}$  also satisfies the  $i^{\text{th}}$  linear equation of (5.17). Thus, any vector  $\mathbf{c} = c_1 \tilde{\mathbf{c}}$ , where  $c_1$  is any constant, is a left null space vector of the matrix  $\mathbf{A}$ .

Furthermore, instead of employing an arbitrary vector  $\mathbf{v}$  to modify the right hand side vector (as in (5.14)), we use the left null space vector  $\mathbf{c}$  for such modification. More precisely, we consider

$$\mathbf{A} \boldsymbol{\phi} = \mathbf{f} - \alpha \mathbf{c}, \quad (5.19)$$

where  $\mathbf{c}$  is the normalized left null space vector (*i.e.*,  $\mathbf{c}^T \mathbf{c} = 1$ ). Applying the solvability condition yields  $\alpha = \mathbf{c}^T \mathbf{f}$ . It can be observed, by multiplying both side of (5.19) by the transpose matrix  $\mathbf{A}^T$ , that the solution of (5.19) is effectively a least square solution. However, in this work, we find that the iterative solution (with ILU) of the straightforward least square problem converges much slower than the iterative solution of system (5.19). The reason for this is likely due to the fact that the condition number of  $\mathbf{A}^T \mathbf{A}$  grows substantially faster than that of  $\mathbf{A}$  (in this case the condition number is defined as a ratio of the maximum singular to the minimum nonzero singular value). Note that we use the ILU of  $\tilde{\mathbf{A}}$  as the preconditioner for nonstationary iterative solvers when solving (5.17) for the left null space vector and the ILU of  $\mathbf{A}^*$  (see 5.13 for definition) as a preconditioner when solving the system (5.19) for  $\phi$ .

## 5.5 2-D lid-driven cavity

The physical configuration of the 2-D lid-driven cavity is that of a square cavity of width  $L$  with a sliding top wall and all other walls being rigid, and which is filled with an incompressible Newtonian fluid. The equations governing the flow problem in dimensionless form are given by (5.1) and (5.2) (without the buoyancy term). The fluid is set in motion by moving the top boundary (lid) at constant  $U$  horizontal velocity. Note that in this case, the reference length, velocity, and time scale correspond respectively to the cavity width  $L$ , velocity of the lid  $U$ , and  $L/U$  respectively and note also that all flow quantities are defined on the dimensionless domain  $\Omega = (0, 1)^2$ . The boundary conditions are defined by

$$\mathbf{u}(x, y, t) = \mathbf{f}(x, y, t) \quad \text{on } \partial\Omega \quad (5.20)$$

where

$$\mathbf{f}(x, y, t) \equiv \begin{cases} (1, 0)^T & \text{on } y = 1, \\ (0, 0)^T & \text{on } x = 0, 1, \text{ and } y = 0, \end{cases} \quad (5.21)$$

and the initial condition is given by  $\mathbf{u}(\mathbf{x}, 0) = \mathbf{0}$ .

In the solution procedure via the fractional step method, the boundary condition for  $\hat{\mathbf{u}}$  are taken to be identical to those of  $\mathbf{u}$ :

$$\hat{\mathbf{u}} = \mathbf{u} = \mathbf{f}^{m+1}, \quad \text{on } \partial\Omega. \quad (5.22)$$

Now due to this choice of boundary conditions, we have that  $\hat{\mathbf{u}} \cdot \mathbf{n} = \mathbf{f}^{m+1} \cdot \mathbf{n} = \mathbf{0}$ , which along with (5.6) results in the following homogeneous Neumann boundary conditions for  $\phi$ :

$$\nabla\phi \cdot \mathbf{n} = 0 \quad \text{on } \partial\Omega. \quad (5.23)$$

It can easily be seen from (5.6) that

$$\int_{\partial\Omega} \mathbf{u}^{n+1} \cdot \mathbf{n} \, d\Gamma = \int_{\partial\Omega} \hat{\mathbf{u}} \cdot \mathbf{n} \, d\Gamma - \Delta t \int_{\partial\Omega} \nabla\phi \cdot \mathbf{n} \, d\Gamma. \quad (5.24)$$

By making use of (5.22) and (5.23), it can be seen that the global mass constraint  $\int_{\partial\Omega} \mathbf{u}^{n+1} \cdot \mathbf{n} \, d\Gamma = 0$  is identically satisfied (and thus ensures satisfaction of the compatibility condition for the continuum case).

In all numerical calculations of this particular problem, we discretize the convection term using the 2<sup>nd</sup> order Adam-Bashforth scheme. The provisional pressure  $\tilde{p}$  is set equal to zero. The adaptive method with the basis of order  $p = 4$  and threshold parameters  $\epsilon = \{5 \times 10^{-4}, 5 \times 10^{-4}\}$ , and  $\epsilon_a = 2\epsilon$  are used <sup>1</sup>. The deriva-

---

<sup>1</sup>Note that in the grid construction of these calculations, neighboring points are added surrounding the grid points whose associated wavelet amplitude are greater than  $\epsilon$ . Since neighboring points are not added near every essential point (only near those whose their amplitude are also

tive approximations are obtained via nine-point finite-difference stencils ( $n = 4$ ). The number of resolution levels is limited to  $J - j_0 = 6$  with  $j_0 = 3$  (9 points in each direction at the coarsest scale). This results in a finest grid spacing of size  $\Delta x = \Delta y = 1/512$ . Such choice of  $J$  is selected for the purpose of comparison with benchmark results. Note that if the goal would have been to obtain the best solutions that can be produced by the adaptive method, then  $J$  would have been allowed to vary to whatever number of levels necessary (*i.e.*, the parameter determining the accuracy is threshold parameters not the finest level set), as done for example in the previous chapter. Time integration is performed until steady state, defined when  $\|\mathbf{u}^{m+1} - \mathbf{u}^m\|_\infty < 5 \times 10^{-5}$ , is reached.

Figure 5.1 shows streamlines and active collocation points obtained for  $Re = 1000$  at different dimensionless times as well as at steady state. Formation of eddies can be observed very distinctly and the pattern agrees qualitatively quite well with that given in [73]. The distribution of grid points clearly demonstrates that the dynamically adaptive algorithm is able to track the moving structures properly. As a results, the number of grid points in  $\mathcal{V}^m$ ,  $N = \dim \mathcal{V}^m$ , varies with the local demands of the solution for fixed threshold parameters. For these particular threshold values, the maximum number of grid points required during the simulation is found to be approximately  $N = 4400$  points. At steady state, the compression ratio (the ratio of the number of active points to that corresponding to a uniform mesh of equivalent resolution) is approximately 0.017. Note that due to the velocity singularities in the upper corners of the cavity, the adaptive algorithm demands a considerable number of points in confined neighborhoods of these locations. Solving the corresponding de-singularized problem would reduce the number of such points substantially [21, 119], but is not done here.

---

grater than  $\epsilon_a$ ), this results in an irregular grid with fewer number of grid points.

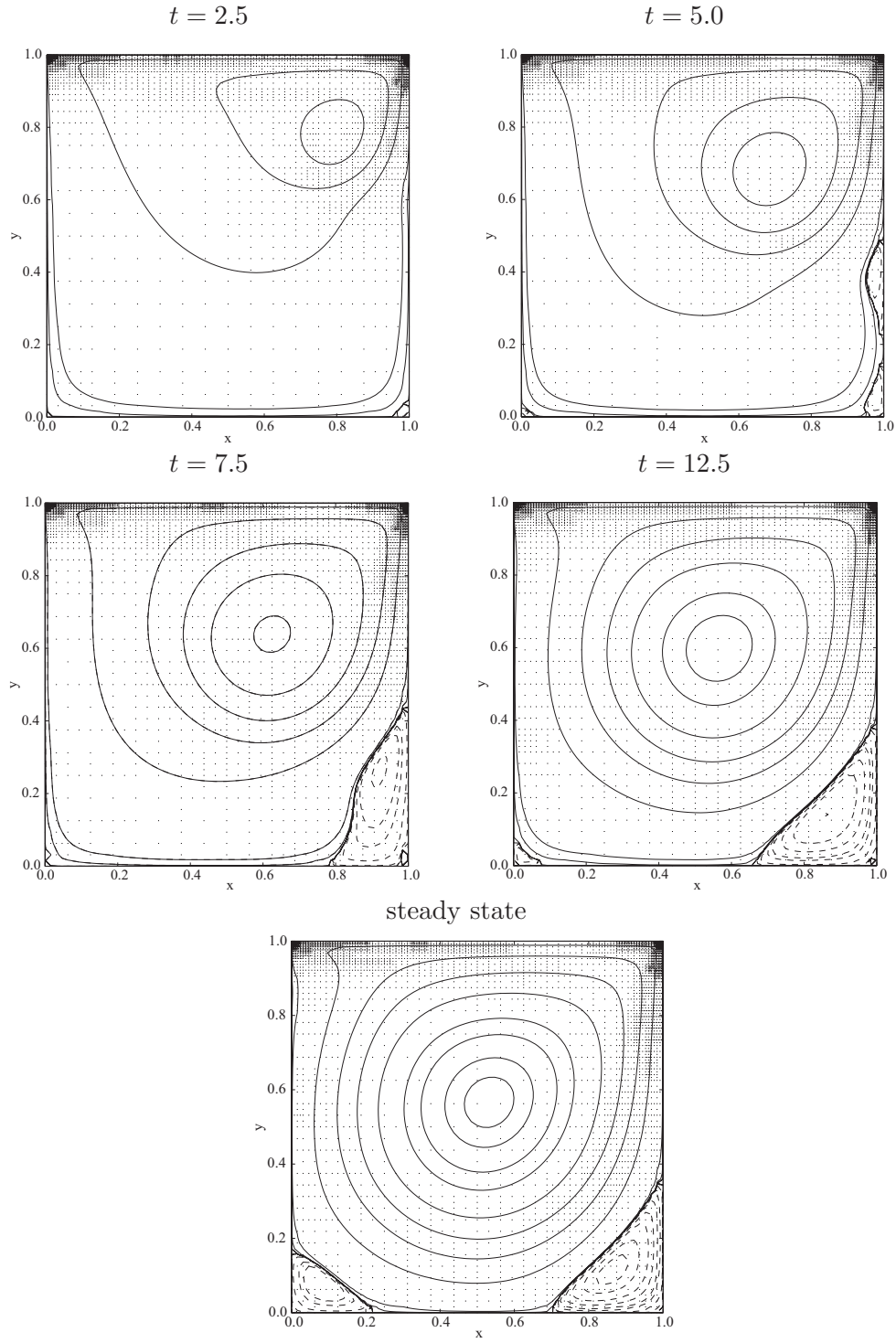


Figure 5.1: Evolutions of streamlines and dynamically adaptive grids for  $Re = 1000$  at  $t = 2.5, 5.0, 7.5, 12.5$  and steady state. The number of grid points at each time are respectively  $N = 3378, 3910, 4075, 4180$  and  $4372$ .

The streamlines and grid points of the solutions at steady state for  $Re = 400$  and 3200 are shown in Figure 5.2.

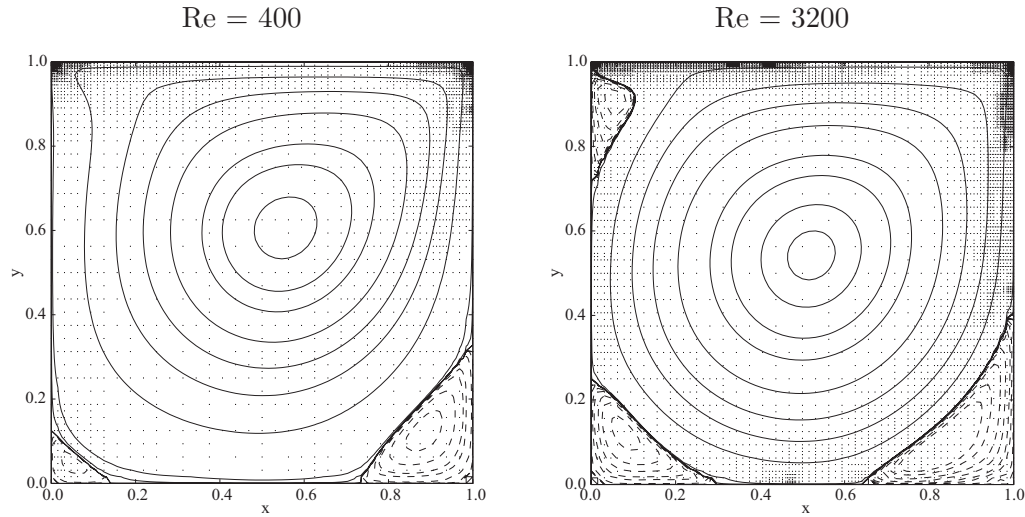


Figure 5.2: Streamlines and dynamically adaptive grids at steady state for  $Re = 400$  and 3200.

For  $Re = 3200$ , as can be seen, an additional secondary vortex appears near the upper left corner. It is worth noting that in the case of  $Re = 400$ , the algorithm yields an inaccurate structure of the secondary vortex at the lower left corner. This may be explained by the fact that such vortex is fairly weak and the values of threshold parameters used are not sufficiently small. The use of threshold parameters with appropriately small values would remedy such issue. Alternatively, in this work, the issue is overcome by using a particular strategy which takes advantage of the knowledge of the (previously reported) solution. Such strategy consists of setting the threshold parameters such that the adaption

criteria is more sensitive near the lower left corner. Specifically, adapt the grid based upon  $w(\mathbf{x}_{j,\lambda})d_{j,\lambda}$  instead of  $d_{j,\lambda}$  with the Gaussian weight function  $w(\mathbf{x}_{j,\lambda}) = 1 + 9 \exp(-25\mathbf{x}_{j+1,2\mathbf{k}+\mathbf{e}}^2)$ . However, this is not surprising since the size of this eddy (0.1237 in width and 0.1081 in length [64]) is smaller than the coarsest resolution. In addition, its strength is fairly weak as the maximum value of absolute vorticity is of the order of 0.05 (based on our computation with larger  $j_0$  and [64]).

Comparisons of steady-state velocity distributions along the mid-sections of the cavity for  $Re = 400, 1000$ , and  $3200$  with those of [64] are shown respectively in Figures 5.3. Tables 5.1-5.3 compare the intensities of the primary vortex, the lower left corner secondary vortex, and lower right corner secondary vortex with those given in [64] obtained by the use of a multigrid technique, in [21] by using a Chebyshev spectral collocation method for the de-singularized problem, in [131] by employing a block-implicit multigrid technique, and in [69] by using the finite element method. In the tables,  $\psi_{min}$  and  $\omega_{v,c}$  represent respectively the value of the streamfunction and vorticity at the vortex center. In addition, code letters after each reference indicate the type of grid (mesh) and formulation used: U-uniform grid, C-Chebyshev grid, G-graded mesh (a mesh that is a result of clustering near boundaries), US-unsteady stream function-vorticity formulation, UP-unsteady primitive variable formulation, and SP-steady primitive variable formulation. Note that resolutions of  $257 \times 257$  grids ( $129 \times 129$  for  $Re = 1000$ ) were used in the calculations reported in [64],  $160 \times 160$  modes (and subsequently collocation points) in those in [21],  $321 \times 321$  grids in those in [131], and  $129 \times 129$  nodes in those in [69]. The total number of active wavelet collocation points required at steady state in the present computations are  $N = 3165, 4180$ , and  $6170$  for  $Re = 400, 1000$ , and  $3200$ , respectively. Table 5.1-5.3 indicate that, while the

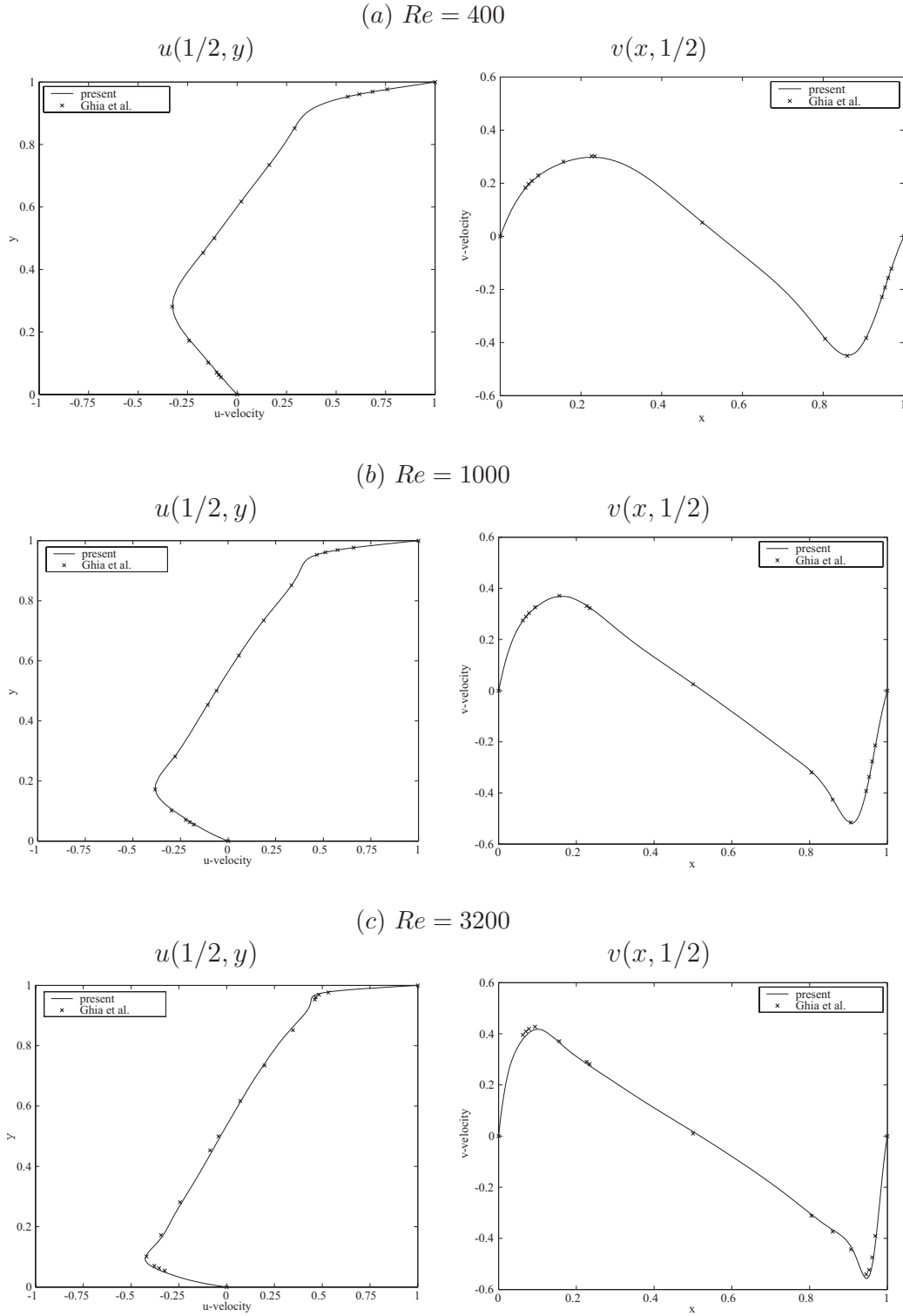


Figure 5.3: Comparison of steady state velocity profiles  $u(1/2, y)$  and  $v(x, 1/2)$  with those of [64] for (a)  $Re = 400$ , (b)  $Re = 1000$ , and (c)  $Re = 3200$ .



number of grid points are relatively small (3-32 times fewer) compared with those of the above calculations, the results obtained from the adaptive method compare favorably with that of benchmark results, which are the best results available in the literature.

## 5.6 2-D differentially-heated cavity

In this section, we consider the problem of fluid flow in a differentially-heated cavity with adiabatic top and bottom walls. Such a problem has served over the years as an excellent test-problem for assessing the accuracy of numerical methods designed for the integration of the Navier-Stokes equations in the Boussinesq limit. The reason for this is partially because boundary conditions are free of singularities (in contrast with the lid-driven cavity problem). Numerical results for a square cavity filled with air have been presented for a wide range of Rayleigh numbers,  $Ra$ . A set of benchmark solutions obtained by using Richardson extrapolation for  $10^3 \leq Ra \leq 10^6$  has been given in [30, 55]. As a result of a decrease in the boundary-layer thickness, there have been fewer accurate numerical computations performed at larger values of  $Ra$ . For  $Ra = 10^7$  and  $10^8$ , accurate results are provided in [95] where a spectral method with a relatively large number of modes is used and in [122] where a spectral Petrov-Galerkin method with divergence-free trial functions is used. It has been found by direct numerical simulations [96, 107] that the solution becomes unsteady at  $Ra \approx 2 \times 10^8$ . The flow becomes chaotic when  $Ra$  is further increased, and it is found that the quasi-periodic route of transition is followed from the onset of instability to the chaotic regime. We

TABLE 5.1

COMPARISONS OF THE INTENSITIES OF THE PRIMARY VORTEX.

$Re$	Investigators	Primary vortex		
		$\psi_{min}$	$\omega_{v,c}$	Location $(x, y)$
400	Present	-0.1131	2.3087	(0.5547, 0.6094)
	Ref. [64] (U,US)	-0.1139	2.2947	(0.5547, 0.6055)
	Ref. [131] (U,SP)	-0.1136	—	(0.5563, 0.6000)
1000	Present	-0.1173	2.0476	(0.5313, 0.5703)
	Ref. [64] (U,US)	-0.1179	2.0497	(0.5313, 0.5643)
	Ref. [21] (C,UP)	-0.1189	2.0678	(0.5384, 0.5652)
	Ref. [131] (U,SP)	-0.1173	—	(0.5438, 0.5625)
	Ref. [69] (G,UP)	-0.114	—	—
3200	Present	-0.1171	1.8594	(0.5234, 0.5390)
	Ref. [64] (U,US)	-0.1204	1.9886	(0.5117, 0.5352)
	Ref. [69] (G,UP)	-0.118	—	—

Note: U—uniform grid, C—Chebushev grid, G—graded mesh, US—unsteady stream function-vorticity formulation, UP—unsteady primitive variable formulation, SP—steady primitive variable formulation.

TABLE 5.2  
COMPARISONS OF THE INTENSITIES OF THE LOWER LEFT  
CORNER SECONDARY VORTEX.

$Re$	Investigators	Lower left corner vortex		
		$\psi_{max}$	$\omega_{v,c}$	Location $(x, y)$
400	Present	$1.5953 \times 10^{-5}$	-0.06295	(0.0547, 0.0469)
	Ref. [64] (U,US)	$1.4195 \times 10^{-5}$	-0.05697	(0.0508, 0.0469)
	Ref. [131] (U,UP)	$1.46 \times 10^{-5}$	—	(0.0500, 0.0500)
1000	Present	$2.2027 \times 10^{-4}$	-0.35628	(0.0859, 0.0781)
	Ref. [64] (U,US)	$2.3113 \times 10^{-4}$	-0.36175	(0.0859, 0.0781)
	Ref. [21] (C,UP)	$2.3345 \times 10^{-4}$	-0.35228	(0.0833, 0.0781)
	Ref. [131] (U,SP)	$2.24 \times 10^{-4}$	—	(0.0750, 0.0813)
	Ref. [69] (G,UP)	$2.0 \times 10^{-4}$	—	—
3200	Present	$9.8933 \times 10^{-4}$	-1.02891	(0.0781, 0.1172)
	Ref. [64] (U,US)	$9.7823 \times 10^{-4}$	-1.06301	(0.0859, 0.1094)
	Ref. [69] (G,UP)	$1.20 \times 10^{-3}$	—	—

Note: U-uniform grid, C-Chebyshev grid, G-graded mesh, US-unsteady stream function-vorticity formulation, UP-unsteady primitive variable formulation, SP-steady primitive variable formulation.

TABLE 5.3  
COMPARISONS OF THE INTENSITIES OF THE LOWER RIGHT  
CORNER SECONDARY VORTEX

$Re$	Investigators	Lower right corner vortex		
		$\psi_{max}$	$\omega_{v,c}$	Location $(x, y)$
400	Present	$6.4099 \times 10^{-4}$	-0.46804	(0.8828, 0.1250)
	Ref. [64] (U,US)	$6.4235 \times 10^{-4}$	-0.43352	(0.8906, 0.1250)
	Ref. [131] (U,SP)	$6.45 \times 10^{-4}$	—	(0.8875, 0.1188)
1000	Present	$1.7669 \times 10^{-3}$	-1.13399	(0.8594, 0.1094)
	Ref. [64] (U,US)	$1.7510 \times 10^{-3}$	-1.15465	(0.8594, 0.1094)
	Ref. [21] (C,UP)	$1.7297 \times 10^{-3}$	-1.10979	(0.8640, 0.1118)
	Ref. [131] (U,SP)	$1.74 \times 10^{-3}$	—	(0.8625, 0.1063)
	Ref. [69] (G,UP)	$1.76 \times 10^{-3}$	—	—
3200	Present	$2.8584 \times 10^{-3}$	-2.11655	(0.8281, 0.0859)
	Ref. [64] (U,US)	$3.1396 \times 10^{-3}$	-2.27365	(0.8125, 0.0859)
	Ref. [69] (G,UP)	$3.29 \times 10^{-3}$	—	—

Note: U—uniform grid, C—Chebushev grid, G—graded mesh, US—unsteady stream function-vorticity formulation, UP—unsteady primitive variable formulation, SP—steady primitive variable formulation.

apply the adaptive wavelet method to solve problems at sufficiently large values of  $Ra$  in order to study the applicability of the method to transitional and chaotic flows.

The differentially-heated cavity problem considers a rectangular cavity of height  $H$ , width  $W$  and filled with a Newtonian fluid. The flow is assumed to be two-dimensional and the temperature along the left and right walls of the cavity are maintained isothermally at  $T_h$  and  $T_c$ , respectively, where  $T_h > T_c$ . The top and bottom walls are considered adiabatic. The temperature difference  $\Delta T \equiv T_h - T_c$  is sufficiently small so that the Boussinesq approximation is valid. For a square cavity ( $H = W$ ), the governing equations in dimensionless form are given by (5.1)-(5.3) where all quantities are defined in the dimensionless domain  $\Omega = (0, 1)^2$ . The boundary conditions are given by

$$\begin{aligned} \mathbf{u} &= 0, \quad \text{on } x = 0, 1 \quad \text{and } y = 0, 1, \\ T &= \frac{1}{2} - x, \quad \text{on } x = 0, 1, \quad \text{and} \quad \frac{\partial T}{\partial y} = 0 \quad \text{on } y = 0, 1. \end{aligned} \tag{5.25}$$

In this problem, we use  $H$ ,  $U = \sqrt{\beta g \Delta T H}$ ,  $\Delta T = T_h - T_c$ , and  $H/U$  as reference length, velocity, temperature, and time scales, respectively and  $T_r = (T_h + T_c)/2$  [33]. With these choices of reference quantities, the following relations hold for dimensionless parameters,

$$Re^2 = Gr = \frac{Ra}{Pr}, \tag{5.26}$$

where  $Ra = \beta g \Delta T H^3 / (\alpha \nu)$  denotes the Rayleigh number, which is a parameter that well suited for the study of this type of flow. Hence, in this case, simulating the flow with a specific value of  $Ra$  using a computer code written for (5.1)-(5.3)

amounts to using  $Re$  according to the above formula. Note also that the maximum velocity is normalized with the above reference quantities, *i.e.*, a magnitude of velocity is of order one with respect to  $Ra$  (as opposed to those used in [30, 55], where the magnitude of velocity changes significantly with  $Ra$ ). Thus, it is not necessary to adjust the value of the threshold parameter associated with velocity as  $Ra$  is increased.

The adaptive algorithm is applied to compute the free convection flow of air ( $Pr = 0.71$ ) in a closed square cavity at moderate to transitional Rayleigh numbers. Since the boundary layers at vertical walls are fairly thin (more precisely they vary as  $O(Ra^{-1/4})$  [30, 95]), it is more suitable to use a grid with points clustered near the boundaries. Such grid can be obtained through the following mapping:

$$y = \frac{(2\alpha + \beta) \left( \frac{\beta + 1}{\beta - 1} \right)^{\frac{\eta - \alpha}{1 - \alpha}} + 2\alpha - \beta}{(2\alpha + 1) \left[ \left( \frac{\beta + 1}{\beta - 1} \right)^{\frac{\eta - \alpha}{1 - \alpha}} + 1 \right]}, \quad \eta \in [0, 1], \quad (5.27)$$

with  $0 < \alpha, \beta < \infty$ . The identical mapping is also used for  $x$ , *i.e* replacing  $y$  by  $x$  and  $\eta$  by  $\xi$ . The equations are then solved in the  $\xi$ - $\eta$  plane.

In the numerical simulations, (5.4) is augmented by the boundary condition  $\partial T^{m+1}/\partial y = 0$  at  $y = 0, 1$ , and  $T^{m+1} = 1/2 - x$  at  $x = 0$  and  $1$ . The boundary condition on  $\hat{\mathbf{u}}$  are taken to be the same as those on  $\mathbf{u}^{m+1}$ , *i.e*  $\hat{\mathbf{u}} = \mathbf{u}^{m+1} = \mathbf{0}$ . We augment the Poisson equation with the condition  $\mathbf{n} \cdot \nabla \phi = 0$  on all boundaries. The fractional time step method used employs the linearized trapezoidal scheme (as outlined in Section 5.2). The provisional pressure  $\tilde{p}$  in (5.5) is obtained by solving

$$\nabla^2 \tilde{p} = \frac{1}{2} \nabla \cdot \{ \mathbf{N}(\tilde{\mathbf{u}}) + \mathbf{N}(\mathbf{u}^m) + (T^{m+1} + T^m) \mathbf{n} \} \quad \text{in } \Omega, \quad (5.28)$$

with boundary conditions

$$\mathbf{n} \cdot \nabla \tilde{p} = \mathbf{n} \cdot \left\{ -\frac{\mathbf{u}^{m+1} - \mathbf{u}^m}{\Delta t} - \frac{1}{2} (N(\tilde{\mathbf{u}}) + N(\mathbf{u}^m)) + \right. \quad (5.29)$$

$$\left. \frac{1}{2} \sqrt{\frac{Pr}{Ra}} \nabla^2 (\tilde{\mathbf{u}} + \mathbf{u}^m) - \frac{1}{2} (T^{m+1} + T^m) \mathbf{n} \right\} \quad \text{on } \partial\Omega, \quad (5.30)$$

where  $N(\mathbf{u}) = (\mathbf{u} \cdot \nabla) \mathbf{u}$  and  $\mathbf{n}$  is a unit vector normal to the boundary. This type of provisional pressure is introduced in [77]. The specific choice alleviates the drawback of a fixed normal pressure gradient on boundaries arising when  $p^n$  is used as the provisional pressure. It is noted that such a claim is fully legitimate when  $p^m$  is computed via (5.9) without the last term on the right hand side, a commonly use formula in the literature. When taking into account the last term on the right hand side, (5.9) permits the normal pressure gradients on boundaries to change (as a function of time). However, numerical experiments shows that there is an improvement in the solutions when the provisional pressure given above is used.

In the following numerical calculations, we use a basis with  $p = 6$  and consistent derivative approximations are obtained via nine-point finite difference stencils ( $n = 4$ ). The resolution is set to  $J - j_0 = 6$  with  $j_0 = 3$  (9 points in each direction at the coarsest level). The mapping parameters in (5.27) are taken to be  $\alpha = 0.5$  and  $\beta = 1.125$ . It is remarked that for these parameters the finest grid spacings in the physical domain corresponds to that of a  $512 \times 512$  grid (the possible finest grid spacing in the computational domain is equivalent to  $\Delta\xi = \Delta\eta = 1/512$ ). Threshold values of  $\varepsilon = \{10^{-3}, 10^{-3}, 10^{-3}\}$  and  $\{5 \times 10^{-3}, 5 \times 10^{-3}, 5 \times 10^{-3}\}$  are used.

We first consider the flow with  $Ra = 10^6, 10^7$  and  $10^8$  in order to compare the solutions with previously published accurate solutions. The initial condition in each case is chosen to be that of a pure conducting quiescent state (*i.e.*  $T(x, y, 0) = 1/2 - x$  and  $\mathbf{u}(x, y, 0) = \mathbf{0}$ ) primarily to demonstrate the robustness and adaptive capabilities of the numerical method. The steady state solution, if it exists, is reached through the time-stepping integration described above. Note that steady state is defined by  $\|f^{m+1} - f^m\|_\infty \leq 5 \times 10^{-5}$  where  $f$  represents  $u, v$  and  $T$ . Figure 5.4 shows streamlines, isotherms, and irregular grids for  $Ra = 10^8$  at different times. The distribution of grid points demonstrates that the adaptive grid, produced automatically, follows the moving and developing structures of the flow. Note that with the use of the pure conducting quiescent state as initial condition, the problem is centro-symmetric and it can be observed that the adaptive grid exhibits centro-symmetry (without enforcing). Note, however, that for  $Ra > 2 \times 10^8$  when instability sets in, any round off error would destroy such symmetry. Figure 5.5 shows the number of grid points  $N$  required during the course of two simulations using different threshold values. It should be noted that the algorithm requires a large  $N$  in the early part of the simulation. In addition, as  $\varepsilon$  is decreased, the number of points  $N$  generated increases automatically. Figures 5.4 and 5.5 demonstrate that the adaptive algorithm provides a means to compute, for fixed threshold parameters, the solution according to local demands. Furthermore, Figure 5.5 also demonstrates the relationship between the accuracy requirement and computational cost, which is  $O(N)$  for the algorithm.

The steady state streamlines and isotherms for different Rayleigh numbers are depicted in Figure 5.6. It is noted that they agree qualitatively with the available numerical results (see [55, 95, 122] for comparisons). Quantitative results are



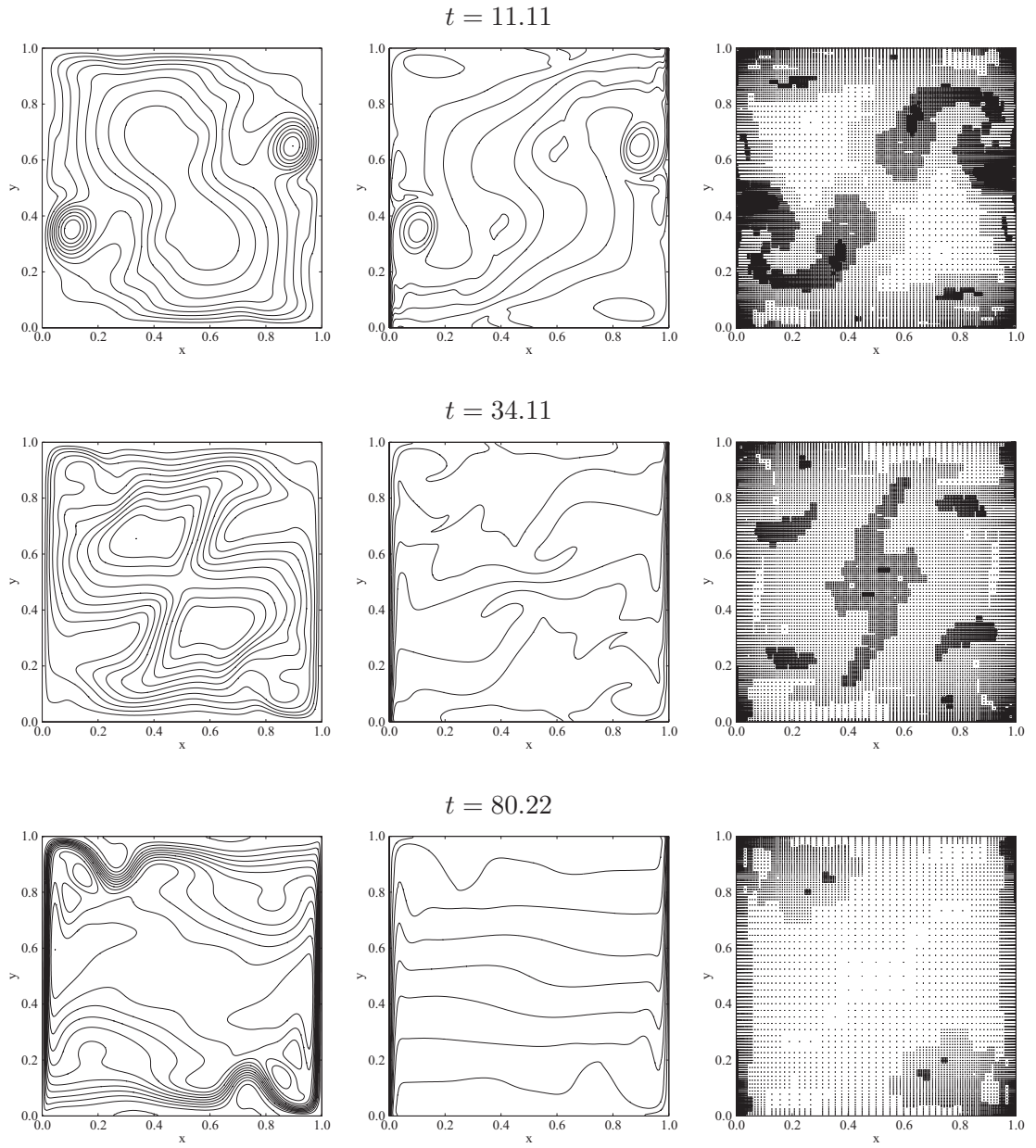


Figure 5.4: Unsteady flow for  $Ra = 10^8$  obtained with  $\varepsilon = \{10^{-3}, 10^{-3}, 10^{-3}\}$  at three different times: stream function (left), isotherms (middle), and dynamically adaptive grid (right).

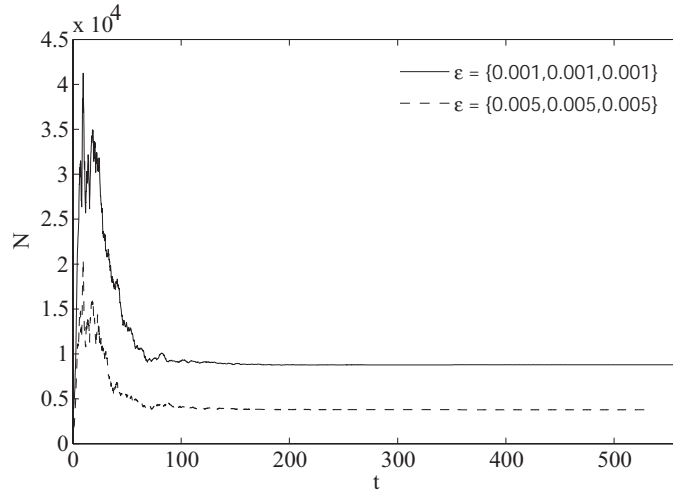


Figure 5.5: Evolution of number of grid points required for  $Ra = 10^8$ .

determined from interpolating values of the obtained solutions on the irregular grid on a grid of  $1001 \times 1001$  equi-distant points. Note that  $N_s$  corresponds to the number of grid points needed by the algorithm at steady state. The results (obtained with different values of the threshold parameter for  $Ra = 10^6, 10^7$  and  $10^8$ ) for the maximum vertical and horizontal velocity components along the mid-sections  $y = 1/2$  and  $x = 1/2$ , and the Nusselt number ( $Nu = \partial T / \partial x|_{x=0}$ ) are compared in Tables 5.4–5.6 with those previously published.

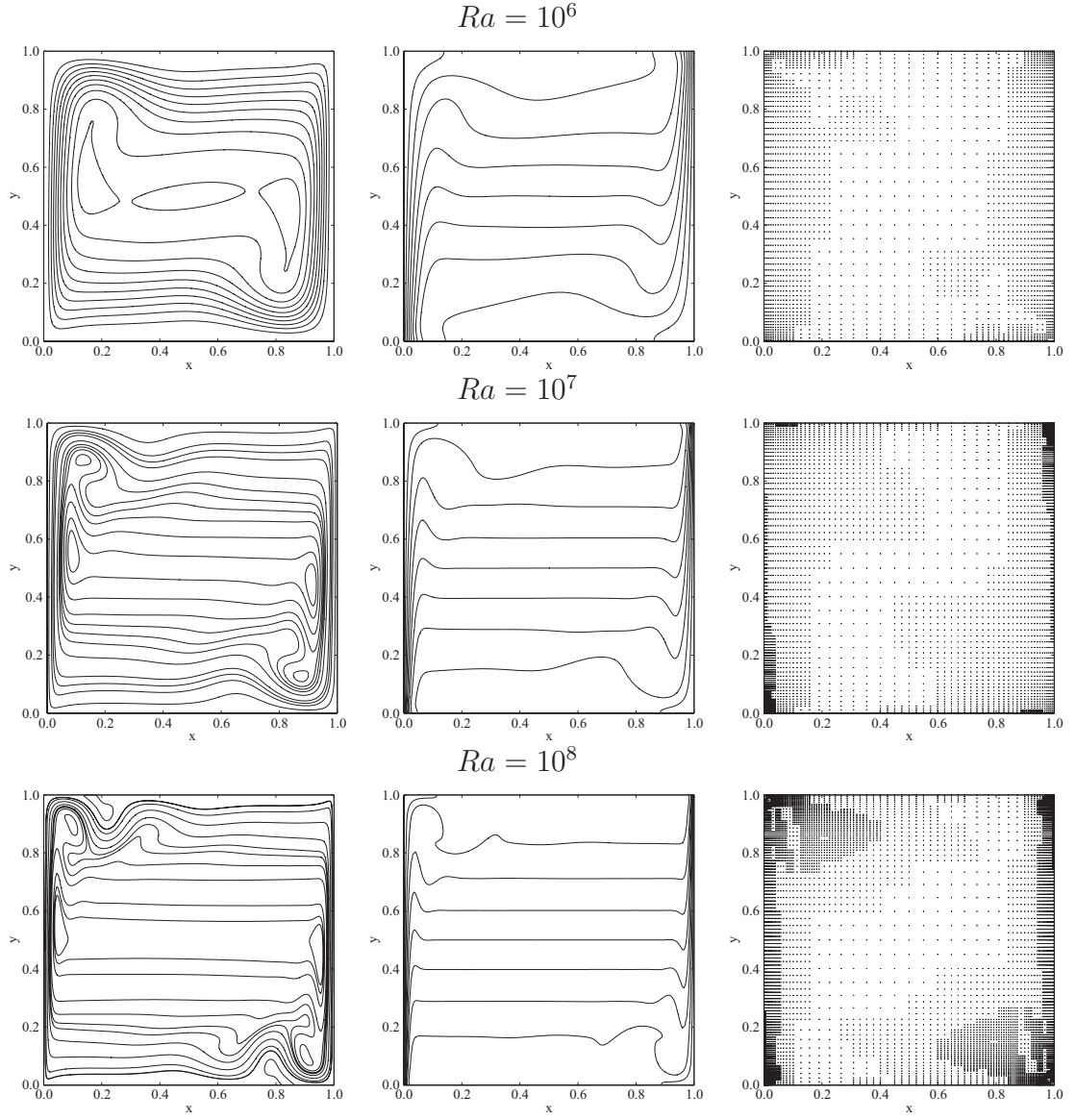


Figure 5.6: Steady state solutions for  $Ra = 10^6$  (top),  $Ra = 10^7$  (middle), and  $Ra = 10^8$  (bottom), obtained with  $\varepsilon = \{10^{-3}, 10^{-3}, 10^{-3}\}$ : stream function (left), isotherms (middle), and dynamically adaptive grid (right).

TABLE 5.4  
RESULTS FOR  $Ra = 10^6$  COMPARED WITH OTHER ACCURATE  
SOLUTIONS.

	$u_{\max} \times 10^2$	$y(u_{\max})$	$v_{\max} \times 10$	$x(v_{\max})$	$Nu_{\max}$	$y(Nu_{\max})$	$Nu_{\min}$	$y(Nu_{\min})$
Present method, $N_s = 1467$ ( $\epsilon = \{0.005, 0.005, 0.005\}$ )	7.6898	0.085	2.6126	0.039	17.225	0.046	1.013	1
Present method, $N_s = 2849$ ( $\epsilon = \{0.001, 0.001, 0.001\}$ )	7.6807	0.085	2.6162	0.038	17.507	0.038	0.987	1
Ref. [55] <sup>a</sup>	7.670	0.085	2.603	0.0379	17.925	0.0378	0.989	1
Ref. [95] <sup>b</sup>	7.6944	0.085	2.6175	0.038	17.5360	0.039	0.979	1
Ref. [122] <sup>c</sup>	7.6944	0.08499	2.6176	0.038				

Note: <sup>a</sup>Finite difference with Richardson extrapolation, <sup>b</sup>pseudo-spectral Chebychev method with  $72 \times 72$  modes, and <sup>c</sup>spectral Petrov-Galerkin method with  $(48 \times 48)/2$  modes.

TABLE 5.5  
RESULTS FOR  $Ra = 10^7$  COMPARED WITH OTHER ACCURATE  
SOLUTION.

	$u_{\max} \times 10^2$	$y(u_{\max})$	$v_{\max} \times 10$	$x(v_{\max})$	$Nu_{\max}$	$y(Nu_{\max})$	$Nu_{\min}$	$y(Nu_{\min})$
Present method, $N_s = 4651$ ( $\varepsilon = \{0.001, 0.001, 0.001\}$ )	5.5487	0.880	2.6095	0.021	39.413	0.019	1.374	1
Present method, $N_s = 6603$ ( $\varepsilon = \{0.001, 0.00025, 0.001\}$ )	5.5785	0.879	2.6237	0.021	39.386	0.018	1.372	1
Ref. [95] <sup>a</sup>	5.5762	0.879	2.6242	0.021	39.395	0.018	1.366	1
Ref. [122] <sup>b</sup>	5.5763	0.8793	2.6245	0.021				
Ref. [130] <sup>c</sup>	5.538	0.888	2.733	0.0237	36.50	0.011	1.416	1
Ref. [94] <sup>d</sup>	5.557	0.874	2.714	0.021	40.15	0.016	1.376	1

*Note:* <sup>a</sup> Pseudo-spectral Chebyshev method with  $80 \times 80$  modes, <sup>b</sup> spectral Petrov-Galerkin method with  $(48 \times 48)/2$  modes, <sup>c</sup> finite element with 168 quadratic elements, and <sup>d</sup> cubic spline.

TABLE 5.6  
RESULTS FOR  $Ra = 10^8$  COMPARED WITH OTHER ACCURATE  
SOLUTION.

	$u_{\max} \times 10^2$	$y(u_{\max})$	$v_{\max} \times 10$	$x(v_{\max})$	$Nu_{\max}$	$y(Nu_{\max})$	$Nu_{\min}$	$y(Nu_{\min})$
Present method, $N_s = 3799$ ( $\epsilon = \{0.005, 0.005, 0.005\}$ )	3.5920	0.920	2.6293	0.012	88.361	0.007	1.873	1
Present Work, $N_s = 8791$ ( $\epsilon = \{0.001, 0.001, 0.001\}$ )	3.7798	0.927	2.6379	0.012	87.514	0.008	1.753	1
Ref. [95] <sup>a</sup>	3.8199	0.928	2.6375	0.012	87.2355	0.008	1.919	1
Ref. [122] <sup>b</sup>	3.8136	0.9277	2.6375	0.012				
Ref. [83] <sup>c</sup>	3.823		2.638					
Ref. [85] <sup>d</sup>	3.698	0.926	2.623	0.012				

*Note:* <sup>a</sup> Pseudo-spectral Chybechev method with  $128 \times 128$  modes, <sup>b</sup> spectral Petrov-Galerkin method with  $(48 \times 48)/2$  modes, <sup>c</sup> finite volume with  $360 \times 360$  grids, and <sup>d</sup> finite-difference with  $200 \times 200$  grids.

The results for  $u_{\max}$ ,  $v_{\max}$  and their locations compare favorably with other accurate results. Some scatters the data of maximum and minimum Nusselt numbers can be observed among the results. The present results are closest to those of [95] which we believe are the most accurate. Perhaps it is not fair to compare the number of degrees of freedom used since the present algorithm, unlike the others, is adaptive. Nevertheless, if we consider only steady state solutions, our method requires a relatively smaller number of degrees of freedom (ranging from 2 to 14 times fewer DOFs, except that of [122] which in fact outperforms every method in term of DOFs required) in order to obtain solutions with errors of less than 1% compared with those of [95] and [122].

A simulation was also performed for  $Ra = 5 \times 10^8$  where the flow is known to be chaotic [107]. In this case we use  $p = 6$ , a consistent nine-point finite difference stencil for derivative calculations, and a grid clustering parameter of  $\beta = 1.085$ . The threshold parameters is set to  $\epsilon = \{10^{-3}, 4 \times 10^{-3}, 4 \times 10^{-3}\}$ . Figure 5.7 shows the solution at the early times and Figure 5.8 the solution after a much longer time. The number of the grid points required in the course of the simulation is depicted in Figure 5.9. The time trace of  $u$  at the point  $\mathbf{x} = (0.0478, 0.9522)$  is shown in Figure 5.10, indicating not only the chaotic behavior of the solution, but also that the solution has become statistically steady at that location by approximately a dimensionless time of 250. It can be observed that in the early part of the simulation, the solution is quite complicated and requires a relative large number of grid points. The flow evolves to produce a stratified and fairly quiescent region in the core of the cavity, with a grid requirement that is substantially smaller.

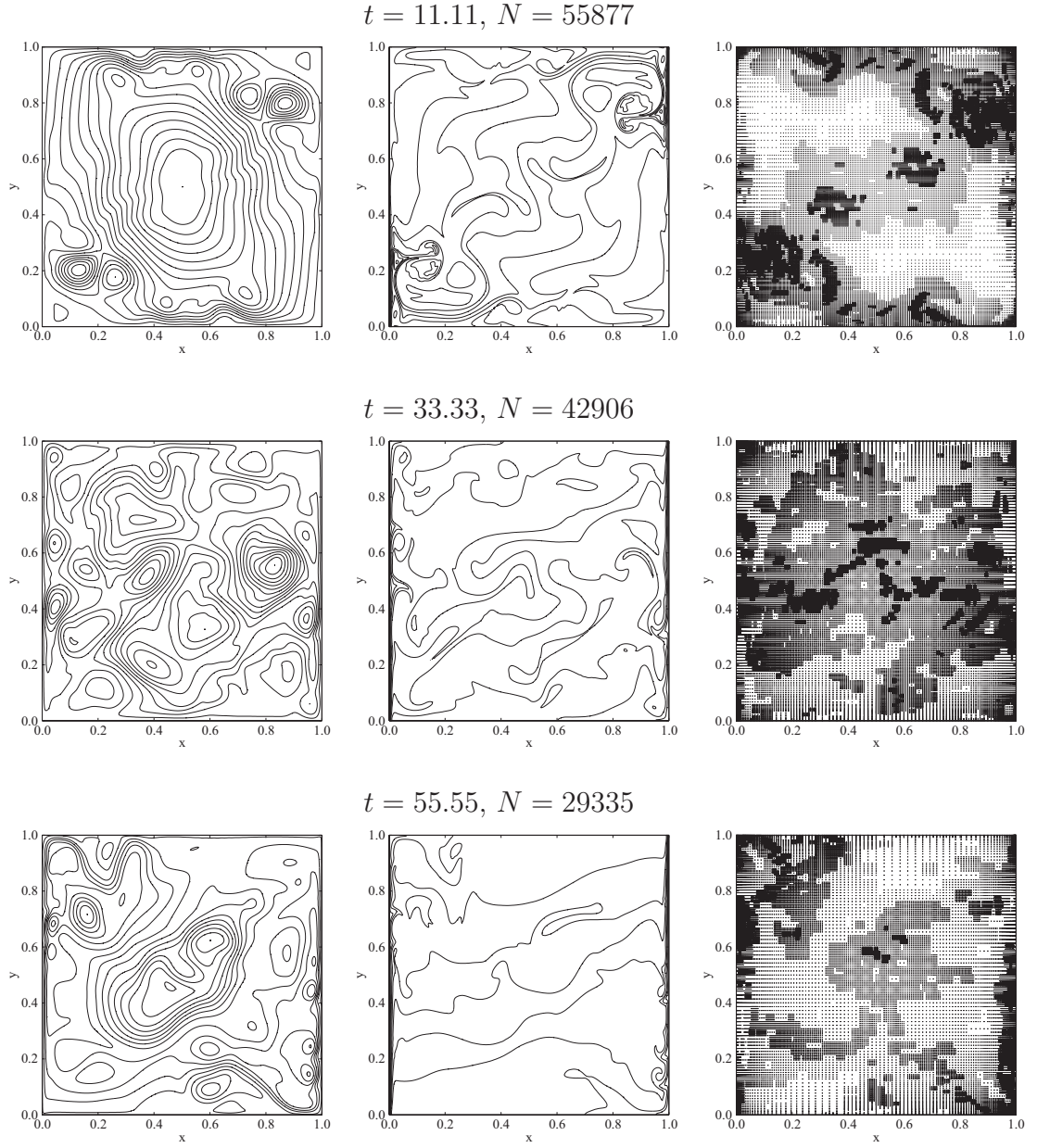


Figure 5.7: Unsteady flow for  $Ra = 5 \times 10^8$  at different times obtained with  $\epsilon = \{10^{-3}, 4 \times 10^{-3}, 4 \times 10^{-3}\}$ : stream function (left), isotherms (middle), and dynamically adaptive grid (right).



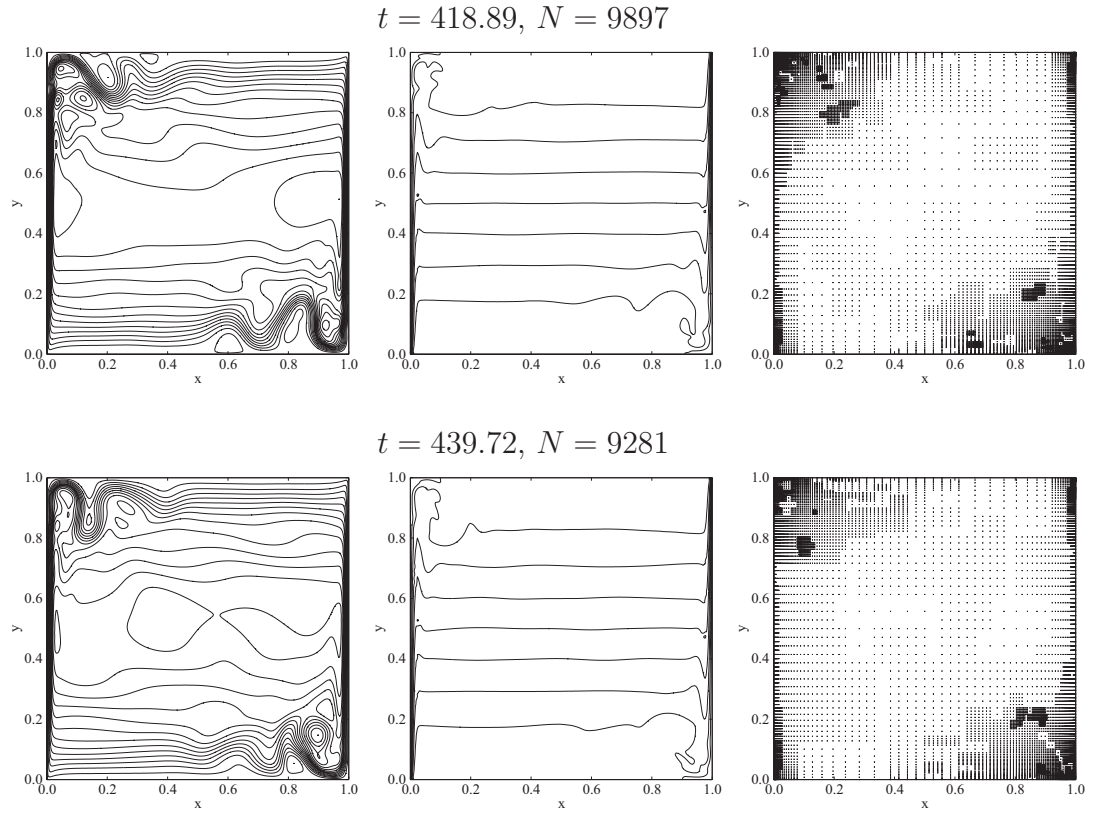


Figure 5.8: Unsteady flow for  $Ra = 5 \times 10^8$  at different times obtained with  $\varepsilon = \{10^{-3}, 4 \times 10^{-3}, 4 \times 10^{-3}\}$ : stream function (left), isotherms (middle), and dynamically adaptive grid (right)

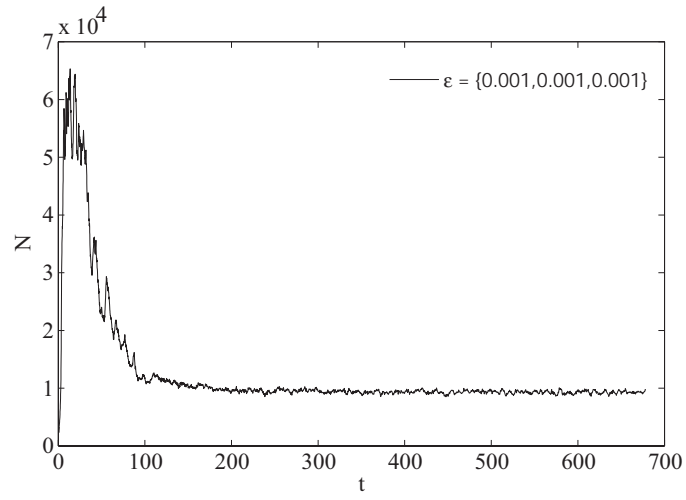


Figure 5.9: Evolution of the number of grid points required for  $Ra = 5 \times 10^8$ .

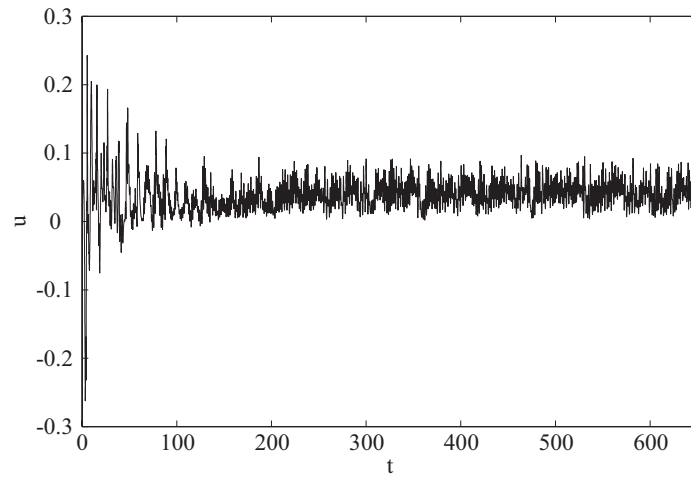


Figure 5.10: Time trace of the  $u$ -velocity component at the point  $\mathbf{x} = (0.0478, 0.9522)$  for  $Ra = 5 \times 10^8$ .

It should be noted that when the flow is statistically steady, most of the grid points are concentrated near the vertical walls and the top-left and bottom-right corners of the cavity. It is noted that the results these are fully consistent with those previously published in [96, 107].

### 5.7 3-D differentially heated cavity

In this section, we use the 3-D differentially heated cavity problem as a test problem for the adaptive method. The schematic diagram of this problem is illustrated in Figure 5.11. In brief, the problem consists of modelling the flow in cubical cavity in which the temperature of the vertical left and right walls are maintain isothermally at  $T_h$  and  $T_c$ . The remaining four walls are considered to be thermally insulated. The governing equations in dimensionless form are given by (5.1)-(5.3) where all quantities are defined in the dimensionless domain  $\Omega = (0, 1)^3$ . In this case, the boundary condition on the walls, are given by

$$\mathbf{u} = 0 \text{ on } x = 0, 1, \ y = 0, 1 \text{ and } z = 0, 1, \quad (5.31)$$

$$T = \frac{1}{2} - x \text{ on } x = 0, 1, \text{ and } \frac{\partial T}{\partial n} = 0 \text{ on } y = 0, 1 \text{ and } z = 0, 1,$$

Note that we use the same reference quantities as in the 2-D problem, namely  $\sqrt{\beta g \Delta T H}$ ,  $H$ ,  $H/U$ , and  $\Delta T = T_h - T_c$  as the reference velocity, length, time, and temperature scales, and  $T_r = (T_h + T_c)/2$  as reference temperature.

The adaptive algorithm is applied to compute the free convection flow of air ( $Pr = 0.71$ ) for Rayleigh numbers ranging from  $10^3$  to  $10^5$ . The initial condition in each case is chosen to be that of a pure conducting quiescent state, *i.e.*  $T(x, y, z, 0) = 1/2 - x$  and  $\mathbf{u}(x, y, z, 0) = \mathbf{0}$ , primarily to demonstrate the

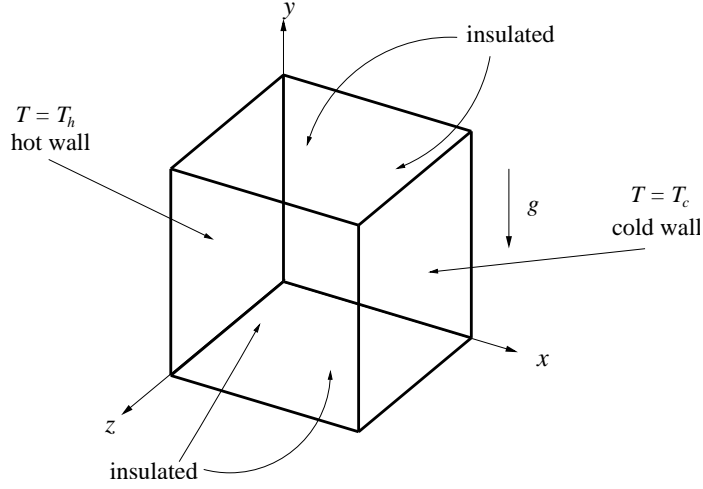


Figure 5.11: Schematic diagram of the differentially-heated cubic cavity.

robustness and adaptive capabilities of the numerical method. The steady state solution, if it exists, is reached when the following criteria is satisfied.

$$\|f^m - f^{m-1}\|_{\mathbf{V}^m, \infty} \leq 5 \times 10^{-5} \quad \text{for all } f, \quad (5.32)$$

where  $f$  represents any component of the velocity field and temperature, and  $m$  refers to the value of  $f$  at the  $m$ -th integration step.

In the following numerical calculations, we use a basis with  $p = 6$  and derivative approximations are obtained via a thirteen-point finite-difference stencil ( $n = 4$ ). The resolution is set to  $(J - j_0) = 4$  with  $J_0 = 3$  (9 points in each direction at the coarsest level). This results in a finest grid spacing of equivalent size  $\Delta x = \Delta y = \Delta z = 1/128$ . We note that, in the presented method, the grid adaption algorithm is inexpensive. However, determining the local finite difference stencils and performing the ILU decomposition (tasks needed when the grid is adapted), although relatively inexpensive, are more expensive than the cost of the

grid adaption itself. Therefore, in the present calculations, to avoid performing such tasks too often, the computational grid is not adapted if the number of essential points,  $\hat{\mathbf{V}}$ , at the current step changes only slightly from that of the previous step, specifically by less than 5%.

Figures 5.12 and 5.13 show respectively the irregular grids generated by the adaptive algorithm and isotherms for the flow corresponding to  $Ra = 10^5$  obtained with threshold parameter  $\varepsilon = \{3.75 \times 10^{-3}, 3.75 \times 10^{-3}, 3.75 \times 10^{-3}, 3.75 \times 10^{-3}\}$  at different values of time.

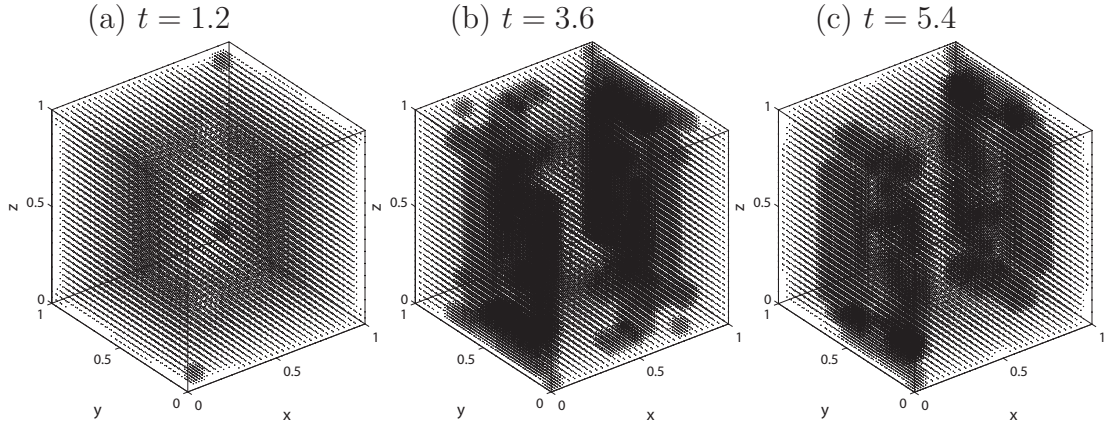


Figure 5.12: Dynamically adaptive grid  $\mathbf{V}$  for  $Ra = 10^5$  obtained with  $\varepsilon = 3.75 \times 10^{-3}$  at three different times; (a)  $t = 1.2$ ,  $N = 30764$ , (b)  $t = 3.6$ ,  $N = 83328$ , (c)  $t = 5.4$ ,  $N = 60800$ .

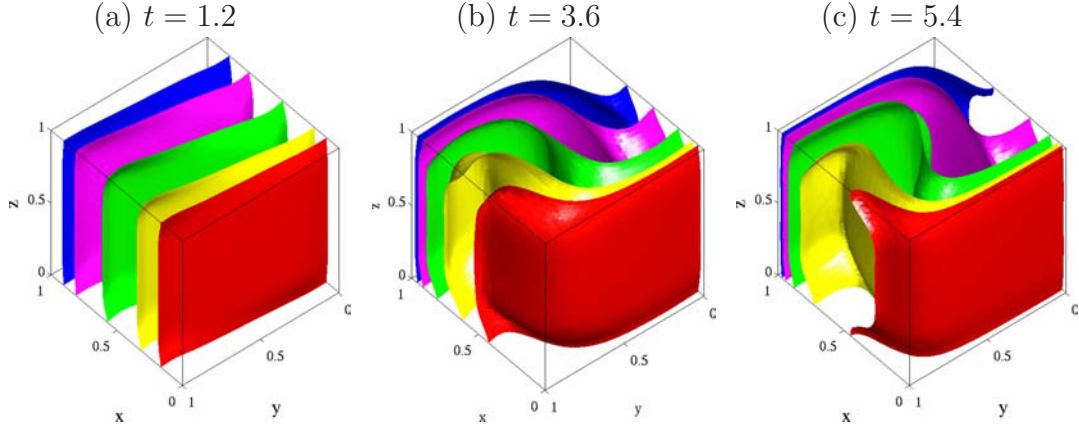


Figure 5.13: Isotherms (contour level: 0.375 (red), 0.25 (yellow), 0 (green),  $-0.25$  (purple),  $-0.375$  (blue)) for  $Ra = 10^5$  obtained with  $\varepsilon = 3.75 \times 10^{-3}$  at three different times: (a)  $t = 1.2$  (b)  $t = 3.6$  (c)  $t = 5.4$ .

Figures 5.14, 5.15 and 5.16 depict respectively the  $u$ - $v$  velocity field in the plane  $z = 1/2$ , the  $u$ - $w$  field in the plane  $y = 1/2$ , and the  $v$ - $w$  field in the plane  $x = 1/2$ . The distribution of grid point in the corresponding three planes are also shown in the figures. It can be noticed that the distribution of grid points the adaptive grid, generated automatically, tracks the moving and developing structures of the flow. Figure 5.17 shows the number of grid points  $N$  required by the adaptive algorithm during the course of two simulations using different threshold values.

It can be seen that the algorithm requires a large  $N$  in the early part of the simulation. Moreover, as the value of  $\varepsilon$  is decreased, the number of points required increases automatically. The steady state isotherms in the cavity and in the plane  $z = 1/2$  as well as the velocity fields in three mid-planes are shown in Figure 5.18. Figure 5.19 shows the computational grid at steady state. It includes the distribution of the corresponding grid points in the three mid-planes.

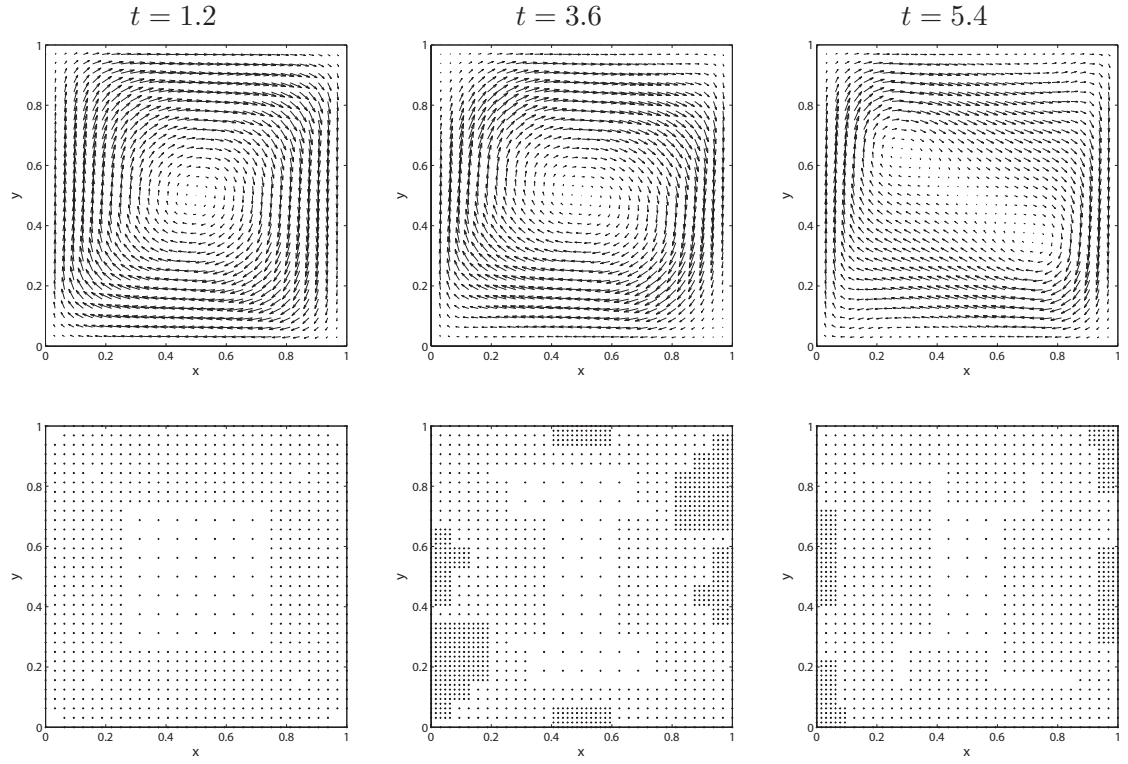


Figure 5.14: Velocity field  $u-v$  (top row) and distribution of grid points (bottom row) in the plane  $z = 1/2$  for  $Ra = 10^5$  obtained with  $\varepsilon = 3.75 \times 10^{-3}$  at three different times.

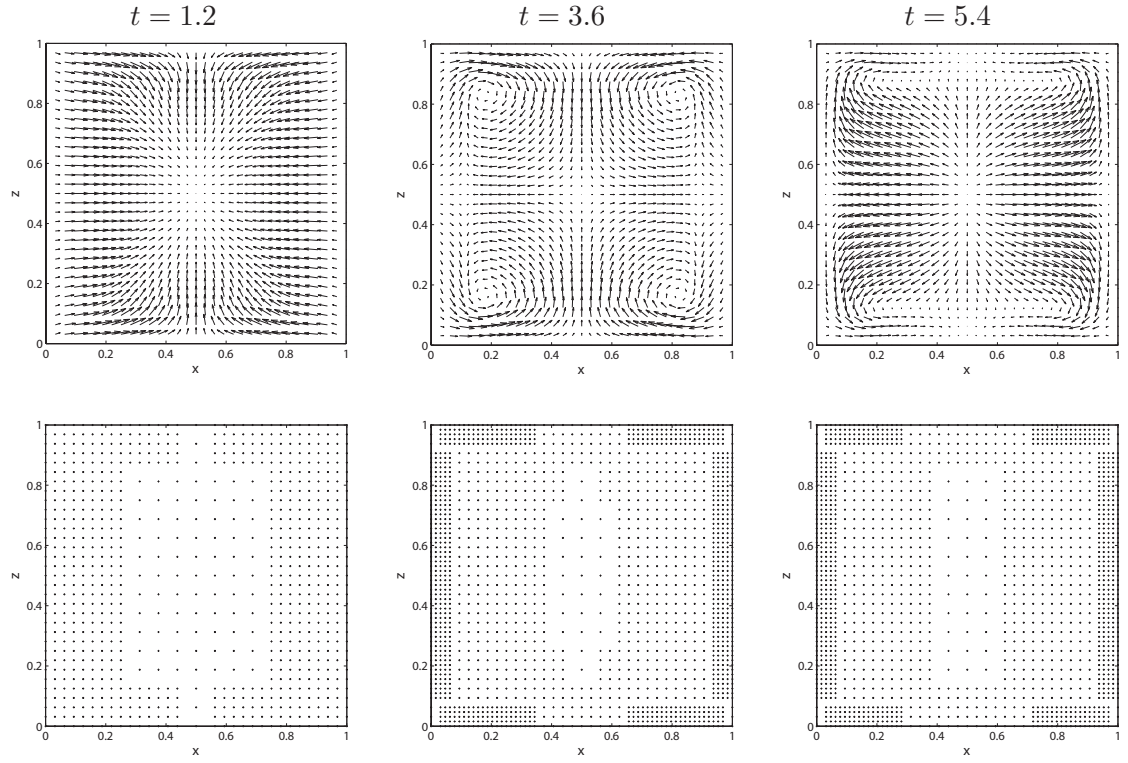


Figure 5.15: Velocity field  $u-w$  (top row) and distribution of grid points (bottom) in the plane  $y = 1/2$  for  $Ra = 10^5$  obtained with  $\varepsilon = 3.75 \times 10^{-3}$  at three different times.



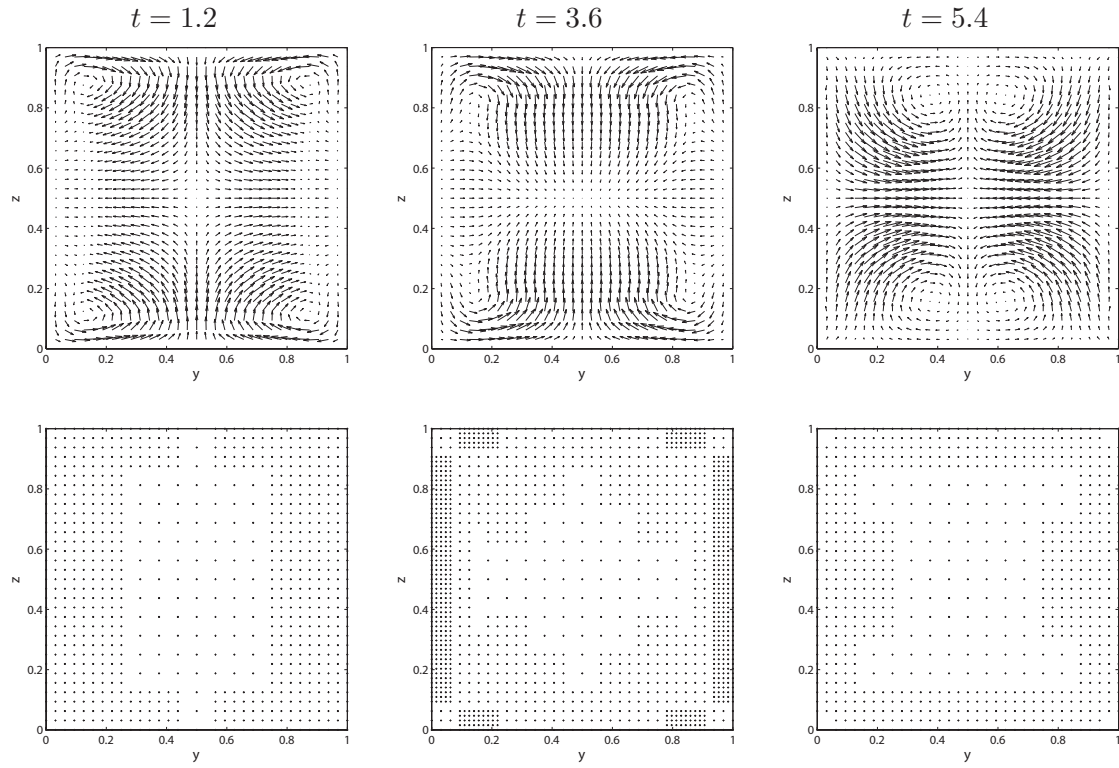


Figure 5.16: Velocity field  $v-w$  (top row) and distribution of grid points (bottom row) in the plane  $x = 1/2$  for  $Ra = 10^5$  obtained with  $\varepsilon = 3.75 \times 10^{-3}$  at three different times.

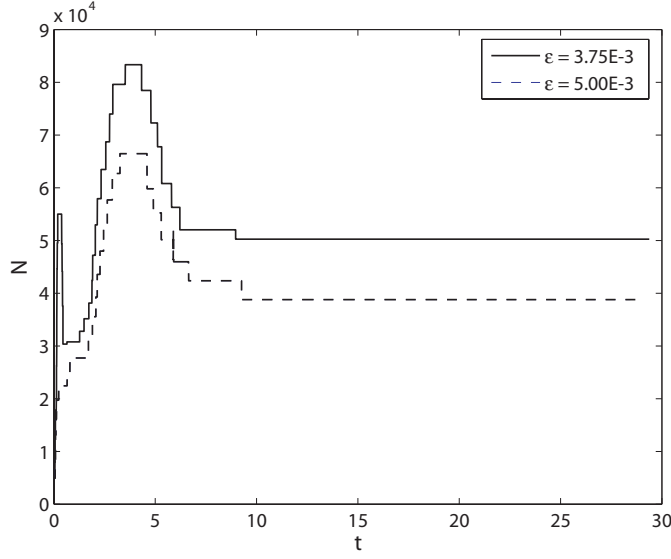


Figure 5.17: Evolution of the number of grid points required by the dynamically adaptive algorithm for  $Ra = 10^5$  using different threshold values: (solid line)  $\varepsilon = 3.75 \times 10^{-3}$ , (dash line)  $\varepsilon = 5.0 \times 10^{-3}$ .

The total number of grid points required at the steady state is  $N = 50245$ . It can be observed that the points are concentrated in the vicinity of the isothermal walls, the area where the boundary layers are fairly thin. This figure along with Figure 5 indicate that the adaptive algorithm distributes the points according to the solution features.

We have performed numerical experiments for  $Ra = 10^3$ ,  $10^4$  and  $10^5$ . All numerical results reach the steady state defined by (5.32). To examine the solutions quantitatively, we compare certain quantities with those previously published: (a) the maximum of the  $u$  component on  $(x, 1/2, 1/2)$  and  $v$  component on  $(1/2, y, 1/2)$  and their locations, and (b) Nusselt numbers  $Nu_w$  and  $Nu_{mp}$  on the hot wall. The Nusselt number on the hot wall is defined as follows:

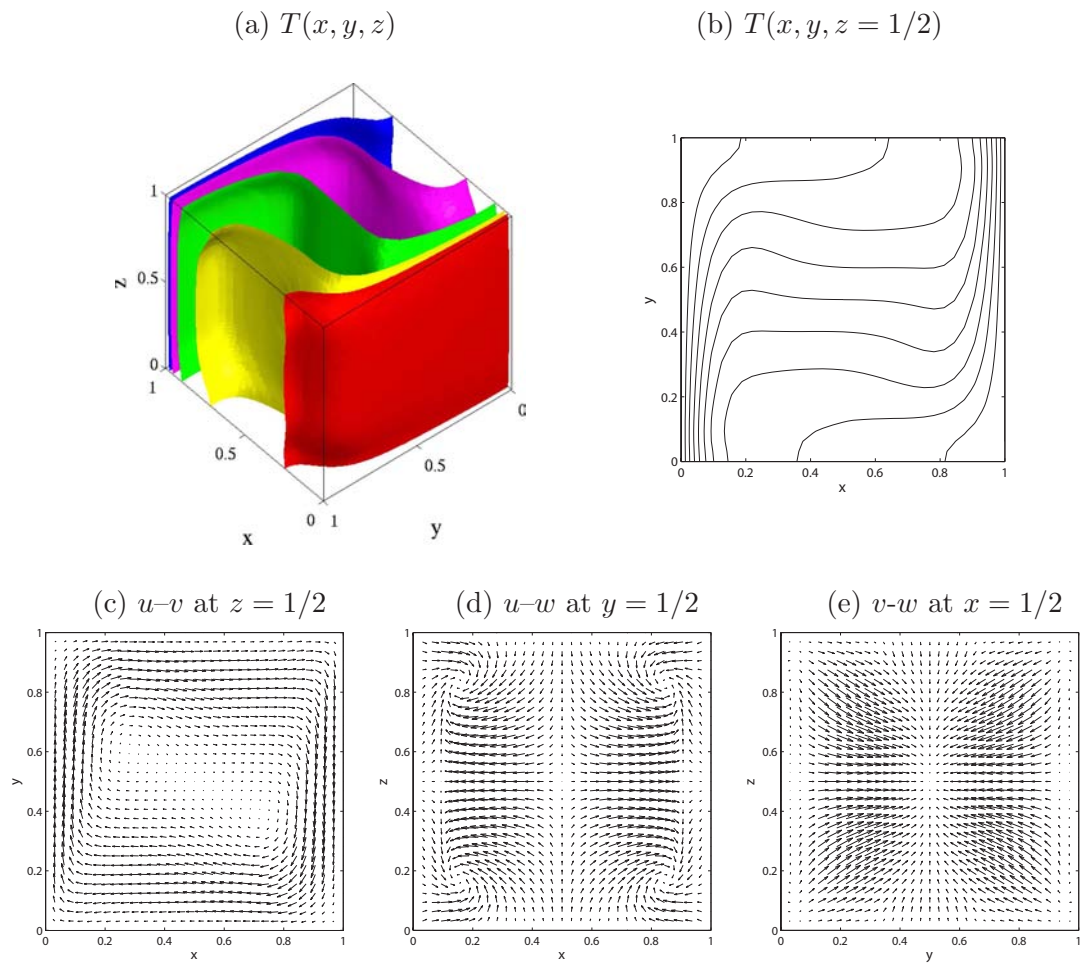


Figure 5.18: Steady flow for  $Ra = 10^5$  obtained with  $\varepsilon = 3.75 \times 10^{-3}$ . (a) temperature contour levels: 0.375 (red), 0.25 (yellow), 0 (green),  $-0.25$  (purple),  $-0.375$  (blue); (b) temperature in the plane  $z = 1/2$ ; (c) velocity field  $u-v$  in the plane  $z = 1/2$ , (d) velocity field  $u-w$  in the plane  $y = 1/2$ , and (e) velocity field  $v-w$  in the plane  $x = 1/2$ .

(a) Irregular grid  $\mathcal{V}$ ,  $N = \dim(\mathcal{V}) = 50245$

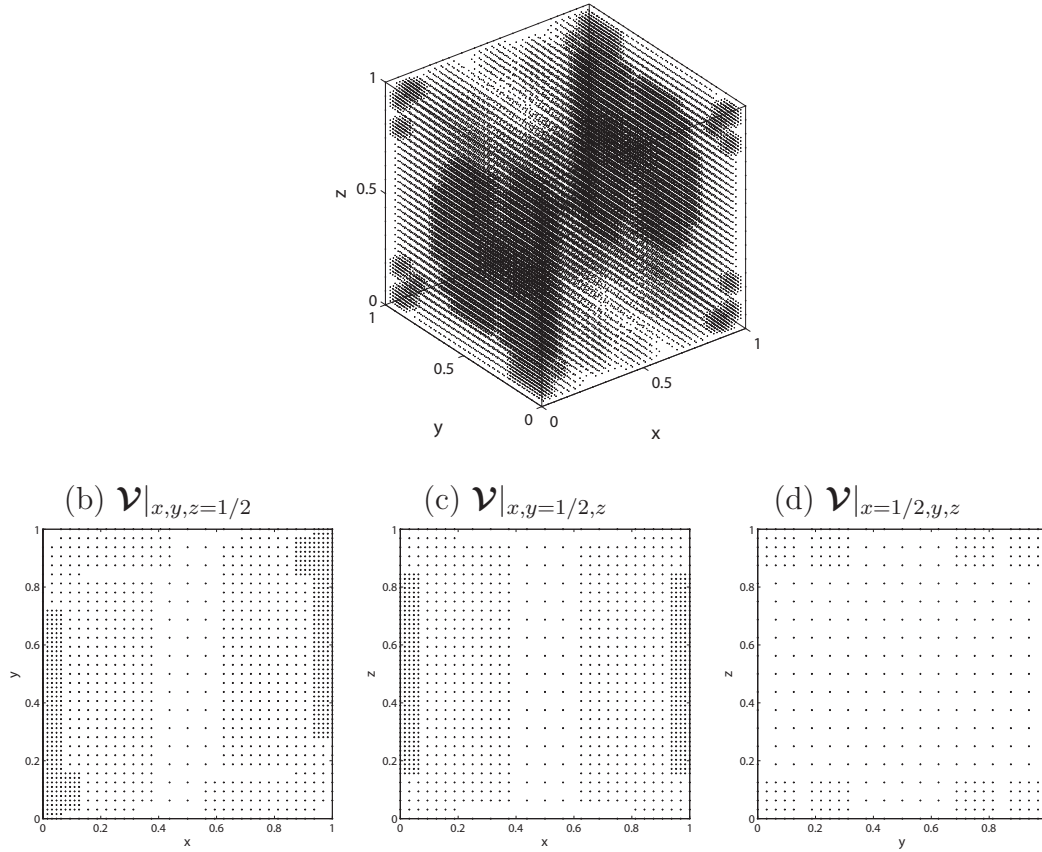


Figure 5.19: Dynamically adaptive grid at steady state for  $Ra = 10^5$  obtained with  $\varepsilon = 3.75 \times 10^{-3}$ : (a) irregular grid in the cavity, (b) irregular grid in the plane  $z = 1/2$ , (c) irregular grid in the plane  $y = 1/2$ , and (d) irregular grid in the plane  $x = 1/2$ .

$$Nu_{local}(y, z) = \partial T(0, y, z)/\partial x, \quad Nu_{mean}(z) = \int_0^1 Nu_{local}(y, z) dy,$$

$$Nu_w = \int_0^1 Nu_{mean}(z) dz, \quad Nu_{mp} = Nu_{mean}(1/2).$$

The data for  $Ra = 10^3$ ,  $10^4$  and  $10^5$ , obtained with different values of threshold  $\epsilon$  are gathered in Tables 5.7, 5.8 and 5.9, respectively. Note that  $N_s$  corresponds to the number of grid points needed by the adaptive algorithm at steady state. For comparison purposes, these tables include also the numerical results provided in [63], [135] and [129]. The results given in [63] are obtained using a control-volume finite difference method with variable spacing of a grid with  $62 \times 62 \times 62$  points. Results report in [135] are obtained by solving the Navier-Stokes in vorticity-vector form using a time-space fourth order finite-difference method with a uniform grid of  $120 \times 120 \times 120$ . In [129], the solutions are obtained with the use of a Chebychev pseudo-spectral method with four different grids with a relatively large number of collocation points ( $51^3$  to  $81^3$ ). In the absence of any singularity, it is known that spectral methods yield highly accurate solutions. We thus believe that these solutions are the most accurate available. It can be seen from the tables that although the number of grid points used in the present method is relatively small, the numerical results for the quantities listed are in good agreement with these accurate results. In addition, as the value of the threshold parameter  $\epsilon$  is decreased the numerical results show an improvement in accuracy (with a corresponding increase in the number of grid points required).

TABLE 5.7  
SPECIFIC RESULTS FOR  $Ra = 10^3$ .

	Ref. [129] <sup>a</sup>	Ref. [63] <sup>b</sup>	Present method	
	81 <sup>3</sup>	32 <sup>3</sup>	$\epsilon = 2.5 \times 10^{-3}$	$\epsilon = 1.0 \times 10^{-3}$
			( $N_s = 4913$ )	( $N_s = 7665$ )
$u_{\max} _{(x,1/2,1/2)} \times 10$	1.328	1.314	1.3085	1.3099
$y$	0.8151	0.800	0.8125	0.8125
$v_{\max} _{(1/2,y,1/2)} \times 10$	1.329	1.320	1.3412	1.3398
$x$	0.1853	0.1667	0.1797	0.1797
$Nu_{mp}$	1.087		1.0859	1.0858
$Nu_w$	1.070	1.085	1.0687	1.0690

Note: <sup>a</sup>Chebyshev spectral method, <sup>b</sup>control-volume finite-difference method.

TABLE 5.8  
SPECIFIC RESULTS FOR  $Ra = 10^4$ .

	Ref. [129] <sup>a</sup>	Ref. [63] <sup>b</sup>	Ref. [135] <sup>c</sup>	Present method	
	81 <sup>3</sup>	62 <sup>3</sup>	120 <sup>3</sup>	$\epsilon = 2.5 \times 10^{-3}$	$\epsilon = 1.0 \times 10^{-3}$
				( $N_s = 16149$ )	( $N_s = 33285$ )
$u_{\max} _{(x,1/2,1/2)} \times 10$	1.9844	2.013	1.984	1.9834	1.9804
$y$	0.8244	0.8167	0.8250	0.8281	0.8242
$v_{\max} _{(1/2,y,1/2)} \times 10$	2.2093	2.252	0.2216	2.2239	2.2164
$x$	0.1198	0.1167	0.1167	0.1133	0.1172
$Nu_{mp}$	2.25	2.302		2.2356	2.2351
$Nu_w$	2.05	2.100	2.06	2.0275	2.034

Note: <sup>a</sup>Chebyshev spectral method, <sup>b</sup>control-volume finite-difference method,  
<sup>c</sup>space-time 4<sup>th</sup> order finite difference method.

TABLE 5.9  
SPECIFIC RESULTS FOR  $Ra = 10^5$ .

	Ref. [129] <sup>a</sup>	Ref. [63] <sup>b</sup>	Ref. [135] <sup>c</sup>	Present method		
	81 <sup>3</sup>	62 <sup>3</sup>	120 <sup>3</sup>	$\epsilon = 3.75 \times 10^{-3}$	$\epsilon = 2.5 \times 10^{-3}$	$\epsilon = 1.5 \times 10^{-3}$
				( $N_s = 50, 245$ )	( $N_s = 63, 453$ )	( $N_s = 92, 209$ )
$u_{\max} _{(1/2,y,1/2)} \times 10$	1.4096	1.468	1.416	1.417	1.4447	1.4297
$y$	0.8535	0.8547	0.8500	0.8594	0.8555	0.8555
$v_{\max} _{(x,1/2,1/2)} \times 10$	2.4473	2.471	2.461	2.4439	2.4494	2.4512
$x$	0.067	0.0647	0.0667	0.0625	0.0625	0.0625
$Nu_{mp}$	4.612	4.646		4.5746	4.5810	4.5868
$Nu_w$	4.337	4.361	4.3713	4.2390	4.2588	4.2919

Note: <sup>a</sup>Chebyshev spectral method, <sup>b</sup>control-volume finite-difference method,  
<sup>c</sup>space-time 4<sup>th</sup> finite difference method.



## CHAPTER 6

### CONCLUDING REMARKS

#### 6.1 Summary

In this work, we describe further development of point-oriented techniques developed originally in [110, 111] for the adaptive solution of time-independent and time-dependent PDEs in  $d$ -spatial dimensions. The techniques are based on  $d$ -D interpolating wavelet bases, which are constructed by considering the decomposition, in a conventional way, of the tensor product of  $d$  one-dimensional interpolating MRA on an interval. Amplitudes of wavelets, which indicate the local regularity of a function, and the connection between interpolating wavelets and dyadic grid points are used in the construction of an adaptively refined grid of irregular points. In the discretization on the irregular grid, finite differences are used in the derivative approximation in order to reduce the number operations required. Important ingredients of the algorithm are the adaptive fast wavelet transform (*AFWT*) and its inverse (*AIWT*, *LAWT*) which map function values on the irregular grid to associated wavelet amplitudes. These algorithms require  $O(N)$  operations. The benefit of these building blocks are two fold: they provide inexpensive means for constructing an adaptive grid and they enable the efficient means (with  $O(N)$  operations) of calculating the discrete version of a differential operator. In the current implementation, any order wavelet and corresponding

finite difference approximation can be prescribed (note that so far the algorithms for  $p > 10$  and  $n > 10$  have not tested). Another feature of the techniques include the ability to handle general boundary conditions.

The adaptive technique for time-independent problems has been tested on 2- and 3-D Poisson and Helmholtz problems with manufactured solutions. Quantitatively, the numerical results clearly indicate that the method refines resolution in areas where the solution changes rapidly and near singularities without knowledge of the solution before-hand. The numerical results indicate that the number of degrees of freedom demanded by the algorithm behave approximately like

$$N = O(\varepsilon^{-d/p}) \quad (6.1)$$

as a results of varying the value of threshold parameter  $\varepsilon$ , where  $p$  is the order of wavelet used, and  $d$  is the spatial dimension of the problem. Note that this relation is similar to that of the sparse wavelet representation of a function (3.32). Of course, the constant associated with (6.1) is larger than that of the wavelet representation of the exact solution. The overall behavior of errors in the numerical solution, in discrete maximum norm, as the threshold values are varied conforms reasonably well with

$$\|u_{exact} - u_{num,\varepsilon}\|_{L_{\mathcal{V},\infty}} = O(N^{-\min(p-2,n)/d}) \quad (6.2)$$

and, in some cases, the error in the numerical solution is proportional to  $O(N^{-p/d})$ . Note that (6.1) and (6.2) are of course related. More precisely, overall behavior of

errors in the numerical solution conforms reasonably with

$$\|u_{exact} - u_{num,\varepsilon}\|_{L_{V,\infty}} = O(\varepsilon^{\min(p-2,n)/p}) \quad (6.3)$$

In some cases (where the error is proportional to  $O(N^{-p/d})$ ), the error is proportional to  $O(\varepsilon)$  with unit constant! Regarding preconditioning of linear systems arising from discretization of elliptic problem, we find that the Jacobi preconditioning performs reasonably well. Numerical experiments on 2-, 3-D problem indicate that the number of iterations required in the preconditioned BiCGSTAB and TFQMR solvers depends mildly the highest level of resolution.

In this study, the adaptive method is extended to more complicated domains by means of a domain transformation technique. We use transfinite interpolation for grid generation. The additional work to set up the grid transformation for one grid point is approximately  $O(5^3)$  for the 2-D case and  $O(9^3)$  for the 3-D case. However, this calculation is done once and thereafter only for newly added grid points. We demonstrate this approach on a 2-D Poisson problem defined on a domain given in Figure 4.25. The method (in terms of the number of grid points generated and the error in the numerical solution as a results of varying threshold values) exhibits a similar behavior as in solving a problem without transformation.

In the adaptive technique applied to time-independent problems, the “time discretization” is used to cast the PDE to a time marching procedure. Of course, the time discretization represents real time in time-dependent problems. The wavelet amplitudes of the current approximate solution is used as an indicator for constructing the adaptive grid for the solution at the next time level. The adaptive technique is used to simulate the 2-D flame ball-vortex interaction (convection-diffusion-reaction) problem. The numerical results clearly indicate that the adap-

tive algorithm adapts computational grids to the evolving structures of the solutions. Subsequently, the number of degrees of freedom varies with the demand of the solution for a fixed threshold value. We observe that the circular flame ball becomes distorted as it rolls up the center of the vortex and eventually forms a snail-like structure. The numerical results are in good agreement with those previously published.

Note that the discussion given above corresponds to the adaptive method that utilizes the interpolating wavelet constructed from the MRA. A similar adaptive algorithm outlined in [70, 89] that utilizes higher-dimensional interpolating wavelets constructed from MRA- $d$ . Note that such basis function is a tensor product of 1-D interpolating wavelets in a strict sense (they are simply basis functions of  $(V_{j_0} \oplus W_{j \geq j_0}^J) \otimes (V_{j_0} \oplus W_{j > j_0}^J) \otimes \cdots \otimes (V_j \oplus W_{j \geq j_0}^J)$ , while in MRA they are basis of  $\mathbf{W}_j = \mathbf{V}_{j+1} \ominus \mathbf{V}_j$ , where  $\mathbf{V}_j = V_j \otimes V_j \cdots \otimes V_j$ ). As a consequence, important ingredients, such as, a fast wavelet transform, its inverse, and the derivative approximation on a  $d$ -D irregular grid reduce to the application of 1-D algorithms to the data in each direction sequentially. This results in elegant algorithms (in term of extension to any spatial dimension). Note that for the same number of degrees of freedom  $N$ , the fast wavelet transform, its inverse, and the derivative approximation for MRA- $d$  are computationally more expensive than those of MRA. However, in general, the two approaches do not yield the same number of degrees of freedom. We verify the implementation of MRA- $d$  in the current computer code by testing it on a 2-D Poisson problem. The MRA- $d$  approach is also applied to simulate the flame ball problem. The numerical results obtained are in good agreement with those of obtained from the MRA approach.

The adaptive method is extended to obtain adaptive solutions of incompress-

ible flows in 2- and 3-D spatial dimensions. In this study, we consider the incompressible Navier-Stokes equations (with the Boussinesq approximation) in the primitive variable formulation. The use of the adaptive algorithm for time-dependent problems is applicable owing to a Chorin-type projection technique for time discretization. Note that the projection method decouples the pressure and velocity by splitting the problem into convection-diffusion and Poisson numerical subproblems. The adaptive technique using the MRA approach is used to simulate the 2-D flow in a lid-driven cavity problem with Reynolds number ranging from 400 to 3200, the 2-D differentially-heated cavity with large Rayleigh number  $10^6$  to  $5 \times 10^8$ , and in the 3-D differentially-heated cavity with Rayleigh numbers ranging from  $Ra = 10^3$  to  $10^5$ . The results clearly demonstrate that the adaptive algorithm adapts the resolution of computational grids according to the flow structures. By comparing specific quantities, the numerical results obtained are found to be in excellent agreement with the most accurate results available in the literature. Application of the adaptive method to flow in the 2-D differentially-heated cavity for  $5 \times 10^8$  (which is known to be chaotic) demonstrates the feasibility of using the present algorithm to simulate complicated flows.

In summary, the contributions of the present work are:

- An adaptive algorithm for solving PDEs in  $d$ -dimension is developed. The connection between the order of wavelets, the order of finite differences, the threshold values, and the accuracy of the numerical solution is numerically investigated. It is found that the relationships between these quantities behave similarly to those associated with the sparse wavelet approximation of a function and with derivative approximation on irregular an grid (with a larger constant).

- The extension of the adaptive wavelet method to problems defined on more complicated domains by means of domain transformation techniques is developed. Here, we use an algebraic grid generation of transfinite interpolation for grid generation.
- We consider a simple Jacobi preconditioning technique, similar to multilevel preconditioning, for elliptic problems.
- We apply the adaptive technique(s) to solve a flame ball-vortex interaction (a diffusion-convection-reaction) problem which constitutes a challenging problem due to the evolving thin layer(s) in the temperature profile.
- We present an accurate and robust adaptive wavelet algorithm for the solution of viscous incompressible flows. The algorithm consists of a Chorin projection with an implicit second order scheme, with linearization of the nonlinear term, to reduce the original problem to subproblems of convection-diffusion and Poisson-Neumann subproblems. A variant of a least-square solver for solving the Poisson-Neumann is presented. The numerical results on 2-D and 3-D flow problems demonstrate that the adaptive algorithm is very promising.

## 6.2 Recommendations for future work

Research that would immediately improve the current algorithm is listed below:

- A better approach for the solution of the Poisson-Neumann problem required in the projection step of the incompressible flow solver must be considered. The main issue is a proper and efficient means for modifying the linear system such that it is solvable (the right hand side of the linear system arising

from discretization of the Poisson-Neumann problem is not in the column space of the linear system; therefore, the right hand side of the system must be modified such that the system is solvable. In this work, we compute the left null vector explicitly and subsequently use it to modify the right hand side vector so that it belongs to the column space (see Section 5.4). Although this approach allows us to develop a robust algorithm (implying that the approach is proper), it is rather expensive; it is as expensive (as it usually requires in general slightly more iterations in the preconditioned-TFQMR or BiCGTAB solver) as solving the (solvable) linear system with modified right hand side.

- A better strategy for defining the neighboring region should be considered in order to further reduce the number of degrees of freedom.
- Elliptic grid generation (although much more expensive) may be considered as an alternative of the algebraic grid generation of transfinite interpolation (which is known to produce folds in the transformation when geometries are complex).
- Take advantage of available parallel linear solvers to (potentially) speed up the solution of linear systems.

The following task (which are challenging in their own right) should also be considered:

- Domain decomposition techniques to provide more versatility to treat complicate geometries, as well as to achieve scalable parallelization and accompanying for load balancing (with techniques such as space-filling curve [115]) in parallel computing architectures.

- Extend the algorithm to solve low-Mach number and compressible (reactive) equations in  $d$ -dimensions.



## BIBLIOGRAPHY

1. R. Abgrall and A. Harten. Multiresolution representation in unstructured meshes. *SIAM Journal on Numerical Analysis*, 35(6):2128–2146, 1994.
2. K. Amaratunga, J. R. Williams, S. Qian, and J. Weiss. Wavelet-galerkin solutions for one-dimensional partial equations. *International Journal for Numerical Methods in Engineering*, 37:2703.
3. P. Angot, C.-H. Bruneau, and P. Fabrie. A penalization method to take into account obstacles in incompressible viscous flows. *Numerische Mathematik*, 81:497–520, 1999.
4. O. Axelsson. *Iterative Solution Methods*. Cambridge University Press, 1994.
5. A. Barinka, T. Barsch, P. Charton, S. Dahlke, W. Dahmen, and K. Urban. Adaptive wavelet schemes for elliptic problems: Implementation and numerical experiments. *SIAM Journal on Scientific Computing*, 23:910–939, 2001.
6. A. Barinka, W. Dahmen, and R. Schneider. Fast computation of adaptive wavelet expansions. Ipgm report, RWTH Aachen, 2004.
7. R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. M. Danato, J. Dongara, V. Eijkhout, P. R. C. Romine, and H. V. der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, 1994.
8. M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82:64–68, 1989.
9. S. Berrone and L. Emmel. A realization of a wavelet galerkin method on nontrivial domains. *Mathematical Models and Methods in Applied Sciences*, 12:1525–1554, 2002.
10. S. Berrone and K. Urban. Adaptive Wavelet Galerkin Methods on Distorted Domains: Setup of the Algebraic System. IMPGM Report 178, RWTH Aachen, 1999.

11. S. Bertoluzza. Adaptive wavelet collocation method for the solution of burgers equation. *Transport Theory and Statistical Physics*, 25:339–359, 1996.
12. S. Bertoluzza. Adaptive wavelet collocation for the solution of steady-state equations. In *Wavelet Applications II: 17-21 April 1995, Orlando, Florida*, volume 2491 of *Proceedings of SPIE-the International Society for Optical Engineering*, pages 947–956, 1995.
13. S. Bertoluzza. An adaptive collocation method based on interpolating wavelets. In W. D. et al., editor, *Multiscale Wavelet Methods for Partial Differential Equations*, volume 6 of *Wavelet Analysis and Its Applications*, pages 109–135. Academic Press, 1997.
14. S. Bertoluzza and G. Naldi. A wavelet collocation method for the numerical solution of partial differential equation. *Applied and Computational Harmonic Analysis*, 25:339–359, 1996.
15. S. Bertoluzza, G. Naldi, and J. C. Ravel. Wavelet methods for the numerical solution of boundary value problems on the interval. In C. K. C. et al., editor, *Wavelets: Theory, Algorithms, and Applications*, volume 5, pages 425–448. Academic Press Inc., 1994.
16. S. Bertoluzza, C. Canuto, and K. Urban. On the adaptive computation of integrals of wavelets. *Applied Numerical Mathematics*, 34:13–38, 2000.
17. G. Beylkin. On wavelet-based algorithm for solving differential equation. In J. J. Benedetto and M. W. Frazier, editors, *Wavelets: Mathematics and Applications*, pages 449–466. CRC Press, 1994.
18. G. Beylkin. On the representation of operators in bases of compactly supported wavelets. 6:1716–1740, 1992.
19. B. L. Bihari and A. Harten. Multiresolution schemes for conservation laws with viscosity. *Journal of Computational Physics*, 35(6):315–354, 1997.
20. H. Bockhorn, J. Fröhlich, and K. Schneider. An adaptive wavelet-vaguelette algorithm for the computational of flame ball. *Combustion and Modelling*, 3:177–198, 1999.
21. O. Botella and R. Peyret. Benchmark spectral results on the lid-driven cavity flow. *Computers & Fluids*, 27:421–433, 1998.
22. R. W. Briley. Numerical method for predicting three-dimensional steady viscous flow in ducts. *Journal of Computational Physics*, 14:13–, 1974.

23. W. Cai and J. Wang. Adaptive wavelet collocation methods for initial value boundary problems of nonlinear pdes. ICASE Report 93-48, NASA, 1993.
24. W. Cai and J. Wang. Adaptive multiresolution collocation methods for initial boundary value problems of nonlinear pdes. *SIAM Journal on numerical Analysis*, 33:937–970, 1996.
25. W. Cai and W. Zhang. An adaptive spline wavelet adi (sw-adi) method for two-dimensional reaction-diffusion equations. *Journal of Computational Physics*, 139:92–126, 1998.
26. C. Canuto, A. Tabacco, and K. Urban. Numerical solution of elliptic problems by the wavelet element method, 1997.
27. C. Canuto, A. Tabacco, and K. Urban. The wavelet element method, part i. construction and analysis. *Applied and Computational Harmonic Analysis*, 6:1–52, 1999.
28. C. Canuto, A. Tabacco, and K. Urban. The wavelet element method, part ii. realization and additional features in 2d and 3d. 8:123–165, 2000.
29. P. Charton and V. Perrier. A pseudo-wavelet scheme for the two-dimensional navier-stokes equation. *Computational and Applied Mathematics*, 15:139–160, 1996.
30. D. R. Chenoweth and S. Paolucci. Natural convection in an enclosed vertical air layer with large horizontal temperature differences. *Journal of Fluid Mechanics*, 169:173–210, 1986.
31. G. Chiavassa and R. Donat. Point value multiscale algorithms for 2d compressible flows. *SIAM Journal on Scientific Computing*, 23(3):805–823, 1997.
32. G. Chiavassa and J. Liandrat. On the effective construction of compactly supported wavelets satisfying homogeneous boundary conditions on the interval. *Applied Computational Harmonic Analysis*, 4:62–73, 1997.
33. M. A. Christon, P. M. Gresho, and S. B. Sutton. Computational predictability of time-dependent natural convection flows in enclosures (including a benchmark solution). *International Journal for Numerical Methods in Fluids*, 40:953–980, 2002.
34. A. Cohen. Adaptive methods for pde’s - wavelets or mesh refinement ? Proceedings of the International Conference of Mathematics (Beijing 2002), 2002.

35. A. Cohen and R. Masson. Wavelet adaptive method for second order elliptic problems: Boundary conditions and domain decomposition. *Numerische Mathematik*, 86:193–238, 2000.
36. A. Cohen, I. Daubechies, and J. C. Feauveau. Biorthogonal bases of compactly supported wavelets, 1992.
37. A. Cohen, I. Daubechies, and P. Vial. Wavelets on the interval and fast wavelet transform. *Applied Computational Harmonic Analysis*, 1:54–81, 1993.
38. A. Cohen, W. Dahmen, I. Daubechies, and R. Devore. Tree approximation and optimal encoding. IPGM Report No. 174, RWTH Aachen, 1999.
39. A. Cohen, W. Dahmen, and R. Devore. Adaptive wavelets methods for elliptic operator equations: Convergence rates. *Mathematics of Computation*, 70:27–75, 2000.
40. A. Cohen, S. Kaber, S. Müller, and M. Postel. Fully adaptive multiresolution finite volume schemes for conservation laws. *Mathematics of Computation*, 72:183–225, 2002.
41. A. Cohen, W. Dahmen, and R. DeVore. Sparse evaluation of compositions of function using multiscale expansion. *SIAM J. Math. Anal.*, 35:1230–1262, 2003.
42. S. Dahlke and I. Weinreich. Wavelet-galerkin methods: An adapted biorthogonal wavelet basis. *Constructive Approximation*, 9:237–262, 1993.
43. S. Dahlke, W. Dahmen, R. Hochmuth, and R. Schneider. Stable multiscale bases and local error estimation for elliptic problems. *Applied Numerical Mathematics*, 23:21–47, 1997.
44. S. Dahlke, R. Hocmuth, and K. Urban. Adaptive wavelet methods for saddle point problems. IPGM Report No. 170, RWTH Aachen, 1999.
45. W. Dahmen. Wavelet and multiscale methods for operator equations. *Acta Numerica*, 6:55–228, 1997.
46. W. Dahmen. Wavelet methods for pdes – some recent development. *Journal of Computational and Applied Mathematics*, 128:133–185, 2001.
47. W. Dahmen and A. Kunoth. Appending boundary conditions by lagrange multiplier: Analysis of LBB condition. *Numerische Mathematik*, 88:9–42, 2001.

48. W. Dahmen and A. Kunoth. Multilevel preconditioning. *Numerische Mathematik*, 63:315–344, 1992.
49. W. Dahmen and R. Schneider. Composite wavelets bases for operator equations. *Mathematics of Computation*, 68:1533–1567, 1999.
50. W. Dahmen, S. Prössdorf, and R. Schneider. Multiscale methods for pseudodifferential equations. In L. L. Schumaker and G. Webb, editors, *Recent Advances in Wavelet Analysis*, pages 191–235. Academic Press, 1994.
51. W. Dahmen, A. Kunoth, and R. Schneider. Wavelet least squares methods for boundary value problems. IPGM Report No. 164, RWTH Aachen, 1999.
52. W. Dahmen, R. Schneider, and Y. Xu. Nonlinear functions of wavelet expansion: Adaptive reconstruction and fast evaluation. *Numerische Mathematik*, 86:49–101, 2000.
53. W. Dahmen, A. Kunoth, and J. Vorloeper. Convergence of adaptive wavelet methods for goal-oriented error estimation. Ipgm report, RWTH Aachen, 2006.
54. I. Daubechies. *Ten Lectures on Wavelets*. SIAM, Philadelphia, PA, 1992.
55. G. De Vahl Davis. Natural convection of air in a square cavity: a benchmark numerical solution. *International Journal for Numerical Methods in Fluids*, 3:249–264, 1983.
56. G. Deslauriers and S. Dubuc. Symmetric iterative interpolation processes. *Constructive Approximation*, 5:49–68, 1989.
57. A. Díaz. *International Journal for Numerical Methods in Engineering*, 44: 1599–1616, 1999.
58. D. Donoho. Interpolating Wavelet Transform. Report, Department of Statistics, Stanford University, 1992.
59. M. Farrashkhalvat and J. P. Miles. *Basic structured grid generation*. Butterworth Heinemann, 2003.
60. C. A. J. Fletcher. *Computational Techniques for Fluid Dynamics, Volume 2*. Springer-Verlag, second edition, 1991.
61. J. Fröhlich and K. Schneider. An adaptive wavelet galerkin algorithm for one-dimensional and 2-dimensional flame computations. *European Journal of Mechanics B-Fluids*, 13:439–471, 1994.

62. J. Fröhlich and K. Schneider. An adaptive wavelet-vaguelette algorithm for the solution of pdes. *Journal of Computational Physics*, 130:174–190, 1997.
63. T. Fusegi, J. M. Hyun, K. Kuwahara, and B. Farouk. A numerical study of three-dimensional natural convection in a differentially heated cubical enclosure. *International Journal of Heat and Mass Transfer*, 34:1543–1557, 1991.
64. U. Ghia, K. N. Ghia, and C. T. Shin. High-Re solutions for incompressible flow using Navier-Stokes equations and a multigrid method. *Journal of Computational Physics*, 48:387–411, 1982.
65. R. Glowinski, W. Lawton, M. Ravachol, and E. Tenenbaum. Wavelet solutions of linear and nonlinear elliptic, parabolic and hyperbolic problems in one space dimension. In R. Glowinski, editor, *Computing Methods in Applied Sciences and Engineering*, pages 55–120. SIAM, Philadelphia, PA, .
66. R. Glowinski, T. W. Pan, R. O. W. Jr., and X. Zhou. Wavelet and finite element solutions for the dirichlet problem. Technical report, .
67. R. Glowinski, T. W. Pan, R. O. W. Jr., and X. Zhou. Wavelet and finite element solutions for the neumann problem using fictitious domains. *Journal of Computational Physics*, 126:40–51, 1996.
68. W. J. Gordon and C. A. Hall. Construction of curvilinear coordinate systems and application to mesh generation. *International Journal for Numerical Methods in Engineering*, 7:461–477, 1973.
69. P. M. Gresho, S. T. Chan, R. L. Lee, and C. D. Upson. A modified finite element method for solving the time-dependent, incompressible navier-stokes equations .2. applications.
70. M. Griebel. Adaptive sparse grid multilevel methods for elliptic pdes based on finite differences. *Computing*, 61:151–180, 1998.
71. M. Griebel and F. Koster. Adaptive wavelet solvers for the unsteady incompressible Navier-Stokes equations. In M. J, editor, *Advances in Mathematical Fluid Mechanics*, pages 67–118. Springer Verlag, 2000.
72. M. Griebel and F. Koster. Multiscale methods for the simulation of turbulent flows. In e. E.H. Hirschel, editor, *Numerical Flow Simulation III*.
73. M. Griebel, T. Dornseifer, and T. Neunhoeffter. *Numerical Simulation in Fluid Dynamics: A Practical Introduction*. SIAM, Philadelphia, PA, 1998.

74. A. Harten. Multiresolution algorithms for the numerical solution of hyperbolic conservation laws. *Communications on Pure and Applied Mathematics*, 48:1305–1342, 1995.
75. W. D. Henshaw. A fourth-order accurate method for the incompressible navier-stokes equations on overlapping grids. *Journal of Computational Physics*, 113:13–25, 1994.
76. M. Holmström. Solving hyperbolic PDEs using interpolating wavelets. *SIAM Journal on Scientific Computing*, 21:405–420, 1999.
77. S. Hugues and A. Randriamampianina. An improved projection scheme applied to pseudospectral methods for the incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, 28:501–521, 1998.
78. S. Jaffard. Wavelet methods for fast resolution of elliptic problems. *SIAM Journal on Numerical Analysis*, 29:956–986, 1992.
79. L. Jameson. On the Daubechies-Based Wavelet Differentiation Matrix. ICASE Report 93-95, NASA, 1993.
80. L. Jameson. On the spline-based wavelet differentiation matrix. ICASE Report No. 93-80, 1993.
81. L. Jameson. On the wavelet optimized finite difference method. ICASE Report No. 94-9, 1994.
82. L. Jameson. A wavelet-optimized, very high order numerical method. *SIAM Journal of Scientific Computing*, 19:1980–2013, 1998.
83. R. Janssen. *Instabilities in natural-convection flows in cavities*. PhD Thesis, Technische Universiteit Delft, Delft, The Netherlands, 1994.
84. B. Jawerth and W. Sweldens. An overview of wavelet based multiresolution analyses. *SIAM Review*, 36:337–412, 1994.
85. N. A. Kelson. The laminar boundary layer regime for natural convection of air in a square cavity. Report 1990/FMT/3, School of Mechanical and Industrial Engineering, University of New South Wales, 1990.
86. N. Kevlahan and O. V. Vasilyev. An adaptive wavelet collocation method for fluid-structure interaction at high Reynolds numbers. *SIAM Journal on Scientific Computing*, 26(6):1894–1915, 2005.
87. E. Kita and N. Kamiya. *Eng. Anal. Bound. Elem.*, 25:479–495, 2001.



88. F. Koster. A proof of the consistency of the finite difference technique on sparse grids. *Computing*, 65:247–261, 2000.
89. F. Koster. *Multiskalen-basierte Finite Differenzen Verfahren auf adaptiven dünnen Gittern*. Phd thesis, Institut für Angewandte Mathematik, Universität Bonn, 2002.
90. F. Koster. Preconditioners for sparse grid discretizations. preprint, 2001.
91. F. Koster and M. Griebel. Orthogonal wavelets on the interval. preprint, 1998.
92. A. Kunoht. *Wavelet Methods for Minimization Problems Involving Elliptic Partial Differential Equations*. PhD Thesis, RWTH Aachen, 1999.
93. A. Latto and E. Tenenbaum. Compactly supported wavelets and the numerical-solution of burgers equation. *C.R. Acad. Sci. Paris, Series I-Mathematique*, 311:903–909, 1990.
94. G. Lauriat and I. Altimir. A new formulation of the sadi method for the prediction of natural convection flows in cavities. *Computers Fluids*, 13:141, 1983.
95. P. Le Quéré. Accurate solutions to the square thermally driven cavity at high Rayleigh number. *Computers Fluids*, 20:29–41, 1991.
96. P. Le Quéré and M. Behnia. From onset of unsteadiness to chaos in a differentially heated square cavity. *Journal of Fluid Mechanics*, pages 81–107, 1998.
97. L. Levaggi and A. Tabacco. Wavelets on interval and related topic. *Rend. Sem. Mat. Univ. Pol. Torino*, 57:123–160, 1999.
98. J. Liandrat. Some wavelets algorithms for partial differential equations. In G. E. et al., editor, *Wavelets: Theory and Applications*. Oxford University Press, 1996.
99. J. Liandrat and P. Tchamitchian. Resolution of the 1d regularized burgers equation using a spatial wavelet approximation. ICASE Report 90–83, NASA, 1990.
100. J. Liandrat and P. Tchamitchian. On the fast approximation of some nonlinear operators in nonregular space. *Advances in Computational Mathematics*, 8:179–192, 1998.



101. J. Liandrat, V. Perrier, and P. H. Tchamitchian. Numerical resolution of non-linear partial differential equations using the wavelet approach. In M. B. R. et al., editor, *Wavelets and Their Applications*. Jones and Bartlett Publishers, 1992.
102. J. Mackerle. Error estimates and adaptive finite element methods: A bibliography (1990 - 2000). *Engineering Computations*, 18:802–914, 2001.
103. Y. Maday, V. Perrier, and J.-C. Ravel. Dynamically adaptive using wavelets basis for the approximation of partial differential equations. *C. R. Acad. Sci. Paris, t 312, Série I*, pages 405–410, 1991.
104. Y. Meyer. *Wavelet and Operators: Volume 1*. Cambridge University Press, 1995.
105. W. F. Mitchell. A comparison of adaptive refinement techniques for elliptic problems. *ACM Transactions on Mathematical Software*, 15:326–347, 1989.
106. P. Monasse and V. Perrier. Orthonormal wavelet bases adapted for partial differential equations with boundary conditions. *SIAM Journal on Mathematical Analysis*, 29:1040–1065, 1998.
107. S. Paolucci and D. R. Chenoweth. Transition to chaos in a differentially heated veridical cavity. *Journal of Fluid Mechanics*, 201:379–410, 1989.
108. C. Pozrikidis. A note on the regularization of the discrete poisson-neumann problem. *Journal of Computational Physics*, 172:917–923, 2001.
109. S. Qian and J. Weiss. Wavelets and the numerical solution of partial differential equations. *Journal of Computational Physics*, 106:155–175, 1993.
110. Y. Rastigejev. *Multiscale Computations with a Wavelet Adaptive Algorithm*. PhD Thesis, University of Notre Dame, Notre Dame, IN, 2002.
111. Y. Rastigejev and S. Paolucci. Wavelet based adaptive multiresolution computation of viscous reactive flows. *International Journal for Numerical Methods in Fluids*, 52(7):749–784, 2005.
112. O. Roussel and K. Schneider. An adaptive multiresolution method for combustion problems: application to flame ball-vortex interaction, 2005.
113. O. Roussel, K. Schneider, A. Tsigulin, and H. Bockhorn. A conservative fully adaptive multiresolution algorithm for parabolic pdes. *Journal of Computational Physics*, 188(2):493–523, 2003.
114. Y. Saad. *SPARSKIT: a basic tool kit for sparse matrix computations*, version 2 edition, 1994. See also URL <http://www-users.cs.umn.edu/~saad>.

115. H. Sagan. *Space-Filling Curves*. Springer, 1994.
116. N. Saito and G. Beylkin. Multiresolution representations using the autocorrelation functions of compactly supported wavelets. *IEEE Transactions on Signal Processing*, 41:3584–3590, 1993.
117. K. Schneider and M. Farge. Numerical simulation of a mixing layer in an adaptive wavelet basis. *C. R. Acad. Sci. Paris, t. 328, Série Iib*, pages 263–269, 2000.
118. R. L. Schult and H. W. Wyld. Using wavelets to solve the burgers equations: A comparative study. *Physical Review A*, 46:7953–7958, 1992.
119. J. Shen. Hopf-bifurcation of the unsteady regularized driven cavity flow. *Journal of Computational Physics*, 95:228–245, 1991.
120. B. Sjögreen. Numerical experiments with the multiresolution scheme for the compressible Euler equations. *Journal of Computational Physics*, 117: 251–261, 1995.
121. R. P. Stevenson. Piecewise linear (pre-) wavelets on non-uniform meshes. In W. Hackbusch and G. Wittum, editors, *Multigrid Methods V*, volume 3 of *LNCSE*. Springer, Heidelberg, 1998.
122. S. A. Suslov and S. Paolucci. A Petrov-Galerkin method for the direct simulation of fully enclosed flows. volume HTD-335 of *Proceedings of the ASME Heat Transfer Division, Volume 4*, 1996.
123. W. Sweldens. *The Construction and Application of Wavelets in Numerical Analysis*. Phd thesis, Katholieke Univiersiteit Leuven, Belgium, 1994.
124. W. Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM J. Math. Anal.*, 29(2):511–546, 1997.
125. P. H. Tchamitchian. Wavelets, functions, and operators. In G. E. et al., editor, *Wavelets: Theory and Applications*. Oxford University Press, 1996.
126. J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin. *Numerical Grid Generation: Foundation and Application*. Elsevier Science Publishing Co., Inc., 1985.
127. C. H. Tong, T. F. Chan, and C. C. J. Kuo. A domain decomposition preconditioner base on a change to a multilevel nodal basis. *SIAM Journal on Scientific and Statistical Computing*, 12, 1991.
128. L. N. Trefethen. *Spectral Methods*. SIAM, Philadeldphia, PA, 2000.

129. E. Tric, G. Labrosse, and M. Bertrouni. A first incursion into the 3d structure of natural convection of air in a differentially heated cubic cavity from accurate numerical solutions. *International Journal of Heat and Mass Transfer*, 43:4043–4056, 2000.
130. C. D. Upson, P. M. Gresho, and R. L. Lee. Finite-element simulation of thermally induced convection in an enclosed cavity. Report UCID-18602, Lawrence Livermore National Laboratory, 1980.
131. S. P. Vanka. Block-implicit multigrid solution of navier-stokes equations in primitive variables. *Journal of Computational Physics*, 65:138–158, 1986.
132. O. V. Vasilyev. Solving multi-dimensional evolution problems with localized structures using second generation wavelets. *International Journal of Computational Fluid Dynamics*, 17:151–168, 2003.
133. O. V. Vasilyev and S. Paolucci. Dynamically adaptive multilevel wavelet collocation method for solving partial differential equations in a finite domain. *Journal of Computational Physics*, 125:498–512, 1996.
134. O. V. Vasilyev and S. Paolucci. A fast adaptive wavelet collocation algorithm for multidimensional PDEs. *Journal of Computational Physics*, 138:16–56, 1997.
135. S. Wakashima and T. S. Saitoh. Benchmark solutions for natural convection in a cubic cavity using the high-order time-space method. *International Journal of Heat and Mass Transfer*, 47:853–864, 2004.
136. J. Waldén. Filter bank methods for hyperbolic pdes. *Siam Journal on Numerical Analysis*, 36:1183–1233, 1999.
137. F. A. Williams. *Combustion theory*. Addison Wesley, second edition, 1984.
138. D. Wirasaet and S. Paolucci. An adaptive wavelet method for incompressible flows in complex domains. *Journal of Fluids Engineering*, 127:656–665, 2005.
139. D. Wirasaet and S. Paolucci. Application of an adaptive wavelet method to natural-convection flow in a differentially heated cavity. volume HTD2005-72864 of *Proceedings of the ASME Heat Transfer Conference*, 2005.
140. D. Wirasaet and S. Paolucci. The application of an adaptive wavelet method to the 3-d natural-convection flow in a differentially heated cavity. volume IMECE2006-16095 of *Proceedings of the ASME international Mechanical Engineering Congress and Exposition*, 2006.

- 141. J.-C. Xu and W.-C. Shann. Galerkin-wavelet methods for two-point boundary value problems. *Numerische Mathematik*, pages 123–144, 1992.
- 142. H. Yserentant. On the multi-level splitting of finite element spaces. *Numerische Mathematik*, 49, 1986.

*This document was prepared & typeset with L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>, and formatted with  
NDdiss2<sub>ε</sub> classfile (v3.0[2005/07/27]) provided by Sameer Vijay.*